Thomas Johansson
Phong Q. Nguyen (Eds.)

LNCS 7881

Advances in Cryptology – EUROCRYPT 2013

32nd Annual International Conference
on the Theory and Applications of Cryptographic Techniques
Athens, Greece, May 2013, Proceedings

INTERNATIONAL ASSOCIATION FOR CRYPTOLOGIC RESEARCH

i a c r

Springer

Thomas Johansson   Phong Q. Nguyen (Eds.)

Advances in Cryptology – EUROCRYPT 2013

32nd Annual International Conference
on the Theory and Applications of Cryptographic Techniques
Athens, Greece, May 26-30, 2013
Proceedings

Springer

Volume Editors

Thomas Johansson
Lund University
Dept. of Electrical and Information Technology
P.O. Box 118, 221 00 Lund, Sweden
E-mail: thomas.johansson@eit.lth.se

Phong Q. Nguyen
Ecole normale supérieure
Dépt. d'informatique
45, rue d'Ulm, 75230 Paris Cedex 05, France
E-mail: phong.nguyen@ens.fr

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-38347-2 e-ISBN 978-3-642-38348-9
DOI 10.1007/978-3-642-38348-9
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2013937954

CR Subject Classification (1998): E.3, F.2, D.4.6, K.6.5, G.2, C.2, J.1

LNCS Sublibrary: SL 4 – Security and Cryptology

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

These are the proceedings of Eurocrypt 2013, the 32nd annual IACR Eurocrypt conference on the theory and applications of cryptographic techniques. The conference was held May 26–30, 2013, in Athens, Greece, and sponsored by the International Association for Cryptologic Research (IACR). The General Chair was Aggelos Kiayias, from the University of Athens. The Eurocrypt 2013 Program Committee (PC) consisted of 33 members. There were 202 papers submitted to the conference, of which one was eventually withdrawn. Each paper was assigned to at least three PC members, while submissions co-authored by PC members were reviewed by at least five other PC members. Papers were refereed anonymously. There were indeed a large number of high-quality submissions and the review process was a challenge. The PC was helped by reports from more than 250 external reviewers, producing a total of more than 670 reviews in all. After the reviews were submitted, the PC discussed the reviews for many weeks, before making a final decision. All of our deliberations were aided by the Web Submission and Review Software written by Shai Halevi and the server was hosted by IACR. We would like to thank Shai for setting up the service on the server and helping us whenever needed.

The PC eventually selected 41 submissions for presentation during the conference and these are the articles that are included in this volume. Two of the accepted papers came from merging two pairs of submitted papers. Note that these proceedings contain the revised versions of the selected papers. Since the revisions were not checked again before publication, the authors (and not the committee) bear full responsibility for the contents of their papers. The PC decided to give the Best Paper Award to Sanjam Garg, Craig Gentry, and Shai Halevi for their paper "Candidate Multilinear Maps from Ideal Lattices." We were greatly honored that the conference program included the 2013 IACR Distinguished Lecture talk by Eli Biham, entitled "How to Make a Difference: 25 Years of Differential Cryptanalysis," as well as an invited lecture by the designers of Keccak (Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche). We would like to thank them all for accepting our invitation and for their contribution to the program of Eurocrypt 2013.

We wish to thank all the authors who submitted their papers. The hard task of reading, commenting, debating, and eventually selecting the papers to be accepted for the conference fell on the PC members. We are very grateful to all the committee members and all external reviewers for their hard and conscientious work. It has been a great honor to chair the PC for Eurocrypt 2013.

May 2013

Thomas Johansson
Phong Q. Nguyen

Eurocrypt 2013

General Chair

Aggelos Kiayias	University of Athens, Greece

Program Chairs

Thomas Johansson	Lund University, Sweden
Phong Nguyen	INRIA, France and Tsinghua University, China

Program Committee

Frederik Armknecht	Universität Mannheim, Germany
Andrey Bogdanov	KU Leuven, Belgium
Melissa Chase	Microsoft Research, USA
Jung Hee Cheon	Seoul National University, Korea
Steven Galbraith	University of Auckland, New Zealand
Rosario Gennaro	City College of New York, USA
Louis Goubin	Université Versailles Saint-Quentin, France
Vipul Goyal	Microsoft Research, India
Jens Groth	University College London, UK
Martin Hirt	ETH Zurich, Switzerland
Jonathan Katz	University of Maryland, USA
Nathan Keller	Bar Ilan University, Israel
Dmitry Khovratovich	Microsoft Research, USA
Eike Kiltz	Ruhr University Bochum, Germany
Xuejia Lai	Shanghai Jiao Tong University, China
Gregor Leander	Technical University of Denmark
Arjen K. Lenstra	EPFL, Switzerland
Gaëtan Leurent	Université du Luxembourg
Vadim Lyubashevsky	INRIA and ENS, France
Subhamoy Maitra	Indian Statistical Institute
Daniele Micciancio	UCSD, USA
Jesper Buus Nielsen	Aarhus University, Denmark
Miyako Ohkubo	NICT, Japan
Kenny Paterson	Royal Holloway, University of London, UK
Giuseppe Persiano	Università di Salerno, Italy
Leonid Reyzin	Boston University, USA
Matt Robshaw	Orange Labs, France

Phil Rogaway	UC Davis, USA
Yu Sasaki	NTT, Japan
Yannick Seurin	ANSSI, France
Abhi Shelat	University of Virginia, USA
Nigel P. Smart	University of Bristol, UK
John P. Steinberger	Tsinghua University, China

Additional Reviewers

Michel Abdalla
Masayuki Abe
Martin Albrecht
Joël Alwen
Prabhanjan Ananth
Elena Andreeva
Gilad Asharov
Gille van Assche
Jean-Philippe Aumasson
Gildas Avoine
Abhishek Banerjee
Subhadeep Banik
Rana Barua
Lejla Batina
Aurelie Bauer
Stephanie Bayer
Anja Becker
Mihir Bellare
David Bernhard
Rishiraj Bhattacharyya
Gaetan Bisson
Olivier Blazy
Julia Borghoff
Joppe Bos
Ioana Boureanu
Elette Boyle
Christina Brzuska
Sebastien Canard
Angelo De Caro
David Cash
Pyrros Chaidos
Andre Chailloux
Anupam Chattopadhyay
Shan Chen
Yuanmi Chen
Ashish Choudhury
Sherman Chow
Sherman S. M. Chow
Hee Won Chung
Sandro Coretti
Cas Cremers
Boureanu Ioana Cristina
Edouard Cuvelier
Dana Dachman-Soled
Özgür Dagdelen
Morten Dahl
Ivan Damgård
M. Prem Laxman Das
Gregory Demay
Ning Ding
Itai Dinur
Maria Dubovitskaya
Leo Ducas
Orr Dunkelman
Alex Escala
Pooya Farshim
Sebastian Faust
Nelly Fazio
Serge Fehr
Dario Fiore
Marc Fischlin
Tore Kasper Frederiksen
Georg Fuchsbauer
Eiichiro Fujisaki
Benjamin Fuller
Philippe Gaborit
Tommaso Gagliardoni
Sebastian Gajek
Nicolas Gama
Pierrick Gaudry
Craig Gentry
Essam Ghadafi
Phillip Gibbons
Benedikt Gierlichs
Zheng Gong
Sergey Gorbunov
Matthew Green
Divya Gupta
Shai Halevi
Fabrice Ben Hamouda
Gerhard Hancke
Kristiyan Haralambiev
Avinatan Hassidim
Jens Hermans
Jason Hinek
Viet Tung Hoang
Dennis Hofheinz
Hyunsook Hong
Jialin Huang
Yun Huang
Jialin Huang
Pavel Hubacek
Jim Hughes
Jung Yeon Hwang
William E. Skeith III
Vincenzo Iovino
Tetsu Iwata
Tibor Jager
Abhishek Jain
Thomas P. Jakobsen
Stanislaw Jarecki
Dimitar Jetchev
Min Young Jun
Charanjit Jutla
Aniket Kate
Aggelos Kiayias
Jihye Kim
Jinsu Kim

Sungwook Kim
Taechan Kim
Ilya Kizhvatov
Thorsten Kleinjung
Simon Knellwolf
Lars R. Knudsen
Markulf Kohlweiss
Matthias Krause
Soonhak Kwon
Martin Lauridsen
Hyung Tae Lee
Jooyoung Lee
Moon Sung Lee
Mun-Kyu Lee
Allison Lewko
Wei Li
Benoit Libert
Rachel Lin
Tingting Lin
Helger Lipmaa
Feng-Hao Liu
Jacob Loftus
Yu Long
Adriana Lopez-Alt
Stefan Lucks
Hans Löhr
Changshe Ma
Arpita Maitra
Lior Malka
Damian Markham
Takahiro Matsuda
Christian Matt
Sophie Mawet
Alex May
Sarah Meiklejohn
Florian Mendel
Xianrui Meng
Bart Mennink
Andrea Miele
Amir Moradi
Pratyay Mukherjee
Yusuke Naito
Ivica Nikolić
Ryo Nishimaki
Adam O'Neill
Tatsuaki Okamoto
Cristina Onete
Claudio Orlandi
David Oswald
Dan Page
Pascal Paillier
Omkant Pandey
Tapas Pandit
Omer Paneth
Charalampos Papamanthou
Periklis Papkonstantinou
Goutam Paul
Chris Peikert
Olivier Pereira
Milinda Perera
Edoardo Persichetti
Andreas Peter
Christiane Peters
Duong Hieu Phan
Krzysztof Pietrzak
Benny Pinkas
Stefano Piranio
Bertram Poettering
Joop van de Pol
Emmanuel Prouff
Baodong Qin
Ananth Raghunathan
Vanishree H. Rao
Pavel Raykov
Mariana Raykova
Christian Rechberger
Yanli Ren
Renato Renner
Alfredo Rial
Bill Rosgen
Yannis Rouselakis
Hansol Ryu
Carla Ràfols
Rei Safavi-Naini
Amit Sahai
Louis Salvail
Santanu Sarkar
Takakazu Satoh
Alessandra Scafuro
Christian Schaffner
Dominique Schroeder
Jacob Schuldt
Sven Schäge
Gil Segev
Jae Hong Seo
Anna Shcherbakova
Vladimir Shpilrain
Tom Shrimpton
Jamie Sikora
Alice Silverberg
Benjamin Smith
Francois-Xavier Standaert
Pantelimon Stanica
Douglas Stebila
Damien Stehle
John Steinberger
Ron Steinfeld
Marc Stevens
Koutarou Suzuki
Björn Tackmann
Katsuyuki Takashima
Keisuke Tanaka
Aris Tentes
Stefano Tessaro
Enrico Thomae
Susan Thomson
Mehdi Tibouchi
Elmar Tischhauser
Joana Treger
Daniel Tschudi
Vinod Vaikuntanathan
Serge Vaudenay
Daniele Venturi
Frederik Vercauteren
Vincent Verneuil
Ivan Visconti
Lei Wang
Bogdan Warinschi
Brent Waters
Carolyn Whitnall
Daniel Wichs
Michael Wiener
Peter Winkler

Christopher Wolf
Keita Xagawa
Hong Xu
Weijia Xu
Kan Yasuda
Lisa Yin
Kazuki Yoneyama
Aaram Yun
Mark Zhandry
Liangfeng Zhang
Zongyang Zhang
Yunlei Zhao
Vassilis Zikas
Angela Zottarel

Table of Contents

Candidate Multilinear Maps from Ideal Lattices

Sanjam Garg[1,⋆], Craig Gentry[2,⋆⋆], and Shai Halevi[2,⋆⋆]

[1] UCLA
[2] IBM Research

Abstract. We describe plausible lattice-based constructions with properties that approximate the sought-after multilinear maps in hard-discrete-logarithm groups, and show an example application of such multi-linear maps that can be realized using our approximation. The security of our constructions relies on seemingly hard problems in ideal lattices, which can be viewed as extensions of the assumed hardness of the NTRU function.

1 Introduction

Bilinear maps are extremely useful tools in cryptography. After being used to construct non-interactive key agreement [SOK00], tripartite Diffie-Hellman [Jou00], and identity-based encryption [BF01], the applications of bilinear maps have become too numerous to name. Boneh and Silverberg [BS03] argued that multilinear maps would have even more interesting applications, including multi-partite Diffie-Hellman and very efficient broadcast encryption. They attempted to construct multilinear maps from abelian varieties (extending known techniques for constructing bilinear maps), but they identified serious obstacles, and concluded that "such maps might have to either come from outside the realm of algebraic geometry, or occur as 'unnatural' computable maps arising from geometry".

Since then, the persistent absence of cryptographically useful multilinear maps as not stopped researchers from proposing applications of them. For example, Rückert and Schröder [RS09] use multilinear maps to construct efficient aggregate and verifiably encrypted signatures without random oracles. Papamanthou et al. [PTT10] show that compact multilinear maps give very efficient authenticated data structures. Rothblum [Rot13] uses multilinear maps to construct

⋆ Research conducted while at the IBM Research, T.J. Watson funded by NSF Grant No.1017660.

⋆⋆ This work was supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior National Business Center (DoI/NBC) contract number D11PC20202. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoI/NBC, or the U.S. Government.

a counterexample to the conjecture that all bit-encryption schemes are KDM-secure (secure when given bit-encryptions of the secret key).

Here, we construct multilinear maps from ideal lattices. Our multilinear maps are "noisy" and bounded to polynomial degree. For very high degree, the "noisiness" overwhelms the signal, somewhat like for ciphertexts in somewhat homomorphic encryption schemes. In light of their noisiness, one could say that our multilinear maps are indeed "unnatural" computable maps arising from geometry. Our candidate multilinear maps differ quite substantially from the "ideal" multilinear maps envisioned by Boneh and Silverberg, in particular some problems that are hard relative to contemporary bilinear maps are easy with our construction (see Section 4.2). Nonetheless, the multilinear analog of the decision Diffie-Hellman problem appears hard in our construction, which gives cause for optimism about its applications in cryptography. In this paper we only demonstrate the applicability of our candidate to the "obvious" application of multipartite Diffie-Hellman key exchange, but other applications are surly possible.

The boundedness of our encodings has interesting consequences, both positive and negative. On the positive side, it hinders an attack based on Boneh and Lipton's subexponential algorithm for solving the discrete logarithm in black box fields [BL96]. This attack cannot be used to solve the "discrete log" problem in our setting, since their algorithm requires exponentiations with exponential degree. On the negative size, the dependence between the degree and parameter-size prevents us from realizing applications such that Papamanthou et al. [PTT10] that needs "compact" maps. Similarly, so far we were not able to use our maps to realize Rothblum's counterexample to the KDM-security of bit encryption conjecture [Rot13]: That counterexample requires degree that is polynomial, but a polynomial that is always just out of our reach of our parameters.

The security of the multilinear-DDH problem in our constructions relies on new hardness assumptions, and we provide an extensive cryptanalysis to validate these assumptions. To make sure that our constructions are not "trivially" insecure, we prove that our constructions are secure against adversaries that merely run a straight-line program. We also analyze our constructions with respect to the best known averaging, algebraic and lattice attacks. Many of these attacks have been published before [CS97, HKL$^+$00, Gen01, GS02, Szy03, HGS04, NR06] in cryptanalysis of the NTRU [HPS01, HHGP$^+$03] and GGH [GGH97] signature schemes. We also present new attacks on *principal* ideal lattices, which arise in our constructions, that are more efficient than (known) attacks on general ideal lattices. Our constructions remain secure against all of the attacks that we present, both old and new. However, we feel that more cryptanalysis needs to be done, and this is partly why we have tried to write our cryptanalysis sections as a thorough survey that will serve as a useful starting point for cryptanalysts.

A Brief Overview. Our constructions work in polynomial rings and use principal ideals in these rings (and their associated lattices). In a nutshell, an instance of our construction has a secret short ring element $\mathbf{g} \in R$, generating a principal ideal $\mathcal{I} = \langle \mathbf{g} \rangle \subset R$. In addition, it has an integer parameter q and another secret $\mathbf{z} \in R/qR$, which is chosen at random (and hence is not small).

We think of a term like g^x in a discrete-log system as an "encoding" of the "plaintext exponent" x. In our case the role of the "plaintext exponents" is played by the elements in $R/\mathcal{I}$ (i.e. cosets of $\mathcal{I}$), and we "encode" it via division by $\mathbf{z}$ in R_q. In a few more details, our system provides many levels of encoding, where a level-i encoding of the coset $\boldsymbol{e}_\mathcal{I} = \boldsymbol{e} + \mathcal{I}$ is an element of the form $\boldsymbol{c}/\mathbf{z}^i \bmod q$ where $\boldsymbol{c} \in \boldsymbol{e}_\mathcal{I}$ is short. It is easy to see that such encodings can be both added and multiplied, so long as the numerators remain short. More importantly, we show that it is possible to publish a "zero testing parameter" that enables to test if two elements encode the same coset at a given level, without violating security (e.g., it should still be hard to compute x from an encoding of x at higher levels). Namely, we add to the public parameters an element of the form $\mathbf{p}_{zt} = h \cdot z^\kappa / g \bmod q$ for a not-too-large h, and show that multiplying an encoding of 0 by $\mathbf{p}_{zt}$ (mod q) yields a small element, while multiplying an encoding of a non-zero by $\mathbf{p}_{zt}$ (mod q) yields a large element. Hence we can distinguish zero from non-zero, and by subtraction we can distinguish two encodings of the same element from encodings of two different elements.

Our schemes are somewhat analogous to graded algebras, hence we sometimes call them *graded encoding schemes*. Our schemes are quite flexible, and for example can be modified to support the analog of asymmetric maps by using several different z's. On the other hand, other variants such as composite-order groups turn out to be insecure with our encodings (at least when implemented in a straightforward manner).

Organization. We define the general notion of encoding that we use in Section 2, as well an abstract notion of our main hardness assumption (which is a multilinear analog of DDH). Then in Section 3 we provide some background on ideal lattices, and in Section 4 we describe our construction.

Applications. In the full version [GGH12]we describe the application to multipartite key agreement. Using our multilinear maps [GGH$^+$13] have provided a construction of an attribute based encryption (ABE) scheme for general circuits. Concurrently and independently Gorbunov, Vaikuntanathan, and Wee [GVW13] also achieve ABE for circuits. One nice feature of their result is that they reduce security to the Learning with Errors (LWE) problem. Goldwasser, Kalai, Popa, Vaikuntanathan, and Zeldovich [GKP$^+$13] show how one can use such an ABE scheme along with fully homomorphic encryption to construct a succinct single use functional encryption scheme. This in turn implies results for reusable Yao garbled circuits and other applications. Subsequent to our work, using our multilinear maps, Garg, Gentry, Sahai, and Waters [GGSW13] constructed a witness encryption scheme where a user's decryption key need not be an actual "key" at all, but rather can be a witness for some arbitrary NP relation specified by the encrypter (the encrypter itself may not know a witness).

2 Multilinear Maps and Graded Encoding Systems

Below we define formally our notion of a "approximate" multilinear maps, which we call *graded encoding schemes* (termed after the notion of graded algebra). Before presenting our notion, we briefly recall *cryptographic multilinear maps* as defined by Boneh and Silverberg [BS03].

For $\kappa+1$ cyclic groups $G_1, \ldots, G_\kappa, G_T$ (written additively) of the same order p, a *κ-multilinear map* $e : G_1 \times \cdots \times G_\kappa \to G_T$ is non-degenerate (in the sense that if $\{g_i \in G_i\}_{i=1,\ldots,\kappa}$ are all generators of their respective groups, then $e(g_1, \ldots, g_\kappa)$ is a generator of G_T), and it satisfies

$$e(g_1, \ldots, \alpha \cdot g_i, \ldots, g_\kappa) = \alpha \cdot e(g_1, \ldots, g_\kappa),$$

for any elements $\{g_i \in G_i\}_{i=1,\ldots,\kappa}$, index $i \in [\kappa]$ and integer $\alpha \in \mathbb{Z}_p$. (Boneh and Silverberg considered in [BS03] only the *symmetric* case $G_1 = \cdots = G_\kappa$, the asymmetric case with different G_i's was considered, e.g., by Rothblum in [Rot13].)

Cryptographic multilinear maps come with efficient procedures for generating parameters of such κ-multilinear map and for computing the group operation in each group and the map itself. They also come with some hardness properties, at least the discrete logarithm must be hard in the respective groups. Other hardness assumptions include the multilinear-DDH (MDDH) assumption (among others), asserting that given $\kappa+1$ random elements in the source group, it is hard to compute (or even recognize) the target-group element whose discrete logarithm is the product of the logarithms of all these $\kappa+1$ elements.

2.1 Graded Encoding Schemes

The starting point for our new notion is viewing group elements in multilinear-map schemes as just a convenient mechanism of encoding the exponent: Typical applications of bilinear (or multilinear) maps use $\alpha \cdot g_i$ as an "obfuscated encoding" of the "plaintext integer" $\alpha \in \mathbb{Z}_p$. This encoding supports limited homomorphism (i.e., linear operations and a limited number of multiplications) but no more. In our setting we retain this concept of a somewhat homomorphic encoding, and have an algebraic ring (or field) R playing the role of the exponent space $\mathbb{Z}_p$. However we dispose of the multitude of algebraic groups, replacing them with "unstructured" sets of encodings of ring elements.

Perhaps the biggest difference between our setting and the setting of cryptographic multilinear maps, is that our encoding is randomized, which means that the same ring-element can be encoded in many different ways. (We do not even insist that the "plaintext version" of a ring element has a unique representation.) This means that checking if two strings encode the same element may not be trivial, indeed our constructions rely heavily on this check being feasible for some encodings and not feasible for others.

Another important difference is that our system lets us multiply not only batches of κ encodings at the time, but in fact any subset of encodings.

This stands in stark contrast to the sharp threshold in multi-linear maps, where you can multiply exactly κ encodings, no more and no less.

A consequence of the ability to multiply any number of encodings is that we no longer have a single target group, instead we have a different "target group" for any number of multiplicands. This yields a richer structure, roughly analogous to *graded algebra*. In its simplest form (analogous to symmetric maps with a single source group), we have levels of encodings: At level zero we have the "plaintext" ring elements $\alpha \in R$ themselves, level one corresponds to $\alpha \cdot g$ in the source group, and level-i corresponds to a product of i level-1 encodings (so level-κ corresponds to the target group from multilinear maps).

Definition 1 (κ-Graded Encoding System). *A κ-Graded Encoding System consists of a ring R and a system of sets $\mathcal{S} = \{S_i^{(\alpha)} \subset \{0,1\}^* : \alpha \in R,\ 0 \le i \le \kappa,\}$, with the following properties:*

1. *For every fixed index i, the sets $\{S_i^{(\alpha)} : \alpha \in R\}$ are disjoint (hence they form a partition of $S_i \stackrel{\text{def}}{=} \bigcup_\alpha S_{\boldsymbol{v}}^{(\alpha)}$).*
2. *There is an associative binary operation '$+$' and a self-inverse unary operation '$-$' (on $\{0,1\}^*$) such that for every $\alpha_1, \alpha_2 \in R$, every index $i \le \kappa$, and every $u_1 \in S_i^{(\alpha_1)}$ and $u_2 \in S_i^{(\alpha_2)}$, it holds that*
$$u_1 + u_2 \in S_i^{(\alpha_1+\alpha_2)} \quad \text{and} \quad -u_1 \in S_i^{(-\alpha_1)}$$
where $\alpha_1 + \alpha_2$ and $-\alpha_1$ are addition and negation in R.
3. *There is an associative binary operation '$\times$' (on $\{0,1\}^*$) such that for every $\alpha_1, \alpha_2 \in R$, every i_1, i_2 with $i_1 + i_2 \le \kappa$, and every $u_1 \in S_{i_1}^{(\alpha_1)}$ and $u_2 \in S_{i_2}^{(\alpha_2)}$, it holds that $u_1 \times u_2 \in S_{i_1+i_2}^{(\alpha_1 \cdot \alpha_2)}$. Here $\alpha_1 \cdot \alpha_2$ is multiplication in R, and $i_1 + i_2$ is integer addition.*

Clearly, Definition 1 implies that if we have a collection of n encodings $u_j \in S_{i_j}^{(\alpha_j)}$, $j = 1, 2 \ldots, n$, then as long as $\sum_j i_j \le \kappa$ we get $u_1 \times \cdots \times u_n \in S_{i_1+\cdots+i_n}^{(\prod_j \alpha_j)}$.

Efficient Procedures, the Dream Version. To be useful, we need efficient procedures for manipulating encodings well as as hard computational tasks. To ease the exposition, we first describe a "dream version" of the efficient procedures (which we do not know how to realize), and then explain how to modify them to deal with technicalities that arise from our use of lattices in the realization.

Instance Generation. The randomized $\mathsf{InstGen}(1^\lambda, 1^\kappa)$ takes as inputs the parameters λ, κ, and outputs $(\mathsf{params}, \mathbf{p}_{zt})$, where params is a description of a κ-Graded Encoding System as above, and $\mathbf{p}_{zt}$ is a zero-test parameter for level κ (see below).

Ring Sampler. The randomized $\mathsf{samp}(\mathsf{params})$ outputs a "level-zero encoding" $a \in S_0^{(\alpha)}$ for a nearly uniform element $\alpha \in_R R$. (Note that we require that the "plaintext" $\alpha \in R$ is nearly uniform, but not that the encoding a is uniform in $S_0^{(\alpha)}$.)

Encoding. The (possibly randomized) $\mathsf{enc}(\mathsf{params}, i, a)$ takes a "level-zero" encoding $a \in S_0^{(\alpha)}$ for some $\alpha \in R$ and index $i \leq \kappa$, and outputs the "level-i" encoding $u \in S_i^{(\alpha)}$ for the same α.

Addition and negation. Given params and two encodings relative to the same index, $u_1 \in S_i^{(\alpha_1)}$ and $u_2 \in S_i^{(\alpha_2)}$, we have $\mathsf{add}(\mathsf{params}, i, u_1, u_2) = u_1 + u_2 \in S_i^{(\alpha_1+\alpha_2)}$, and $\mathsf{neg}(\mathsf{params}, i, u_1) = -u_1 \in S_i^{(-\alpha_1)}$.

Multiplication. For $u_1 \in S_{i_1}^{(\alpha_1)}$, $u_2 \in S_{i_2}^{(\alpha_2)}$ such that $i_1 + i_2 \leq \kappa$, we have $\mathsf{mul}(\mathsf{params}, i_1, u_1, i_2, u_2) = u_1 \times u_2 \in S_{i_1+i_2}^{(\alpha_1 \cdot \alpha_2)}$.

Zero-test. The procedure $\mathsf{isZero}(\mathsf{params}, u)$ output 1 if $u \in S_\kappa^{(0)}$ and 0 otherwise. Note that in conjunction with the subtraction procedure, this lets us test if $u_1, u_2 \in S_\kappa$ encode the same element $\alpha \in R$.

Extraction. This procedure extracts a "canonical" and "random" representation of ring elements from their level-κ encoding. Namely $\mathsf{ext}(\mathsf{params}, \mathbf{p}_{zt}, u)$ outputs (say) $s \in \{0,1\}^\lambda$, such that:

(a) For any $\alpha \in R$ and two $u_1, u_2 \in S_\kappa^{(\alpha)}$, $\mathsf{ext}(\mathsf{params}, \mathbf{p}_{zt}, u_1) = \mathsf{ext}(\mathsf{params}, \mathbf{p}_{zt}, u_2)$,

(b) The distribution $\{\mathsf{ext}(\mathsf{params}, \mathbf{p}_{zt}, u) : \alpha \in_R R, u \in S_\kappa^{(\alpha)}\}$ is nearly uniform over $\{0,1\}^\lambda$.

Efficient Procedures, the Real-Life Version. Our realization of the procedures above over ideal lattices uses noisy encodings, where the noise increases with every operation and correctness is only ensured as long as it does not increase too much. We therefore modify the procedures above, letting them take as input (and produce as output) also a bound on the noise magnitude of the encoding in question. The procedures are allowed to abort if the bound is too high (relative to some maximum value which is part of the instance description params). Also, they provide no correctness guarantees if the bound on their input is "invalid." (When B is a noise-bound for some encoding u, we say that it is "valid" if it is at least as large as the bound produced by the procedure that produced u itself, and moreover any encoding that was used by that procedure (if any) also came with a valid noise bound.) Of course we also require that these procedure do not always abort, i.e. they should support whatever set of operations that the application calls for, before the noise becomes too large. Finally, we also relax the requirements on the zero-test and the extraction routines. Some more details are described next:

Zero-test. We sometime allow false positives for this procedure, but not false negatives. Namely, $\mathsf{isZero}(\mathsf{params}, \mathbf{p}_{zt}, u) = 1$ for every $u \in S_\kappa^{(0)}$, but we may have $\mathsf{isZero}(\mathsf{params}, \mathbf{p}_{zt}, u) = 1$ also for some $u \notin S_\kappa^{(0)}$. The weakest functionality requirement that we make is that for a uniform random choice of $\alpha \in_R R$, we have

$$\Pr_{\alpha \in_R R}\left[\exists\, u \in S_\kappa^{(\alpha)} \text{ s.t } \mathsf{isZero}(\mathsf{params}, \mathbf{p}_{zt}, u) = 1\right] = \text{negligible}(\lambda). \qquad (1)$$

Additional requirements are considered security features (that a scheme may or may not possess), and are discussed later in this section.

Extraction. Our construction from Section 4 does not support full canonicalization. Instead, we settle for $\mathsf{ext}(\Lambda, \mathbf{p}_{zt}, u)$ that has a good chance of producing the same output when applied to different encoding of the same elements. Specifically, we replace properties (a)-(b) from above by the weaker requirements:

(a$'$) For a randomly chosen $a \leftarrow \mathsf{samp}(\mathsf{params})$, if we run the encoding algorithm twice to encode a at level κ and then extract from both copies then we get:

$$\Pr\left[\begin{array}{l}\mathsf{ext}(\mathsf{params}, \mathbf{p}_{zt}, u_1)\\ = \mathsf{ext}(\mathsf{params}, \mathbf{p}_{zt}, u_2)\end{array} : \begin{array}{l} a \leftarrow \mathsf{samp}(\mathsf{params})\\ u_1 \leftarrow \mathsf{enc}(\mathsf{params}, \kappa, a)\\ u_2 \leftarrow \mathsf{enc}(\mathsf{params}, \kappa, a)\end{array}\right] \geq 1 - \text{negligible}(\lambda).$$

(b$'$) The distribution $\{\mathsf{ext}(\mathsf{params}, \mathbf{p}_{zt}, u) : a \leftarrow \mathsf{samp}(\mathsf{params}), u \leftarrow \mathsf{enc}(\mathsf{params}, \kappa, a)\}$ is nearly uniform over $\{0,1\}^\lambda$.

We typically need these two conditions to hold even if the noise bound that the encoding routine takes as input is larger than the one output by samp (upto some maximum value).

Hardness Assumptions. Our hardness assumptions are modeled after the discrete-logarithm and DDH assumptions in multilinear groups. For example, the most direct analog of the discrete-logarithm problem is trying to obtain a level-zero encoding $a \in S_0^{(\alpha)}$ for $\alpha \in R$ from an encoding relative to some other index $i > 0$.

The analog of DDH in our case roughly says that it is hard to recognize encoding of products, except relative to indexes upto κ. In other words, given $\kappa + 1$ level-one encoding of random elements it should be infeasible to generate a level-κ encoding of their product, or even to distinguish it from random. One way to formalize it is by the following process. (Below we suppress the noise bounds for readability):

```
1. $(\mathsf{params}, \mathbf{p}_{zt}) \leftarrow \mathsf{InstGen}(1^\lambda, 1^\kappa)$
2. For $i = 1, \ldots, \kappa + 1$:
3.     Choose $a_i \leftarrow \mathsf{samp}(\mathsf{params})$   // level-0 encoding of random $\alpha_i \in_R R$
4.     Set $u_i \leftarrow \mathsf{enc}(\mathsf{params}, 1, a_i)$   // level-1 encoding of the $\alpha_i$'s
5. Set $\tilde{a} = \prod_{i=1}^{\kappa+1} a_i$   // level-0 encoding of the product
6. Choose $\hat{a} \leftarrow \mathsf{samp}(\mathsf{params})$   // level-0 encoding of a random element
7. Set $\tilde{u} \leftarrow \mathsf{enc}(\mathsf{params}, \kappa, \tilde{a})$   // level-$\kappa$ encoding of the product
8. Set $\hat{u} \leftarrow \mathsf{enc}(\mathsf{params}, \kappa, \hat{a})$   // level-$\kappa$ encoding of random
```

(We note that with the noise bound, it may be important that the encoding routines for both $\tilde{a}$ and $\hat{a}$ get as input the same bound, i.e., the largest of the bounds for $\tilde{a}$ and $\hat{a}$.) The GDDH distinguisher gets all the level-one u_i's and either $\tilde{u}$ (encoding the right product) or $\hat{u}$ (encoding a random element), and it needs to decide which is the case. In other words, the GDDH assumption says that for any setting of the parameters, the following two distributions, defined over the experiment above, are computationally indistinguishable:

$$\mathcal{D}_{GDDH} = \{(\mathsf{params}, \mathbf{p}_{zt}, \{u_i\}_i, \tilde{u})\} \text{ and } \mathcal{D}_{RAND} = \{(\mathsf{params}, \mathbf{p}_{zt}, \{u_i\}_i, \hat{u})\}.$$

3 Preliminaries

Lattices. A lattice $L \subset \mathbb{R}^n$ is an additive discrete sub-group of $\mathbb{R}^n$. Every (nontrivial) lattice has bases: a basis for a full-rank lattice is a set of n linearly independent points $\boldsymbol{b}_1, \dots, \boldsymbol{b}_n \in L$ such that $L = \{\sum_{i=1}^n z_i \boldsymbol{b}_i : z_i \in \mathbb{Z}\ \forall i\}$. If we arrange the vectors $\boldsymbol{b}_i$ as the columns of a matrix $B \in \mathbb{R}^{n \times n}$ then we can write $L = \{B\boldsymbol{z} : \boldsymbol{z} \in \mathbb{Z}^n\}$. If B is a basis for L then we say that B spans L. For a lattice $L \subset \mathbb{R}^n$, its *dual lattice* consists of all the points in span(L) that are orthogonal to L modulo one, namely $L^* = \{\boldsymbol{y} \in \mathrm{span}(L) : \forall \boldsymbol{x} \in L, \langle \boldsymbol{x}, \boldsymbol{y} \rangle \in \mathbb{Z}\}$

Gaussians. For a real $\sigma > 0$, define the (spherical) Gaussian function on $\mathbb{R}^n$ with parameter σ as $\rho_\sigma(\boldsymbol{x}) = \exp(-\pi\|\boldsymbol{x}\|^2/\sigma^2)$ for all $\boldsymbol{x} \in \mathbb{R}^n$. This generalizes to ellipsoid Gaussians, where the different coordinates are jointly Gaussian but not independent, where we replace the parameter $\sigma \in \mathbb{R}$ by the (square root of the) covariance matrix $\Sigma \in \mathbb{R}^{n \times n}$. For a rank-$n$ matrix $S \in \mathbb{R}^{m \times n}$, the ellipsoid Gaussian function on $\mathbb{R}^n$ with parameter S is defined by $\rho_S(\boldsymbol{x}) = \exp\big(-\pi \boldsymbol{x}^T (S^T S)^{-1} \boldsymbol{x}\big)\ \ \forall \boldsymbol{x} \in \mathbb{R}^n$. Obviously this function only depends on $\Sigma = S^T S$ and not on the particular choice of S. It is also clear that the spherical case can be obtained by setting $S = \sigma I_n$, with I_n the n-by-n identity matrix.

The *ellipsoid discrete Gaussian distribution* over lattice L with parameter S is $\forall\ \boldsymbol{x} \in L, D_{L,S}(\boldsymbol{x}) = \rho_S(\boldsymbol{x})/\rho_S(L)$, where $\rho_S(L)$ denotes $\sum_{\boldsymbol{x} \in L} \rho_S(\boldsymbol{x})$. In other words, the probability $D_{L,S}(\boldsymbol{x})$ is simply proportional to $\rho_S(\boldsymbol{x})$, the denominator being a normalization factor. The same definitions apply to the spherical case, $D_{L,\sigma}(\cdot)$.

Smoothing parameter. Micciancio and Regev defined in [MR07] the *smoothing parameter* for a lattice L and real $\epsilon > 0$, denoted $\eta_\epsilon(L)$, as the smallest s such that $\rho_{1/s}(L^* \setminus \{\boldsymbol{0}\}) \leq \epsilon$. Intuitively, for a small enough ϵ, the number $\eta_\epsilon(L)$ is sufficiently larger than L's fundamental parallelepiped so that sampling from the corresponding Gaussian "wipes out the internal structure" of L. It is easy to show that the size of vectors drawn from $D_{L,S}$ is roughly bounded by the largest singular value of S. (Recall that the largest and least singular values of a full rank matrix $X \in \mathbb{R}^{m \times n}$ are defined as $\sigma_1(X) = \sup(U_X)$ and $\sigma_n(X) = \inf(U_X)$, respectively, where $U_X = \{\|X\boldsymbol{u}\| : \boldsymbol{u} \in \mathbb{R}^n, \|\boldsymbol{u}\| = 1\}$.)

Lemma 1. *For a rank-n lattice L, constant* $0 < \epsilon < 1$ *and matrix S s.t.* $\sigma_n(S) \geq \eta_\epsilon(L)$, *we have* $\Pr_{\boldsymbol{v} \leftarrow \mathcal{D}_{L,S}}\big(\|\boldsymbol{v}\| \geq \sigma_1(S)\sqrt{n}\big) \leq \frac{1+\epsilon}{1-\epsilon} \cdot 2^{-n}$.

Sum of Discrete Gaussians. A recent work [AGHS12] considered the process that begins by choosing "once and for all" m points in a lattice L, drawn independently from a "wide discrete Gaussian" $\boldsymbol{x}_i \leftarrow D_{L,S}$. Once the $\boldsymbol{x}_i$'s are fixed, they are arranged as the rows of an m-by-n matrix $X = (\boldsymbol{x}_1|\boldsymbol{x}_2|\dots|\boldsymbol{x}_m)^T$, and we consider the distribution $\mathcal{D}_{X,\sigma}$, induced by choosing an integer vector $\boldsymbol{v}$ from a discrete spherical Gaussian over $\mathbb{Z}^m$ with parameter σ and outputting

$\boldsymbol{y} = X^T\boldsymbol{v}$, $\mathcal{E}_{X,\sigma} \overset{\text{def}}{=} \{X^T\boldsymbol{v} : \boldsymbol{v} \leftarrow D_{\mathbb{Z}^m,\sigma}\}$. [AGHS12] proved that with high probability over the choice of X, the distribution $\mathcal{D}_{X,\sigma}$ is statistically close to the ellipsoid Gaussian $D_{L,\sigma X}$, and moreover the singular values of X are of size roughly $\sigma\sqrt{m}$:

Theorem 1 ([AGHS12]). *Let L be a full-rank lattice $L \subset \mathbb{R}^n$ and B a matrix whose rows form a basis of L, and denote $\chi = \sigma_1(B)/\sigma_n(B)$. Also let ϵ be negligible in n, and let m, s, s' be parameters such that $s \geq \eta_\epsilon(\mathbb{Z}^n)$, $m \geq 10n\log(8(mn)^{1.5}s\chi)$ and $s' \geq 4mn\chi\ln(1/\epsilon)$.*

Then, when choosing the rows of an m-by-n matrix X from the spherical Gaussian over L, $X \leftarrow (\mathcal{D}_{L,s})^m$, we have with all but probability $2^{-O(m)}$ over the choice of X, that the statistical distance between $\mathcal{E}_{X,s'}$ and the ellipsoid Gaussian $\mathcal{D}_{L,s'X}$ is bounded by 2ϵ.

Lemma 2 ([AGHS12]). *There exists a universal constant $K > 1$ such that for all $m \geq 2n$, $\epsilon > 0$ and every n-dimensional real lattice $L \subset \mathbb{R}^n$, the following holds: Choosing the rows of an m-by-n matrix X independently at random from a spherical discrete Gaussian on L with parameter $\rho > 2K\eta_\epsilon(L)$, $X \leftarrow (D_{L,\rho})^m$, we have*

$$\Pr\left[s\sqrt{2\pi m}/K < \sigma_n(X) \leq \sigma_1(X) < \rho K\sqrt{2\pi m}\right] > 1 - (4m\epsilon + O(\exp(-m/K))).$$

Ideal Lattices. For n a power of two, we consider the $2n$'th cyclotomic polynomial ring $R = \mathbb{Z}[X]/(X^n+1)$, and identify an element $\boldsymbol{u} \in R$ with the coefficient vector[1] of the degree-$(n-1)$ integer polynomial that represents $\boldsymbol{u}$. In this way, R is identified with the integer lattice $\mathbb{Z}^n$. Additionally we sometimes consider also the ring $R_q = R/qR = \mathbb{Z}_q[X]/(X^n+1)$ for a (large enough) integer q. Obviously, addition in these rings is done component-wise in their coefficients, and multiplication is polynomial multiplication modulo the ring polynomial $X^n + 1$. In some cases we also consider the corresponding number field $\mathbb{K} = \mathbb{Q}[X]/(X^n+1)$, which is likewise associated with the linear space $\mathbb{Q}^n$.

For an element $\mathbf{g} \in R$, let $\langle\mathbf{g}\rangle$ be the principal ideal in R generated by $\mathbf{g}$ (alternatively, the sub-lattice of $\mathbb{Z}^n$ corresponding to this ideal), namely $\langle\mathbf{g}\rangle = \{\mathbf{g}\cdot\boldsymbol{u} : \boldsymbol{u} \in R\}$. We call $\langle\mathbf{g}\rangle$ an *ideal lattice* to stress its dual interpretation as both an ideal and a lattice. Let $B(\mathbf{g})$ denote the basis of the lattice $\langle\mathbf{g}\rangle$ consisting of the vectors $\{\mathbf{g}, X\mathbf{g}, X^2\mathbf{g}, \ldots, X^{n-1}\mathbf{g}\}$.

For an arbitrary element $\boldsymbol{u} \in R$, denote by $[\boldsymbol{u}]_\mathbf{g}$ the reduction of $\boldsymbol{u}$ modulo the fundamental cell of $B(\mathbf{g})$, which is *symmetric around the origin.* To wit, $[\boldsymbol{u}]_\mathbf{g}$ is the unique element $\boldsymbol{u}' \in R$ such that $\boldsymbol{u} - \boldsymbol{u}' \in \langle\mathbf{g}\rangle$ and $\boldsymbol{u}' = \sum_{i=0}^{n-1}\alpha_i X^i\mathbf{g}$ where all the α_i's are in the interval $[-\frac{1}{2}, \frac{1}{2})$. We use the similar notation $[t]_p$ for integers t, p to denote the reduction of t modulo p into the interval $[-p/2, p/2)$.

[1] Other representations of polynomials are also possible, for example representing a polynomial by its canonical embedding is sometimes preferable to the coefficient representation. Here we stick to coefficient representation for simplicity.

4 The New Encoding Schemes

An instance of our basic construction is parametrized by the security parameter λ and the required multi-linearity level $\kappa \leq$poly(λ). Based on these parameters, we choose a cyclotomic ring $R = \mathbb{Z}[X]/(X^n + 1)$ (where n is large enough to ensure security), a modulus q that defines $R_q = R/qR$ (with q large enough to support functionality), and another parameter m (chosen so that we can apply Theorem 1). The specific constraints that these parameters must satisfy are discussed at the end of this section, an approximate setting to keep in mind is $n = \tilde{O}(\kappa\lambda^2)$, $q = 2^{n/\lambda}$ and $m = O(n^2)$.

4.1 The Basic Graded Encoding Scheme

An instance of our scheme relative to the parameters above encodes elements of a quotient ring $QR = R/\mathcal{I}$, where $\mathcal{I}$ is a principal ideal $\mathcal{I} = \langle \mathbf{g} \rangle \subset R$, generated by a short vector $\mathbf{g}$. Namely, the "ring elements" that are encoded in our scheme are cosets of the form $\boldsymbol{e} + \mathcal{I}$ for some vector $\boldsymbol{e}$. The short generator $\mathbf{g}$ itself is kept secret, and no "good" description of $\mathcal{I}$ is made public in our scheme. In addition, our system depends on another secret element $\mathbf{z}$, which is chosen at random in R_q (and hence is not short).

A level-zero ("plaintext") encoding of a coset $\boldsymbol{e}+\mathcal{I} \in R/\mathcal{I}$ is just a short vector in that coset (which must exist, since the generator $\mathbf{g}$ is short and therefore the basic cell of $\mathcal{I}$ is quite small). For higher-level encodings, a level-i encoding of the same coset is a vector of the form $\boldsymbol{c}/\mathbf{z}^i \in R_q$ with $\boldsymbol{c} \in \boldsymbol{e}+\mathcal{I}$ short. Specifically, for $i \in \{0, 1, \ldots, \kappa\}$ the set of all level-i encodings is $S_i = \{\boldsymbol{c}/\mathbf{z}^i \in R_q : \|\boldsymbol{c}\| < q^{1/8}\}$, and the set of levle-$i$ encodings of the "plaintext element" $\boldsymbol{e} + \mathcal{I}$ is $S_i^{(\mathbf{e}+\mathcal{I})} = \{\boldsymbol{c}/\mathbf{z}^i \in R_q : \boldsymbol{c} \in \boldsymbol{e} + \mathcal{I},\ \|\boldsymbol{c}\| < q^{1/8}\ \}$. Throughout the construction we use the size of the numerator as the "noise level" in the encoding. Namely, with each level-i encoding $\boldsymbol{c}/\mathbf{z}^i$ we produce also an upper bound on $\|\boldsymbol{c}\|$.

Instance generation: $(\mathsf{params}, \mathbf{p}_{zt}) \leftarrow \mathsf{InstGen}(1^\lambda, 1^\kappa)$. Our instance-generation procedure chooses at random the ideal-generator $\mathbf{g}$ and denominator $\mathbf{z}$, as well as several other vectors that are used in the other procedures and are described later in the section. The denominator $\mathbf{z}$ is chosen uniformly at random in R_q. For technical reasons, the generator $\mathbf{g} \in R$ should be chosen so that both $\mathbf{g}$ and $\mathbf{g}^{-1} \in \mathbb{K}$ are short. (Recall that we denote $\mathbb{K} = \mathbb{Q}[X]/(X^n + 1)$. The reason that we need $\mathbf{g}^{-1} \in \mathbb{K}$ to be short is explained when we describe the zero-testing procedure.) We simply draw $\mathbf{g}$ from a discrete Gaussian over $\mathbb{Z}^n$, say $\mathbf{g} \leftarrow D_{\mathbb{Z}^n,\sigma}$ with $\sigma = \tilde{O}(\sqrt{n})$. Clearly $\mathbf{g}$ itself is short (of size less than $\sigma\sqrt{n}$), and we claim that with good probability its inverse in the field of fractions is also rather short. To see this, notice that with probability $1 - o(1/n)$, evaluating $\mathbf{g}$ at any complex n'th root of unity $\zeta \in \mathbb{C}$ yields $\mathbf{g}(\zeta)$ which is not too tiny, say larger than $1/n$. Hence with probability $1 - o(1)$ we have $\mathbf{g}^{-1}(\zeta) = 1/\mathbf{g}(\zeta) < n$ for all the primitive $2n$'th roots of unity ζ, which means that $\mathbf{g}^{-1}$ itself is not too large, say $\|1/\mathbf{g}\| < n^2$. We can draw repeatedly until we get this condition to hold.

Once we have $\mathbf{g}, \mathbf{z}$, we choose and publish some other elements in R_q that will be used for the various procedures below. Specifically we have $m+1$ elements $rand_1, \ldots, \mathbf{x}_m, \mathbf{y}$ that are used for encoding, and an element $\mathbf{p}_{zt}$ that is used as a zero-testing parameter. These elements are described later. finally we also choose a random seed s for a strong randomness extractor. The instance-generation procedure outputs $\mathsf{params} = (n, q, \mathbf{y}, \{\mathbf{x}_i\}_i, s)$ and $\mathbf{p}_{zt}$.

Sampling level-zero encodings: $\boldsymbol{d} \leftarrow \mathsf{samp}(\mathsf{params})$. To sample a level-zero encoding of a random coset, we just draw a random short element in R, $\boldsymbol{d} \leftarrow D_{\mathbb{Z}^n,\sigma'}$, where $\sigma' = \sigma n$ (for σ that was used to sample $\mathbf{g}$). Since whp $\sigma' \geq \eta_{2^{-\lambda}}(\mathcal{I})$, then the induced distribution over the cosets of $\mathcal{I}$ is close to uniform, upto a negligible distance. Also the size of this level-zero encoding is bounded by $\sigma'\sqrt{n}$ (and we use this as our noise-bound for this encoding).

Encodings at higher levels: $\boldsymbol{u}_i \leftarrow \mathsf{enc}(\mathsf{params}, i, \boldsymbol{d})$. To allow encoding of cosets at higher levels, we publish as part of our instance-generation a level-one encoding of $1+\mathcal{I}$, namely an element $\mathbf{y} = [\mathbf{a}/\mathbf{z}]_q$ where $\mathbf{a} \in 1+\mathcal{I}$ is short. A simplistic method of doing that is drawing $\mathbf{a} \leftarrow D_{1+\mathcal{I},\sigma'}$, then computing $\mathbf{y}$ from $\mathbf{a}$. (Later we describe a somewhat more involved procedure, which we believe is more secure.) Given a level-zero encoding $\boldsymbol{d}$ as above, we can multiply it by $\mathbf{y}$ over R_q to get $\boldsymbol{u}_1 := [\mathbf{y}\boldsymbol{d}]_q$. Note that $\boldsymbol{u}_1 = [\boldsymbol{d}\mathbf{a}/\mathbf{z}]_q$, where $\boldsymbol{d}\mathbf{a} \in \boldsymbol{d}+\mathcal{I}$ as needed, and the size of the numerator is bounded by $\|\boldsymbol{d}\| \cdot \|\mathbf{a}\| \cdot \sqrt{n} = \text{poly}(n)$. More generally we can generate a level-i encoding as $\boldsymbol{u}_i := [\boldsymbol{d}\mathbf{y}^i]_q = [\boldsymbol{d}\mathbf{a}^i/\mathbf{z}^i]_q$. The numerator $\boldsymbol{d}\mathbf{a}^i$ is obviously in $\boldsymbol{d}+\mathcal{I}$, and its size is at most $\|\boldsymbol{d}\| \cdot \|\mathbf{a}\|^i \cdot n^{i/2}$.

The above encoding is insufficient, however, since from $\boldsymbol{u}_1$ and $\mathbf{y}$ it is easy to get back $\boldsymbol{d}$ by simple division in R_q. We therefore include in the public parameters also the "randomizers" $\mathbf{x}_i$, these are just random encodings of zero, namely $\mathbf{x}_i = [\mathbf{b}_i/\mathbf{z}]_q$ where the $\mathbf{b}_i$'s are short elements in $\mathcal{I}$. A simplistic procedure for choosing these randomizers would be to draw short these elements as $\mathbf{b}_i \leftarrow D_{\mathcal{I},\sigma'}$ and publish $\mathbf{x}_i = [\mathbf{b}_i/\mathbf{z}]_q$. As we note in the full version [GGH12], we have reasons to suspect that this simplistic method is insecure so instead we use a somewhat more involved sampling procedure, see details in the full version [GGH12]. Below we denote by $\mathbf{X}$ the matrix with the vectors $\mathbf{x}_i$ as rows, namely $\mathbf{X} = (\mathbf{x}_1|\ldots|\mathbf{x}_m)^T$. We also use $\mathbf{B}$ to denote the matrix with the numerators $\mathbf{b}_i$ as rows, i.e., $\mathbf{B} = (\mathbf{b}_1|\ldots|\mathbf{b}_m)^T$.

We use the $\mathbf{x}_i$'s to randomize level-one encodings: Given $\boldsymbol{u}' = [\boldsymbol{c}'/\mathbf{z}]_q$ with noise-bound $\|\boldsymbol{c}'\| < \gamma$, we draw an m-vector of integer coefficients $\boldsymbol{r} \leftarrow D_{\mathbb{Z}^m,\sigma^*}$ for large enough σ^* (e.g. $\sigma^* = 2^\lambda\gamma$), and output

$$\boldsymbol{u} \ := \ [\boldsymbol{u}' + \mathbf{X}\boldsymbol{r}]_q \ = \ [\boldsymbol{u}' + \sum_{i=1}^{m} r_i\mathbf{x}_i]_q \ (= \ [\frac{\boldsymbol{c}' + \sum_i r_i\mathbf{b}_i}{\mathbf{z}}]_q).$$

We write $\mathbf{B}\boldsymbol{r}$ as a shorthand for $\sum_i r_i\mathbf{b}_i$ and similarly $\mathbf{X}\boldsymbol{r}$ as a shorthand for $\sum_i r_i\mathbf{x}_i$.

Since all the $\mathbf{b}_i$'s are in the ideal $\mathcal{I}$, then obviously $\boldsymbol{c}' + \sum_i r_i\mathbf{b}_i$ is in the same coset of $\mathcal{I}$ as $\boldsymbol{c}'$ itself. Moreover since $\|\mathbf{b}_i\| < \text{poly}(n)$ then $\|\mathbf{B}\boldsymbol{r}\| < \sigma^*\text{poly}(m,n)$. If indeed $\|\boldsymbol{c}'\| < \gamma$, then $\|\boldsymbol{c}' + \mathbf{B}\boldsymbol{r}\| < \gamma + \sigma^*\text{poly}(m,n)$. We also claim that the distribution of $\boldsymbol{u}$ is nearly independent of original $\boldsymbol{u'}$ (except of course its coset). To see why, note that if the $\mathbf{b}_i$'s are chosen from a wide enough spherical distribution then we can use Theorem 1 to conclude that $\mathbf{B}\boldsymbol{r}$ is close to a wide ellipsoid Gaussian. With our choice of σ^* the "width" of that distribution is much larger than the original $\boldsymbol{c'}$, hence the distribution of $\boldsymbol{c}' + \mathbf{B}\boldsymbol{r}$ is nearly independent of $\boldsymbol{c'}$, except in the coset that it belongs to.

A different approach is to re-randomize $\mathbf{y}$, setting $\mathbf{y}' := \mathbf{y} + \mathbf{X}\boldsymbol{r}$ and then encode via $\boldsymbol{u}_1 := [\mathbf{y}'\boldsymbol{d}]_q$. This does not have the information-theoretic same-distribution guarantee as above (since the distributions $[\mathbf{y}'\boldsymbol{d}]_q$ and $[\mathbf{y}'\boldsymbol{d'}]_q$ may differ, even if $\boldsymbol{d}, \boldsymbol{d'}$ are both short and in the same coset). But on the plus side, it is more convenient to use this re-randomization method for encoding at high levels $i > 1$: After computing the randomized $\mathbf{y}'$, we can use it by setting $\boldsymbol{u}_i := [\boldsymbol{d}(\mathbf{y}')^i]_q$.

Remark 1. Note that in the above description we used the matrix $\mathbf{X}$ to randomize level-one encodings. Using similar pubic parameter $\mathbf{X}_i$ we can generalize the re-randomization procedure to work at any level $i \leq \kappa$. In particular we abstract this procedure as $\mathsf{reRand}(\mathbf{y}, i, \boldsymbol{u}')$: Given $\boldsymbol{u}' = [\boldsymbol{c}'/\mathbf{z}^i]_q$ with noise-bound $\|\boldsymbol{c}'\| < \gamma$, we draw an m-vector of integer coefficients $\boldsymbol{r} \leftarrow D_{\mathbb{Z}^m,\sigma^*}$ for large enough σ^* (e.g. $\sigma^* = 2^\lambda\gamma$), and output $\boldsymbol{u} := [\boldsymbol{u}' + \mathbf{X}_i\boldsymbol{r}]_q$ as a re-randomized version of $\boldsymbol{u}$. Using the same argument as above we can conclude that the distribution generated in this way will be independent of $\boldsymbol{c}'$, except in the coset that it belongs to.

Note that for some applications it might be useful to use the re-randomization operation multiple times. We consider the case in which a constant number of re-randomizations is needed. In this case, with the ℓ^{th} re-randomization (for any constant ℓ) we can generate an encoding by choosing $\boldsymbol{r}$ from $D_{\mathbb{Z}^m,\sigma^*}$ where $\sigma^* = 2^{\lambda^\ell}$ and re-randomizing as above. Since the addition and multiplication of constant number of terms increases noise by a small factor we can claim that each re-randomization wipes the structure that was present previously (even with multiple additions and multiplications).

We define a canonicalizing encoding algorithm $\mathsf{cenc}_\ell(\mathsf{params}, i, \boldsymbol{u}')$ which takes as input an encoding of $\boldsymbol{u}'$ and generates another encoding according with a noise factor of 2^{λ^ℓ}.

Adding and multiplying encodings. It is easy to see that the encoding as above is additively homomorphic, in the sense that adding encodings yields an encoding of the sum. This follows since if we have many short $\boldsymbol{c}_j$'s then their sum is still short, $\|\sum_j \boldsymbol{c}_j\| \ll q$, and therefore the sum $\boldsymbol{c} = \sum_j \boldsymbol{c}_j = [\sum_j \boldsymbol{c}_j]_q \in R_q$ belong to the coset $\sum_j(\boldsymbol{c}_j + \mathcal{I})$. Hence, if we denote $\boldsymbol{u}_j = \boldsymbol{c}_j/\mathbf{z} \in R_q$ then each $\boldsymbol{u}_j$ is an encoding of the coset $\boldsymbol{c}_j + \mathcal{I}$, and the sum $[\sum_j \boldsymbol{u}_j]_q$ is of the form $\boldsymbol{c}/\mathbf{z}$ where $\boldsymbol{c}$ is still a short element in the sum of the cosets.

Moreover, since $\mathcal{I}$ is an ideal then multiplying upto κ encodings can be interpreted as an encoding of the product, by raising the denominator to the appropriate power. Namely, for $\boldsymbol{u}_j = \boldsymbol{c}_j/\mathbf{z} \in R_q$ as above, we have

$$\boldsymbol{u} = \prod_{j=1}^{\kappa} \boldsymbol{u}_j = \frac{\prod_j \boldsymbol{c}_j}{\mathbf{z}^\kappa} \quad \text{(all the operations in } R_q\text{)}.$$

As long as the $\boldsymbol{c}_j$'s are small enough to begin with, we still have $\|\prod_j \boldsymbol{c}_j\| \ll q$, which means that $[\prod_j \boldsymbol{c}_j]_q = \prod_j \boldsymbol{c}_j$ (operations in R), hence $[\prod_j \boldsymbol{c}_j]_q$ belongs to the product coset $\prod_j(\boldsymbol{c}_j + \mathcal{I})$.

Thus, if each $\boldsymbol{u}_j$ is a level-1 encoding of the coset $\boldsymbol{c}_j + \mathcal{I}$ with short-enough numerator, then their product is a level-κ encoding of the product coset. We note that just like level-1 encoding, level-κ encoding still offers additive homomorphism.

Zero testing: $\mathsf{isZero}(\mathsf{params}, \mathbf{p}_{zt}, \boldsymbol{u}_\kappa) \overset{?}{=} 0/1$**.** Since the encoding is additively homomorphic, we can test equality between encodings by subtracting them and comparing to zero. To enable zero-testing, we generate the zero-testing parameter as follows: We draw a "somewhat small" ring element $\boldsymbol{h} \leftarrow D_{\mathbb{Z}^n,\sqrt{q}}$, and the zero-testing parameter is set as $\mathbf{p}_{zt} = [\boldsymbol{h}\mathbf{z}^\kappa/\mathbf{g}]_q$. To test if a level-$\kappa$ encoding $\boldsymbol{u} = [\boldsymbol{c}/\mathbf{z}^\kappa]_q$ is an encoding of zero, we just multiply it in R_q by $\mathbf{p}_{zt}$ and check whether the resulting element $\boldsymbol{w} = [\mathbf{p}_{zt} \cdot \boldsymbol{u}]_q$ is short (e.g., shorter than $q^{3/4}$). Namely, we use the test

$$\mathsf{isZero}(\mathsf{params}, \mathbf{p}_{zt}, \boldsymbol{u}) \;=\; \begin{cases} 1 \text{ if } \;\|[\mathbf{p}_{zt}\boldsymbol{u}]_q\|_\infty < q^{3/4} \\ 0 \text{ otherwise} \end{cases} \tag{2}$$

To see why this works, note that

$$\boldsymbol{w} \;=\; \mathbf{p}_{zt} \cdot \boldsymbol{u} \;=\; \frac{\boldsymbol{h}\mathbf{z}^\kappa}{\mathbf{g}} \cdot \frac{\boldsymbol{c}}{\mathbf{z}^\kappa} \;=\; \boldsymbol{h} \cdot \boldsymbol{c}/\mathbf{g} \quad \text{(all the operations in } R_q\text{)}.$$

If $\boldsymbol{u}$ is an encoding of zero then $\boldsymbol{c}$ is a short vector in $\mathcal{I}$, which means that it is divisible by $\mathbf{g}$ in R. Hence the element $\boldsymbol{c}/\mathbf{g} \in R_q$ is the same as the element $\boldsymbol{c} \cdot \mathbf{g}^{-1} \in \mathbb{K}$, which means that it has size at most $\|\boldsymbol{c}\| \cdot \|\mathbf{g}^{-1}\| \cdot \sqrt{n} = \|\boldsymbol{c}\| \cdot \mathrm{poly}(n)$. This, in turn, implies that $\|\boldsymbol{w}\| \le \|\boldsymbol{h}\| \cdot \|\boldsymbol{c}\| \cdot \mathrm{poly}(n)$, which for our choice of parameter is $q^{1/2} \cdot q^{1/8} \cdot \mathrm{poly}(n) < q^{3/4}$.

If $\boldsymbol{u}$ is an encoding of a different coset, then $\boldsymbol{c}$ is a short vector in some coset of $\mathcal{I}$. In this case we have $\boldsymbol{w} = [\boldsymbol{c} \cdot \boldsymbol{h}/\mathbf{g}]_q$, where $\boldsymbol{c}, \mathbf{g}$ are small (and $\boldsymbol{h}$ is "somewhat small"). Intuitively, since $\boldsymbol{h}/\mathbf{g}$ is large whp then for a "random enough" $\boldsymbol{c}$ we expect the size of $\boldsymbol{w}$ to be large. More formally, we argue below that when choosing a uniformly random coset of $\mathcal{I} = \langle \mathbf{g} \rangle$, there are *no short elements* $\boldsymbol{c}$ in that coset such that $[\boldsymbol{c} \cdot \boldsymbol{h}/\mathbf{g}]_q$ is small.

Lemma 3. *Let* $\boldsymbol{w} = [\boldsymbol{c} \cdot \boldsymbol{h}/\mathbf{g}]_q$ *and suppose* $\|\mathbf{g} \cdot \boldsymbol{w}\|$ *and* $\|\boldsymbol{c} \cdot \boldsymbol{h}\|$ *are each at most* $q/2$*. Suppose* $\langle \mathbf{g} \rangle$ *is a prime ideal. Then, either* $\mathbf{c}$ *or* $\boldsymbol{h}$ *is in the ideal* $\langle \mathbf{g} \rangle$*.*

Proof. Since $\mathbf{g} \cdot \boldsymbol{w} = \boldsymbol{c} \cdot \boldsymbol{h} \bmod q$, and since $\|\mathbf{g} \cdot \boldsymbol{w}\|$ and $\|\boldsymbol{c} \cdot \boldsymbol{h}\|$ are each at most $q/2$, we have $\mathbf{g} \cdot \boldsymbol{w} = \boldsymbol{c} \cdot \boldsymbol{h}$ exactly. We also have an equality of ideals $\langle \mathbf{g} \rangle \cdot \langle \boldsymbol{w} \rangle = \langle \boldsymbol{c} \rangle \cdot \langle \boldsymbol{h} \rangle$, and, since $\langle \mathbf{g} \rangle$ is a prime ideal and our cyclotomic ring is a unique factorization domain, we have that $\langle \mathbf{g} \rangle$ divides either $\langle \boldsymbol{c} \rangle$ or $\langle \boldsymbol{h} \rangle$ (or both). The result follows.

Lemma 4. *Let n, q, σ be as in our parameter setting, suppose $q = n^{\omega(1)}$, and consider drawing $\mathbf{g} \leftarrow D_{\mathbb{Z}^n,\sigma'}$ subject to $\langle \mathbf{g} \rangle$ being prime and $\boldsymbol{h} \leftarrow D_{\mathbb{Z}^n,\sqrt{q}}$ not being in $\langle \mathbf{g} \rangle$. Then, there is no $\epsilon > 0$ and $\boldsymbol{c}$ in a nonzero coset of $\mathcal{I}$ such that $\|\boldsymbol{c}\| < q^{1/8}$ and $\|[\boldsymbol{c} \cdot \boldsymbol{h}/\mathbf{g}]_q\| < q^{1-\epsilon}$.*

Proof. This follows directly from Lemma 3, our parameter setting (with $\|\mathbf{g}\| = \mathsf{poly}(n)$) and the fact that in the coefficient embedding $\|\mathbf{a} \cdot \mathbf{b}\| \leq n \cdot \|\mathbf{a}\| \cdot \|\mathbf{b}\|$.

Extraction: $s \leftarrow \mathsf{ext}(\mathsf{params}, \mathbf{p}_{zt}, u_\kappa)$. To extract a "canonical" and "random" representation of a coset from an encoding $\boldsymbol{u} = [\boldsymbol{c}/\mathbf{z}^\kappa]_q$, we just multiply by the zero-testing parameter $\mathbf{p}_{zt}$, collect the $(\log q)/4 - \lambda$ most-significant bits of each of the n coefficients of the result, and apply a strong randomness extractor to the collected bits (using the seed from the public parameters). Namely

$$\mathsf{ext}(\mathsf{params}, \mathbf{p}_{zt}, \boldsymbol{u}) = \text{EXTRACT}_s(\mathsf{msbs}([\boldsymbol{u} \cdot \mathbf{p}_{zt}]_q)) \quad (\mathsf{msbs} \text{ of coefficient representation}).$$

This works because for any two encodings $\boldsymbol{u}, \boldsymbol{u}'$ of the same coset we have

$$\|\mathbf{p}_{zt}\boldsymbol{u} - \mathbf{p}_{zt}\boldsymbol{u}'\| = \|\mathbf{p}_{zt}(\boldsymbol{u} - \boldsymbol{u}')\| < q^{3/4},$$

so we expect $\mathbf{p}_{zt}\boldsymbol{u}$, $\mathbf{p}_{zt}\boldsymbol{u}'$ to agree on their $(\log q)/4 - \lambda$ most significant bits. (There is a negligible (in λ) chance that $\boldsymbol{u}$ and $\boldsymbol{u}'$ are such that $\mathbf{p}_{zt}\boldsymbol{u}$ and $\mathbf{p}_{zt}\boldsymbol{u}'$ are on opposite sides of a boundary, such that they have different MSBs.) On the other hand, by Lemma 4, we know that we cannot have $\|\mathbf{p}_{zt}(\boldsymbol{u} - \boldsymbol{u}')\| < q^{1-\epsilon}$ when $\boldsymbol{u} - \boldsymbol{u}'$ encodes something nonzero, and therefore (since $\lambda \ll \log q/4$) the values $\mathbf{p}_{zt}\boldsymbol{u}$ and $\mathbf{p}_{zt}\boldsymbol{u}'$ cannot agree on their $(\log q)/4 - \lambda$ MSBs.

This means, however, that no two points in the basic cell of $\mathcal{I}$ agree on their collected bits when multiplied by $\mathbf{p}_{zt}$, so the collected bits from an encoding of a random coset have min-entropy at least $\log |R/\mathcal{I}|$. We can therefore use a strong randomness extractor to extract a nearly uniform bit-string of length (say) $\lfloor \log |R/\mathcal{I}| \rfloor - \lambda$.

4.2 Security of Our Constructions

The security of our graded encoding systems relies on new, perhaps unconventional assumptions, and at present it seems unlikely that they can be reduced to more established assumptions, such as learning-with-errors (LWE) [Reg05], or even the NTRU hardness assumption [HPS98]. Given that the construction of multilinear maps has been a central open problem now for over a decade, we feel that exploring unconventional assumptions for this purpose is well worth the effort, as long as this exploration is informed by extensive cryptanalysis.

We attempted an extensive cryptanalysis of our scheme, including some new extensions of tools from the literature that we devised in the course of this work. These attempts are described at length in the full version [GGH12].

Easiness of other Problems. In light of the apparent hardness of our CDH/DDH analog, we could optimistically hope to get also the analog of other hardness assumptions in bilinear maps, such as decision-linear, subgroup membership, etc. Unfortunately, these problems turn out to be easy in our setting, at least with the simple encoding methods from above.

To see why, observe that publishing level-1 encodings of 0 and 1 enables some "weak discrete log" computation at any level strictly smaller than κ. Specifically, consider one particular encoding of zero $\mathbf{x}_j = [\mathbf{b}_j/\mathbf{z}]_q$ (where $\mathbf{b}_j = \boldsymbol{c}_j\mathbf{g}$ for some $\boldsymbol{c}_j$), which is given in the public parameters together with an encoding of one $\mathbf{y} = [\mathbf{a}/\mathbf{z}]_q$ and the zero-testing parameter $\mathbf{p}_{zt} = [\boldsymbol{h}\mathbf{z}^\kappa/\mathbf{g}]_q$. Given a level-$i$ encoding with $1 \le i \lneq \kappa$, $\boldsymbol{u} = [\boldsymbol{d}/\mathbf{z}^i]_q$, we can multiply it by $\mathbf{x}_j$, $\mathbf{p}_{zt}$, and some power of $\mathbf{y}$ to get

$$\begin{aligned}\boldsymbol{f} = [\boldsymbol{u} \cdot \mathbf{x}_j \cdot \mathbf{p}_{zt} \cdot \mathbf{y}^{\kappa-i-1}]_q \;&=\; \left[\frac{\boldsymbol{d}}{\mathbf{z}^i} \cdot \frac{\boldsymbol{c}_j \cdot \mathbf{g}}{\mathbf{z}} \cdot \frac{\boldsymbol{h}\mathbf{z}^\kappa}{\mathbf{g}} \cdot \frac{\mathbf{a}^{\kappa-i-1}}{\mathbf{z}^{\kappa-i-1}}\right]_q \\ = \underbrace{\boldsymbol{d} \cdot \boldsymbol{c}_j \cdot \boldsymbol{h} \cdot \mathbf{a}^{\kappa-i-1}}_{\ll q} \;&=\; \boldsymbol{d} \cdot \underbrace{\boldsymbol{c}_j \cdot \boldsymbol{h}}_{\Delta_j} \pmod{\mathcal{I}}.\end{aligned}$$

We stress that the right-hand-side of the equality above is *not reduced modulo q*. This means that from a level-i encoding $\boldsymbol{u}$ of an element $\boldsymbol{d} + \mathcal{I}$, we can get a "plaintext version" of $\boldsymbol{d}\cdot\Delta_j$ from some fixed Δ_j (that depends only on the public parameters but not on $\boldsymbol{u}$). This "plaintext version" is not small enough to be a valid level-zero encoding (because Δ_j is roughly the size of $\boldsymbol{h}$, so in particular $\Delta_j > \sqrt{q}$). Nonetheless, we can still use it in attacks.

For starters, we can apply the above procedure to many of the level-one encodings of zero from the public parameters, thereby getting many elements in the ideal $\mathcal{I}$ itself. This by itself still does not yield a basis of $\mathcal{I}$ (since all these elements have the extra factor of h), but in the full version [GGH12] we show how to remove this extra factor and nonetheless compute a basis for $\mathcal{I}$. This is not a small basis of course, but it tells us that we cannot hope to hide the plaintext space $R/\mathcal{I}$ itself.

Next, consider the subgroup membership setting, where we have $\mathbf{g} = \mathbf{g}_1 \cdot \mathbf{g}_2$, we are given a level-1 encoding $\boldsymbol{u} = [\boldsymbol{d}/\mathbf{z}]_q$ and need to decide if $\boldsymbol{d} \in \langle \mathbf{g}_1 \rangle$. Using the procedure above we can get $\boldsymbol{f} = \boldsymbol{d} \cdot \Delta_j$, which belongs to the ideal $\langle \mathbf{g}_1 \rangle$ if $\boldsymbol{d}$ does. Taking the GCD of the ideals $\langle \boldsymbol{f} \rangle$ and $\mathcal{I}$ will then give us the factor $\langle \mathbf{g}_1 \rangle$ with high probability. It follows that the subgroup membership problem is easy for the encoding method above.

Finally, consider getting a matrix of elements $A = (\boldsymbol{a}_{i,j})_{i,j}$, all encoded at some level $i \lneq \kappa$. Using the method above we can get a "plaintext version" of $\Delta_j \cdot M$, which has the same rank as A. Since the decision linear problem is essentially a matrix rank problem, this means that this problem too is easy for this encoding method.

At this point it is worth stressing again that these attacks do not seem to apply to the GDDH problem, specifically because in that problem we need to make a decision about a level-κ encoding, and the "weak discrete log" procedure from above only applies to encoding at levels strictly below κ. The attacks above make it clear that providing encodings of zero in the public parameters (in conjunction with the zero-testing parameter) gives significant power to the adversary. One interesting direction to counter these attacks is to find different randomization tools that can be applied even when we do not have these encodings of zero in the public parameters.

References

[AGHS12] Agrawal, S., Gentry, C., Halevi, S., Sahai, A.: Sampling discrete gaussians efficiently and obliviously. Cryptology ePrint Archive, Report 2012/714 (2012), `http://eprint.iacr.org/`

[BF01] Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)

[BL96] Boneh, D., Lipton, R.J.: Algorithms for black-box fields and their application to cryptography (extended abstract). In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 283–297. Springer, Heidelberg (1996)

[BS03] Boneh, D., Silverberg, A.: Applications of multilinear forms to cryptography. Contemporary Mathematics 324, 71–90 (2003)

[CS97] Coppersmith, D., Shamir, A.: Lattice attacks on ntru. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 52–61. Springer, Heidelberg (1997)

[Gen01] Gentry, C.: Key recovery and message attacks on ntru-composite. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 182–194. Springer, Heidelberg (2001)

[GGH97] Goldreich, O., Goldwasser, S., Halevi, S.: Public-key cryptosystems from lattice reduction problems. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 112–131. Springer, Heidelberg (1997)

[GGH12] Garg, S., Gentry, C., Halevi, S.: Candidate multilinear maps from ideal lattices. Cryptology ePrint Archive, Report 2012/610 (2012), `http://eprint.iacr.org/`

[GGH$^+$13] Garg, S., Gentry, C., Halevi, S., Sahai, A., Waters, B.: Attribute-based encryption for circuits from multilinear maps. Cryptology ePrint Archive, Report 2013/128 (2013), `http://eprint.iacr.org/`

[GGSW13] Garg, S., Gentry, C., Sahai, A., Waters, B.: Witness encryption and its applications. In: STOC (2013)

[GKP$^+$13] Goldwasser, S., Kalai, Y., Popa, R.A., Vaikuntanathan, V., Zeldovich, N.: Succinct functional encryption and applications: Reusable garbled circuits and beyond. In: STOC (2013)

[GS02] Gentry, C., Szydlo, M.: Cryptanalysis of the revised ntru signature scheme. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 299–320. Springer, Heidelberg (2002)

[GVW13] Gorbunov, S., Vaikuntanathan, V., Wee, H.: Predicate encryption for circuits. In: STOC (2013)

[HGS04] Howgrave-Graham, N., Szydlo, M.: A method to solve cyclotomic norm equations. In: Buell, D.A. (ed.) ANTS 2004. LNCS, vol. 3076, pp. 272–279. Springer, Heidelberg (2004)

[HHGP$^+$03] Hoffstein, J., Howgrave-Graham, N., Pipher, J., Silverman, J.H., Whyte, W.: Ntrusign: Digital signatures using the ntru lattice. In: Joye, M. (ed.) CT-RSA 2003. LNCS, vol. 2612, pp. 122–140. Springer, Heidelberg (2003)

[HKL$^+$00] Hoffstein, J., Kaliski, B.S., Lieman, D.B., Robshaw, M.J.B., Yin, Y.L.: Secure user identification based on constrained polynomials. US Patent 6,076,163 (2000)

[HPS98] Hoffstein, J., Pipher, J., Silverman, J.H.: Ntru: A ring-based public key cryptosystem. In: Buhler, J.P. (ed.) ANTS 1998. LNCS, vol. 1423, pp. 267–288. Springer, Heidelberg (1998)

[HPS01] Hoffstein, J., Pipher, J., Silverman, J.H.: Nss: An ntru lattice-based signature scheme. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 211–228. Springer, Heidelberg (2001)

[Jou00] Joux, A.: A one round protocol for tripartite diffie-hellman. In: Bosma, W. (ed.) ANTS 2000. LNCS, vol. 1838, pp. 385–394. Springer, Heidelberg (2000)

[MR07] Micciancio, D., Regev, O.: Worst-case to average-case reductions based on gaussian measures. SIAM J. Computing 37(1), 267–302 (2007)

[NR06] Nguyên, P.Q., Regev, O.: Learning a parallelepiped: Cryptanalysis of ggh and ntru signatures. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 271–288. Springer, Heidelberg (2006)

[PTT10] Papamanthou, C., Tamassia, R., Triandopoulos, N.: Optimal authenticated data structures with multilinear forms. In: Joye, M., Miyaji, A., Otsuka, A. (eds.) Pairing 2010. LNCS, vol. 6487, pp. 246–264. Springer, Heidelberg (2010)

[Reg05] Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: STOC, pp. 84–93 (2005)

[Rot13] Rothblum, R.D.: On the circular security of bit-encryption. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 579–598. Springer, Heidelberg (2013)

[RS09] Rückert, M., Schröder, D.: Aggregate and verifiably encrypted signatures from multilinear maps without random oracles. In: Park, J.H., Chen, H.-H., Atiquzzaman, M., Lee, C., Kim, T.-h., Yeo, S.-S. (eds.) ISA 2009. LNCS, vol. 5576, pp. 750–759. Springer, Heidelberg (2009)

[SOK00] Sakai, R., Ohgishi, K., Kasahara, M.: Cryptosystems based on pairing. In: SCIS 2000, Okinawa, Japan (January 2000)

[Szy03] Szydlo, M.: Hypercubic lattice reduction and analysis of ggh and ntru signatures. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 433–448. Springer, Heidelberg (2003)

Lossy Codes and a New Variant of the Learning-With-Errors Problem

Nico Döttling and Jörn Müller-Quade

Karlsruhe Institute of Technology, Karlsruhe, Germany
{doettling,mueller-quade}@kit.edu

Abstract. The hardness of the Learning-With-Errors (LWE) Problem has become one of the most useful assumptions in cryptography. It exhibits a worst-to-average-case reduction making the LWE assumption very plausible. This worst-to-average-case reduction is based on a Fourier argument and the errors for current applications of LWE must be chosen from a gaussian distribution. However, sampling from gaussian distributions is cumbersome.

In this work we present the first worst-to-average case reduction for LWE with uniformly distributed errors, which can be sampled very efficiently. This new worst-to-average-case connection comes with a slight drawback and we need to use a bounded variant of the LWE problem, where the number of samples is fixed in advance. Most applications of LWE can be based on the bounded variant. The proof is based on a new tool called *lossy codes*, which might be of interest in the context other lattice/coding-based hardness assumptions.

Keywords: Learning-With-Errors, Worst-Case Reduction, Uniform Interval Error-Distribution.

1 Introduction

The Learning-with-Errors (LWE) Problem asks to recover an unknown vector $\mathbf{x} \in \mathbb{Z}_q^n$, given a random matrix $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ and a *noisy-codeword* $\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{e}$, where $\mathbf{e} \in \mathbb{Z}_q^m$ is chosen from an error-distribution χ^m. This problem has had a significant impact in cryptography since its conception in 2005 [Reg05]. Maybe the most intriguing feature of this problem is its worst-to-average case connection [Reg05, Pei09]. This basically allows to transform an efficient adversary solving LWE on average, into an efficient (quantum) algorithm solving lattice problems in the worst case. Beyond this very strong hardness-guarantee, the problem has unmatched cryptographic versatility. It allows for IND-CPA and IND-CCA secure encryption [Reg05, GPV08, Pei09], lossy-trapdoor functions [PW08], (hierarchical) identity-based encryption [CHKP10, ABB10], fully homomorphic encryption [BV11, BGV12, Bra12] and many more. The worst-to-average-case reductions [Reg05, Pei09] crucially rely on the Fourier-properties of gaussian error-distributions. This has the consequence that the cryptographic applications also need to use a gaussian error-distribution. For the above-mentioned encryption-schemes, sampling from a gaussian error-distribution is usually the computationally heaviest step (which occurs mostly during key-generation). It would thus be desirable to have a variant of the LWE problem enjoying the same worst-to-average-case connection, but that comes with an easier-to-sample error-distribution. Micciancio and Mol [MM11a] write:

> "Can lattice-based hardness results for search LWE be extended to noise distributions other than Gaussian? Can we show similar lattice-based hardness results if the noise is distributed uniformly at random modulo 2^i? The latter case is very attractive from a practical viewpoint since arithmetic modulo 2 and sampling from uniform distributions can be implemented very efficiently."

1.1 Our Contribution

In this work we present the first instantiation of the LWE problem with worst-to-average case connection where the error-distribution is the uniform distribution on a small interval $[-r, r]$ (call this distribution $\mathcal{U}([-r, r])$). In particular, setting $r = 2^i$, this answers the question of [MM11a]. Rather than proving a new worst-to-average case reduction, we will build ours on top of existing ones. More precisely, the gaussian error-distributions will appear in the hardness-reduction, but not in the LWE instantiation itself. Our main-lever to achieve this is a technique which we call *lossy codes*. Roughly speaking, lossy codes are pseudorandom codes that seem to be good codes. However, encoding messages with a lossy code and adding certain errors *provably* annihilates the message (on average). On the other hand, encoding the same message using a truly random code and adding the same type of error preserves the message, i.e. the message can be recovered *information theoretically* (yet not efficiently). Using a proof-strategy pioneered by Peikert and Waters [PW08], we conclude that recovering the message when encoding with a random code and adding noise must be computationally hard. If this was not the case, lossy codes could be efficiently distinguished from random codes, contradicting the pseudorandomness-property of lossy codes. The main-part of this work is devoted to proving that a very simple construction of lossy codes for LWE *actually is lossy* for the error-distribution $\mathcal{U}([-r, r])$. The key-insight for this construction is that the standard LWE problem with gaussian error-distribution allows us to implant many very short vectors into a random looking lattice. Our resulting worst-to-average case connection-factor for LWE with error-distribution $\mathcal{U}([-r, r])$ depends on the number of samples provided by LWE (while those for standard LWE [Reg05, Pei09] do not). We will therefore consider an m-bounded LWE problem $\mathsf{LWE}(n, m, q, \mathcal{U}([-r, r]))$, where the number of samples m has a fixed $\mathsf{poly}(n)$ upper bound (rather than being arbitrary $\mathsf{poly}(n)$ depending on the adversary, like in the standard LWE problem). As lossy codes are basically an information-theoretical technique, this seems unavoidable. However, this drawback is still quite mild compared to the super-polynomial inapproximability assumptions made in other works [GKPV10, BV11, Bra12]. We now state our main-theorem.

Theorem 1 (Main Theorem). *Let n be a security parameter and let $\sigma \in (0,1)$ be an arbitrarily small constant. Let $q = q(n)$ be a modulus and $m = m(n) = \mathsf{poly}(n)$ be a integer with $m \geq 3n$. Let $\rho = \rho(n) \in (0, 1/10)$ be such that $\rho q \geq 2n^{0.5+\sigma}m$. If there exists a PPT-algorithm that solves $\mathsf{LWE}(n, m, q, \mathcal{U}([-\rho q, \rho q]))$ with non-negligible probability, then there exists an efficient quantum-algorithm that approximates the decision-version of the shortest vector problem (GAPSVP) and the shortest independent vectors problem (SIVP) to within $\tilde{O}(n^{1+\sigma}m/\rho)$ in the worst case.*

Applying the search-to-decision reduction of [MM11b], we can conclude as a corollary that the decisional variant $\mathsf{DLWE}(n, m, q, \mathcal{U}([-\rho q, \rho q]))$ is also hard. Finally, we believe that the notion of lossy codes might also be useful to transform other lattice/coding-based hardness-assumptions.

1.2 Outline of the Techniques

We will briefly outline the construction and the proof of our main results. The Learning-With-Errors Problem is basically the decoding-problem for q-ary lattices: Given a randomly chosen generator-matrix $\mathbf{A}$ and a vector $\mathbf{y}$, find the nearest lattice point (or codeword) $\mathbf{Ax}$, under the promise that $\mathbf{y}$ was generated by drawing a random point from the lattice and adding an error by some specified distribution. We want to show that this decoding-problem is hard if the error is component-wise chosen by $\mathcal{U}([-r, r])$, i.e. from the uniform distribution on some interval $[-r, r]$. Assume that we knew that there exists a distribution of *lossy* matrices $\mathbf{A}'$ such that that the decoding-problem has no unique solution if the errors come from $\mathcal{U}([-r, r])$, i.e., adding noise to a lattice-point $\mathbf{A}'\mathbf{x}$ loses information about $\mathbf{x}$. If distinguishing such matrices from truly random matrices is hard, we can conclude that the decoding-problem must be hard for truly random matrices. Otherwise, given a decoder for random matrices we can distinguish random matrices from lossy matrices. The distinguisher samples random challenges for the decoder. If the decoder succeeds significantly often, i.e. if it outputs the same $\mathbf{x}$ that was used to sample the instance, then the given matrix must come from the random distribution, as this behavior is impossible for the lossy distribution. Thus, our task is to construct a distribution of lossy codes for the error-distribution $\mathcal{U}([-r, r])$. Our starting-point to find such a distribution is the observation that the standard LWE-problem allows us to construct pseudorandom matrices that generate lattices which contain many vectors that are significantly shorter than one would expect for lattices generated by truly random matrices. Let $\mathbf{G} \in \mathbb{Z}_q^{m \times n}$ be component-wise chosen according to a (short) discretized gaussian distribution $\bar{\Psi}_\alpha$. We want to set the parameters α and r such that the lattice generated by $\mathbf{G}$ is "bad" on average against errors from $\mathcal{U}([-r, r])$. Put differently, if $\mathbf{y} = \mathbf{Gx} + \mathbf{e}$, where $\mathbf{x}$ is chosen uniformly at random and $\mathbf{e}$ is chosen from $\mathcal{U}([-r, r])^m$, we want that, with overwhelming probability, there exist at least one more "admissible" $\mathbf{x}' \neq \mathbf{x}$ and $\mathbf{e}' \in [-r, r]^m$ such that $\mathbf{y} = \mathbf{Gx}' + \mathbf{c}'$. As $\mathbf{e}$ is distributed uniformly on the volume $[-r, r]^m$, each $\mathbf{x}'$ will have the same posterior-probability given $\mathbf{G}$ and $\mathbf{y}$. If there is at least one such

$\mathbf{x}'$, then $\mathbf{y}$ statistically hides at least one bit of information about $\mathbf{x}$ and we can implement the distinguisher sketched above. To make this lossy code pseudorandom, we *hide* the matrix $\mathbf{G}$ in a bigger matrix $\mathbf{A}$. This can be achieved in a pretty standard way. Let $\mathbf{A}' \in \mathbb{Z}_q^{m \times n}$ be chosen uniformly at random. Define $\mathbf{B} = (\mathbf{A}' \| \mathbf{G})$ as the concatenation of $\mathbf{A}'$ and $\mathbf{G}$. $\mathbf{B}$ now contains the $\mathbf{G}$ as a submatrix. Thus, $\mathbf{B}$ has a *lossy sub-code.* As having a lossy sub-code is sufficient to be lossy, $\mathbf{A}$ is also lossy. We can randomize the generator-matrix $\mathbf{B} = (\mathbf{A}' \| \mathbf{G})$ by applying the transformation

$$\mathbf{T} = \begin{pmatrix} \mathbf{I} & \mathbf{T}' \\ \mathbf{0} & \mathbf{I} \end{pmatrix},$$

for a $\mathbf{T}' \in \mathbb{Z}_q^{n \times n}$ chosen uniformly at random. This yields the randomized generator $\mathbf{A} = \mathbf{B}\mathbf{T} = (\mathbf{A}' \| \mathbf{A}'\mathbf{T}' + \mathbf{G})$ for the same code. By the LWE-assumption (for specific parameters), the matrix $\mathbf{A}$ is pseudorandom.

Assume that $\bar{\Psi}_\alpha$ is B-bounded, where $B \ll r$. We still need to show that a matrix $\mathbf{G}$ chosen from $\bar{\Psi}_\alpha^{m \times n}$ is (with high probability) lossy for the error-distribution $\mathcal{U}([-r, r])$. By linearity, it is sufficient to show that for an $\mathbf{e}$ chosen from $\mathcal{U}([-r, r])^m$ there exist $\mathbf{x}' \neq 0$ and $\mathbf{e}' \in [-r, r]^m$ such that $\mathbf{e} = \mathbf{G} \cdot \mathbf{x}' + \mathbf{e}'$, with high probability. Then $\mathbf{e}$ can be reached from either $\mathbf{x}_1 = 0$ or $\mathbf{x}_2 = \mathbf{x}' \neq 0$ and we have established 1-bit loss (which is sufficient for the above construction).

Consider a slightly simpler problem, namely when $\mathbf{G}$ only consists of a single column $\mathbf{g}$. We first observe that errors $\mathbf{e}$ drawn from $\mathcal{U}([-r, r])^m$ show the following *typical* behavior: If we draw $\mathbf{g}$ according to $\bar{\Psi}_\alpha^m$, then there is a substantial chance (over the choice of $\mathbf{g}$) that it holds $\mathbf{e} - \mathbf{g} \in [-r, r]^m$. An $\mathbf{e}$ drawn from $\mathbf{U}([-r, r])^m$ has this property with high probability. To see this, note that there are not too many components e_i of $\mathbf{e}$ that have distance less than B from the boundaries of the interval $[-r, r]$. Call a component e_i with this property (i.e. $e_i \notin [-r + B, r - B]$) bad. For each component e_i, the probability e_i is bad is B/r. Thus, the expected number of components e_i too close to the boundaries is $m \cdot B/r$, which is < 1 for an appropriate choice of m, B and r. We can use a tail-bound for this type of Bernoulli-distribution to show that with overwhelming probability, the number of bad components e_i of $\mathbf{e}$ is less than $\log(n)/2$. Now fix an $\mathbf{e}$ with less than $\log(n)/2$ bad components. For each good component e_i of $\mathbf{e}$, it holds that $e_i + g_i \in [-r, r]$ as g_i is $B-$*bounded.* For all bad components e_i, the probability that $e_i - g_i \in [-r, r]$ is at least $1/2$, as $\bar{\Psi}_\alpha$ is symmetric and thus there is only a $1/2$ chance that g_i *goes the wrong way.* All together, it holds that $\mathbf{e} + \mathbf{g} \in [-r, r]^m$ with probability at least $\left(\frac{1}{2}\right)^{\log(n)/2} = \frac{1}{\sqrt{n}}$, which is substantial.

Now, return to the original problem. Fix an $\mathbf{e}$ that is typical in the above sense. As $\mathbf{G}$ has n columns $\mathbf{g}_1, \ldots, \mathbf{g}_n$ independently chosen from $\bar{\Psi}_\alpha^m$, the probability that there is at least one $\mathbf{g}_i$ such that $\mathbf{e} - \mathbf{g}_i \in [-r, r]^m$ is at least $1 - e^{-\sqrt{n}}$, which is overwhelming. Thus there exists an $\mathbf{x}' \neq 0$ (which is the i-th unit-vector) such that $\mathbf{e} = \mathbf{G}\mathbf{x}' + \mathbf{e}'$ and we are done.

1.3 Related Work

Recently, there has been a growing interest to instantiate new LWE variants. In [GKPV10] an LWE variant was introduced where the secret $\mathbf{x}$ is chosen by a distribution with a sufficient amount of min-entropy, rather than uniformly at random. Lyubashevsky et. al [LPR10] introduced the Ring-LWE problem and provided a worst-to-average-case reduction from the GAPSVP problem in ideal lattices to Ring-LWE. Applebaum et al. [AIK11] noticed, that if the LWE-modulus q is super-polynomial, then the gaussian error-distribution can be "overridden" by a sufficiently (super-polynomially) wider rectangular uniform distribution. This however requires the underlying worst-case lattice-problems to be hard to approximate to within a super-polynomial factor. In [BPR12] an LWE-variant called Learning-With-Rounding (LWR) was introduced. LWR-samples are of the form $(\mathbf{a}, \lfloor \langle \mathbf{a}, \mathbf{x} \rangle \cdot p/q \rceil)$ (for two moduli p and q). Remarkably, the problem inherits the worst-to-average case connection from the corresponding standard LWE problem modulo q, without making use of a gaussian error-distribution by itself. To establish worst-case hardness of LWR, [BPR12] need to assume that the underlying worst-case lattice-problems are hard to approximate to within a super-polynomial factor. Bellare et al. [BKPW12] construct identity-based lossy trapdoor functions based on the hardness of the decisional-linear problem and LWE. The LWE-based construction of lossy trapdoor functions in [BKPW12] has some similarities with the lossy-codes construction in this work, though the technical details and analysis are incomparable. Finally, Pietrzak [Pie12] gave an adaptively secure instantiation of LWE called Subspace-LWE, where the adversary is allowed to learn inner products of the secret $\mathbf{x}$ after it has been projected on an adversarially chosen subspace.

1.4 Concurrent Independent Work

Concurrently and independently of our work, Micciancio and Peikert [MP13] established a worst-case connection for LWE with short uniform errors. Specifically, [MP13] shows that a family of instantiations of LWE with short uniform errors, at most linear number of samples and polynomial modulus are as hard as approximating standard worst-case lattice problems to withing a factor of $\tilde{O}(\sqrt{n}q)$. For instance, their result can be instantiated with binary errors and $n \cdot (1 + \Omega(1/\log(n)))$ samples or polynomial errors (n^ϵ for some small ϵ) and a linear number of samples ($m = (1 + \epsilon/3)n$).

The main similarity of [MP13] and our work is on a conceptual level. In both [MP13] and our work a lossiness-argument is essential to establish the main result. To prove their result, Micciancio and Peikert restate the LWE problem in terms of SIS (Short Integer Solution) functions. This formulation states that, given a randomly drawn SIS-function $\mathbf{H} \in \mathbb{Z}_q^{(m-n)\times m}$ and $\mathbf{y} = \mathbf{H}\mathbf{x}$, where $\mathbf{x}$ is drawn from an input-distribution χ^n, it is hard to find $\mathbf{x}$. They establish lossiness of a pseudorandom SIS function-family by counting the number of elements in the image of functions $\mathbf{H}$ chosen from that family (on average). For appropriate parameter-choices, they can conclude that the image $\mathbf{H}(X)$ contains noticeably fewer elements than the domain X, thus $\mathbf{H}$ must be lossy for the uniform distribution on its domain X.

For comparison, in the language of [MP13] our results might be restated as follows. We first construct pseudorandom SIS-functions $\mathbf{H}$ with domain $[-r, r]^m$ that have short vectors $\mathbf{g}_1, \ldots, \mathbf{g}_k$ (drawn from a gaussian distribution) in their kernel. Next, we show that elements $\mathbf{e}$ randomly chosen from $[-r, r]^m$ are *well behaved* in the sense that (with overwhelming probability) there exists a $\mathbf{g}_i$ such that $\mathbf{e} + \mathbf{g}_i \in [-r, r]^m$. Thus, $\mathbf{e}$ and $\mathbf{e} + \mathbf{g}_i$ form a collision for $\mathbf{H}$ (as $\mathbf{H}(\mathbf{e} + \mathbf{g}_i) = \mathbf{He} + \mathbf{Hg}_i = \mathbf{He}$) and we conclude that $\mathbf{H}$ loses at least 1-bit of information on the uniform distribution on $[-r, r]^m$.

2 Preliminaries

We will use the notation $(\mathbf{A}\|\mathbf{B})$ for the horizontal concatenation of two matrices $\mathbf{A}$ and $\mathbf{B}$ and $(\mathbf{x}, \mathbf{y})$ for the vertical concatenation of two vectors $\mathbf{x}$ and $\mathbf{y}$. Let $\mathsf{sgn}(x)$ be the signum-function, i.e. $\mathsf{sgn}(x) = 1$ if $x > 0$, $\mathsf{sgn}(x) = -1$ if $x < 0$ and $\mathsf{sgn}(x) = 0$ if $x = 0$. We denote computational indistinguishability of two distributions $\mathcal{X}$ and $\mathcal{Y}$ by $\mathcal{X} \approx_c \mathcal{Y}$.

2.1 Norms

We will use the $\|\cdot\|_2$- and the $\|\cdot\|_\infty$-norm in this work. The $\|\cdot\|_2$-norm on $\mathbb{R}^n$ is defined by $\|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^n x_i^2}$, the $\|\cdot\|_\infty$-norm on $\mathbb{R}^n$ is defined by $\|\mathbf{x}\|_\infty = \max_{i=1,\ldots,n} |x_i|$. Norms $\|\cdot\|$ are multiplicative and obey the triangle-inequality, i.e. for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ and $\alpha \in \mathbb{R}$ it holds that $\|\alpha\mathbf{x}\| = |\alpha|\|\mathbf{x}\|$ and $\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$. The set $C = \{\mathbf{x} \in \mathbb{R}^n | \|\mathbf{x}\|_\infty \leq r\}$ forms a hypercube of dimension n, i.e. $C = [-r, r]^n$.

2.2 Min-Entropy

Let χ be a probability distribution with finite support and let X be distributed according to χ. Define the *min-entropy* as $H_\infty(X) = -\log(\max_\xi(\Pr[X = \xi]))$. Let Y be random-variable (possibly correlated with X) and let $\tilde{y}$ be a measurement or outcome of Y. The *conditional min-entropy* $H_\infty(X|Y = \tilde{y})$ is defined as $H_\infty(X|Y = \tilde{y}) = -\log(\max_\xi(\Pr[X = \xi|Y = \tilde{y}]))$. Instead of using the *conditional average min-entropy* [DORS08], we will directly derive laws of the form $\Pr_{\tilde{y}}[H_\infty(X|Y = \tilde{y}) \geq \delta] \geq 1 - \epsilon$, i.e. $H_\infty(X|Y = \tilde{y})$ is at least δ, except with probability ϵ over the choice of the measurement $\tilde{y}$. This will enable a more fine-grained analysis of the lossiness of our constructions (the average conditional min-entropy $\tilde{H}_\infty(X|Y)$ would be lower-bounded by $\tilde{H}_\infty(X|Y) \geq -\log(2^{-\delta} + \epsilon)$, i.e. it compresses δ and ϵ into one scalar).

2.3 Binomial Distributions

Let $X_i \in \{0, 1\}$ for $i = 1, \ldots, n$ be iid. random variables with $\Pr[X_i = 1] = p$. Then $X = \sum_{i=1}^n X_i$ is *binomially* distributed with $\Pr[X = k] = \binom{n}{k} p^k (1-p)^{n-k}$.

The binomial-distribution assumes its maximum at an index $k_{max} = \lfloor p(n+1) \rfloor$. The tails of a binomial-distribution can be bounded by the Chernoff-bound, which states that $\Pr[X \leq (1-\delta)\mathsf{E}[X]] \leq e^{-\delta^2 \mathsf{E}[X]/2}$, where the expectation is $\mathsf{E}[X] = p \cdot n$

2.4 Gaussian Distributions

We denote Φ_s the normal-distribution with variance $s^2/(2\pi)$, i.e. if X is distributed according to Φ_s, then X has the probability-density function $p_X(x) = e^{-\pi x^2/s^2}/s$. A standard tail-bound for Gaussians is $\Pr[|X| > t \cdot s] < e^{-\pi t^2}$. Following [Reg05], we denote by $\bar{\Psi}_\alpha$ the discretized gaussian distribution over $\mathbb{Z}$ (or $\mathbb{Z}_q$) with variance $(\alpha q)^2/(2\pi)$, where q is given by the context. More precisely, $\bar{\Psi}_\alpha$ is sampled by taking a sample from $\Phi_{\alpha q}$ and rounding it to the nearest integer. Let Y be distributed according to $\bar{\Psi}_\alpha$, i.e. let $Y = \lceil X \rfloor$ with X distributed by $\Phi_{\alpha q}$. If $t\alpha q \geq 2$, we can derive the tail-bound $\Pr[|Y| > t\alpha q] \leq \Pr[|X| > t\alpha q - 1] \leq \Pr[|X| > t\alpha q/2] \leq e^{-\pi t^2/4}$.

2.5 Lattices

Let $\mathbf{B} \in \mathbb{Z}^{m \times n}$ be a full rank-matrix. The *lattice* $\Lambda(\mathbf{B})$ is defined as $\Lambda(\mathbf{B}) = \{\mathbf{Bx} \in \mathbb{Z}^m : \mathbf{x} \in \mathbb{Z}^n\}$, i.e. the lattice $\Lambda(\mathbf{B})$ is the set of all integer-linear combination of columns of $\mathbf{B}$. Let $q \geq 2$ be an integer. The q-ary lattice $\Lambda_q(\mathbf{B})$ is defined as $\Lambda_q(\mathbf{B}) = \{\mathbf{y} \in \mathbb{Z}^m : \exists \mathbf{x} \in \mathbb{Z}^n : \mathbf{y} \equiv \mathbf{Bx} \mod q\}$. Observe that the lattice $\Lambda_q(\mathbf{B})$ contains $q\mathbb{Z}^m$ as a sublattice, therefore $\Lambda_q(\mathbf{B})$ is always full-rank. Moreover, it holds that $\Lambda(\mathbf{B}) \subseteq \Lambda_q(\mathbf{B})$, as $\mathbf{x} \in \Lambda_q(\mathbf{B})$, for each $\mathbf{x} \in \mathsf{columns}(\mathbf{B})$.

We will generally assume that elements of $\mathbb{Z}_q$ are given in the central residue-class representation, i.e. if $x' \in \mathbb{Z}_q$, we will identify $x' = x \mod q$ with an integer x in $\{-\lfloor q/2 \rfloor, \ldots, \lceil q/2 \rceil - 1\}$. We can thus generically lift x' from $\mathbb{Z}_q$ to $\mathbb{Z}$. Moreover, with this we can define a meaningful "norm" on $\mathbb{Z}_q$ by $\|\mathbf{x} \mod q\|_\infty = \|\mathbf{x}\|_\infty$.

2.6 Learning-With-Errors

As mentioned above, we will consider an m-bounded LWE-problem, where the adversary is given $m(n) = \mathsf{poly}(n)$ samples (which we can write conveniently in matrix-form).

Problem 1. **m-bounded LWE Search-Problem, Average-Case Version.** Let n be a security parameter, let $m = m(n) = \mathsf{poly}(n)$ and $q = q(n) \geq 2$ be integers and χ be a distribution on $\mathbb{Z}_q$. Let $\mathbf{x} \in \mathbb{Z}_q^n$ be chosen uniformly at random, let $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ be chosen uniformly at random an let $\mathbf{e}$ be chosen according to χ^m. The goal of the $\mathsf{LWE}(n, m, q, \chi)$ problem is, given $(\mathbf{A}, \mathbf{Ax} + \mathbf{e})$, to find $\mathbf{x}$.

We remark that most cryptographic applications of the LWE problem require only an a-priori fixed number of samples. For those applications, the formulation of Problem 1 poses no restriction. The notable exception to this are the

KDM-secure encryption scheme in [ACPS09] and the pseudorandom functions in [BPR12]. For both schemes the number of LWE-samples required is determined adversarially. Regev [Reg05] and Peikert [Pei09] established worst-to-average-case connections between worst-case lattice problems and Problem 1 for suitable parameter-choices. For our construction, we will rely on the theorem of Regev [Reg05].

Theorem 2 (Worst-to-Average Case Reduction [Reg05]). *Let n be a security parameter and $q = q(n)$ be a modulus, let $\alpha = \alpha(n) \in (0, 1)$ be such that $\alpha q > 2\sqrt{n}$. If there exists a PPT-algorithm solving $\mathsf{LWE}(n, m, q, \bar{\Psi}_\alpha)$ with non-negligible probability, then there exists an efficient quantum-algorithm that approximates the decision-version of the shortest vector problem (GAPSVP) and the shortest independent vectors problem (SIVP) to within $\tilde{O}(n/\alpha)$ in the worst case.*

The LWE distinguishing-problem DLWE asks to distinguish the distribution of Problem 1 from uniform random. Thus, the hardness of the DLWE problem states that the LWE-distribution is pseudorandom.

Problem 2. **m-bounded LWE Distinguishing-Problem** Let n be a security parameter, let $m = m(n) = \mathsf{poly}(n)$ and $q = q(n) \geq 2$ be integers and χ be a distribution on $\mathbb{Z}_q$. Let $\mathbf{x} \in \mathbb{Z}_q^n$ be chosen uniformly at random, let $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ be chosen uniformly at random an let $\mathbf{e}$ be chosen according to χ^m. The goal of the $\mathsf{DLWE}(n, m, q, \chi)$ problem is, given $(\mathbf{A}, \mathbf{y})$, to decide whether $\mathbf{y}$ is distributed by $\mathbf{Ax} + \mathbf{e}$ or chosen uniformly at random from $\mathbb{Z}_q^m$.

There are several search-to-decision reductions basing the hardness of Problem 2 on the hardness of Problem 1 [Reg05, Pei09, MP12, MM11b]. The one most suitable for our instantiation is due to Micciancio and Mol [MM11a, MM11b]. Their search-to-decision reduction works for any error-distribution χ and is *sample-preserving* (i.e. the distinguisher requires the same amount of samples as the search-adversary).

Theorem 3 (Search-to-Decision [MM11b]). *Let $q = q(n) = \mathsf{poly}(n)$ be a prime modulus and let χ be* any *distribution over $\mathbb{Z}_q$. Assume there exists a PPT-distinguisher $\mathcal{D}$ that distinguishes $\mathsf{DLWE}(n, m, q, \chi)$ with non-negligible advantage, then there exists a PPT-adversary $\mathcal{A}$ that inverts $\mathsf{LWE}(n, m, q, \chi)$ with non-negligible success-probability.*

Finally, we need a *matrix-version* of Problem 2. The hardness of the matrix-version can be easily established using a hybrid-argument (see e.g. [ACPS09]).

Lemma 1. *Let $m(n), k(n) = \mathsf{poly}(n)$. Assume that $\mathsf{DLWE}(n, m, q, \chi)$ is pseudorandom. Then the distribution $(\mathbf{A}, \mathbf{AX} + \mathbf{E})$ is also pseudorandom, where $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ and $\mathbf{X} \in \mathbb{Z}_q^{n \times k}$ are chosen uniformly at random and $\mathbf{E}$ is chosen according to $\bar{\Psi}_\alpha^{m \times k}$.*

3 Lossy Codes

In this section, we introduce the main technical tool of this work, lossy codes, and show that the existence of lossy codes implies that the associated decoding problems for random codes are hard.

Definition 1 (Families of Lossy Codes). *Let n be a security parameter, let $q = q(n)$ be a modulus, let $m = m(n) = \mathsf{poly}(n)$ and $\gamma = \gamma(n)$. Let $\{\mathcal{C}_{n,m,q}\}$ be a family of distributions where $\mathcal{C}_{n,m,q}$ is defined on $\mathbb{Z}_q^{m\times n}$ and let χ be a distribution on $\mathbb{Z}_q$. Finally, let $\mathcal{U}(\mathbb{Z}_q^{m\times n})$ be the uniform distribution on $\mathbb{Z}_q^{m\times n}$. We say that $\{\mathcal{C}_{n,m,q}\}$ is γ-lossy for the error-distribution χ, if the following 3 properties hold.*

1. *$\mathcal{C}_{n,m,q}$ **is pseudorandom:** It holds that $\mathcal{C}_{n,m,q} \approx_c \mathcal{U}(\mathbb{Z}_q^{m\times n})$.*
2. *$\mathcal{C}_{n,m,q}$ **is lossy:** Let $\mathbf{A}$ be distributed according to $\mathcal{C}_{n,m,q}$, $\mathbf{y} = \mathbf{A}\cdot\tilde{\mathbf{x}} + \tilde{\mathbf{e}}$ (where $\tilde{\mathbf{x}}$ is chosen uniformly from $\mathbb{Z}_q^n$ and $\tilde{\mathbf{e}}$ is distributed according to χ^m), let $\mathbf{x}$ be chosen uniformly from $\mathbb{Z}_q^n$ and let $\mathbf{e}$ be chosen according to χ^m. Then it holds that $\Pr_{(\mathbf{A},\mathbf{y})}[H_\infty(\mathbf{x}|\mathbf{Ax}+\mathbf{e}=\mathbf{y}) \geq \gamma] \geq 1 - \mathsf{negl}(n)$.*
3. *$\mathcal{U}(\mathbb{Z}_q^{m\times n})$ **is non-lossy:** Let $\mathbf{A}$ be distributed according to $\mathcal{U}(\mathbb{Z}_q^{m\times n})$, $\mathbf{y} = \mathbf{A}\cdot\tilde{\mathbf{x}}+\tilde{\mathbf{e}}$ (where $\tilde{\mathbf{x}}$ is chosen uniformly from $\mathbb{Z}_q^n$ and $\tilde{\mathbf{e}}$ is distributed according to χ^m), let $\mathbf{x}$ be chosen uniformly from $\mathbb{Z}_q^n$ and let $\mathbf{e}$ be chosen according to χ^m. Then it holds that $\Pr_{(\mathbf{A},\mathbf{y})}[H_\infty(\mathbf{x}|\mathbf{Ax}+\mathbf{e}=\mathbf{y}) = 0] \geq 1 - \mathsf{negl}(n)$.*

Remark. Notice that while we require the error-distribution χ to be efficiently samplable, we do not require the distribution $\mathcal{C}_{n,m,q}$ of lossy codes to be efficiently samplable. In our construction in the next section however, $\mathcal{C}_{n,m,q}$ will be efficiently samplable.

Our main motivation for defining lossy codes is proving that the decoding-problem of recovering $\mathbf{x}$ given a matrix $\mathbf{A}$ and a noisy codeword $\mathbf{Ax}+\mathbf{e}$, where $\mathbf{A}$ and $\mathbf{x}$ are chosen uniformly and $\mathbf{e}$ is chosen from χ^m, is computationally hard, even though $\mathbf{x}$ is information-theoretically (with overwhelming probability) uniquely defined.

Theorem 4. *Let n be a security-parameter, let $m = m(n) = \mathsf{poly}(n)$ and let $q = q(n)$ be a modulus. Let the distribution χ on $\mathbb{Z}_q$ be efficiently samplable.*

1. *Let χ be a uniform distribution with efficiently decidable support. Then the problem $\mathsf{LWE}(n,m,q,\chi)$ is hard, given that there exists a family of 1-lossy codes $\mathcal{C}_{n,m,q} \in \mathbb{Z}_q^{m\times n}$ for the error-distribution χ.*
2. *Let $\gamma = \gamma(n) = \omega(\log(n))$. Then $\mathsf{LWE}(n,m,q,\chi)$ is hard, given that there exists a family of γ-lossy codes $\mathcal{C}_{n,m,q} \in \mathbb{Z}_q^{m\times n}$ for the error-distribution χ.*

Proof. First notice that due to the non-lossiness property of $\mathcal{U}(\mathbb{Z}_q^{m\times n})$, instances of $\mathsf{LWE}(n,m,q,\chi)$ have a unique solution, except with negligible probability. For contradiction, let $\mathcal{A}$ be a PPT-adversary that solves $\mathsf{LWE}(n,m,q,\chi)$ with non-negligible probability ϵ.

We will begin with the first statement of the theorem. Using $\mathcal{A}$, we will construct a PPT-distinguisher $\mathcal{D}$ that distinguishes $\mathcal{C}_{n,m,q}$ and $\mathcal{U}(\mathbb{Z}_q^{m\times n})$ with non-negligible advantage. Say that a solution $\mathbf{x}$ for an instance $(\mathbf{A}, \mathbf{y})$ is valid, if $\mathbf{y} - \mathbf{A} \cdot \mathbf{x}$ is in the support of the error-distribution χ.

There are two different behaviors that algorithm $\mathcal{A}$ could exhibit when receiving inputs of the form $(\mathbf{A}, \mathbf{y})$, where $\mathbf{A}$ is chosen from $\mathcal{C}_{n,m,q}$ and $\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{e}$. In the first case, the probability that $\mathcal{A}$ outputs a valid solution $\mathbf{x}$ is negligible. In the second case, there exists a non-negligible ϵ' such that the probability that $\mathcal{A}$ outputs a valid solution $\mathbf{x}$ with probability at least ϵ'.

In the first case we can construct $\mathcal{D}$ as follows. Let $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ be $\mathcal{D}$'s input. It first samples $\mathbf{x}$ uniformly at random, samples $\mathbf{e}$ according to χ^m and sets $\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{e}$. It then runs $\mathcal{A}$ on input $(\mathbf{A}, \mathbf{y})$. If $\mathcal{A}$ outputs $\mathbf{x}$, $\mathcal{D}$ outputs 1, otherwise $\mathcal{D}$ outputs 0. Clearly, if $\mathbf{A}$ is chosen according to $\mathcal{U}(\mathbb{Z}_q^{m \times n})$, then $\mathcal{A}$ recovers $\mathbf{x}$ with probability at least ϵ. On the other hand, if $\mathbf{A}$ is chosen according to $\mathcal{C}_{n,m,q}$, then $\mathcal{A}$ recovers $\mathbf{x}$ only with negligible probability. Thus it holds that $\mathsf{Adv}(\mathcal{D}) = |\Pr[\mathcal{D}(\mathcal{U}(\mathbb{Z}_q^{m \times n})) = 1] - \Pr[\mathcal{D}(\mathcal{C}_{n,m,q}) = 1]| = \epsilon(n) - \mathsf{negl}(n)$, which is non-negligible.

In the second case, we construct $\mathcal{D}$ differently. Let $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ be $\mathcal{D}$'s input. $\mathcal{D}$ samples $\mathbf{x}$ uniformly at random, $\mathbf{e}$ according to χ^m and sets $\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{e}$. It then runs $\mathcal{A}$ on input $(\mathbf{A}, \mathbf{y})$. If $\mathcal{A}$ outputs an $\mathbf{x}' \neq \mathbf{x}$ such that $\mathbf{e}' = \mathbf{y} - \mathbf{A}\mathbf{x}'$ is in the support of χ^m, it outputs 1, otherwise 0. First, observe that such a *collision* $\mathbf{x}' \neq \mathbf{x}$ cannot exist (except with negligible probability) if $\mathbf{A}$ is chosen according to the uniform distribution $\mathcal{U}(\mathbb{Z}_q^{m \times n})$. This is due to the non-lossiness property of $\mathcal{U}(\mathbb{Z}_q^{m \times n})$. On the other hand, consider that $\mathbf{A}$ is chosen according to $\mathcal{C}_{n,m,q}$. Then it holds (with overwhelming probability) that $H_\infty(\mathbf{x} | \mathbf{A}\mathbf{x} + \mathbf{e} = \mathbf{y}) \geq 1$. Thus it holds (even for an unbounded $\mathcal{A}$) that $\mathcal{A}$ outputs the same $\mathbf{x}$ that was chosen by $\mathcal{D}$ with probability at most $1/2$, conditioned that $\mathcal{A}$ outputs a valid $\mathbf{x}$. Thus, conditioned that $\mathcal{A}$ gives a valid output, there is a chance of $1/2$ that $\mathcal{A}$ outputs a valid $\mathbf{x}' \neq \mathbf{x}$. As $\mathcal{A}$ gives a valid output with probability at least ϵ', $\mathcal{A}$ outputs a collision $\mathbf{x}'$ with probability at least $\epsilon'/2$. Thus $\mathcal{D}$ distinguishes $\mathcal{U}(\mathbb{Z}_q^{m \times n})$ from $\mathcal{C}_{n,m,q}$ with advantage at least $\epsilon'/2$, which is non-negligible.

We now turn to the second statement of the theorem. In this case the construction of the distinguisher $\mathcal{D}$ is straightforward. Let $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ be $\mathcal{D}$'s input. As before, $\mathcal{D}$ samples $\mathbf{x}$ uniformly at random, $\mathbf{e}$ according to χ^m, sets $\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{e}$ and runs $\mathcal{A}$ on input $(\mathbf{A}, \mathbf{y})$. If $\mathcal{A}$ outputs $\mathbf{x}$ it outputs 1, otherwise 0. Again, if $\mathbf{A}$ was chosen from $\mathcal{U}(\mathbb{Z}_q^{m \times n})$, then $\mathcal{A}$ outputs $\mathbf{x}$ (which is in this case unique) with probability at least ϵ. On the other hand, if $\mathbf{A}$ comes from the distribution $\mathcal{C}_{n,m,q}$, then $\mathcal{A}$ finds $\mathbf{x}$ with probability at most $2^{-H_\infty(\mathbf{x}|\mathbf{A}\mathbf{x}+\mathbf{e}=\mathbf{y})} \leq 2^{-\gamma(n)}$ (this holds with overwhelming probability in the choice of $\mathbf{A}$ and $\mathbf{y}$), which is negligible (as $\gamma(n) = \omega(\log(n))$). All together, $\mathcal{D}$ distinguishes $\mathcal{U}(\mathbb{Z}_q^{m \times n})$ from $\mathcal{C}_{n,m,q}$ with advantage at least $\epsilon - 2^{-\gamma}$, which is non-negligible.

4 Construction of Lossy Codes for Uniform Errors from Standard-LWE

We will now provide the details of the construction outlined in Section 1.2.

Construction 1. *Let n be a security parameter, let $q = q(n)$ be a modulus, $m = m(n) = \mathsf{poly}(n)$ and $k = k(n) \leq n$. The distribution $\mathcal{C}_{n,m,q,k,\alpha}$ defined on $\mathbb{Z}_q^{m\times n}$ is specified as follows. Choose $\mathbf{A}' \in \mathbb{Z}_q^{m\times(n-k)}$ uniformly at random, choose $\mathbf{T}' \in \mathbb{Z}_q^{(n-k)\times(n-k)}$ uniformly at random and sample $\mathbf{G} \in \mathbb{Z}_q^{m\times k}$ from $\bar{\Psi}_\alpha^{m\times k}$. Output $\mathbf{A} = (\mathbf{A}' \| \mathbf{A}'\mathbf{T}' + \mathbf{G})$.*

We will show that, for certain parameter choices, Construction 1 yields a lossy code for the error-distribution $\mathcal{U}([-r,r])$. The pseudorandomness of the distribution $\mathcal{C}_{n,m,q,k,\alpha}$ can be established directly from Lemma 1, assuming the hardness of $\mathsf{LWE}(n,m,q,\bar{\Psi}_\alpha)$.

Lemma 2. *Let n be a security-parameter, let $q = q(n)$ be a modulus, let $m = m(n) = \mathsf{poly}(n)$, let $k = \lceil \beta n \rceil$ for some constant $\beta \in (0,1)$ and let $\alpha = \alpha(n) \in (0,1)$ with $\alpha q \geq 2\sqrt{n}$. Assuming that $\mathsf{LWE}(n,m,q,\bar{\Psi}_\alpha)$ is hard, it holds that $\mathcal{C}_{n,m,q,k,\alpha} \approx_c \mathcal{U}(\mathbb{Z}_q^{m\times n})$.*

The non-lossiness of the truly random distribution $\mathcal{U}(\mathbb{Z}_q^{m\times n})$ can be established by a simple Gilbert-Varshamov-type argument.

Lemma 3. *Let n be a security parameter, let $\tau > 0$ be a constant, $r = \mathsf{poly}(n)$, $q > (4r+1)^{1+\tau}$ and $m > (1+2/\tau)n$. Let $\mathbf{A}$ be chosen from $\mathcal{U}(\mathbb{Z}_q^{m\times n})$. Then the shortest vector of the lattice $\Lambda_q(\mathbf{A})$ has length (in the $\|\cdot\|_\infty$-norm) greater than $2r$, except with negligible probability. Thus it holds that $\Pr_{(\mathbf{A},\mathbf{y})}[H_\infty(\mathbf{x}|\mathbf{A}\mathbf{x}+\mathbf{e} = \mathbf{y}) = 0] \geq 1 - \mathsf{negl}(n)$.*

Proof. Let $\mathbf{A} \in \mathbb{Z}_q^{m\times n}$ be chosen uniformly at random. Clearly, it holds that $H_\infty(\mathbf{x}|\mathbf{A}\mathbf{x}+\mathbf{e} = \mathbf{y}) = 0$ if the length of the shortest vector in $\Lambda_q(\mathbf{A})$ (in the $\|\cdot\|_\infty$-norm) is greater than $2r$. Now fix a vector $\mathbf{x} \neq 0 \in \mathbb{Z}_q^n$. Then the vector $\mathbf{A}\cdot\mathbf{x}$ is distributed uniformly at random in $\mathbb{Z}_q^m$. Thus it holds that $\Pr_\mathbf{A}[\|\mathbf{A}\cdot\mathbf{x}\|_\infty \leq 2r] \leq \left(\frac{4r+1}{q}\right)^m$. Thus, a union-bound yields that $\Pr[\exists x \neq 0 \in \mathbb{Z}_q^n : \|\mathbf{A}\mathbf{x}\|_\infty \leq r] \leq \frac{(4r+1)^m}{q^{m-n}}$. This expression is negligible whenever $(m-n)\log(q) - m\log(4r+1) > \omega(\log(n))$. This is certainly the case if $r = \mathsf{poly}(n)$, $q > (4r+1)^{1+\tau}$ and $m > (1+2/\tau))n$ for some constant $\tau > 0$.

We now turn to showing that $\mathcal{C}_{n,m,q,k,\alpha}$ fulfills the lossiness-requirement.

Definition 2. *We say that a vector $\mathbf{y} \in \mathbb{Z}_q^m$ is* N-ambiguous *for a matrix $\mathbf{A}$ and a distance r, if $|\{\mathbf{x} \in \mathbb{Z}_q^n | \|\mathbf{y} - \mathbf{A}\cdot\mathbf{x}\|_\infty \leq r\}| \geq N$. If $\mathbf{A}$ and r are clear by context, we just say that $\mathbf{y}$ is N-ambiguous.*

Notice that if $\mathbf{y}$ is N-ambiguous, then for every $\mathbf{z} \in \mathbb{Z}_q^n$ by linearity it holds that $\mathbf{y} + \mathbf{A}\mathbf{z}$ is also N-ambiguous. Since we want to establish lossiness for the uniform distribution $\mathcal{U}([-r,r])$, counting the number of possible preimages is sufficient, as each preimage is equally likely. This is formalized in the following lemma.

Lemma 4. *Let n be a security parameter, let $q = q(n)$ be a modulus and let $r = r(n)$ and $N = N(n)$. Fix a matrix $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$. Let $\mathbf{y} \in \mathbb{Z}_q^m$ be N-ambiguous for the matrix $\mathbf{A}$ and distance r. Let $\mathbf{x} \in \mathbb{Z}_q^n$ be chosen uniformly at random and $\mathbf{e}$ be distributed according to $\mathcal{U}([-r,r])^m$. Then it holds that $H_\infty(\mathbf{x}|\mathbf{A}\mathbf{x} + \mathbf{e} = \mathbf{y}) \geq \log(N)$.*

Proof. Since $\mathbf{x}$ and $\mathbf{e}$ are drawn from uniform distributions, $p := \Pr[\mathbf{x} = \tilde{\mathbf{x}}, \mathbf{e} = \tilde{\mathbf{e}}]$ is the same for all $\tilde{\mathbf{x}} \in \mathbb{Z}_q^n$ and $\tilde{\mathbf{e}} \in [-r,r]^m$. Let $X := \{\mathbf{z} \in \mathbb{Z}_q^n | \|\mathbf{y} - \mathbf{A}\mathbf{z}\| \leq r\}$. As $\mathbf{y}$ is N-ambiguous it holds that $|X| \geq N$, thus

$$\Pr[\mathbf{A}\mathbf{x}+\mathbf{e} = \mathbf{y}] = \sum_{\mathbf{z} \in \mathbb{Z}_q^n} \Pr[\mathbf{A}\mathbf{x}+\mathbf{e} = \mathbf{y}, \mathbf{x} = \mathbf{z}] = \sum_{\mathbf{z} \in X} \Pr[\mathbf{e} = \mathbf{y}-\mathbf{A}\mathbf{z}, \mathbf{x} = \mathbf{z}] \geq p \cdot N.$$

Thus it holds for all $\mathbf{z} \in \mathbb{Z}_q^n$ that

$$\Pr[\mathbf{x} = \mathbf{z}|\mathbf{A}\mathbf{x} + \mathbf{e} = \mathbf{y}] = \frac{\Pr[\mathbf{x} = \mathbf{z}, \mathbf{A}\mathbf{x} + \mathbf{e} = \mathbf{y}]}{\Pr[\mathbf{A}\mathbf{x} + \mathbf{e} = \mathbf{y}]} \leq \frac{1}{N}.$$

This immediately implies $H_\infty(\mathbf{x}|\mathbf{A}\mathbf{x} + \mathbf{e} = \mathbf{y}) \geq \log(N)$, which concludes the proof.

The following lemma shows that if we sample $\mathbf{e}$ from $\mathcal{U}([-r,r])^m$, then with overwhelming probability $\mathbf{e}$ is such that if we add a sample $\mathbf{g}$ from an appropriately bounded distribution χ^m, $\mathbf{e} - \mathbf{g}$ is, with substantial probability over the choice of $\mathbf{g}$, also in $[-r,r]^m$. Say that a distribution χ is strictly B-bounded if the support of χ is contained in $[-B,B]$.

Lemma 5. *Let $n, m, B > 0$ be integers, let $r > (m+1)B$ and let $\epsilon < 1/2$. Let χ be a strictly B-bounded symmetrical distribution on $\mathbb{Z}$. Let $\mathbf{e}$ be chosen uniformly at random from $[-r,r]^m$ and let $\mathbf{g}$ be distributed according to χ^m. Then it holds that*

$$\Pr_{\mathbf{e}}\left[\Pr_{\mathbf{g}}[\|\mathbf{e} - \mathbf{g}\|_\infty \leq r] \geq \epsilon\right] \geq 1 - m \cdot \epsilon^{\log(r/(m \cdot B))}.$$

Proof. We will first bound the probability that it holds for more than $k = \lfloor -\log(\epsilon) \rfloor$ components e_i of $\mathbf{e}$ that $|e_i| > r - B$, i.e. that e_i is not in the interval $[-r+B, r-B]$. For $i = 1, \ldots, m$ let Z_i be a random-variable that is 1 if $|e_i| > r - B$ and 0 otherwise. As $e_1, \ldots, e_m$ are iid., $Z_1, \ldots, Z_m$ are also iid. Thus let $p = \Pr[Z_1 = 1] = \cdots = \Pr[Z_m = 1]$. As e_1 is distributed by $\mathcal{U}([-r,r])$ and $p = \Pr[Z_1 = 1] = \Pr[|e_1| > r - B]$ it holds that $(B-1)/r \leq p \leq B/r$. Set $Z = \sum_{i=1}^m Z_i$. Clearly, Z is the number of components of $\mathbf{e}$ that are not in the interval $[-r+B, r-B]$ and it is binomially distributed. We can bound the probability $\Pr[Z > k]$ by

$$\Pr[Z > k] = \sum_{i=k+1}^m \binom{m}{i} p^i (1-p)^{m-i} \overset{(1)}{\leq} m \underbrace{\binom{m}{k+1}}_{\leq m^{k+1}} \underbrace{p^{k+1}}_{\leq (B/r)^{k+1}} \underbrace{(1-p)^{m-k-1}}_{\leq 1}$$

$$\leq m \cdot \left(\frac{m \cdot B}{r}\right)^{k+1} \overset{(2)}{<} m \cdot \left(\frac{m \cdot B}{r}\right)^{-\log(\epsilon)} = m \cdot \epsilon^{\log(r/(m \cdot B))}.$$

Inequality (1) holds, as $\binom{m}{i}p^i(1-p)^{m-i}$ is monotonically decreasing for $i \geq \lfloor (m+1)p \rfloor \geq \lfloor (m+1)(B-1)/r \rfloor = 0$. Inequality (2) holds as $m \cdot B/r < 1$ and $k+1 > -\log(\epsilon)$.

Now, fix an $\mathbf{e}$ and assume that it holds that it holds for at most k components $e_{i_1}, \ldots, e_{i_k}$ of $\mathbf{e}$ that $|e_{i_j}| > r - B$. Let $i \in \{i_1, \ldots, i_k\}$. If $\mathsf{sgn}(g_i) = \mathsf{sgn}(e_i)$, then it holds that $|e_i - g_i| = |e_i| - |g_i| \leq |e_i| \leq r$. As χ is a symmetrical distribution, it holds that $\Pr[\mathsf{sgn}(g_i) = \mathsf{sgn}(e_i)] \geq \frac{1}{2}$. Therefore, it holds that $\Pr[|e_i - g_i| \leq r] \geq \frac{1}{2}$. For all other indexes $j \notin \{i_1, \ldots, i_k\}$ it holds that $|e_j| \leq r - B$. The triangle-inequality yields $|e_j - g_j| \leq |e_j| + |g_j| \leq r - B + B = r$. Therefore, we have that $\Pr[|e_j - g_j| \leq r] = 1$. Putting this together, we get that

$$\Pr[\|\mathbf{e} - \mathbf{g}\|_\infty \leq r] = \prod_{i=1}^{m} \Pr[|e_i - g_i| \leq r] \geq 2^{-k} \geq \epsilon.$$

All together, it holds that

$$\Pr_{\mathbf{e}}[\Pr_{\mathbf{g}}[\|\mathbf{e} - \mathbf{g}\|_\infty \leq r] \geq \epsilon] \geq 1 - m \cdot \epsilon^{\log(r/(m \cdot B))},$$

which concludes the proof.

We can now show that Construction 1 also suffices the lossiness-condition, for appropriate parameters.

Lemma 6. *Let n be a security-parameter, let $m = m(n) = \mathsf{poly}(n)$, let $k = k(n) = \lceil \beta n \rceil$ for some constant $\beta \in (0,1)$ and let $c \in (0,1)$ be a constant. Let $q = q(n)$ be a modulus, $\alpha = \alpha(n) \in (0,1)$, let $B = B(n)$ and assume that the distribution $\bar{\Psi}_\alpha$ is B-bounded, except with negligible probability. Finally let $r = r(n) > 0$ be such that $r \geq m \cdot B n^c$.*

Let $\mathbf{G}$ be chosen according to $\bar{\Psi}_\alpha^{m \times k}$, let the matrix $\mathbf{A}'$ be distributed according to $\mathcal{U}(\mathbb{Z}_q^{m \times (n-k)})$, $\mathbf{T}'$ be distributed according to $\mathcal{U}(\mathbb{Z}_q^{(n-k) \times (n-k)})$ and let $\mathbf{A} = (\mathbf{A}' \| \mathbf{A}'\mathbf{T}' + \mathbf{G})$. Let $\mathbf{y} = \mathbf{A}\mathbf{x}' + \mathbf{e}'$, with $\mathbf{x}'$ chosen uniformly from $\mathbb{Z}_q^n$ and $\mathbf{e}'$ chosen from $\mathcal{U}([-r,r])^m$. Also let $\mathbf{x}$ be chosen uniformly from $\mathbb{Z}_q^n$ and $\mathbf{e}$ be chosen from $\mathcal{U}([-r,r])^m$. Then it holds that $\Pr_{(\mathbf{A},\mathbf{y})}[H_\infty(\mathbf{x} | \mathbf{A}\mathbf{x} + \mathbf{e} = \mathbf{y}) \geq 1] \geq 1 - \mathsf{negl}(n)$.

Proof. Assume first that $\mathbf{G}$ was chosen from $\chi^{m \times k}$, where χ is a symmetrical strictly B-bounded distribution. Fix an $\mathbf{e}'$ with $\Pr_{\mathbf{g}}[\|\mathbf{e}' - \mathbf{g}\|_\infty \leq r] \geq n^{-c}$, where $\mathbf{g}$ is distributed according to χ^m. Let $\mathbf{g}_1, \ldots, \mathbf{g}_k$ be the columns of $\mathbf{G}$, thus each $\mathbf{g}_i$ is distributed according to χ^m. We first show that $\mathbf{e}'$ is 2-ambiguous for the matrix $\mathbf{G}$ and distance r with high probability over the choice of $\mathbf{G}$. If there is at least one column $\mathbf{g}_i$ of $\mathbf{G}$ such that $\|\mathbf{e}' - \mathbf{g}_i\|_\infty \leq r$, then $\|\mathbf{e}' - \mathbf{G}\mathbf{x}^{(i)}\|_\infty \leq r$ (where $\mathbf{x}^{(i)}$ is the i-th unit vector) and we have that $\mathbf{e}'$ is 2-ambiguous. Here $\mathbf{x}_1 = 0$ and $\mathbf{x}_2 = \mathbf{x}^{(i)}$ are two different points satisfying $\|\mathbf{e}' - \mathbf{G} \cdot \mathbf{x}\|_\infty \leq r$.

The probability of the event that it holds for all $i = 1, \ldots, k$ that $\|\mathbf{e}' - \mathbf{g}_i\|_\infty > r$ is at most $(1 - n^{-c})^k \leq e^{-k \cdot n^{-c}} \leq e^{-\beta \cdot n^{1-c}}$. Thus we have that $\Pr[\mathbf{e}' \text{ 2-ambiguous for } \mathbf{G}] \geq 1 - e^{-\beta n^{1-c}}$. The same holds for the matrix

$\mathbf{A} = (\mathbf{A}' \| \mathbf{A}'\mathbf{T}' + \mathbf{G})$, as we can obtain $\mathbf{A}$ from $\mathbf{G}$ by appending extra columns and applying a basis-change. Both operations straightforwardly do not decrease the ambiguity. Therefore it holds $\Pr[\mathbf{e}'$ 2-ambiguous for $\mathbf{A}] \geq 1 - e^{-\beta n^{1-c}}$

Now let $\mathbf{e}'$ be distributed by $\mathcal{U}([-r, r])^m$. It holds that $r \geq m \cdot B \cdot n^c > (m+1)B$ and $\epsilon := n^{-c} < 1/2$ for sufficiently large n. Thus the above and Lemma 5 imply that $\Pr_{\mathbf{e}'}[\Pr_{\mathbf{A}}[\mathbf{e}'$ 2-ambiguous for $\mathbf{A}] \geq 1 - e^{-\beta n^{1-c}}] \geq 1 - m \cdot n^{-c \cdot \log(r/(m \cdot B))} = 1 - m \cdot n^{-c^2 \log(n)}$. This immediately yields $\Pr_{\mathbf{A},\mathbf{e}'}[\mathbf{e}'$ 2-ambiguous for $\mathbf{A}] \geq 1 - e^{-\beta n^{1-c}} - m \cdot n^{-c^2 \log(n)} = 1 - \mathsf{negl}(n)$. By linearity, this holds also if we shift $\mathbf{e}'$ by $\mathbf{A}\mathbf{x}'$ for any $\mathbf{x}' \in \mathbb{Z}_q^n$. Thus we get $\Pr_{\mathbf{A},\mathbf{y}}[\mathbf{y}$ 2-ambiguous for $\mathbf{A}] \geq 1 - \mathsf{negl}(n)$. Now, since $\bar{\Psi}_\alpha$ is statistically close to symmetrical strictly B-bounded distribution χ (which can be sampled by rejecting samples of $\bar{\Psi}_\alpha$ greater than B), this probability drops at most by a negligible amount if we sample $\mathbf{G}$ from $\bar{\Psi}_\alpha^{m \times k}$. Applying Lemma 4 yields $\Pr_{(\mathbf{A},\mathbf{y})}[H_\infty(\mathbf{x} | \mathbf{A}\mathbf{x} + \mathbf{e} = y) \geq 1] \geq 1 - \mathsf{negl}(n)$, which concludes the proof.

We will summarize the statements of Lemmas 2, 3 and 6 in the following theorem.

Theorem 5. *Let n be a security-parameter and let $\sigma \in (0,1)$, $\beta \in (0,1)$ and $\tau \geq 1$ be constants. Let $q = q(n)$, $m = m(n) = \mathsf{poly}(n)$, $k = \lceil \beta n \rceil$, $\alpha = \alpha(n) \in (0,1)$ and $r = r(n)$ be such that the following holds*

- $m \geq (1 + 2/\tau)n$
- $r \geq 2mn^{0.5+\sigma}$
- $q > (4r+1)^\tau$
- $2\sqrt{n} \leq \alpha q \leq \frac{r}{mn^\sigma}$

Then the distribution $\mathcal{C}_{n,m,q,k,\alpha}$ given in Construction 1 is 1-lossy for the error-distribution $\mathcal{U}[-r, r]$, provided that $\mathsf{LWE}(n, m, q, \bar{\Psi}_\alpha)$ is hard.

Proof. It is straightforward that this parameter-set satisfies the requirements of Lemmas 2 and 3, thus both lemmas holds. A gaussian tail-bound yields that $\bar{\Psi}_\alpha$ is $B = \alpha q n^{\sigma/2}$-bounded, except with probability $e^{-\pi n^\sigma/4}$, which is negligible. Item 4 above implies that $r \geq mn^\sigma \alpha q = mBn^{\sigma/2}$, thus setting $c = \sigma/2$ we can apply Lemma 6 and the claim follows.

4.1 LWE with Uniform Errors

Using Theorems 4 and 5 we can translate the worst-case connection for standard-LWE (Theorem 2) to LWE with uniform errors. For simplicity, we will set $\tau = 1$, $\beta = 1/2$ and $r = \rho \cdot q$, for a parameter $\rho = \rho(n) \in (0, 1/10)$.

Theorem 6 (Main Theorem). *Let n be a security parameter and let $\sigma \in (0,1)$ be an arbitrarily small constant. Let $q = q(n)$ be a modulus and $m = m(n) = \mathsf{poly}(n)$ be a integer with $m \geq 3n$. Let $\rho = \rho(n) \in (0, 1/10)$ be such that $\rho q \geq 2n^{0.5+\sigma}m$. If there exists a PPT-algorithm that solves $\mathsf{LWE}(n, m, q, \mathcal{U}([-\rho q, \rho q]))$ with non-negligible probability, then there exists an efficient quantum-algorithm* *that approximates the decision-version of the shortest vector problem (*GAPSVP*) and the shortest independent vectors problem (*SIVP*) to within* $\tilde{O}(n^{1+\sigma}m/\rho)$ *in the worst case.*

Proof. Set $\alpha = \alpha(n) = \frac{\rho}{mn^\sigma}$. Then the requirements of Theorem 5 are fulfilled:

- For $\tau = 1$ it holds that $m \geq 3n$
- $r = \rho q \geq 2mn^{0.5+\sigma}$
- $q > 4r + 1$ is equivalent to $r < (q-1)/4$, which holds for $r = \rho q < q/10$ and $q \geq 2$.
- $\alpha q = \frac{\rho q}{mn^\sigma} \geq 2\sqrt{n}$ and $\alpha q = \frac{\rho q}{mn^\sigma} = \frac{r}{mn^\sigma}$.

Thus by Theorem 5 there exists a 1-lossy code $\mathcal{C}$ for the error-distribution $\mathcal{U}([-\rho q, \rho q])$, provided that $\mathsf{LWE}(n, m, q, \bar{\Psi}_\alpha)$ is hard. The uniform distribution $\mathcal{U}([-\rho q, \rho q])^m$ clearly has efficiently decidable support, and so the first statement of Theorem 4 yields that $\mathsf{LWE}(n, m, q, \mathcal{U}[-\rho q, \rho q])$ is at least as hard as $\mathsf{LWE}(n, m, q, \bar{\Psi}_\alpha)$. Thus, setting $\alpha = \frac{\rho}{mn^\sigma}$ the claim follows by Theorem 2.

Using the search-to-decision reduction of Theorem 3, we can establish the hardness of the decisional LWE problem with error-distribution $\mathcal{U}([-\rho q, \rho q])$. We therefore need to restrict q to be a polynomially small prime integer.

Corollary 1. *Let* n *be a security parameter and let* $\sigma \in (0, 1)$ *be an arbitrarily small constant. Let* $q = q(n)$ *be a modulus and* $m = m(n) = \mathsf{poly}(n)$ *be a integer with* $m \geq 3n$*. Let* $\rho = \rho(n) \in (0, 1/10)$ *be such that* $\rho q \geq 2n^{0.5+\sigma}m$*. If there exists a PPT-distinguisher that distinguishes* $\mathsf{DLWE}(n, m, q, \mathcal{U}([-\rho q, \rho q]))$ *with non-negligible advantage, then there exists an efficient quantum-algorithm that approximates the decision-version of the shortest vector problem (*GAPSVP*) and the shortest independent vectors problem (*SIVP*) to within* $\tilde{O}(n^{1+\sigma}m/\rho)$ *in the worst case.*

5 Conclusion

This work presented the first worst-to-average-case reduction for an LWE variant with polynomial modulus and uniformly distributed errors, thereby answering a question from Micciancio and Mol from Crypto 2011. The factor of this worst-to-average-case connection depends on the number of samples given to the adversary and we have to use a bounded LWE assumption where this number is fixed in advance. Overcoming this limitation poses an interesting open problem. The main ingredient in our proof is a new tool called lossy codes, i.e., codes which lose information when decoding noisy code words. Another interesting question is, if these techniques carry over to hardness assumptions for binary codes.

Acknowledgement. The authors would like to thank Daniele Micciancio, Chris Peikert and the anonymous reviewers of Eurocrypt 2013 for valuable comments on this work. The authors are especially grateful to one particular reviewer who suggested the first statement of Theorem 4, thereby allowing a major simplification in the proof of Lemma 6. Nico Döttling was supported by IBM Research and Development Germany within the Homer-Project.

References

[ABB10] Agrawal, S., Boneh, D., Boyen, X.: Lattice Basis Delegation in Fixed Dimension and Shorter-Ciphertext Hierarchical IBE. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 98–115. Springer, Heidelberg (2010)

[ACPS09] Applebaum, B., Cash, D., Peikert, C., Sahai, A.: Fast Cryptographic Primitives and Circular-Secure Encryption Based on Hard Learning Problems. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 595–618. Springer, Heidelberg (2009)

[AIK11] Applebaum, B., Ishai, Y., Kushilevitz, E.: How to garble arithmetic circuits. In: FOCS, pp. 120–129 (2011)

[BGV12] Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. In: ITCS, pp. 309–325 (2012)

[BKPW12] Bellare, M., Kiltz, E., Peikert, C., Waters, B.: Identity-based (Lossy) trapdoor functions and applications. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 228–245. Springer, Heidelberg (2012)

[BPR12] Banerjee, A., Peikert, C., Rosen, A.: Pseudorandom Functions and Lattices. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 719–737. Springer, Heidelberg (2012)

[Bra12] Brakerski, Z.: Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP. In: Safavi-Naini, R. (ed.) CRYPTO 2012. LNCS, vol. 7417, pp. 868–886. Springer, Heidelberg (2012)

[BV11] Brakerski, Z., Vaikuntanathan, V.: Efficient Fully Homomorphic Encryption from (Standard) LWE. In: FOCS, pp. 97–106 (2011)

[CHKP10] Cash, D., Hofheinz, D., Kiltz, E., Peikert, C.: Bonsai Trees, or How to Delegate a Lattice Basis. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 523–552. Springer, Heidelberg (2010)

[DORS08] Dodis, Y., Ostrovsky, R., Reyzin, L., Smith, A.: Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data. SIAM J. Comput. 38(1), 97–139 (2008)

[GKPV10] Goldwasser, S., Kalai, Y.T., Peikert, C., Vaikuntanathan, V.: Robustness of the Learning with Errors Assumption. In: ICS, pp. 230–240 (2010)

[GPV08] Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: STOC, pp. 197–206 (2008)

[LPR10] Lyubashevsky, V., Peikert, C., Regev, O.: On Ideal Lattices and Learning with Errors over Rings. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 1–23. Springer, Heidelberg (2010)

[MM11a] Micciancio, D., Mol, P.: Pseudorandom Knapsacks and the Sample Complexity of LWE Search-to-Decision Reductions. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 465–484. Springer, Heidelberg (2011)

[MM11b] Micciancio, D., Mol, P.: Pseudorandom Knapsacks and the Sample Complexity of LWE Search-to-Decision Reductions. IACR Cryptology ePrint Archive, 2011:521 (2011)

[MP12] Micciancio, D., Peikert, C.: Trapdoors for Lattices: Simpler, Tighter, Faster, Smaller. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 700–718. Springer, Heidelberg (2012)

[MP13] Micciancio, D., Peikert, C.: Hardness of SIS and LWE with Small Parameters. IACR Cryptology ePrint Archive, 2013:069 (2013)

[Pei09] Peikert, C.: Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In: STOC, pp. 333–342 (2009)

[Pie12] Pietrzak, K.: Subspace LWE. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 548–563. Springer, Heidelberg (2012)

[PW08] Peikert, C., Waters, B.: Lossy trapdoor functions and their applications. In: STOC, pp. 187–196 (2008)

[Reg05] Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: STOC, pp. 84–93 (2005)

A Toolkit for Ring-LWE Cryptography

Vadim Lyubashevsky[1,⋆], Chris Peikert[2,⋆⋆], and Oded Regev[3,⋆⋆⋆]

[1] INRIA and École Normale Supérieure, Paris
[2] Georgia Institute of Technology
[3] Courant Institute, New York University

Abstract. Recent advances in lattice cryptography, mainly stemming from the development of ring-based primitives such as ring-LWE, have made it possible to design cryptographic schemes whose efficiency is competitive with that of more traditional number-theoretic ones, along with entirely new applications like fully homomorphic encryption. Unfortunately, realizing the full potential of ring-based cryptography has so far been hindered by a lack of practical algorithms and analytical tools for working in this context. As a result, most previous works have focused on very special classes of rings such as power-of-two cyclotomics, which significantly restricts the possible applications.

We bridge this gap by introducing a toolkit of fast, modular algorithms and analytical techniques that can be used in a wide variety of ring-based cryptographic applications, particularly those built around ring-LWE. Our techniques yield applications that work in *arbitrary* cyclotomic rings, with *no loss* in their underlying worst-case hardness guarantees, and very little loss in computational efficiency, relative to power-of-two cyclotomics. To demonstrate the toolkit's applicability, we develop two illustrative applications: a public-key cryptosystem and a "somewhat homomorphic" symmetric encryption scheme. Both apply to arbitrary cyclotomics, have tight parameters, and very efficient implementations.

1 Introduction

The past few years have seen many exciting developments in lattice-based cryptography. Two such trends are the development of schemes whose efficiency is competitive with traditional number-theoretic ones (e.g., [27] and follow-ups),

⋆ Part of this work was performed while at Tel Aviv University and also while visiting Georgia Tech. Partially supported by a European Research Council Starting Grant.

⋆⋆ This material is based upon work supported by the National Science Foundation under CAREER Award CCF-1054495, by DARPA under agreement number FA8750-11-C-0096, and by the Alfred P. Sloan Foundation. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation, DARPA or the U.S. Government, or the Sloan Foundation.

⋆⋆⋆ Supported by a European Research Council Starting Grant. Part of the work done while the author was with the CNRS, DI, ENS, Paris.

and the breakthrough work of Gentry [14, 13] (followed by others) on fully homomorphic encryption. While these two research threads currently occupy opposite ends of the efficiency spectrum, they are united by their use of algebraically structured *ideal lattices* arising from polynomial rings. The most efficient and advanced systems in both categories rely on the ring-LWE problem [26], an analogue of the standard *learning with errors* problem [31]. Informally (and a bit inaccurately), in a ring $R = \mathbb{Z}[X]/(f(X))$ for monic irreducible $f(X)$ of degree n, and for an integer modulus q defining the quotient ring $R_q := R/qR = \mathbb{Z}_q[X]/(f(X))$, the ring-LWE problem is to distinguish pairs $(a_i, b_i = a_i \cdot s + e_i) \in R_q \times R_q$ from uniformly random pairs, where $s \in R_q$ is a random secret (which stays fixed over all pairs), the $a_i \in R_q$ are uniformly random and independent, and the error (or "noise") terms $e_i \in R$ are independent and "short."

In all applications of ring-LWE, and particularly those related to homomorphic encryption, a main technical challenge is to control the sizes of the noise terms when manipulating ring-LWE samples under addition, multiplication, and other operations. For correct decryption, q must be chosen large enough so that the final accumulated error terms do not "wrap around" modulo q and cause decryption error. On the other hand, the *error rate* (roughly, the ratio of the noise magnitude to the modulus q) of the original published ring-LWE samples and the dimension n trade off to determine the theoretical and concrete hardness of the ring-LWE problem. Tighter control of the noise growth therefore allows for a larger initial error rate, which permits a smaller modulus q and dimension n, which leads to smaller keys and ciphertexts, and faster operations for a given level of security.

Regarding the choice of ring, the class of *cyclotomic* rings $R \cong \mathbb{Z}[X]/\Phi_m(X)$, where $\Phi_m(X)$ is the mth cyclotomic polynomial (which has degree $n = \varphi(m)$ and is monic and irreducible over the rationals), has many attractive features that have proved very useful in cryptography. For example, the search/decision equivalence for ring-LWE in arbitrary cyclotomics [26] relies on their special algebraic properties, as do many recent works that aim for more efficient fully homomorphic encryption schemes (e.g., [32, 8, 17, 18, 16]). In particular, *power-of-two* cyclotomics, i.e., where the index $m = 2^k$ for some $k \geq 1$, are especially nice to work with, because (among other reasons) $n = m/2$ is also a power of two, $\Phi_m(X) = X^n + 1$ is maximally sparse, and polynomial arithmetic modulo $\Phi_m(X)$ can be performed very efficiently using just a slight tweak of the classical n-dimensional FFT (see, e.g., [25]). Indeed, power-of-two cyclotomics have become the dominant and preferred class of rings in almost all recent ring-based cryptographic schemes (e.g., [25, 24, 21, 14, 15, 26, 33, 9, 8, 17, 18, 22, 5, 28, 20, 16]), often to the exclusion of all other rings.

While power-of-two cyclotomic rings are very convenient to use, there are several reasons why it is essential to consider other cyclotomics as well. The most obvious, practical reason is that powers of two are sparsely distributed, and the desired concrete security level for an application may call for a ring dimension much smaller than the next-largest power of two. So restricting to powers of two could lead to key sizes and runtimes that are at least twice as large as necessary. A more fundamental reason is that certain applications, such as the above-mentioned works that aim for more efficient (fully) homomorphic encryption, *require* the use of non-power-of-two cyclotomic rings. This is because power-of-two cyclotomics lack the requisite algebraic properties needed to implement features like SIMD operations on "packed" ciphertexts, or plaintext spaces isomorphic to finite fields of characteristic two (other than $\mathbb{F}_2$ itself). A final important reason is diversification of security assumptions. While some results are known [16] that relate ring-LWE in cyclotomic rings when one index m divides the other, no other connections appear to be known. So while we might conjecture that ring-LWE and ideal lattice problems are hard in *every* cyclotomic ring (of sufficiently high dimension), some rings might turn out to be significantly easier than others.

Unfortunately, working in non-power-of-two cyclotomics is rather delicate, and the current state of affairs is unsatisfactory in several ways. Unlike the special case where m is a power of two, in general the cyclotomic polynomial $\Phi_m(X)$ can be quite "irregular" and dense, with large coefficients. While in principle, polynomial arithmetic modulo $\Phi_m(X)$ can still be done in $O(n \log n)$ scalar operations (on high-precision complex numbers), the generic algorithms for achieving this are rather complex and hard to implement, with large constants hidden by the $O(\cdot)$ notation.

Geometrically, the non-power-of-two case is even more problematic. If one views $\mathbb{Z}[X]/(\Phi_m(X))$ as the set of polynomial residues of the form $a_0 + a_1 X + \cdots + a_{n-1}X^{n-1}$, and uses the naïve "coefficient embedding" that views them as vectors $(a_0, a_1, \ldots, a_{n-1}) \in \mathbb{Z}^n$ to define geometric quantities like the ℓ_2 norm, then both the concrete and theoretical security of cryptographic schemes depend heavily on the form of $\Phi_m(X)$. This stems directly from the fact that multiplying two polynomials with small norms can result in a polynomial residue having a *much* larger norm. The growth can be quantified by the "expansion factor" [23] of $\Phi_m(X)$, which unfortunately can be very large, up to $n^{\Omega(\log n)}$ in the case of highly composite m [12]. Later works [17] circumvented such large expansion by using tricks like lifting to the larger-dimensional ring $\mathbb{Z}[X]/(X^m - 1)$, but this still involves a significant loss in the tolerable noise rates as compared with the power-of-two case.

In [30, 26] a different geometric approach was used, which avoided any dependence on the form of the polynomial modulus $\Phi_m(X)$. In these works, the norm of a ring element is instead defined according to its *canonical embedding* into $\mathbb{C}^n$, a classical concept from algebraic number theory. This gives a much better way of analyzing expansion, since both addition and multiplication in the canonical embedding are simply coordinate-wise. Working with the canonical embedding, however, introduces a variety of practical issues, such as how to efficiently generate short noise terms having appropriate distributions over the ring. More generally, the focus of [26] was on giving an abstract mathematical definition of ring-LWE and proving its hardness under worst-case ideal lattice assumptions; in particular, it did not deal with issues related to practical efficiency, bounding noise growth, or designing applications in non-power-of-two cyclotomics.

1.1 Contributions

Our main contribution is a toolkit of modular algorithms and analytical techniques that can be used in a wide variety of ring-based cryptographic applications, particularly those built around ring-LWE. The high-level summary is that using our techniques, one can design applications to work in *arbitrary* cyclotomic rings, with *no loss* in their underlying worst-case hardness guarantees, and very little loss in computational efficiency, relative to the best known techniques in power-of-two cyclotomics. In fact, our analytical techniques even improve the state of the art for the power-of-two case.

In more detail, our toolkit includes fast, specialized algorithms for all the main cryptographic operations in arbitrary cyclotomic rings. Among others, these include: addition, multiplication, and conversions among various useful representations of rings elements; generation of noise terms under probability distributions that guarantee both worst-case and concrete hardness; and decoding of noise terms as needed in decryption and related operations. Our algorithms' efficiency and quality guarantees stem primarily from our use of simple but non-obvious representations of ring elements, which differ from their naïve representations as polynomial residues modulo $\Phi_m(X)$. (See the second part of Section 1.2 for more details.) On the analytical side, we give tools for tightly bounding noise growth under operations like addition, multiplication, and round-off/discretization. (Recall that noise growth is the main factor determining an application's parameters and noise rates, and hence its key sizes, efficiency, and concrete security.) Some attractive features of the toolkit include:

- All the algorithms for arbitrary cyclotomics are simple, modular, and highly parallel, and work by elementary reductions to the (very simple) prime-index case. In particular, they do not require any polynomial reductions modulo $\Phi_m(X)$ – in fact, they never need to compute $\Phi_m(X)$ at all! The algorithms work entirely on vectors of dimension $n = \varphi(m)$, and run in $O(n \log n)$ or even $O(nd)$ scalar operations (with small hidden constants), where d is the number of distinct primes dividing m. With the exception of continuous noise generation, all scalar operations are low precision, i.e., they involve small integers. In summary, the algorithms are very amenable to practical implementation. (Indeed, we have implemented all the algorithms from scratch, which will be described in a separate work.)
- Our algorithm for decoding noise, used primarily in decryption, is fast (requiring $O(n \log n)$ or fewer small-integer operations) and correctly recovers from optimally large noise rates. (See the last part of Section 1.2 for details.) This improves upon prior techniques, which in general have worse noise tolerance by anywhere between an $m/2$ and super-polynomial $n^{\omega(1)}$ factor, and are computationally slower and more complex due to polynomial reduction modulo $\Phi_m(X)$, among other operations.
- Our bounds on noise growth under ring addition and multiplication are exactly the same in *all* cyclotomic rings; no ring-dependent "expansion factor" is incurred. (For discretizing continuous noise distributions, our bounds are

the same up to very small $1 + o(1)$ factors, depending on the primes dividing m.) This allows applications to use essentially the same underlying noise rate as a function of the ring dimension n, and hence be based on the same worst-case approximation factors, for all cyclotomics. Moreover, our bounds improve upon the state of the art even for power-of-two cyclotomics: e.g., our (average-case, high probability) expansion bound for ring multiplication improves upon the (worst-case) expansion-factor bound by almost a $\sqrt{n}$ factor.

To illustrate the toolkit's applicability, in Section 5 we construct an efficient and compact public-key cryptosystem, which is essentially the "two element" system outlined in [26], but generalized to arbitrary cyclotomics, and with tight parameters. Further applications are given in the full version of the paper.

A final contribution of independent interest is a new "regularity lemma" for arbitrary cyclotomics, i.e., a bound on the smoothing parameter of random q-ary lattices over the ring. Such a lemma is needed for porting many applications of standard SIS and LWE to the ring setting, including SIS-based signature schemes [19, 10, 7, 28], the "primal" [31] and "dual" [19] LWE cryptosystems, chosen ciphertext-secure encryption schemes [29, 28], and (hierarchical) identity-based encryption schemes [19, 10, 1]. In terms of generality and parameters, our lemma essentially subsumes a prior one of Micciancio [27] for the ring $\mathbb{Z}[X]/(X^n - 1)$, and an independent one of Stehlé et al. [34] for power-of-two cyclotomics. (See Section 4 for further discussion.)

1.2 Techniques

The tools we develop in this work involve several novel applications of classical notions from algebraic number theory. In summary, our results make central use of: (1) the *canonical embedding* of a number field, which endows the field (and its subrings) with a nice and easy-to-analyze geometry; (2) the decomposition of arbitrary cyclotomics into the *tensor product* of prime-power cyclotomics, which yields both simpler and faster algorithms for computing in the field, as well as geometrically nicer bases; and (3) the "*dual*" ideal $R^\vee$ and its "*decoding*" basis $\vec{d}$, for fast noise generation and optimal noise tolerance in decryption and related operations. We elaborate on each of these next.

The Canonical Embedding. As in the previous works [30, 26], our analysis relies heavily on using the *canonical embedding* $\sigma \colon K \to \mathbb{C}^n$ (rather than, say, the naïve coefficient embedding) for defining all geometric quantities, such as Euclidean norms and inner products. For example, under the canonical embedding, the "expansion" incurred when multiplying by an element $a \in K$ is characterized exactly by $\|\sigma(a)\|_\infty$, its ℓ_∞ norm under the canonical embedding; no (worst-case) ring-dependent "expansion factor" is needed. So in the average-case setting, where the multiplicands are random elements from natural noise distributions, for each multiplication we get at least a $\tilde{\Omega}(\sqrt{n})$ factor improvement over using the expansion factor in *all* cyclotomics (including those with power-of-two index), and up to a super-polynomial $n^{\omega(1)}$ factor improvement in cyclotomics having highly composite indices. In our analysis of the noise tolerance of decryption, we also get an additional $\tilde{\Omega}(\sqrt{n})$ factor savings over more simplistic analyses that only use norm information, by using the notion of *subgaussian* random variables. These behave under linear transformations in essentially the same way as Gaussians, and have Gaussian tails. (This builds upon prior works that use subgaussianity in lattice cryptography, e.g., [2, 28].)

Tensorial Decomposition. An important fact at the heart of this work is that the mth cyclotomic number field $K = \mathbb{Q}(\zeta_m) \cong \mathbb{Q}[X]/(\Phi_m(X))$ may instead be viewed as (i.e., is isomorphic to) the *tensor product* of prime-power cyclotomics:

$$K \cong \bigotimes_\ell K_\ell = \mathbb{Q}(\zeta_{m_1}, \zeta_{m_2}, \ldots),$$

where $m = \prod_\ell m_\ell$ is the prime-power factorization of m and $K_\ell = \mathbb{Q}(\zeta_{m_\ell})$. Equivalently, in terms of polynomials we may view K as the multivariate field

$$K \cong \mathbb{Q}[X_1, X_2, \ldots]/(\Phi_{m_1}(X_1), \Phi_{m_2}(X_2), \ldots), \tag{1}$$

where there is one indeterminant X_ℓ and modulus $\Phi_{m_\ell}(X_\ell)$ per prime-power divisor of m. Similar decompositions hold for the ring of integers $R \cong \mathbb{Z}[X]/\Phi_m(X)$ and other important objects in K, such as the dual ideal $R^\vee$ (described below).

Adopting the polynomial interpretation of K from Equation (1) for concreteness, notice that a natural $\mathbb{Q}$-basis is the set of multinomials $\prod_\ell X_\ell^{j_\ell}$ for each choice of $0 \leq j_\ell < \varphi(m_\ell)$. We call this set the "powerful" basis of K (and of R). For non-prime-power m, under the field isomorphism with $\mathbb{Q}[X]/(\Phi_m(X))$ that maps each $X_\ell \to X^{m/m_\ell}$, the powerful basis does *not* coincide with the standard "power" basis $1, X, X^2, \ldots, X^{\varphi(m)-1}$ usually used to represent the univariate field. It turns out that in general, the powerful basis has much nicer computational and geometric properties than the power basis, as we outline next.

Computationally, the tensorial decomposition of K (with the powerful basis) allows us to modularly reduce essentially all operations in K (or R, or powers of $R^\vee$) to their counterparts in much simpler prime-power cyclotomics (which themselves easily reduce to the prime-index case). We can therefore completely avoid all the many algorithmic complications associated with working with polynomials modulo $\Phi_m(X)$. In particular, we obtain novel, simple and fast algorithms, similar to the FFT, for converting between the multivariate "polynomial" representation (i.e., the powerful basis) and the "evaluation" or "Chinese remainder" representation, in which addition and multiplication are essentially linear time. Similarly, we obtain linear-time (or nearly so) algorithms for switching between the polynomial representation and "decoding" representation used in decryption (described below), and for generating noise terms in the decoding representation. A final advantage of the tensorial representation is that it yields trivial linear-time algorithms for computing the *trace* function to subfields of K, which is at the heart of the "ring-switching" technique from [16].

The tensorial representation also comes with important geometrical advantages. In particular, under the canonical embedding the powerful basis is better-conditioned than the power basis, i.e., the ratio of its maximal and minimal singular values can be much smaller. This turns out to be important when bounding the additional error introduced when discretizing (rounding off) field elements in noise-generation and modulus-reduction algorithms, among others.

The Dual Ideal $R^\vee$ and Its Decoding Basis. Under the canonical embedding, the cyclotomic ring R of index m embeds as a lattice which, unlike $\mathbb{Z}^n$, is in general not self-dual. Instead, its dual lattice corresponds to a fractional ideal $R^\vee \subset K$ satisfying $R \subseteq R^\vee \subseteq m^{-1}R$, where the latter inclusion is nearly an equality. (In fact, $R^\vee$ is a scaling of R exactly when m is a power of two, in which case $R = (m/2)R^\vee$.) In [26] it is shown that the "right" definition of the ring-LWE distribution, which arises naturally from the worst-case to average-case reduction, involves the dual ideal $R^\vee$: the secret belongs to the quotient $R_q^\vee = R^\vee/qR^\vee$, and ring-LWE samples are of the form $(a, b = a \cdot s + e \bmod qR^\vee)$ for uniformly random $a \in R_q$ and error e which is essentially spherical in the canonical embedding.

While it is possible [11] to simplify the ring-LWE definition by replacing every instance of $R^\vee$ with R, while retaining essentially spherical error (but scaled up by about m, corresponding to the approximate ratio of R to $R^\vee$), in this work we show that *it is actually advantageous to retain $R^\vee$ and expose it in applications.*[1] The reason is that in general, $R^\vee$ supports correct bounded-distance decoding—which is the main operation performed in decryption—under a larger error rate than R does.[2] In fact, $R^\vee$'s error tolerance is *optimal* for the simple, fast lattice decoding algorithm used implicitly in essentially all decryption procedures, namely Babai's "round-off" algorithm [4]. The reason is that when decoding a lattice Λ using some basis $\{\mathbf{b}_i\}$, the error tolerance depends inversely on the Euclidean lengths of the vectors dual to $\{\mathbf{b}_i\}$. For $R^\vee$, there is a particular "*decoding*" basis whose dual basis is optimally short (relative to the determinant of R), whereas for R no such basis exists in general.[3] In fact, the decoding basis of $R^\vee$ is simply the dual of the powerful basis described above!

In addition to its optimal error tolerance, we also show that the decoding basis has good computational properties. In particular, there are linear-time (or nearly so) algorithms for converting to the decoding basis from the other bases of $R^\vee$ or $R_q^\vee$ that are more appropriate for other computational tasks. And Gaussian errors (especially spherical ones) can be sampled in (near-)linear time in the decoding basis.

[1] This is unless m is a power of two, in which case nothing is lost by simply scaling up by exactly $m/2$ to replace $R^\vee$ with R.

[2] By "error rate" here we mean the ratio of the error (in, say, ℓ_2 norm) to the dimension-normalized determinant $\det(\Lambda)^{1/n}$ of the lattice Λ, so exact scaling has no effect on the error rate.

[3] We note that decoding by "lifting" R to the larger-dimensional ring $\mathbb{Z}[X]/(X^m - 1)$, as done in [17], still leads to an m or $m/2$ factor loss in error tolerance overall, because some inherent loss is already incurred when replacing $R^\vee$ with R, and a bit more is lost in the lifting procedure.

2 Preliminaries

For a positive integer k, we denote by $[k]$ the set $\{0, \ldots, k-1\}$. For a real $a \in \mathbb{R}$, define $\lfloor a \rceil = \lfloor a + \frac{1}{2} \rfloor \in \mathbb{Z}$. For any $\bar{a} \in \mathbb{R}/\mathbb{Z}$, we let $[\![\bar{a}]\!] \in \mathbb{R}$ denote the unique representative $a \in (\bar{a} + \mathbb{Z}) \cap [-1/2, 1/2)$. Similarly, for $\bar{a} \in \mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$ we let $[\![\bar{a}]\!]$ denote the unique representative $a \in (\bar{a} + q\mathbb{Z}) \cap [-q/2, q/2)$. We extend $\lfloor \cdot \rceil$ and $[\![\cdot]\!]$ entrywise to vectors and matrices. The *radical* of a positive integer m, denoted $\mathrm{rad}(m)$, is the product of all primes dividing m. We also define $\hat{m} = m/2$ whenever m is even, and $\hat{m} = m$ otherwise. For a vector $\mathbf{x}$ over $\mathbb{R}$ or $\mathbb{C}$, we define the ℓ_2 norm as $\|\mathbf{x}\|_2 = (\sum_i |x_i|^2)^{1/2}$, and the ℓ_∞ norm as $\|\mathbf{x}\|_\infty = \max_i |x_i|$. When the subscript is omitted, we mean the ℓ_2 norm.

Throughout this paper, the entries of a vector over a domain D are always indexed (in no particular order) by some finite set S, and we write D^S to denote the set of all such vectors. Similarly, the rows and columns of an "R-by-C matrix" over D are indexed by some finite sets R and C, respectively. All the standard matrix and vector operations, including the Kronecker (or tensor) product, are defined in the natural way.

2.1 The Space H

When working with cyclotomic number fields and ideal lattices, it is convenient to work with the subspace $H \subseteq \mathbb{C}^{\mathbb{Z}_m^*}$ for integer $m \geq 2$, defined as

$$H = \{\mathbf{x} \in \mathbb{C}^{\mathbb{Z}_m^*} \ : \ x_i = \overline{x_{m-i}}, \ \forall \, i \in \mathbb{Z}_m^*\}.$$

Letting $n = \varphi(m)$, it is not difficult to verify that H (with the inner product induced on it by $\mathbb{C}^{\mathbb{Z}_m^*}$) is isomorphic to $\mathbb{R}^{[n]}$ as an inner product space. For $m = 2$ this is trivial, and for $m > 2$ this can seen via the $\mathbb{Z}_m^*$-by-$[n]$ unitary basis matrix $\frac{1}{\sqrt{2}} \begin{pmatrix} I & \sqrt{-1}J \\ J & -\sqrt{-1}I \end{pmatrix}$ of H, where here the $\mathbb{Z}_m^*$-indexed rows are in increasing order according to their canonical representatives in $\{1, \ldots, m-1\}$, the $[n]$-indexed columns are in increasing order by index, I is the identity matrix, and J is the reversal matrix (obtained by reversing the rows of I).

We equip H with the ℓ_2 and ℓ_∞ norms induced on it from $\mathbb{C}^{\mathbb{Z}_m^*}$. Namely, for $\mathbf{x} \in H$ we have $\|\mathbf{x}\|_2 = \sum_i (|x_i|^2)^{1/2} = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle}$, and $\|\mathbf{x}\|_\infty = \max_i |x_i|$.

2.2 Gaussians and Subgaussian Random Variables

For $s > 0$, define the Gaussian function $\rho_s \colon H \to (0, 1]$ as $\rho_s(\mathbf{x}) = \exp(-\pi \|\mathbf{x}\|^2 / s^2)$. By normalizing this function we obtain the *continuous* Gaussian probability distribution D_s of parameter s, whose density is given by $s^{-n} \cdot \rho_s(\mathbf{x})$.

For much of our analysis it is convenient to use the standard notion of *subgaussian* random variables, relaxed slightly as in [28]. For any $\delta \geq 0$, we say that a random variable X (or its distribution) over $\mathbb{R}$ is *δ-subgaussian* with parameter $s > 0$ if for all $t \in \mathbb{R}$, the (scaled) moment-generating function satisfies

$$\mathbb{E}[\exp(2\pi t X)] \leq \exp(\delta) \cdot \exp(\pi s^2 t^2).$$

Notice that the $\exp(\pi s^2 t^2)$ term on the right is exactly the (scaled) moment-generating function of the one-dimensional Gaussian distribution of parameter s over $\mathbb{R}$.

Decoding. In many applications we need to perform the following algorithmic task, which is essentially that of bounded-distance decoding. Let Λ be a known fixed lattice, and let $\mathbf{x} \in H$ be an unknown short vector. The goal is to recover $\mathbf{x}$, given $\mathbf{x} \bmod \Lambda$. Although there are several possible algorithms for this task, here we focus on a slight extension of the so-called "round-off" algorithm, originally due to Babai [4]. This is due to its high efficiency and because for our lattices it performs optimally (or nearly so). The algorithm is very simple: let $\{\mathbf{v}_i\}$ be a fixed set of n short, linearly independent vectors in the dual lattice $\Lambda^\vee$. Denote the dual vectors of $\{\mathbf{v}_i\}$ by $\{\mathbf{b}_i\}$, and let $\Lambda' \supseteq \Lambda$ be the (super)lattice generated by $\{\mathbf{b}_i\}$. Given an input $\mathbf{t} = \mathbf{x} \bmod \Lambda$, we express $\mathbf{t} \bmod \Lambda'$ in the basis $\{\mathbf{b}_i\}$ as $\sum_i \bar{a}_i \mathbf{b}_i$ where $\bar{a}_i \in \mathbb{R}/\mathbb{Z}$ (so $\bar{a}_i = \langle \mathbf{x}, \mathbf{v}_i \rangle \bmod 1$), and output $\sum_i [\![\bar{a}_i]\!] \mathbf{b}_i \in H$.

Lemma 2.1. *Let $\Lambda \subset H$ be a lattice, let $\{\mathbf{v}_i\}$ be a set of n linearly independent vectors in its dual, and let $d_{\max} = \max_i \|\mathbf{v}_i\|$. For any $\mathbf{x}$ of length less than $1/(2d_{\max})$, the above round-off algorithm succeeds in recovering $\mathbf{x}$ from $\mathbf{x} \bmod \Lambda$. Moreover, for any $\delta > 0$, if $\mathbf{x}$ is a random vector such that $\langle \mathbf{x}, \mathbf{v}_i \rangle$ is δ-subgaussian with parameter s for every i (in particular, if $\mathbf{x}$ itself is δ-subgaussian with parameter $s/d_{\max}$), then the round-off algorithm succeeds with probability at least $1 - 2n\exp(\delta - \pi/(2s)^2)$, which is $1 - \mathrm{negl}(n)$ when $\delta = O(1)$ and $s = 1/\omega(\sqrt{\log n})$.*

Discretization. We now consider another algorithmic task related to the one in the previous subsection. This task shows up in applications, such as when converting a continuous Gaussian into a discrete Gaussian-like distribution. Given a lattice $\Lambda = \mathcal{L}(\mathbf{B})$ represented by a "good" basis $\mathbf{B} = \{\mathbf{b}_i\}$, a point $\mathbf{x} \in H$, and a point $\mathbf{c} \in H$ representing a lattice coset $\Lambda + \mathbf{c}$, the goal is to discretize $\mathbf{x}$ to a point $\mathbf{y} \in \Lambda + \mathbf{c}$, written $\mathbf{y} \leftarrow \lfloor \mathbf{x} \rceil_{\Lambda+\mathbf{c}}$, so that the length (or subgaussian parameter) of $\mathbf{y} - \mathbf{x}$ is not too large. To do this, we sample a relatively short offset vector $\mathbf{f}$ from the coset $\Lambda + \mathbf{c}' = \Lambda + (\mathbf{c} - \mathbf{x})$, and output $\mathbf{y} = \mathbf{x} + \mathbf{f}$. We require that the method used to choose $\mathbf{f}$ be efficient and depend only on the desired coset $\Lambda + \mathbf{c}'$, not on the particular representative used to specify it. In the full version of the paper, we describe several valid ways of sampling $\mathbf{f}$, offering tradeoffs between efficiency and output guarantees.

2.3 Algebraic Number Theory Background

Cyclotomic Number Fields and Polynomials. For a positive integer m, the mth *cyclotomic number field* is a field extension $K = \mathbb{Q}(\zeta_m)$ obtained by adjoining an element ζ_m of order m (i.e., a primitive mth root of unity) to the rationals. (Note that we view ζ_m as an abstract element, and not, for example, as any particular value in $\mathbb{C}$.) The minimal polynomial of ζ_m is the mth *cyclotomic polynomial*

$$\Phi_m(X) = \prod_{i \in \mathbb{Z}_m^*} (X - \omega_m^i) \in \mathbb{Z}[X], \tag{2}$$

where $\omega_m \in \mathbb{C}$ is any primitive mth root of unity in $\mathbb{C}$, e.g., $\omega_m = \exp(2\pi\sqrt{-1}/m)$. Therefore, there is a natural isomorphism between K and $\mathbb{Q}[X]/(\Phi_m(X))$, given by $\zeta_m \mapsto X$. Since $\Phi_m(X)$ has degree $n = |\mathbb{Z}_m^*| = \varphi(m)$, we can view K as a vector space of degree n over $\mathbb{Q}$, which has $\{1, \zeta_m, \ldots, \zeta_m^{n-1}\}$ as a basis. This is called the *power basis* of K.

For the mth cyclotomic number field $K = \mathbb{Q}(\zeta_m)$ of degree $n = \varphi(m)$, the *ring of integers* is $R = \mathbb{Z}[\zeta_m] \cong \mathbb{Z}[X]/\Phi_m(X)$, and hence has the power basis $\{\zeta_m^j\}_{j\in[n]}$ as a $\mathbb{Z}$-basis.

Non-Prime-Power Cyclotomics. Let m have prime-power factorization $m = \prod_\ell m_\ell$, i.e., the m_ℓ are powers of distinct primes. Then $K = \mathbb{Q}(\zeta_m)$ may be seen as the *tensor product* $\bigotimes_\ell K_\ell$ of the fields $K_\ell = \mathbb{Q}(\zeta_{m_\ell})$, in the following way. First, view each K_ℓ as a subfield of K, via the ring embedding $\zeta_{m_\ell} \mapsto \zeta_m^{m/m_\ell}$. Then viewing K and K_ℓ as vector spaces over $\mathbb{Q}$, the tensor product $\bigotimes_\ell K_\ell$ is isomorphic to K, under the map $(\otimes_\ell\, a_\ell) \mapsto \prod_\ell a_\ell$.[4] In particular, if B_ℓ are $\mathbb{Q}$-bases of K_ℓ respectively (e.g., the power bases), then their tensor product $\bigotimes_\ell B_\ell = \{\prod_\ell b_\ell \in K : b_\ell \in B_\ell\}$ is a $\mathbb{Q}$-basis of K. Moreover, endowing $\bigotimes_\ell K_\ell$ with the multiplication operation induced by the mixed-product property $(\otimes_\ell\, a_\ell)\cdot(\otimes_\ell\, b_\ell) = \otimes_\ell\,(a_\ell \cdot b_\ell)$ also makes the above mapping from $\bigotimes_\ell K_\ell$ to K a field isomorphism, as desired.

Equivalently, in terms of polynomial rings we may view $K \cong \mathbb{Q}[X]/(\Phi_m(X))$ instead as

$$K \cong \mathbb{Q}[X_1, X_2, \ldots]/(\Phi_{m_1}(X_1), \Phi_{m_2}(X_2), \ldots), \tag{3}$$

where there is one indeterminant X_ℓ and modulus $\Phi_{m_\ell}(X_\ell)$ per prime divisor of m, and where $X_\ell \mapsto X^{m/m_\ell}$ defines an isomorphism with $\mathbb{Q}[X]/(\Phi_m(X))$. Notice that by tensoring the power bases $\{X_\ell^j\}_{j\in[\varphi(m_\ell)]}$ of each K_ℓ, we get the basis $\{X_1^{j_1} X_2^{j_2} \cdots\}_{j_\ell \in[\varphi(m_\ell)]}$. Mapping this basis to $\mathbb{Q}[X]/(\Phi_m(X))$ yields the basis $\{X^{\sum_\ell (m/m_\ell) j_\ell}\}_{j_\ell\in[\varphi(m_\ell)]}$, which is *not* necessarily the power basis $\{X^j\}_{j\in[\varphi(m)]}$, since the powers of X appearing in each basis can be different modulo m. (For example, take $m = 3 \cdot 5$.)

Embeddings and Geometry. Here we describe the *embeddings* of a cyclotomic number field, which induce a 'canonical' geometry on it.

The mth cyclotomic number field $K = \mathbb{Q}(\zeta_m)$ of degree $n = \varphi(m)$ has exactly n ring homomorphisms (embeddings) $\sigma_i\colon K \to \mathbb{C}$ that fix every element of $\mathbb{Q}$. Concretely, for each $i \in \mathbb{Z}_m^*$ there is an embedding σ_i defined by $\sigma_i(\zeta_m) = \omega_m^i$, where $\omega_m \in \mathbb{C}$ is some fixed primitive mth root of unity. Clearly, the embeddings come in pairs of complex conjugates, i.e., $\sigma_i = \overline{\sigma_{m-i}}$. The *canonical embedding* $\sigma\colon K \to \mathbb{C}^{\mathbb{Z}_m^*}$ is defined as

$$\sigma(a) = (\sigma_i(a))_{i\in\mathbb{Z}_m^*}.$$

[4] The tensor product of two vector spaces K, L over a common base field can be defined as the set of all finite sums of pure tensors $a \otimes b$ for $a \in K$, $b \in L$, where $\otimes$ is bilinear. The tensor product of multiple vector spaces is defined similarly.

When K is viewed as the tensor product of subfields K_ℓ, $\sigma = \bigotimes_\ell \sigma^{(\ell)}$ is the tensor product of the canonical embeddings $\sigma^{(\ell)}$ of K_ℓ. In this case, the index set of σ is $\prod_\ell \mathbb{Z}^*_{m_\ell}$, which corresponds to $\mathbb{Z}^*_m$ via the Chinese remainder theorem.

The *trace* $\mathrm{Tr} = \mathrm{Tr}_{K/\mathbb{Q}} \colon K \to \mathbb{Q}$ can be defined as the sum of the embeddings: $\mathrm{Tr}(a) = \sum_i \sigma_i(a)$. Clearly, $\mathrm{Tr}(a + b) = \mathrm{Tr}(a) + \mathrm{Tr}(b)$ and $\mathrm{Tr}(c \cdot a) = c \cdot \mathrm{Tr}(a)$ for all $a, b \in K$ and $c \in \mathbb{Q}$. Moreover,

$$\mathrm{Tr}(a \cdot b) = \sum_i \sigma_i(a)\sigma_i(b) = \langle \sigma(a), \overline{\sigma(b)} \rangle.$$

Duality. For any fractional ideal $\mathcal{I}$ in K, its *dual* is defined as

$$\mathcal{I}^\vee = \{a \in K \ : \ \mathrm{Tr}(a\mathcal{I}) \subseteq \mathbb{Z}\}.$$

It is easy to verify that $\mathcal{I}^\vee$ is a fractional ideal, and that $(\mathcal{I}^\vee)^\vee = \mathcal{I}$.

For any $\mathbb{Q}$-basis $B = \{b_j\}$ of K, we denote its dual basis by $B^\vee = \{b_j^\vee\}$, which is characterized by $\mathrm{Tr}(b_i \cdot b_j^\vee) = 1$ if $i = j$, and 0 otherwise. It is immediate that $(B^\vee)^\vee = B$, and if B is a $\mathbb{Z}$-basis of some fractional ideal $\mathcal{I}$, then $B^\vee$ is a $\mathbb{Z}$-basis of its dual ideal $\mathcal{I}^\vee$. An important fact is that if $a = \sum_j a_j \cdot b_j$ (where $a_j \in \mathbb{R}$) is the unique representation of some $a \in K_\mathbb{R}$ in basis B, then $a_j = \mathrm{Tr}(a \cdot b_j^\vee)$ by linearity of Tr.

Except in the trivial number field $K = \mathbb{Q}$, the ring of integers R is not self-dual, nor are an ideal and its inverse dual to each other. However, an ideal and its inverse *are* related by multiplication with the dual ideal $R^\vee$ of the ring: for any fractional ideal $\mathcal{I}$, its dual is $\mathcal{I}^\vee = \mathcal{I}^{-1} \cdot R^\vee$. (Notice that for $\mathcal{I} = R$ this holds trivially, since $R^{-1} = R$.) A standard fact is that $R^\vee = \langle t^{-1} \rangle$ is a principal ideal generated by t^{-1} for some (non-unique) $t \in R$. When $R \cong \bigotimes_\ell R_\ell$ is viewed as the tensor product of rings of integers $R_\ell \subset K_\ell$ (where $K \cong \bigotimes_\ell K_\ell$), its dual ideal has an analogous tensorial form, as $R^\vee = \bigotimes_\ell R_\ell^\vee$.

2.4 Ring-LWE

We now provide the formal definition of the ring-LWE problem and recall the worst-case hardness result shown in [26]. We remark that our definition here differs very slightly from the one used in [26]: we scale the b component by a factor of q, so that it is an element of $K_\mathbb{R}/qR^\vee$ and not $K_\mathbb{R}/R^\vee$ as in [26]. This is done for convenience when later discretizing the b component, and the two definitions are easily seen to be equivalent.

Definition 2.2 (Ring-LWE Distribution). *For a "secret" $s \in R_q^\vee$ (or just $R^\vee$) and a distribution ψ over $K_\mathbb{R}$, a sample from the ring-*LWE *distribution $A_{s,\psi}$ over $R_q \times (K_\mathbb{R}/qR^\vee)$ is generated by choosing $a \leftarrow R_q$ uniformly at random, choosing $e \leftarrow \psi$, and outputting $(a, b = a \cdot s + e \bmod qR^\vee)$.*

Definition 2.3 (Ring-LWE, Average-Case Decision). *The* average-case decision *version of the ring-*LWE *problem, denoted R-$\mathsf{DLWE}_{q,\psi}$, is to distinguish with non-negligible advantage between independent samples from $A_{s,\psi}$, where $s \leftarrow R_q^\vee$ is uniformly random, and the same number of uniformly random and independent samples from $R_q \times (K_\mathbb{R}/qR^\vee)$.*

Theorem 2.4. *Let K be the mth cyclotomic number field having dimension $n = \varphi(m)$ and $R = \mathcal{O}_K$ be its ring of integers. Let $\alpha = \alpha(n) > 0$, and let $q = q(n) \geq 2$, $q = 1 \bmod m$ be a* $\mathrm{poly}(n)$*-bounded prime such that $\alpha q \geq \omega(\sqrt{\log n})$. Then there is a polynomial-time quantum reduction from $\tilde{O}(\sqrt{n}/\alpha)$-approximate* SIVP *(or* SVP*) on ideal lattices in K to the problem of solving R-$\mathsf{DLWE}_{q,\psi}$ given only ℓ samples, where ψ is the Gaussian distribution $D_{\xi q}$ for $\xi = \alpha \cdot (n\ell/\log(n\ell))^{1/4}$.*

In cryptographic applications it is often useful to work with a version of ring-LWE whose error distribution is discrete. In the full version of the paper, we show that for a wide family of discrete error distributions, it is easy to deduce the hardness of the discrete version from that of the continuous one. Another important variant of ring-LWE, known as the "normal form," is the one in which the secret, instead of being uniformly distributed, is chosen from the error distribution (discretized to $R^\vee$). Showing that this variant of ring-LWE is as hard as the original one follows from the techniques of [3].

3 The Powerful, CRT, and Decoding Bases

In this section we study certain $\mathbb{Z}$-bases of certain (fractional) ideals $\mathcal{I}$ in $K = \mathbb{Q}(\zeta_m)$, which are also $\mathbb{Z}_q$-bases of the quotients $\mathcal{I}_q = \mathcal{I}/q\mathcal{I}$ for any positive integer q. Fixing such a basis $\vec{b}$ and viewing it as a (column) vector over K, we can represent any $a \in \mathcal{I}$ uniquely as $a = \langle \vec{b}, \mathbf{a} \rangle = \vec{b}^T \cdot \mathbf{a}$ for some coefficient vector $\mathbf{a}$ over $\mathbb{Z}$. Similarly, any $\bar{a} \in \mathcal{I}_q$ is represented uniquely as $\bar{a} = \langle \vec{b}, \bar{\mathbf{a}} \rangle$ for some $\bar{\mathbf{a}}$ over $\mathbb{Z}_q$. Our algorithms that work with field elements simply store and operate on these coefficient vectors, while also keeping track of the corresponding basis, which will be among the few we consider below. Notice that by linearity, if we have some $a \in \mathcal{I}$ represented by coefficient vector $\mathbf{a}$ in basis $\vec{b}$, then $\mathbf{a}$ is also the representation of $ra \in r\mathcal{I}$ in the basis $r\vec{b}$, so we can switch between the two values at essentially no cost.

3.1 The Powerful Basis

Here we define a certain useful $\mathbb{Q}$-basis of K, and $\mathbb{Z}$-basis of R. We call it the "powerful" basis, due to its decomposition in terms of the power bases of K_ℓ, and the fast algorithms associated with it. (We are aware of only one occurrence in the literature of this basis; it coincides with what Bosma [6] calls the "canonical" basis of R.)

Definition 3.1. *The* powerful *basis $\vec{p}$ of $K = \mathbb{Q}(\zeta_m)$ and $R = \mathbb{Z}[\zeta_m]$ is defined as follows:*

– *For a prime power m, define $\vec{p}$ to be the power basis $(\zeta_m^j)_{j \in [\varphi(m)]}$, treated as a vector over $R \subset K$.*
– *For m having prime-power factorization $m = \prod_\ell m_\ell$, define $\vec{p} = \bigotimes_\ell \vec{p_\ell}$, the tensor product of the power(ful) bases $\vec{p_\ell}$ of each $K_\ell = \mathbb{Q}(\zeta_{m_\ell})$.*

For any power $\mathcal{I} = (R^\vee)^k$ of $R^\vee = \langle t^{-1} \rangle$, the powerful basis of $\mathcal{I}$ is $t^{-k} \cdot \vec{p}$.

By definition of the tensor product, $\vec{p}$ is a vector with index set $\prod_\ell [\varphi(m_\ell)]$. So to specify an entry of $\vec{p}$ we need one index $j_\ell \in [\varphi(m_\ell)]$ per prime divisor of m, and the specified entry is $p_{(j_\ell)} = \prod_\ell \zeta_{m_\ell}^{j_\ell}$. Note that because $\zeta_{m_\ell} = \zeta_m^{m/m_\ell} \in K$, it is possible to "flatten" the index set to a size-$\varphi(m)$ subset of $[m]$, where index (j_ℓ) maps to $j = \sum_\ell (m/m_\ell) \cdot j_\ell \bmod m$, and $p_j = \zeta_m^j$. We note that unless m is a prime power, the flattened index set is *not* $[\varphi(m)]$, so the powerful basis differs from the power basis, although it still consists of powers of ζ_m. For instance, for $m = 15$ and $\zeta = \zeta_{15}$, the powerful basis consists of $\zeta^0, \zeta^3, \zeta^5, \zeta^6, \zeta^8, \zeta^9, \zeta^{11}$, and ζ^{14}. For our purposes, it is preferable to maintain the structured index set.

In the full version we prove the following lemma describing the good geometric properties of the powerful basis.

Lemma 3.2. *The length of each element p_j of $\vec{p}$ in ℓ_2 norm is $\|p_j\| = \sqrt{\varphi(m)} = \sqrt{n}$, and in ℓ_∞ norm is $\|p_j\|_\infty = 1$. The largest singular value of $\sigma(\vec{p}^T)$ is $s_1(\vec{p}) = \sqrt{\hat{m}}$, and the smallest singular value is $s_n(\vec{p}) = \sqrt{m/\operatorname{rad}(m)}$, where $\hat{m} = m/2$ if m is even, and $\hat{m} = m$ otherwise.*

We point out while the *power* basis elements also all have ℓ_2 and ℓ_∞ norms $\sqrt{n}$ and 1 (respectively), the power basis can be poorly conditioned. E.g., for $m = 1155 = 3 \cdot 5 \cdot 7 \cdot 11$ its ratio of largest to smallest singular value is $\approx 21.4\sqrt{m}$, whereas for the powerful basis it is exactly $\sqrt{m}$.

3.2 The CRT Basis and Fast Operations

In ring-LWE and its applications, we work in R_q and $R_q^\vee$, and sometimes in $\mathcal{I}_q$ for $\mathcal{I} = (R^\vee)^k$, where $q = 1 \bmod m$ is a prime integer. Here we define Chinese remainder (CRT) bases for these quotients, and describe how they yield fast addition and multiplication.

Recalling that $R \cong \bigotimes_\ell R_\ell$ where $m = \prod_\ell m_\ell$ is the prime-power factorization of m and R_ℓ is the m_ℓth cyclotomic ring, it is easy to verify that the quotient ring $R_q \cong \bigotimes_\ell (R_\ell / qR_\ell)$. Therefore we may focus on the case of prime-power m. A standard fact is that the ideal $\langle q \rangle \subset R$ factors into the product of n distinct prime ideals $\mathfrak{q}_i$, for $i \in \mathbb{Z}_m^*$.

Definition 3.3. *For a positive integer m, the Chinese remainder (or CRT) $\mathbb{Z}_q$-basis $\vec{c}$ of R_q is as follows:*

- *For a prime power m, $\vec{c} = (c_i)_{i \in \mathbb{Z}_m^*}$ is characterized by $c_i = 1 \bmod \mathfrak{q}_i$ and $c_i = 0 \bmod \mathfrak{q}_j$ for $i \neq j$. (Its existence is guaranteed by the Chinese Remainder Theorem.)*
- *For m having prime-power factorization $m = \prod_\ell m_\ell$, define $\vec{c} = \bigotimes_i \vec{c_\ell}$, the tensor product of the CRT bases $\vec{c_\ell}$ of each R_ℓ / qR_ℓ.*

For any power $\mathcal{I} = (R^\vee)^k$ of $R^\vee = \langle t^{-1} \rangle$, the CRT $\mathbb{Z}_q$-basis of $\mathcal{I}_q$ is $t^{-k} \cdot \vec{c}$.

Similarly to the powerful basis, $\vec{c}$ is a vector over R_q having the Cartesian product $\prod_\ell \mathbb{Z}_{m_\ell}^*$ as its index set, which may be flattened to the set $\mathbb{Z}_m^*$ using the bijective correspondence $(j_\ell) \leftrightarrow j = \sum_\ell (m/m_\ell) \cdot j_\ell \in \mathbb{Z}_m^*$. But it is usually more convenient to retain the structured index set.

In the full version of the paper we give a novel, fast "CRT transformation" algorithm for converting between the powerful and CRT bases of R_q, or more generally $\mathcal{I}_q$ for $\mathcal{I} = (R^\vee)_q^k$. The algorithm is analogous to a combination of the Cooley-Tukey and Good-Thomas (mixed radix) FFT algorithms, but specialized to evaluate at only the *primitive* mth roots of unity in a ring. The algorithm is simpler and more efficient than converting between the *power* and CRT bases, which involves reducing modulo the cyclotomic polynomial $\Phi_m(X)$.

Working in the CRT basis yields very fast arithmetic operations. Suppose that m is a prime power. Since $c_i^2 = c_i \in R_q$ and $c_i \cdot c_{i'} = 0 \in R_q$ for distinct $i, i' \in \mathbb{Z}_m^*$, the CRT basis has the property that if $a, b \in R_q$ have coefficient vectors $\mathbf{a}, \mathbf{b}$ (respectively) over $\mathbb{Z}_q$ in the CRT basis—i.e., $a = \langle \vec{c}, \mathbf{a} \rangle$ and $b = \langle \vec{c}, \mathbf{b} \rangle$—then the coefficient vector of $a \cdot b \in R_q$ is the componentwise product $\mathbf{a} \odot \mathbf{b}$ over $\mathbb{Z}_q$. (Addition is componentwise as well, simply by linearity.) Moreover, this extends immediately to powers of $R^\vee$: if $\mathbf{a}, \mathbf{b}$ are the respective coefficient vectors of $a \in (R^\vee)_q^{k_1}$, $b \in (R^\vee)_q^{k_2}$ in the respective CRT bases $t^{-k_1} \cdot \vec{c}$ and $t^{-k_2} \cdot \vec{c}$, then $\mathbf{a} \odot \mathbf{b}$ is the coefficient vector of $a \cdot b \in (R^\vee)_q^k$ in the CRT basis $t^{-k} \cdot \vec{c}$, where $k = k_1 + k_2$.

3.3 The Decoding Basis of $R^\vee$

When working with ring-LWE we need to perform a variety of operations over $R^\vee = \langle t^{-1} \rangle$ or $R_q^\vee$. For certain operations it is best to use the following important $\mathbb{Z}$-basis of $R^\vee$ (and $\mathbb{Z}_q$-basis of $R_q^\vee$).

Definition 3.4. *The decoding basis of $R^\vee$ is $\vec{d} = \vec{p}^\vee$, the dual of the powerful basis $\vec{p}$ of R.*[5]

The decoding basis therefore has the same index set as $\vec{p}$. When m is a prime power, $\vec{d}$ is simply the dual of the power basis $\vec{p} = (\zeta_m^j)_{j \in [\varphi(m)]}$ of R. In general, because $\vec{p}$ is the tensor product of the power bases for prime-power cyclotomics R_ℓ, and $(\vec{a} \otimes \vec{b})^\vee = (\vec{a}^\vee \otimes \vec{b}^\vee)$, it follows that $\vec{d}$ is the tensor product of the decoding bases for each $R_\ell^\vee$.

In the full version of the paper, we prove several important and useful properties of the decoding basis, summarized as follows:

– There are very fast linear transformations (requiring fewer than nd scalar additions, where d is the number of prime divisors of m) for converting between the decoding basis $\vec{d}$ and the powerful basis $t^{-1}\vec{p}$ of $R^\vee$.

[5] Note that unlike the powerful and CRT bases, we do not define a decoding basis for any other power of $R^\vee$; see Section 3.4 for discussion.

– Short elements (as always, in the sense of the canonical embedding) of K have optimally small coefficients with respect to $\vec{d}$, making it a best choice for decoding $R^\vee$. Moreover, $\vec{d}$ also yields (nearly) optimal decoding in higher powers of $R^\vee$.
– Continuous Gaussians (especially spherical ones) as represented in the decoding basis can be sampled very simply and efficiently.

The first fact, combined with the fast CRT transformation, means that we can efficiently convert among the decoding, power, and CRT bases of $R^\vee$ (or $R_q^\vee$) as needed. The latter two facts mean that the decoding basis is an excellent choice for generating and decoding error terms (e.g., in encryption and decryption, respectively). By contrast, the power(ful) basis and other natural bases of R or $R^\vee$ do not typically enjoy the above properties (except when m is a power of 2), and while they can in principle be used for all the same tasks, it would come at a potentially large loss in tightness and/or computational efficiency.

3.4 Decoding $R^\vee$ and Its Powers

Recall from Section 2.2 the "round-off" decoding procedure, which uses short linearly independent vectors in a dual lattice $\Lambda^\vee$ to recover a sufficiently short $\mathbf{x}$ given $\mathbf{x} \bmod \Lambda$. To decode from $K/R^\vee$ to K, we apply the procedure using the decoding basis $\vec{d}$ of $R^\vee$; i.e., the linearly independent dual elements (in $(R^\vee)^\vee = R$) are those of the powerful basis $\vec{p}$. Recall from Lemma 2.1 that the tolerable decoding distance (or subgaussian parameter) depends inversely on the maximum length of the dual elements, and that by Lemma 3.2, every p_j in the powerful basis has length $\|p_j\| = \sqrt{n}$. From this we get corresponding bounds on the decoding operation, as summarized below in Lemma 3.6. We remark that the decoding basis is an optimal choice here.

In some applications (e.g., homomorphic encryption), we need to solve the more general problem of decoding $K/\mathcal{I}$ to K, where $\mathcal{I} = (R^\vee)^k = \langle t^{-k} \rangle$ for some (usually small) $k \geq 1$. The naïve way to do this would be to apply the round-off procedure with the $\mathbb{Z}$-basis $t^{1-k}\vec{d}$ of $\mathcal{I}$. This, however, turns out to be highly suboptimal for many values of m, because the elements of the dual basis $t^{k-1}\vec{p}$ might be much longer than the shortest nonzero elements of $\mathcal{I}^\vee = \langle t^{k-1} \rangle$.

Instead, in the round-off algorithm we use the *scaled decoding basis* $\hat{m}^{1-k}\vec{d}$, which generates the superideal $\mathcal{J} = \hat{m}^{1-k}R^\vee = t^{-k}g^{1-k} \supseteq \mathcal{I}$, and whose dual elements are $\hat{m}^{k-1}\vec{p} \subseteq \mathcal{I}^\vee$. (Recall that $\hat{m} = m/2$ if m is even, and $\hat{m} = m$ otherwise. It is easy to show that $\hat{m} = t \cdot g$ for some $g \in R$; see the full version.) The lengths of the dual elements are therefore $\hat{m}^{k-1}\sqrt{n}$, from which one gets the bounds summarized in Lemma 3.6 below.

We summarize the above discussion in the following definition and lemma. As it will be more convenient for applications, here we consider a "scaled up and discretized" version of the decoding procedure, where we decode from $\mathcal{I}_q$ to $\mathcal{I}$ for some $q \geq 1$. So the unknown short element is guaranteed to be in $\mathcal{I}$, and the output is also expected to be in $\mathcal{I}$. The only difference this makes in the above procedure (apart from the obvious scaling by q) is that for $k \geq 2$, since the scaled decoding basis may generate a strict superideal of $\mathcal{I}$, when the round-off procedure fails to decode correctly it might produce an element that is not in $\mathcal{I}$. In such a case we just consider the output to be undefined.

Definition 3.5 (Decoding $\mathcal{I}_q$ to $\mathcal{I}$). *For $\bar{a} \in \mathcal{I}_q$ where $\mathcal{I} = (R^\vee)^k$ for some $k \geq 1$, let $\bar{a} = \langle \hat{m}^{1-k}\vec{d}, \bar{\mathbf{a}} \rangle \bmod q\mathcal{J}$ for some $\bar{\mathbf{a}}$ over $\mathbb{Z}_q$, where $\mathcal{J} = \hat{m}^{1-k}R^\vee$. Define $[\![\bar{a}]\!]$ to be $\langle \hat{m}^{1-k}\vec{d}, [\![\bar{\mathbf{a}}]\!] \rangle$ if it is in $\mathcal{I}$, otherwise $[\![\bar{a}]\!]$ is undefined (where $[\![\bar{\mathbf{a}}]\!]$ is a vector over $\mathbb{Z}$, as defined in the beginning of Section 2).*

Lemma 3.6. *For any $k \geq 1$ let $\mathcal{I} = (R^\vee)^k$ and let $q \geq 1$ be arbitrary. Then for any $a \in \mathcal{I}$ of length less than $q/(2\hat{m}^{k-1}\sqrt{n})$, we have $[\![a \bmod q\mathcal{I}]\!] = a$. Moreover, if a is δ-subgaussian with parameter s, then for any $b \in (R^\vee)^\ell$ where $\ell \geq 0$, we have $[\![a \cdot b \bmod q(R^\vee)^{k+\ell}]\!] = a \cdot b$ except with probability at most*

$$2n \exp(\delta - \pi q^2/(2s \cdot \hat{m}^{k+\ell-1}\|b\|_2)^2).$$

4 Regularity

In this section we state a certain "regularity theorem" (whose proof appears in the full version) that is useful in cryptographic applications of ring-LWE, such as when adapting the 'dual' cryptosystem and IBEs of Gentry et al. [19] and others. Independently, a closely related statement (specialized to power-of-2 cyclotomics) was recently shown in [33] with a different proof.

The theorem says the following. Assume we are working with the mth cyclotomic of degree $n = \varphi(m)$, and let $q \geq 1$ be a prime integer. Let $a_1, \ldots, a_{\ell-1}$ be chosen uniformly and independently from R_q. Then, with high probability over the choice of the a_i's, the distribution of $b_0 + \sum_{i=1}^{\ell-1} b_i a_i$ is within statistical distance $2^{-\Omega(n)}$ of uniform, where the b_i are chosen from a discrete Gaussian distribution on R of width essentially $nq^{1/\ell}$ (in the canonical embedding). Equivalently, the lemma says that if a_0 is any fixed invertible element of R_q and $a_1, \ldots, a_{\ell-1}$ are uniformly and independently chosen from R_q, then $\sum_{i=0}^{\ell-1} b_i a_i$ is within $2^{-\Omega(n)}$ of uniform, where the b_i are chosen as before. The equivalence follows by simply dividing by a_0. (The lemma we prove is actually more general, and applies to the joint distribution of $k \geq 1$ sums as above; see Theorem 4.1 and Corollary 4.2 for the exact statement.)

This regularity statement is already interesting and non-trivial when ℓ is as small as 2, and is close to being tight: for instance, in case m is a power of 2, a width of at least $\sqrt{n}q^{1/\ell}$ is required just for entropy reasons. To see this, recall that R is a rotation of $\sqrt{n}\mathbb{Z}^n$, so roughly speaking, a discrete Gaussian of width t covers $(t/\sqrt{n})^n$ points.

One might wonder about the significance of the b_0 term, and why we do not analyze the regularity of $\sum_{i=0}^{\ell-1} b_i a_i$ when all the a_i are chosen uniformly from R. In fact, a regularity lemma for exactly such sums was shown by Micciancio [27]. (His work is specialized to the ring $R = \mathbb{Z}[x]/\langle x^n - 1\rangle$, but can be extended to other rings, as observed in [34].) Unfortunately, such sums have a much worse regularity property, and in particular require super-constant ℓ to get negligible distance to uniformity. To see why this is the case, assume q is a prime satisfying $q = 1 \bmod m$, so that $\langle q \rangle$ splits completely into n ideals of norm q each. Letting $\mathfrak{q}$ denote one of these prime factors, notice that with probability $q^{-\ell}$, all the a_i are in $\mathfrak{q}$. In this case, $\sum_{i=1}^{m} b_i a_i$ is in $\mathfrak{q}$ with certainty, and its distribution is therefore very far from uniform. By adding the b_0 term we avoid this "common divisor" problem and get much better regularity, providing exponentially small distance to uniformity already for ℓ as small as 2.

The following is the regularity theorem. Here, for a matrix $A \in R_q^{[k]\times[\ell]}$ we define $\Lambda^{\perp}(A) = \{\vec{z} \in R^{[\ell]} : A\vec{z} = 0 \bmod qR\}$, which we identify with a lattice in H^{ℓ}. Its dual lattice (which is again a lattice in H^{ℓ}) is denoted by $\Lambda^{\perp}(A)^{\vee}$.

Theorem 4.1. *Let R be the ring of integers in the mth cyclotomic number field K of degree n, and $q \geq 2$ an integer. For positive integers $k \leq \ell \leq \operatorname{poly}(n)$, let $A = [I_{[k]} \mid \bar{A}] \in (R_q)^{[k]\times[\ell]}$, where $I_{[k]} \in (R_q)^{[k]\times[k]}$ is the identity matrix and $\bar{A} \in (R_q)^{[k]\times[\ell-k]}$ is uniformly random. Then for all $r > 2n$,*

$$\mathbb{E}_{\bar{A}}\left[\rho_{1/r}(\Lambda^{\perp}(A)^{\vee})\right] \leq 1 + 2(r/n)^{-n\ell} q^{kn+2} + 2^{-\Omega(n)}.$$

In particular, if $r > 2n \cdot q^{k/\ell + 2/(n\ell)}$ then $\mathbb{E}_{\bar{A}}[\rho_{1/r}(\Lambda^{\perp}(A)^{\vee})] \leq 1 + 2^{-\Omega(n)}$, and so by Markov's inequality, $\eta_{2^{-\Omega(n)}}(\Lambda^{\perp}(A)) \leq r$ except with probability at most $2^{-\Omega(n)}$.

Using [31, Claim 3.8], we obtain the following corollary, which is often more useful in applications.

Corollary 4.2. *Let R, n, q, k, and ℓ be as in Theorem 4.1. Assume that $A = [I_{[k]} \mid \bar{A}] \in (R_q)^{[k]\times[\ell]}$ is chosen as in Theorem 4.1. Then, with probability $1 - 2^{-\Omega(n)}$ over the choice of $\bar{A}$, the distribution of $A\vec{x} \in R_q^{[k]}$ where each coordinate of $\vec{x} \in R_q^{[\ell]}$ is chosen from a discrete Gaussian distribution of radius $r > 2n \cdot q^{k/\ell + 2/(n\ell)}$ over R, satisfies that the probability of each of the q^{nk} possible outcomes is in the interval $(1 \pm 2^{-\Omega(n)})q^{-nk}$ (and in particular is within statistical distance $2^{-\Omega(n)}$ of the uniform distribution over $R_q^{[k]}$).*

5 Example Cryptosystem

Here we give an example application of our toolkit which works in arbitrary cyclotomic rings. In particular, we give a public-key cryptosystem whose public key and ciphertext each consists of only two ring elements. In the full version, we also give a simple adaptation of the "dual-style" LWE-based public-key cryptosystem of [19], which uses our regularity theorem of Section 4, and which can serve as a foundation for (hierarchical) identity-based encryption. Additionally, in the full version we provide another (much more involved) example of a symmetric-key "somewhat homomorphic" cryptosystem and all the associated "modulus reduction" and "key switching" algorithms.

Let q be a positive integer that is coprime with every odd prime dividing m, and let p be a positive integer coprime with q. The message space is R_p. Let ψ be a continuous LWE error distribution over $K_{\mathbb{R}}$, and let $\lfloor \cdot \rceil$ denote a valid discretization to (cosets of) $R^{\vee}$ or $pR^{\vee}$. The cryptosystem is defined as follows.

- Gen: choose a uniformly random $a \leftarrow R_q$. Choose $s \leftarrow \lfloor\psi\rceil_{R^\vee}$ and $e \leftarrow \lfloor p \cdot \psi\rceil_{pR^\vee}$. Output $(a, b = \hat{m}(a \cdot s + e) \bmod qR) \in R_q \times R_q$ as the public key, and s as the secret key.
- $\mathsf{Enc}_{(a,b)}(\mu \in R_p)$: choose $z \leftarrow \lfloor\psi\rceil_{R^\vee}$, $e' \leftarrow \lfloor p \cdot \psi\rceil_{pR^\vee}$, and $e'' \leftarrow \lfloor p \cdot \psi\rceil_{t^{-1}\mu + pR^\vee}$.
  Let $u = \hat{m}(a \cdot z + e') \bmod qR$ and $v = b \cdot z + e'' \in R_q^\vee$. Output $(u, v) \in R_q \times R_q^\vee$.
- $\mathsf{Dec}_s(u, v)$: compute $v - u \cdot s = \hat{m}(e \cdot z - e' \cdot s) + e'' \bmod qR^\vee$, and decode it to $d = [\![v - u \cdot s]\!] \in R^\vee$ (see Definition 3.5). Output $\mu = t \cdot d \bmod pR$.

Lemma 5.1. *The above cryptosystem is IND-CPA secure assuming the hardness of R-$\mathsf{DLWE}_{q,\psi}$.*

Lemma 5.2. *Suppose that $\lfloor\psi\rceil_{c+R^\vee}$ is δ-subgaussian with parameter $r \geq 1$ and $\delta = O(1)$, for any coset $c + R^\vee$. Then assuming $q > \hat{m}pr^2 \cdot \omega(\sqrt{n} \log n)$, the decryption procedure is correct with probability negligibly close to one (over all the random choices of* Gen *and* Enc*).*

Acknowledgments. We thank Markus Püschel for his help with the sparse decomposition of the "Chinese remainder transform," and Damien Stehlé for useful discussions.

References

[1] Agrawal, S., Boneh, D., Boyen, X.: Efficient lattice (H)IBE in the standard model. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 553–572. Springer, Heidelberg (2010)

[2] Alwen, J., Peikert, C.: Generating shorter bases for hard random lattices. Theory of Computing Systems 48(3), 535–553 (2011); Preliminary version in STACS 2009

[3] Applebaum, B., Cash, D., Peikert, C., Sahai, A.: Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 595–618. Springer, Heidelberg (2009)

[4] Babai, L.: On Lovász' lattice reduction and the nearest lattice point problem. Combinatorica 6(1), 1–13 (1986); Preliminary version in Mehlhorn, K. (ed.) STACS 1985. LNCS, vol. 182, pp. 13–20. Springer, Heidelberg (1984)

[5] Banerjee, A., Peikert, C., Rosen, A.: Pseudorandom functions and lattices. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 719–737. Springer, Heidelberg (2012)

[6] Bosma, W.: Canonical bases for cyclotomic fields. Appl. Algebra Eng. Commun. Comput. 1, 125–134 (1990)

[7] Boyen, X.: Lattice mixing and vanishing trapdoors: A framework for fully secure short signatures and more. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 499–517. Springer, Heidelberg (2010)

[8] Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. In: ICTS, pp. 309–325 (2012)

[9] Brakerski, Z., Vaikuntanathan, V.: Fully homomorphic encryption from ring-LWE and security for key dependent messages. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 505–524. Springer, Heidelberg (2011)

[10] Cash, D., Hofheinz, D., Kiltz, E., Peikert, C.: Bonsai trees, or how to delegate a lattice basis. J. Cryptology 25(4), 601–639(2010); Preliminary version in Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 523–552. Springer, Heidelberg (2010)

[11] Ducas, L., Durmus, A.: Ring-LWE in polynomial rings. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 34–51. Springer, Heidelberg (2012)

[12] Erdös, P.: On the coefficients of the cyclotomic polynomial. Bulletin of the American Mathematical Society 52(2), 179–184 (1946)

[13] Gentry, C.: A fully homomorphic encryption scheme. PhD thesis, Stanford University (2009), `http://crypto.stanford.edu/craig`

[14] Gentry, C.: Fully homomorphic encryption using ideal lattices. In: STOC, pp. 169–178 (2009)

[15] Gentry, C.: Toward basing fully homomorphic encryption on worst-case hardness. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 116–137. Springer, Heidelberg (2010)

[16] Gentry, C., Halevi, S., Peikert, C., Smart, N.P.: Ring switching in BGV-style homomorphic encryption. In: Visconti, I., De Prisco, R. (eds.) SCN 2012. LNCS, vol. 7485, pp. 19–37. Springer, Heidelberg (2012), Full version at `http://eprint.iacr.org/2012/240`

[17] Gentry, C., Halevi, S., Smart, N.P.: Fully homomorphic encryption with polylog overhead. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 465–482. Springer, Heidelberg (2012)

[18] Gentry, C., Halevi, S., Smart, N.P.: Homomorphic evaluation of the AES circuit. In: Safavi-Naini, R. (ed.) CRYPTO 2012. LNCS, vol. 7417, pp. 850–867. Springer, Heidelberg (2012)

[19] Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: STOC, pp. 197–206 (2008)

[20] Güneysu, T., Lyubashevsky, V., Pöppelmann, T.: Practical lattice-based cryptography: A signature scheme for embedded systems. In: Prouff, E., Schaumont, P. (eds.) CHES 2012. LNCS, vol. 7428, pp. 530–547. Springer, Heidelberg (2012)

[21] Lyubashevsky, V.: Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 598–616. Springer, Heidelberg (2009)

[22] Lyubashevsky, V.: Lattice signatures without trapdoors. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 738–755. Springer, Heidelberg (2012)

[23] Lyubashevsky, V., Micciancio, D.: Generalized compact knapsacks are collision resistant. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006, Part II. LNCS, vol. 4052, pp. 144–155. Springer, Heidelberg (2006)

[24] Lyubashevsky, V., Micciancio, D.: Asymptotically efficient lattice-based digital signatures. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 37–54. Springer, Heidelberg (2008)

[25] Lyubashevsky, V., Micciancio, D., Peikert, C., Rosen, A.: SWIFFT: A modest proposal for FFT hashing. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 54–72. Springer, Heidelberg (2008)

[26] Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 1–23. Springer, Heidelberg (2010)

[27] Micciancio, D.: Generalized compact knapsacks, cyclic lattices, and efficient one-way functions. Computational Complexity 16(4), 365–411 (2002); Preliminary version in FOCS 2002

[28] Micciancio, D., Peikert, C.: Trapdoors for lattices: Simpler, tighter, faster, smaller. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 700–718. Springer, Heidelberg (2012)

[29] Peikert, C.: Public-key cryptosystems from the worst-case shortest vector problem. In: STOC, pp. 333–342 (2009)

[30] Peikert, C., Rosen, A.: Lattices that admit logarithmic worst-case to average-case connection factors. In: STOC, pp. 478–487 (2007)

[31] Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. J. ACM 56(6), 1–40 (2005); Preliminary version in STOC

[32] Smart, N.P., Vercauteren, F.: Fully homomorphic SIMD operations. Cryptology ePrint Archive, Report 2011/133 (2011), `http://eprint.iacr.org/`

[33] Stehlé, D., Steinfeld, R.: Making NTRU as secure as worst-case problems over ideal lattices. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 27–47. Springer, Heidelberg (2011)

[34] Stehlé, D., Steinfeld, R., Tanaka, K., Xagawa, K.: Efficient public key encryption based on ideal lattices. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 617–635. Springer, Heidelberg (2009)

Regularity of Lossy RSA on Subdomains and Its Applications

Mark Lewko[1], Adam O'Neill[2], and Adam Smith[3]

[1] University of California, Los Angeles
mlewko@math.ucla.edu
[2] Boston University
amoneill@bu.edu
[3] Pennsylvania State University
asmith@cse.psu.edu

Abstract. We build on an approach of Kiltz et al. (CRYPTO '10) and bring new techniques to bear on the study of how "lossiness" of the RSA trapdoor permutation under the Φ-Hiding Assumption (ΦA) can be used to understand the security of classical RSA-based cryptographic systems. In particular, we show that, under ΦA, several questions or conjectures about the security of such systems can be reduced to bounds on the regularity (the distribution of the primitive e-th roots of unity mod N) of the "lossy" RSA map (where e divides $\phi(N)$). Specifically, this is the case for: (i) showing that large consecutive runs of the RSA input bits are simultaneously hardcore, (ii) showing the widely-deployed PKCS #1 v1.5 encryption is semantically secure, (iii) improving the security bounds of Kiltz et al. for RSA-OAEP. We prove several results on the regularity of the lossy RSA map using both classical techniques and recent estimates on Gauss sums over finite subgroups, thereby obtaining new results in the above applications. Our results deepen the connection between "combinatorial" properties of exponentiation in $\mathbb{Z}_N$ and the security of RSA-based constructions.

Keywords: RSA encryption, PKCS #1 v1.5, Lossy trapdoor functions, Φ-Hiding Assumption, Gauss sums.

1 Introduction

Cryptographic systems built from the RSA trapdoor permutation [34] are ubiquitous in practice. Though these schemes are simple and highly efficient, they are typically only proven secure in the random oracle model [3], if at all.[1] An important research direction is to prove their security under better-understood assumptions. For example, consider the "simple embedding" RSA-based encryption scheme specified by RSA PKCS #1 v1.5, which is still in widespread use: roughly, the encryption of a plaintext x is $f_{N,e}(x,r) = (x\|r)^e \bmod N$, where r

[1] There are more recent constructions without random oracles, e.g., [19,20], but they are less efficient and seem unlikely to be used in practice in the near future.

is a random string of appropriate length and '$\|$' denotes string concatenation.[2] There was until now no proof of security of this scheme under a standard and well-studied assumption on RSA.[3] In this paper, we show that the security of this and related constructions can be analyzed under natural assumptions without the need for the random oracle model. Our analysis relies on new connections between the security of RSA and "combinatorial" properties of arithmetic in $\mathbb{Z}_N$ (namely, the regularity of exponentation on arithmetic sequences and bounds on the magnitude of Gauss sums).

Key tool: Lossiness. A keyed trapdoor function family f_{pk} on k-bits is L-*lossy* [32] if there are two algorithms, or "modes", for generating public keys: one which generates key *pairs* (pk, sk) such that f_{pk} is injective and can be inverted efficiently given sk, and one which generates keys pk for which f_{pk} is "L-lossy", meaning the image of f_{pk} has at most 2^{k-L} points. The security requirement is that the two modes be computationally indistinguishable. The concept has found applications in many areas of cryptography, for example in [32,6,1,2,31].

Lossiness is a powerful tool, since it allows one to prove security with respect to the lossy mode, where information-theoretic techniques often apply. As concrete example, one can show that a lossy function family admits many simultaneously hard-core bits: in the lossy mode, the (average) min-entropy of a uniformly random input X given $f_{pk}(X)$ is at least L, and hence one can use an appropriate randomness extractor Ext, such as a 2-wise-independent hash family, to obtain a $L - 2\log(1/\varepsilon)$-bit string that is ε-close to uniform even given $f_{pk}(X)$ and the seed. In the injective mode, this string will be *pseudorandom* given $f_{pk}(X)$ and the seed, since a distinguisher for the extracted string would imply a distinguisher that tells apart lossy/injective keys. The existence of many simultaneously hardcore bits allows for the design of efficient, semantically-secure encryption schemes (say, by using these bits as a one-time pad).

Using Lossiness to Analyze Classical RSA-based Constructions. Lossiness has mostly been used in the literature as a tool for designing new cryptographic systems. Recently, however, Kiltz et al. [25] showed that the concept also sheds light on existing, widely-used constructions. Specifically, they showed that RSA-OAEP [4] is semantically secure under the ϕ-Hiding Assumption. The ϕ-Hiding Assumption, abbreviated ΦA, states (roughly) that given an RSA modulus $N = pq$, it is hard to distinguish primes that divide $\phi(N) = (p-1)(q-1)$ from those that do not. ΦA has been used as the basis for a number of efficient protocols [10,9,14,18]. It has also attracted attention of cryptanalysis: the current best attack uses Coppersmith's techniques [12] and applies when $e \le p^{1/2-\varepsilon}$; other attacks [35] are for moduli of a special form that does not include RSA. Kiltz et al. [25] observed that the RSA map $x \mapsto x^e$ in $\mathbb{Z}_N^*$ is $\log(e)$-*lossy* under

[2] In practice x and r are typically switched and some bits of r are a fixed constant; however, this won't affect our results.

[3] We clarify that the parameters (i.e., the RSA modulus and exponent length) supported by our security proof are not practical (see the discussion at the end of the Intro for more details). However, prior work does not provide a proof for *any* parameter settings.

ΦA: Valid RSA keys (N, e), for which $\gcd(e, \phi(N)) = 1$, are computationally indistinguishable from "lossy" pairs (N, e) for which e divides $p - 1$, and the map $x \mapsto x^e$ in $\mathbb{Z}_N^*$ is e-to-one when e divides $(p - 1)$.

Thus, *ΦA* implies that the RSA map hides $\log(e)$ bits of information (on average) about x. But without additional information about *which* bits are hidden, it seems we can only analyze constructions which have extractor-like objects (such as keyed hash functions), explicitly built in. For example, Kiltz et al. [25] analyzed the OAEP padding scheme by modeling the random oracles as keyed, t-wise independent functions. One can similarly exploit the observation above about simultaneous hardcore bits to replace the random oracle in the "Simple RSA" or "RSA-KEM" scheme proposed by Shoup [36]. However, this methodology does not apply to schemes that do not use keyed hash functions or do not use hash functions at all.

1.1 Our Contributions

We show specific, natural functions that are hidden by RSA in the lossy case, and use these results to obtain proofs of several natural conjectured security properties of RSA-based constructions, assuming *ΦA*. Roughly, the three main applications are: (i) Any run of $\log(e) = \Omega(\log N)$ consecutive physical bits are *simultaneously* hardcore for RSA. (The assumption that RSA is hard to invert implies only that $\log \log N$ physical bits are simultaneously hardcore.) (ii) PKCS #1 v1.5 (described above) is semantically secure against chosen *plaintext* attacks (CPA). (iii) Improved parameters for the CPA security of RSA-OAEP (improving on the reduction in [25]).

These results emanate from our core technical contribution: showing that, in the lossy setting (when e divides $\phi(N)$), exponentiation by e is nearly *regular* on certain subdomains $\mathcal{K} \subseteq \mathbb{Z}_N$. Regularity means that all (or most) points in the image of $x \mapsto x^e$ have approximately the same number of preimages in $\mathcal{K}$. This implies in turn that X^e is approximately uniform on its image when X is uniform on $\mathcal{K}$. Consider, for example, the natural conjecture that the t most significant bits of the input are hardcore for exponentiation. To prove this conjecture under *ΦA*, it suffices to show that, for every fixed string $z \in \{0,1\}^t$, the value $(z \| R)^e$ is nearly uniform, where $R \leftarrow \{0,1\}^{\lceil \log N \rceil - t}$ (this implies that any two settings z, z' of the hardcore bits are statistically indistinguishable in the lossy mode, and computationally indistinguishable in the usual injective mode). This question corresponds to the regularity of exponentiation by e on the arithmetic progression $\mathcal{K} = \{z 2^{\lceil \log N \rceil - t} + r : r = 0, ..., 2^{\lceil \log N \rceil - t} - 1\}$.

Below, we explain our results in more detail.

Results on Regularity of Exponentiation. We prove several results on the regularity of lossy exponentiation on subdomains of $\mathbb{Z}_N$ when $N = pq$. The subdomains we consider have some additive structure, which somehow breaks up the multiplicative structure of exponentiation. Consider a subdomain $\mathcal{K} \subseteq \mathbb{Z}_N$. Let X be uniform on $\mathcal{K}$ and U be uniform on $\mathbb{Z}_N$. Note that for X^e to be close to U^e, the set $\mathcal{K}$ must have size at least $\phi(N)/e \approx N/e$.

Regularity for Random Translations of a Set: Our first result considers random translations of an arbitrary subdomain $\mathcal{K}$. Specifically, we show that the pair $(C, (C+X)^e)$ is statistically close to (C, U^e) when C is uniform on $\mathbb{Z}_n$, as long as $\mathcal{K}$ has size larger than N/e. The proof relies on a careful collision probability argument. One key piece of that argument is the observation that the random offset and exponentiation *together* behave like a universal hash function: for any two values $a, b \in \mathbb{Z}_N$, the ratio $\left(\frac{a+C}{b+C}\right)^e$ is nearly uniformly distributed over e-th residues, conditioned on the denominator being invertible. The other main piece is a careful accounting of the probability of noninvertible elements; this is delicate because the conversion from collision probability to $\mathcal{L}_1$-distance can amplify very small irregularities.

This first result achieves optimal parameters but it takes advantage of "averaging" in two senses: first, it averages over translations of the initial set $\mathcal{K}$ and second, it only implies regularity on average over points in the pre-image (since we show that the $\mathcal{L}_1$-distance between the resulting distributions is small). This turns out to be insufficient for some applications, motivating our second result.

Regularity for Arithmetic Progressions and the Relation to Gauss Sums: Our second result is more specific: we show that if $\mathcal{K}$ is a sufficiently long arithmetic progression (with period relatively prime to N), then exponentiation is regular on $\mathcal{K}$ in a strong sense: the number of preimages of *every* point in the image is approximately the same.

The main tool in our analysis is a reduction from the question of regularity to bounds on Gauss sums. Given a prime p, and integers a, d, consider the sum $\mathcal{G}_p(a,d) := \sum_{x=1}^{p} \omega^{ax^d}$, where $\omega = \exp(2\pi i/p)$ is a primitive p-th root of unity and the arithmetic is in $\mathbb{C}$. We show that if e divides $p-1$ and $N = pq$, then

$$\left|\Pr(X^e = a) - \frac{e}{N}\right| \le \max_{b \neq 0} \left|\mathcal{G}_p(b, \tfrac{p-1}{e})\right| \cdot O\left(\frac{e \log K}{pK}\right)$$

where K is the length of $\mathcal{K}$. One can think of the Gauss sums $\mathcal{G}_p(\cdot, d)$ as the Fourier coefficients of the function $x \mapsto x^d$ over $\mathbb{Z}_p$. The proof of our main lemma uses Fourier analysis over $Z_N = \mathbb{Z}_p \times \mathbb{Z}_q$ to connect regularity to the magnitude of the sum.

Leveraging the rich literature on bounds on Gauss sums, we obtain regularity results for different ranges of e (relative to p). These results show that exponentiation (when e divides $\phi(N)$) is a *deterministic extractor* for sources whose support is a sufficiently long arithmetic progression. Moreover, the output distributions are close to uniform not only in $\mathcal{L}_1$-distance but also in the stronger $\mathcal{L}_\infty$ sense. Given the state of our knowledge of bounds for Gauss sums, this second class of results yields weaker (but still useful) bounds on uniformity than our first result. (For a comparison of the bounds, see the end of Section 4.)

Applications to RSA-Based Cryptosystems. Our regularity bounds imply the following new security results for RSA-based constructions under the ϕ-Hiding Assumption (ΦA):

Natural hardcore bits for RSA: Any run of about $\log(e)$ consecutive physical bits of x are simultaneously hardcore for RSA. Specifically, let $f(x)$ denote a run of $\log(e) - 4\log(1/\varepsilon)$ physical bits of the input x. Our result on random translations implies (with some additional work) that the pair $(f(C), C^e)$ is ε-close to $(f(C), U^e)$ when e divides $p-1$ and C, U are uniform in $\mathbb{Z}_n$. If we consider either the most significant bits or least significant bits of x, then one can improve the run length to $\log(e) - 3\log(1/\varepsilon)$ (that is, we get $\log(1/\varepsilon)$ additional hardcore bits).

Semantic security of PKCS #1 v1.5: Our bounds on regularity on arithmetic progressions imply that PKCS #1 v1.5, which encrypts m as $(0002_{16}\|m\|00_{16}\|R)^e$ where R is uniform, is semantically secure (aka. chosen-plaintext secure or IND-CPA) [16] for certain parameters. Indeed, note that the space of pre-images for a given message m forms an arithmetic progression with period 1 (variants of this scheme mentioned in Footnote 2 give progression with period 2^t for $t > 1$, which is still relatively prime to N). Under ΦA, with appropriately long R, the ciphertext is thus indistinguishable from uniform, for every fixed message m.

Improved Security for RSA-OAEP: Finally, our regularity bounds for arithmetic sequences also give tighter standard-model security proofs for the IND-CPA security of RSA-OAEP, improving on the results of [25]. The RSA-OAEP scheme as per PKCS #1 v2.1 encrypts m as $(0002_{16}\|\mathsf{OAEP}(m))^e$ where OAEP is some randomized, invertible transformation. Recall [25] were able to obtain their best security bound in the case that the lossy RSA map is (close to) regular on the subdomain $\{0002_{16}\|x \mid x \in \{0,1\}^{k-16}\}$ and left it as an open problem to prove this. As this subdomain forms an arithmetic progression, we resolve this positively.

Discussion and concrete parameters. As mentioned above, our result on regularity for random translations averages in two senses. The first sense is sufficient to obtain results on hardcore bits, but not security of PKCS # 1 v1.5 encryption or RSA-OAEP. Roughly, this is because the elements in the subdomains contain fixed messages and constants as substrings. Getting better regularity bounds without additional randomness (i.e., eliminating the first sense) remains an interesting open problem. This is primarily a concern for the PKCS #1 v1.5 application, since we need regularity on arithmetic progressions of length $2^{\rho} = 2^{\Omega(k)}$ where ρ is the length of the random padding. In the RSA-OAEP application we use length $2^{k-16} = 2^{k+O(1)}$, for which our regularity bounds on arithmetic progressions give much better parameters.

To give an idea of the concrete parameters we obtain, for modulus length $k = 2048$ we get about 190 natural hardcore bits. In our security bound for PKCS #1 v1.5, for modulus length $k = 8192$ we support about 128-bit messages. Finally, in the RSA-OAEP application we get significant savings: e.g., secure encryption of about 100-bit longer messages than supported by [25] for modulus length $k = 2048$ (274-bit messages for 80-bit security rather than 160-bits). In terms of practice, we view our results mainly as providing a qualitative, theoretical backing for in-use schemes at some parameter settings. We hope our techniques will prove useful in future work and the bounds will be improved.

1.2 Related Work

Following [25], Kakvi and Kiltz [23] showed that lossiness of RSA under ΦA is also useful to understand security of a classical RSA-based *signature*, giving improved security bounds for the RSA Full-Domain Hash signature scheme [3]. Jager et al. [21] recently provided a standard-model analysis of TLS-DHE, another widely used protocol. Gauss sums also have applications to elliptic-curve cryptography, see e.g. [37,26,38].

Bleichenbacher [5] (see also [22]) gave a well-known chosen-ciphertext attack against PKCS #1 v1.5 encryption, which has since been patched and the scheme is still in widespread use for legacy reasons. Coron et al. [13] gave *chosen-plaintext* attacks on PKCS #1 v1.5 encryption. These do not contradict our results because the attacks of [13] are for different parameter settings. Specifically, they rely on the length of the random padding being quite small. Our results require sufficiently large random padding– at least $\frac{3}{4}\log N$ bits – as well as large e. Interestingly, our analysis implies a plausible setting in which PKCS #1 v1.5 is provably immune to the attacks of [13] under ΦA.

The "large hardcore bit conjecture" for RSA and the security of the simple embedding scheme are mentioned as important open problems by Goldreich [15]. Prior progress was made by Steinfeld et al. [39], who showed that the $1/2 - 1/e - \varepsilon - o(1)$ least significant bits of RSA are simultaneously hardcore under a computational problem related to the work of Coppersmith [12]. This result does not apply as such to PKCS #1 v1.5 because the latter does not use the full RSA domain (some bits are fixed constants). Moreover, we show *chosen-plaintext security*, i.e., security for arbitrary messages, rather than only for random ones (which, disregarding some of the other bits being fixed constants, is equivalent to the message bits being hardcore). The fact that PKCS #1 v1.5 is believed to be CPA-secure but no proof is known is also discussed by Katz and Lindell [24, pg. 363].

2 Preliminaries

Notation. For a probabilistic algorithm A, by $y \leftarrow_{\$} A(x)$ we mean that A is executed on input x and the output is assigned to y, whereas if S is a finite set then by $s \leftarrow_{\$} S$ we mean that s is assigned a uniformly random element of S. Unless otherwise specified, an algorithm may be probabilistic and its running-time includes that of any overlying experiment. We denote by 1^k the unary encoding of the security parameter k. We sometimes surpress dependence on k for readability. For $i \in \mathbb{N}$ we denote by $\{0,1\}^i$ the set of all (binary) strings of length i. If s is a string then $|s|$ denotes its length in bits, whereas if S is a set then $|S|$ denotes its cardinality. By '$\|$' we denote string concatenation. If s is a string then for all $1 \leq i \leq j \leq |s|$ we denote by $s[i \dots j]$ its substring of bits i through j.

We will occasionally use the usual asymptotic notation $X = O(Y)$ or $X \ll Y$ to indicate that $|X| \leq C|Y|$ for some unspecified constant C. We will also make use of the exact asymptotic notation $X = \mathcal{O}(Y)$ to indicate that $|X| \leq |Y|$ without any unspecified constant. We will use $\log(\cdot)$ to denote the natural logarithm and C to denote an absolute constant, which may change at unrelated occurrences. In addition, we use C_ε to denote an absolute constant that depends only on its subscript, ε. If X and Y are random variables on common domain, then their *statistical distance* is $\Delta(X,Y) = 1/2 \sum_x |\Pr[X = x] - \Pr[Y = x]| = 1/2 \cdot ||X - Y||_1$.

We borrow the following notation from [25]. For $i \in \mathbb{N}$ we denote by $\mathcal{P}_i$ the set of all i-bit primes. By RSA_k we denote the set of tuples (N, p, q) where p, q are distinct $k/2$-bit primes and $N = pq$. Let R be a relation on p and q. By $\mathsf{RSA}_k[R]$ we denote the subset of RSA_k for which the relation R holds on p and q. For example, let e be a prime. Then $\mathsf{RSA}_k[p = 1 \bmod e]$ is the set of all (N, p, q), where where $N = pq$ is the product of two distinct $k/2$-bit primes p, q and $p = 1 \bmod e$. That is, the relation $R(p, q)$ is true if $p = 1 \bmod e$ and q is arbitrary. By $(N, p, q) \leftarrow_\$ \mathsf{RSA}_k[R]$ we mean that (N, p, q) is sampled according to the uniform distribution on $\mathsf{RSA}_k[R]$.

RSA, Lossy RSA, and Phi-hiding. Recall that the *RSA function* for modulus N and encryption exponent e is defined as

$$\mathsf{RSA}_{N,e}(x) = x^e \bmod N \ .$$

To specify the RSA trapdoor permutation family we need to give a *parameter-generation algorithm* that specifies how parameters N, e are generated. (We ignore the decryption exponent d for now.) Letting $0 < c < 1$ be a public constant, we define two of them ("Injective RSA" and "Lossy RSA"):

Algorithm $\mathsf{RSA}_{inj}(1^k)$:	**Algorithm** $\mathsf{RSA}_{loss}(1^k)$:
$e \leftarrow_\$ \mathcal{P}_{ck}$	$e' \leftarrow_\$ \mathcal{P}_{ck}$
$(N, p, q) \leftarrow_\$ \mathsf{RSA}_k$	$(N', p', q') \leftarrow_\$ \mathsf{RSA}_k[p' = 1 \bmod e']$
Return (N, e)	Return (N', e')

The *Phi-Hiding Assumption* (ΦA) for c [10] states that (N, e) is computationally indistinguishable from (N', e') where (N, e) is generated via $\mathsf{RSA}_{inj}(1^k)$ and (N', e') is generated via $\mathsf{RSA}_{loss}(1^k)$. More precisely, to a distinguisher D we associate its *ΦA-advantage* defined as

$$\mathbf{Adv}^{\Phi\mathrm{A}}_{D,c}(k) \ = \ \Pr[\, D(N, e) \text{ outputs } 1\,] - \Pr[\, D(N', e') \text{ outputs } 1\,]$$

with inputs generated as above.

As shown by [25], RSA constitutes a *lossy trapdoor permutation* in the sense of [32] under *ΦA* by using the above two parameter generation algorithms. (We avoid giving a formal definition of lossy TDPs in the paper, since our results are specifically tied to ΦA.) We recall that we need $e \leq p^{1/2-\varepsilon}$ to avoid Coppersmith's attack [12,29] on ΦA. More specifically, N can be factored efficiently if $e \geq p^{1/2}$ and in time $O(N^\varepsilon)$ if $c = 1/4 - \varepsilon$ (*i.e.*, $\log e \geq \log N(1/4 - \varepsilon)$). For example, with modulus size $k = 2048$ we can set $\varepsilon = .04$ for 80-bit security (to enforce $k\varepsilon \geq 80$) and obtain $2048 \cdot (1/4 - 0.04) = 430$ bits of lossiness.

3 Approximate Regularity on Subdomains

We start by defining variants of regularity we consider.

Notions of regularity on subdomains. Let $f\colon D \to R$ be a function from domain D to range R. We say that f is *regular* on D if $|f^{-1}(y)| = |D|/|R|$ for every $y \in R$. Suppose f is regular. Let $D' \subseteq D$ be a subdomain.

Definition 1 ($\mathcal{L}_1$-regularity). *We say that f is λ-$\mathcal{L}_1$-*regular on *D' if for all $y \in R$,*

$$\Delta(f(X), f(X')) \le \lambda \ ,$$

where $X \leftarrow_{\$} D$ and $X' \leftarrow_{\$} D'$.

Above "λ" is the approximation factor and "$\mathcal{L}_1$" indicates that we measure the regularity via the $\mathcal{L}_1$-norm.

We also consider the following "worst-case" regularity notion. For $y \in R$ we denote by $f^{-1}(y)[D']$ the preimage set of y restricted to D', that is

$$f^{-1}(y)[D'] := \{x \in D' \mid x \in f^{-1}(y)\} \ .$$

Definition 2 ($\mathcal{L}_\infty$-regularity). *We say that f is ν-$\mathcal{L}_\infty$-*regular on *D' if for all $y \in R$,*

$$\left| \frac{|f^{-1}(y)[D']|}{|D'|} - \frac{1}{|R|} \right| \le \nu \ .$$

Equivalently, f is ν-$\mathcal{L}_\infty$-regular on D' if for all $y \in R$

$$|\Pr[f(X') = y \ : \ X' \leftarrow_{\$} D'] - \Pr[f(X) = y \ : \ X \leftarrow_{\$} D]| \le \nu \ .$$

In other words, $\mathcal{L}_1$-regularity is a bound on the $\mathcal{L}_1$-distance of a random image point from the subdomain from uniform, and $\mathcal{L}_\infty$-regularity is a bound on the $\mathcal{L}_\infty$-distance from uniform. The following proposition (which immediately follows from the definitions) will be useful:

Proposition 3. *Suppose f is ν-$\mathcal{L}_\infty$-regular on D'. Then f is $\nu|R|$-$\mathcal{L}_1$-regular on D'.*

Thus, if f is ν-$\mathcal{L}_\infty$-regular on D' for $\nu \ll 1/|R|$, then f is $o(1)$-$\mathcal{L}_1$-regular on D'.

Main technical question. We can now state the main (informal) technical question of this work. Consider the "lossy RSA" function $\mathsf{RSA}_{N',e'}$ where (N', e') is output by $\mathsf{RSA}_{loss}(1^k)$.

> What is the approximate regularity of $\mathsf{RSA}_{N',e'}$ on subdomains of $\mathbb{Z}_N$ of sufficient size?

In Section 4 we answer this question for a variety of parameter setting and regularity notions in the case that the subdomain of certain forms, in particular those described by *arithmetic progressions*. In Section 5 we give applications of these results to hardcore bits of RSA, RSA PKCS v1.5 encryption, and RSA-OAEP encryption.

4 Bounds on Approximate Regularity of Lossy RSA

We give bounds on the approximate regularity of lossy of RSA for a variety of parameter settings as notions of regularity.

4.1 $\mathcal{L}_1$-Regularity for Random Translations

We consider *expected* $\mathcal{L}_1$-regularity of lossy RSA over random translation of a fixed subset. The following lemma says for any subset of sufficient size, we have good expected $\mathcal{L}_1$-regularity over a random translation of the subset. It can also be viewed as saying that exponentiation with a random offset modulo N is a strong seeded extractor. (However, this interpretation is just for understanding; we do not use the lemma this way.)

Lemma 4. *Let $N = pq$ and e be such that $e \mid p-1$ and $\gcd(e, q-1) = 1$. Let $\mathcal{K} \subseteq \mathbb{Z}_N$ such that $|\mathcal{K}| \geq 4N/(e\alpha^2)$ for some $\alpha \geq \frac{4(p+q-1)}{N}$. Then*

$$(C, (C+X)^e \bmod N) \approx_\alpha (U', U^e \bmod N)$$

where $C, U', U \leftarrow_\$ \mathbb{Z}_N$ and $X \leftarrow_\$ \mathcal{K}$.

The proof relies on a careful collision probability argument. One key piece of that argument is the observation that the random offset and exponentiation *together* behave like a universal hash function: for any two values $a, b \in \mathbb{Z}_N$, the ratio $\left(\frac{a+C}{b+C}\right)^e$ is nearly uniformly distributed over e-th residues, conditioned on the denominator being invertible.

Proof. For ease of notation let $\mathcal{P}$ denote the distribution of $(C, (C+X)^e)$ and $\mathcal{U}$ denote the distribution of (C, U^e) (we omit the "mod N" here and below when it is clear from context). Let $K = |\mathcal{K}|$. Write

$$\mathcal{P} = \mathcal{P}_1 + \mathcal{P}_0 \text{ and } \mathcal{U} = \mathcal{U}_1 + \mathcal{U}_0$$

where $\mathcal{P}_1$ denotes the distribution of $\mathcal{P} \wedge (C+X)^e \in \mathbb{Z}_N^*$, $\mathcal{P}_0$ denotes the distribution of $\mathcal{P} \wedge (C+X)^e \notin \mathbb{Z}_N^*$, $\mathcal{U}_1$ denotes the distribution of $\mathcal{U} \wedge U^e \in \mathbb{Z}_N^*$, and $\mathcal{U}_0$ denotes the distribution of $\mathcal{U} \wedge U^e \notin \mathbb{Z}_N^*$. Note that

$$\Delta(\mathcal{P}, \mathcal{U}) = ||\mathcal{P} - \mathcal{U}||_1 = ||\mathcal{P}_1 - \mathcal{U}_1||_1 + ||\mathcal{P}_0 - \mathcal{U}_0||_1 .$$

We bound each term with the following claims.

Claim.

$$||\mathcal{P}_1 - \mathcal{U}_1||_1 \leq \frac{\alpha}{2} .$$

Proof. We have

$$\begin{aligned} ||\mathcal{P}_1 - \mathcal{U}_1||_1 &\leq \sqrt{\text{supp}(\mathcal{P}_1 - \mathcal{U}_1)} \cdot ||\mathcal{P}_1 - \mathcal{U}_1||_2 \\ &\leq \sqrt{\text{supp}(\mathcal{P}_1 - \mathcal{U}_1) \cdot (||\mathcal{P}_1||_2)^2 - 1} . \qquad (1) \end{aligned}$$

The first line above is by the Cauchy-Schwarz inequality, and the second is due to the fact that $(\mathcal{P}_1 - \mathcal{U}_1) \perp \mathbf{1}$, which follows from the observation that

$$\langle \mathbf{1}, \mathcal{P}_1 \rangle = \frac{p+q-1}{N} = \langle \mathbf{1}, \mathcal{U}_1 \rangle$$

where the first equality is because the marginal distribution of $(C+X)^e$ is that of U^e. To bound Equation (1), note that $\mathrm{supp}(\mathcal{P}_1 - \mathcal{U}_1) = N\phi(N)/e$. Also,

$$(||\mathcal{P}_1||_2)^2 = \Pr\left[\,(C,(C+X)^e) = (C',(C'+Y)^e \wedge z \in \mathbb{Z}_N^*\,\right] = \frac{1}{N} \cdot \Pr\left[\,(C+X)^e = (C+Y)^e \ \wedge z \in \mathbb{Z}_N^*\,\right]$$

where $X, Y \leftarrow_{\$} \mathcal{K}$ and z denotes the common value of $(C+X)^e$ and $(C+Y)^e$. We have

$$\begin{aligned}
&\Pr\left[\,(C+X)^e = (C+Y)^e \ \wedge z \in \mathbb{Z}_N^*\,\right] = \sum_{\omega} \Pr\left[\,(C+X)/(C+Y) = \omega \ \wedge \ z \in \mathbb{Z}_N^*\,\right] \\
&\leq \Pr\left[\,X = Y\,\right] + \sum_{\omega \neq 1} \Pr\left[\,(C+X)/(C+Y) = \omega \ \wedge \ z \in \mathbb{Z}_N^* \ \mid \ X \neq Y\,\right] \cdot \Pr\left[\,X \neq Y\,\right] \\
&\leq \Pr\left[\,X = Y\,\right] + \sum_{\omega \neq 1} \Pr\left[\,C = (\omega Y - X)/(\omega - 1) \ \wedge \ z \in \mathbb{Z}_N^* \ \mid \ X \neq Y\,\right] \cdot \Pr\left[\,X \neq Y\,\right] \\
&\leq \frac{1}{K} + \frac{e-1}{N}\left(1 - \frac{1}{K}\right) .
\end{aligned}$$

where ω is an e-th root of unity modulo N (for which there are e possibilities); for the second-to-last line above we use the fact that $X = Y$ iff $\omega = 1$. Plugging the above into Equation (1) yields

$$\begin{aligned}
||\mathcal{P}_1 - \mathcal{U}_1||_1 &\leq \sqrt{\frac{\phi(N)}{e}\left(\frac{1}{K} + \frac{e-1}{N}\left(1 - \frac{1}{K}\right)\right) - 1} \\
&= \sqrt{\frac{\phi(N)/e}{K} + \left(\frac{e-1}{e} \cdot \frac{\phi(N)}{N} \cdot \frac{K-1}{K} - 1\right)} \ \leq \ \frac{\alpha}{2}
\end{aligned}$$

as desired, where for the last inequality we use the assumption $K \geq 4N/(e\alpha^2)$. ▌

Claim.

$$||\mathcal{P}_0 - \mathcal{U}_0||_1 \leq \frac{\alpha}{2} .$$

Proof. We have

$$||\mathcal{P}_0 - \mathcal{U}_0||_1 \ \leq \ \langle \mathbf{1}, \mathcal{P}_0 \rangle + \langle \mathbf{1}, \mathcal{U}_0 \rangle \ = \ \frac{2(p+q-1)}{N} \ \leq \ \frac{\alpha}{2}$$

where the last inequality uses the assumption that $\alpha \geq \frac{4(p+q-1)}{N}$. ▌▌

4.2 $\mathcal{L}_\infty$-Regularity for Arithmetic Progressions

We next consider the $\mathcal{L}_\infty$-regularity of lossy RSA on subdomains described by *arithmetic progressions.* We start with some definitions.

Arithmetic progressions. Recall that a subset $P \subseteq [1, N]$ is an *arithmetic progression* if it can be expressed as $P = \{\sigma\ell + \tau : 1 \le \ell \le K\}$ for some $\tau, \sigma \neq 0$. Here σ is called the *period* of the arithmetic progression.

Gauss sums. We define a *Gauss sum* as

$$\mathcal{G}_p(a, d) := \sum_{x=1}^{p} e_p(ax^d) \tag{2}$$

where $a, d \in \mathbb{N}$, $e(x) = e^{2\pi i x}$ and $e_p(x) := e(x/p)$. Trivially one has $|\mathcal{G}_p(a, d)| \le p$. There are a variety of tighter bounds available for various choices of parameters which will be discussed later.

Connecting Gauss sums to Lossy RSA. First we show how estimates on Gauss sums imply results about approximate regularity of lossy RSA.

Lemma 5. *Let $N = pq$ and e be such that $e \mid p - 1$ and $\gcd(e, q - 1) = 1$. Assume that*

$$\max_{a\neq 0} \left| \mathcal{G}_p\left(a, \frac{p-1}{e}\right) \right| \le Cp^\theta$$

for some $0 < \theta < 1$. Let $P = \{\sigma\ell + \tau : 1 \le \ell \le K\}$ ($\sigma, \tau \in \mathbb{N}$), $P \subseteq [1, N]$ denote an arithmetic progression with $(\sigma, N) = 1$, and let $\mathcal{K} = \{(x, N) = 1 : x \in P\}$. We then have that (assuming $p, q > 2$ and $|\mathcal{K}| \ge \max(p, q)$)

$$\left| \Pr_{x\leftarrow\mathcal{K}}[x^e = a] - \Pr_{x\leftarrow\mathbb{Z}_N^*}[x^e = a] \right| \le 7|\mathcal{K}|^{-1} + 10Ce|\mathcal{K}|^{-1}p^{\theta-1}\log(|\mathcal{K}|). \tag{3}$$

Proof. If a is not in the range of $x \mapsto x^e$ then $\Pr_{x\leftarrow\mathcal{K}}[x^e = a] = \Pr_{x\leftarrow\mathbb{Z}_N^*}[x^e = a] = 0$, so we assume that a is in the range. Next we recall some elementary number theory. We will identify an element $x \in \mathbb{Z}_m^*$ ($m = p, q$ or N) with its smallest positive integer representative which we will denote $\overline{x}$. The Chinese remainder theorem gives the isomorphism $\mathbb{Z}_N^* \cong \mathbb{Z}_p^* \oplus \mathbb{Z}_q^*$. This isomorphism is explicitly given from $\mathbb{Z}_N^*$ to $\mathbb{Z}_p^* \oplus \mathbb{Z}_q^*$ by the map $a \mapsto (\overline{a} \mod p, \overline{a} \mod q)$. Let $\mathcal{S} := \{x \in \mathbb{Z}_N^* : x^e = a\}$, $a_p = \overline{a} \mod p$, and $a_q = \overline{a} \mod q$. Denote by $\mathcal{S}_p := \{\overline{x} \mod p : x \in \mathcal{S}\} = \{u^e = a_p : u \in \mathbb{Z}_p^*\}$ and $\mathcal{S}_q := \{\overline{x} \mod q : x \in \mathcal{S}\} = \{v^e = a_q : v \in \mathbb{Z}_q^*\}$. Since $(e, q-1) = 1$ we have that that map $v \mapsto v^e$ is a bijection on $\mathbb{Z}_q^*$ and hence $|\mathcal{S}_q| = 1$. We will denote this element as s_q. The map $u \mapsto u^e$ on $\mathbb{Z}_p^*$ is e-to-1, so $|\mathcal{S}_p| = e = |\mathcal{S}|$. Moreover, $\mathcal{S}_p$ is a coset of a subgroup and can be represented as $\mathcal{S}_p = \{x^{\frac{p-1}{e}} b : x \in \mathbb{Z}_p^*\}$, for any $b \in \mathcal{S}_p$. Our goal is to estimate $|\mathcal{K} \cap \mathcal{S}|$. Given a set $S \subseteq \mathbb{Z}_m^*$ (for $m = p, q$ or N), we will denote the associated indicator function as $1_S(x) \mapsto \{0, 1\}$. Thus,

$$|\mathcal{K} \cap \mathcal{S}| = \sum_{x\in\mathcal{K}} 1_{\mathcal{S}}(x) = \sum_{x\in\mathcal{K}} 1_{s_q}(x) 1_{\mathcal{S}_p}(x) = \sum_{\substack{x\in\mathcal{K} \\ x\equiv s_q \mod q}} 1_{\mathcal{S}_p}(x).$$

We can expand $1_{\mathcal{S}_p}(x) = \sum_{\xi\in\mathbb{Z}_p^*} \widehat{1_{\mathcal{S}_p}}(\xi)e_p(x\xi)$ where the Fourier coefficients $\widehat{1_{\mathcal{S}_p}}(\xi)$ are given by

$$\widehat{1_{\mathcal{S}_p}}(\xi) = p^{-1}\sum_{x\in\mathbb{Z}_p} 1_{\mathcal{S}_p}(x)e_p(-x\xi).$$

We have that

$$\widehat{\mathcal{S}_p}(\xi) = p^{-1}\frac{e}{p-1}\mathcal{G}(b\xi, \frac{p-1}{e}).$$

Thus $\widehat{1_{\mathcal{S}_p}}(0) = \frac{e}{p-1}$ and $|\widehat{\mathcal{S}_p}(\xi)| \leq C\frac{e}{p-1}p^{\theta-1}$ for $\xi \neq 0$.

Let $\mathcal{K}' := \{\overline{x} \equiv s_q \mod q : x \in \mathcal{K}\}$. We wish to estimate $|\mathcal{K}'|$ in terms of $|\mathcal{K}|$. In what follows we make use of the assumption that $(\sigma, N) = 1$ which prevents P from degenerating to a point when reduced $\mod p$ or $\mod q$. Noting that $\mathcal{K}$ is obtained from P by sieving out elements congruent to $0 \mod p$ and $0 \mod q$, we have that $|P|(1 - p^{-1} - q^{-1}) + \mathcal{O}(3) = |\mathcal{K}|$. Now if we define $I := \{\overline{x} \equiv s_q \mod q : x \in P\}$ and $E := \{\overline{x} \equiv 0 \mod p : x \in I\}$, then $\mathcal{K}' = I \setminus E$. Moreover, $|E| \leq 1$ since if $b \in E \subseteq \mathbb{Z}_N^*$ then $b \equiv s_q \mod q$ and $b \equiv 0 \mod p$ which uniquely specifies b by the Chinese remainder theorem. Thus $|\mathcal{K}'| = |P|q^{-1} + \mathcal{O}(2)$ where $|P| = (|\mathcal{K}| + \mathcal{O}(3))\frac{qp}{qp-p-q}$, which gives

$$|\mathcal{K}'| = (|\mathcal{K}| + \mathcal{O}(3))\frac{p}{qp-p-q} + \mathcal{O}(2) = \frac{1}{q-1}|\mathcal{K}| + \mathcal{O}(7)$$

where we have used that $\frac{p}{qp-p-q} \leq 1$ (using that $p, q > 2$) and $\frac{p}{qp-p-q} = \frac{1}{q-1-qp^{-1}} = \frac{1}{q-1} + \frac{q-1+qp^{-1}}{(q-1)(q-1-qp^{-1}} = \frac{1}{q-1} + \mathcal{O}(2)$.

We now may express

$$1_{\mathcal{S}_p}(x) = \frac{e}{p-1} + \sum_{\xi\in\mathbb{Z}_p^*} \widehat{1_{\mathcal{S}_p}}(\xi)e_p(x\xi).$$

Thus

$$|\mathcal{K}\cap\mathcal{S}| = \sum_{x\in I\setminus E} 1_{\mathcal{S}_p}(\overline{x}) = \frac{e|\mathcal{K}'|}{(p-1)} + \sum_{\xi\in\mathbb{Z}_p^*}\widehat{1_{\mathcal{S}_p}}(\xi)\sum_{x\in I\setminus E} e_p(x\xi)$$

Using that $|\widehat{\mathcal{S}_p}(\xi)| \leq Cep^{\theta-2}$ we have

$$\left| |\mathcal{K}\cap\mathcal{S}| - \frac{e|\mathcal{K}|}{\phi(N)}\right| \leq 7 + C\frac{e}{p-1}p^{\theta-1}\sum_{\xi\in\mathbb{Z}_p^*}\left|\sum_{x\in I\setminus E} e_p(x\xi)\right|$$

Now I can be expressed as an arithmetic progression, $I = \{xq + b : x = 1, 2, \ldots |I|\}$ so

$$|\sum_{x\in I} e_p(x\xi)| = |\sum_{x=1}^{|I|} e_p(-b\xi)e_p(qx\xi)| = |\sum_{x=1}^{|I|} e_p(qx\xi)| = \left|\frac{\sin(\pi\xi q|I|/p)}{\sin(\pi\xi q/p)}\right| \leq \left|\frac{1}{\sin(\pi q\xi/p)}\right|.$$

Using the inequality $\sin(\pi x) \geq 2x$ for $-1/2 \leq x \leq 1/2$ and denoting the distance of a real number to the nearest integer by $|| \cdot ||$ the quantity above is $\leq 2^{-1}||\xi q p^{-1}||^{-1}$. Thus,

$$\sum_{\xi \in \mathbb{Z}_p^*} \left| \sum_{x \in I \setminus E} e_p(x\xi) \right| \leq 1+2^{-1} \sum_{\xi \in \mathbb{Z}_p^*} ||\xi q p^{-1}||^{-1} \leq 1+2^{-1} p \sum_{n=1}^{|I|} n^{-1} \leq 1+2^{-1} p (1+\log(|I|))$$

where we have used the inequality $\sum_{n=1}^{|I|} n^{-1} \leq 1+\log(|I|)$. Since $\Pr_{x \leftarrow \mathbb{Z}_N^*}[x^e = a] = \frac{e}{\phi(N)}$ and $\Pr_{x \leftarrow \mathcal{K}}[x^e = a] = \frac{|\mathcal{K} \cap \mathcal{S}|}{|\mathcal{K}|}$, putting this all together we have that

$$\left| \Pr_{x \leftarrow \mathcal{K}}[x^e = a] - \Pr_{x \leftarrow \mathbb{Z}_N^*}[x^e = a] \right| \leq 7|\mathcal{K}|^{-1} + C\frac{e}{p-1}|\mathcal{K}|^{-1} p^{\theta - 1} \left(1 + 2^{-1} p \left(\log\left(\frac{|\mathcal{K}|}{q} + 1\right) + 1\right)\right).$$

From our assumptions we have $\log\left(\frac{|\mathcal{K}|}{q} + 1\right) \leq \log(|\mathcal{K}|)$, and $\log\left(\frac{|\mathcal{K}|}{q} + 1\right) + 1 \leq 2\log(|\mathcal{K}|)$. Now $1 + 2^{-1} p \times 2 \log(|\mathcal{K}|) \leq 5 \log(|\mathcal{K}|)$. Using $\frac{1}{p-1} \leq \frac{2}{p}$, we have

$$\left| \Pr_{x \leftarrow \mathcal{K}}[x^e = a] - \Pr_{x \leftarrow \mathbb{Z}_N^*}[x^e = a] \right| \leq 7|\mathcal{K}|^{-1} + 10Ce|\mathcal{K}|^{-1} p^{\theta-1} \log(|\mathcal{K}|).$$

This completes the proof. ∎

Known estimates on Gauss sums. We now summarize some known estimates on $\mathcal{G}_p(a, e)$. Throughout, we assume $1 \leq a < p$.

$$|\mathcal{G}_p(a,d)| \leq \begin{cases} (d-1)p^{1/2} & , 1 \leq d \leq p^{1/3} \\ 2 \cdot 3^{-1/4} d^{5/8} p^{5/8} & , p^{1/3} < d \leq p^{1/2} \\ 2 \cdot 3^{-1/4} d^{3/8} p^{3/4} & , p^{1/2} < d \leq p^{2/3} \\ C_\delta p^{1-\varepsilon(\delta)} & , p^{2/3} < d < p^{\delta}. \end{cases}$$

The first estimate is classical, the second and third are due to Heath-Brown and Konyagin [17] (with the explicit constants given by Cochrane and Pinner [11]), and the fourth is due to Bourgain, Glibichuk, and Konyagin [8]. To the best of our knowledge explicit values of C_δ have not been worked out. Also, see [27] and [7] for some additional refinements. Much more is believed to be true, in particular Montgomery, Vaughan, and Wooley [30] have made the following conjecture

$$|\mathcal{G}_p(a,d)| \leq \min\{(d-1)p^{1/2}, (1+\eta)(2dp\log(dp))^{1/2}\} \tag{4}$$

where $\eta \to 0$ as d and p/d tend to infinity.

Bounds on regularity of Lossy RSA. Combining the known estimates with Lemma 5 gives the following corollary.

Corollary 6. *With the notation and assumptions of Lemma 5 we have*

$$\text{The map } x \mapsto x^e \text{ is } \nu\text{-}\mathcal{L}_\infty\text{-regular on } \mathcal{K} \text{ for} \begin{cases} \nu = C_\delta e p^{-\varepsilon(\delta)} \frac{\log(|\mathcal{K}|)}{|\mathcal{K}|} & , p^\delta \le e \le p^{1/3} \\ \nu = 23 e^{5/8} p^{1/8} \frac{\log(|\mathcal{K}|)}{|\mathcal{K}|} & , p^{1/3} < e \le p^{1/2} \\ \nu = 23 e^{3/8} p^{1/4} \frac{\log(|\mathcal{K}|)}{|\mathcal{K}|} & , p^{1/2} < e \le p^{2/3} \\ \nu = 17 p^{1/2} \frac{\log(|\mathcal{K}|)}{|\mathcal{K}|}. & , p^{2/3} < e < p. \end{cases}$$

Recall from Section 2 that we take $e \le p^{1/2-\varepsilon}$ in our applications, and thus only the first two cases above are applicable. We include bounds for other parameter ranges in case they are useful for future work. (Indeed, we consider finding a relaxation of lossiness that avoids Coppersmith's attack but still allows our proof techniques to go through to be an interesting open problem.) We are also able to obtain improvements for some values of e and K by appealing to estimates for incomplete character sums, see the full version [28] for details.

Consequences of the MVW conjecture and comparison to collision probability bound. One may check that the Montgomery-Vaughan-Wooley (MVW) conjecture mentioned above would give ν-$\mathcal{L}_\infty$-regularity for $\nu = O\left(e^{1/2} \log^{1/2}(p) \frac{\log(|\mathcal{K}|)}{|\mathcal{K}|}\right)$.[4] Thus, disregarding logarithmic factors the MWV conjecture implies λ-$\mathcal{L}_1$-regularity for $\lambda = O(\sqrt{N/eK} \cdot \sqrt{N/K})$, whereas the collision probability bound in Section 4.1 gives $\lambda = O(\sqrt{N/eK})$, which is which is always better since $K \le N$. However, when $K \sim N$ these bounds essentially agree and are both asymptotically around $1/\sqrt{e}$.

5 Applications

We describe several applications of our results.

5.1 Large Consecutive Runs of Hardcore Bits for RSA

We can use our result on expected $\mathcal{L}_1$-regularity of lossy RSA over random translations given in Section 4.1 to derive new results on substrings of the RSA input that are hardcore.

Hardcore substrings. We begin by defining what it means for a substring of the input to be hardcore. For $1 \le i < j \le k$, we say that the (i,j)*-th substring of RSA is simultaneously hardcore* (we omit "simultaneously" below, with it being understood) if the following two distributions are computationally indistinguishable:

$$\mathsf{DistReal} := \{(N, e, x^e \bmod N, x[i \dots j])) \ : \ (N,e) \leftarrow_\$ \mathsf{RSA}_{inj}(1^k) \ ; \ x \leftarrow_\$ \mathbb{Z}_N^*\}$$
$$\mathsf{DistRand} := \{(N, e, x^e \bmod N, r) \ : \ (N,e) \leftarrow_\$ \mathsf{RSA}_{inj}(1^k) \ ; \ x \leftarrow_\$ \mathbb{Z}_N^* \ ; \ r \leftarrow_\$ \{0,1\}^{j-i}\} \ .$$

[4] Note that Parseval's identity gives us that $\sum_{a \in \{x^{(p-1)/d} : x \in Z_p\}} |\mathcal{G}_p(a,d)|^2 \gg d^2 p$ (see section 4 of [30]). Thus, for some a we must have $|\mathcal{G}(p,a)| \gg (dp)^{1/2}$. So (disregarding logarithmic factors) nothing beyond the MWV conjecture is possible.

To a distinguisher D we associate its *hardcore-bits advantage* defined as

$$\mathbf{Adv}^{\mathrm{hcb}}_{i,j,D}(k) \;=\; \Pr\left[\, D(\mathsf{DistReal}) \text{ outputs } 1 \,\right] - \Pr\left[\, D(\mathsf{DistRand}) \text{ outputs } 1 \,\right] .$$

Our result. We are now ready to state our result: Roughly, under ΦA, (1) the $\log e - 3\log(1/\varepsilon)$ most significant bits of RSA are hardcore, (2) the $\log e - 3\log(1/\varepsilon)$ least significant bits of RSA are hardcore, and (3) an arbitrary substring of $\log e - 4\log(1/\varepsilon)$ bits of RSA are hardcore. Typically, in practice the least significant bits are used, see e.g. [39].

Theorem 7. *Assume that ΦA holds for c and let $\varepsilon > 0$. Let $1 \le i < j \le n$ such that $|j-i| \le \log e - 4\log(1/\varepsilon) - 2$. Then for any hardcore-bits distinguisher D against RSA there is a ΦA distinguisher D' such that for all $k \in \mathbb{N}$*

$$\mathbf{Adv}^{\mathrm{hcb}}_{i,j,D}(k) \le \mathbf{Adv}^{\Phi \mathrm{A}}_{c,D'}(k) + 2\varepsilon .$$

The running-time of D is that of D'. In the special cases $i = 1$ or $j = n$ (i.e., for the least significant or most significant bits), we only require $|j-i| \le \log e - 3\log(1/\alpha) - 2$.

The proof is in the full version [28]. The main idea is to note that, in Lemma 4, if we choose X appropriately, the (i,j)-th substrings of X and $X + C$ in the lemma will be the same. To ensure this we need to choose X so that, with high probability, we avoid "overflow" modulo N, or a carry into the the i-th bit position in the addition. Ensuring this is why we pay an extra $2\log(1/\varepsilon)$ bits (versus Lemma 4) in the theorem in general.

Concrete parameters. Recall from Section 2 that with modulus size $k = 2048$ we can take $\log e$ to be roughly 430 bits for 80-bit security. Then, taking $\varepsilon = 2^{-80}$ in Theorem 7, we get 190 natural hardcore bits of RSA (either the 190 most significant bits or the 190 least significant bits). Similarly, with $k = 3072$ we get 688 bits of lossiness and 448 natural hardcore bits.

5.2 IND-CPA Security of PKCS #1 v1.5

We can use our results on $\mathcal{L}_\infty$-regularity of lossy RSA on arithmetic progressions given in Section 4.2 to prove security of PKCS #1 v1.5 encryption.

PKCS #1 v1.5 encryption. Namely, define the "simple embedding" RSA-based encryption scheme defined as follows. Let μ, ρ such that $\mu + \rho + 32 = k$ be integer parameters. Define the randomized encoding function PKCS that takes plaintext $x \in \{0,1\}^\mu$ and coins $r \in \{0,1\}^\rho$ and outputs

$$\mathsf{PKCS}(x; r) = x \| 00_{16} \| r .$$

Define the encryption scheme $\Pi_{\mathsf{PKCS}} = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$ by

```
Alg Kg(1^k)
    (N, e, d) ←$ RSA_inj(1^k)
    Return ((N, e), (N, d))

Alg Enc((N, e), x)
    x' ←$ PKCS(x)
    x'' ← 0002_16 || x'
    y ← (x'')^e mod N
    Return y

Alg Dec((N, d), y)
    x' ← y^d mod N
    0002_16 || x || 00_16 || r ← x'
    Return x
```

Essentially such a scheme was adopted by PKCS #1 v1.5 and is still in widespread use. (In practice, as opposed to in the academic literature, r and x are switched; however, this doesn't affect our results.)

Our result. We show the first positive result about the security of this scheme; namely, for certain parameters it is IND-CPA secure. (The standard definition of IND-CPA security is recalled in Appendix A.)

Theorem 8. *Suppose ΦA holds for c and lossy RSA is ν-$\mathcal{L}_\infty$-regular on arithmetic progressions of length 2^ρ. Then for any IND-CPA adversary A against Π_{PKCS}, there is a distinguisher D against ΦA such that for every $k \in N$*

$$\mathbf{Adv}^{\text{ind-cpa}}_{\Pi_{\mathsf{PKCS}},A}(k) \;\le\; \mathbf{Adv}^{\Phi\text{A}}_{c,D}(k) + \frac{\nu \cdot \phi(N)}{e} .$$

The running-time of D is that of A.

Proof. (Sketch.) The first step of the proof is to replace (N, e) generated via $\mathsf{RSA}_{inj}(1^k)$ in the IND-CPA game with (N', e') generated via $\mathsf{RSA}_{loss}(1^k)$. Now consider the distribution of a ciphertext $C = (0002_{16}\|x\|00_{16}\|R)^{e'} \bmod N'$ for any fixed but arbitrary plaintext $x \in \{0,1\}^\mu$, where $R \in \{0,1\}^\rho$ is uniformly random. (Note that x may depend on N', e' here, which are fixed in the argument below.) Notice subdomain $\{0002_{16}\|x\|00_{16}\|r \mid r \in \{0,1\}^\rho\}$ is described by the arithmetic progression of length 2^ρ, namely $\{2^{\rho+16+\mu}\cdot 2 + 2^{\rho+16}\cdot x + i \mid 1 \le i \le 2^\rho\}$. (Variants of the scheme, e.g. where R and x are switched, are described by an arithmetic progression with a different period.) Proposition 3 tells us that lossy RSA is $\nu(\phi(N)/e)$-$\mathcal{L}_1$-regular on arithmetic progressions of length 2^ρ, and thus $\Delta(C, U^e) \le \nu(\phi(N)/e)$. Noting that U^e is independent of x concludes the proof. ∎

Concrete parameters. We can calculate concrete security bounds for the scheme according to our results on the $\mathcal{L}_\infty$-regularity of lossy RSA on arithmetic progressions given in Section 4.2. As per Section 2 assume $e = p^{1/2-\varepsilon}$. Then according to part 2 of Corollary 6 we have that lossy RSA is ν-$\mathcal{L}_\infty$-regular on arithmetic progressions of length K for

$$\nu = 23e^{5/8}p^{1/8}\frac{\log(K)}{K} .$$

Now $p = N^{1/2}$ so $e = N^{1/4-\varepsilon/2}$ and thus

$$\begin{aligned}\nu \;=\; \frac{23 \cdot N^{7/32-(5/16)\varepsilon}\log K}{K} \;&=\; \frac{e}{\phi(N)}\left(\frac{N}{e}\cdot\frac{23 \cdot N^{7/32-(5/16)\varepsilon}\log K}{K}\right)\\ &=\; \frac{e}{\phi(N)}\left(\frac{23 \cdot N^{31/32+(3/16)\varepsilon}\log K}{K}\right)\end{aligned}$$

Imposing

$$\nu \leq \frac{e}{\phi(N)} N^{-\varepsilon} \implies K \geq 23 \cdot N^{31/32 + 11/8\varepsilon} \cdot \log N \;.$$

Unfortunately, we must use a very long modulus length to get any meaningful concrete bound. For example, consider modulus length $k = 8192$. By taking $\varepsilon = .01$ we get 80-bit security and approximately 128-bit messages. Hence, we view our result as a qualitative one (in the same vein, we note, for example, that the current concrete security reduction for RSA-OAEP requires modulus length 4096 [33] for a meaningful bound) and hope further work will improve the parameters.

5.3 Improved Security Reduction for RSA-OAEP

Finally, we can use our bounds on $\mathcal{L}_\infty$-regularity of lossy RSA on arithmetic progressions given in Section 4.2 to improve the bounds given in [25] on CPA security of RSA-OAEP encryption scheme [4], as adopted by PKCS #1 v2.1 as a replacement for the "simple embedding" scheme described above.

RSA-OAEP ENCRYPTION. Let μ, ρ such that $\mu + \rho + 16 = k$ be integer parameters. Define the randomized encoding function OAEP that takes plaintext $x \in \{0,1\}^\mu$ and coins $r \in \{0,1\}^\rho$ and outputs

$$\mathsf{OAEP}(x; r) = G(r) \oplus x \| H(s) \oplus r$$

where $s = G(r) \oplus x$ and G, H are hash functions. Define scheme $\Pi_{\mathsf{OAEP}} = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$ by

```
Alg Kg($1^k$)
    $(N,e,d) \leftarrow_{\$} \mathsf{RSA}_{inj}(1^k)$
    Return $((N,e),(N,d))$

Alg Enc($(N,e),x$)
    $x' \leftarrow_{\$} 0002_{16} \| \mathsf{OAEP}(x)$
    $y \leftarrow (x')^e \bmod N$
    Return $y$

Alg Dec($(N,d),y$)
    $x' \leftarrow y^d \bmod N$
    $0002_{16} \| s \| t$
    $r \leftarrow t \oplus H(s)$ ; $x \leftarrow s \oplus G(r)$
    Return $x$
```

OUR RESULT. Recall that Theorem 4.2 of [25] provides several bounds on the CPA security of RSA-OAEP. The best bound, namely Part 2 of Theorem 4.2, requires that RSA is (close to) regular on the subdomain where the most significant two bytes of the input are zero. However, they left this as an open problem and resorted to a worse bound for general functions, namely Part 3 of Theorem 4.2. Here we adapt Part 2 to prove our result:

Theorem 9. *Suppose ΦA holds and $\mathsf{RSA}_{N',e'}$ as defined above is $\lambda(e/\phi(N))$-$\mathcal{L}_\infty$-regular on arithmetic progressions of length 2^{k-16}, and G is t-wise independent. Then for every IND-CPA adversary A against Π_{OAEP} there is a distinguisher D against ΦA such that*

$$\mathbf{Adv}^{\text{ind-cpa}}_{\Pi_{\mathsf{OAEP}},A}(k) \;\leq\; \mathbf{Adv}^{\Phi\mathrm{A}}_{c,D}(k) + 2^{-u}$$

where

$$u = \frac{t}{2t+2}(\log \lambda + \rho - s - \log t + 2) - \frac{\mu + s + 2}{t+1} - 1$$

The running-time of D' is that of D.

Specifically, "$\log \lambda$" term in the above theorem was absent in Part 2 of Theorem 4.2 [25], but its presence here follows from their analysis (see "Proof of Part 1", pg. 14, of [25]).

Concrete parameters. Notice that, in this application, as opposed to Section 5.2, we only need regularity on very large arithmetic progressions of length 2^{k-16} (independent of μ, ρ) versus length 2^ρ in Section 5.2. More precisely, we need ν-$\mathcal{L}_\infty$-regularity for say $\nu \leq e/\phi(N) \cdot 1/2$ (rather than $\nu \leq e/\phi(N) \cdot \mathsf{negl}(k)$) to essentially lose nothing in the bound as compared to the bound of Part 2 of Theorem 4.1 in [25] (we lose just $\log 2 = 1$ additional bit). Following our calculations in Section 5.2 we impose

$$\nu \leq \frac{e}{\phi(N)} N^{-\delta} \implies K \geq 23 \cdot N^{31/32+3/8\varepsilon+\delta} \cdot \log N \; .$$

For modulus length $k = 2048$ we take take $\varepsilon = .04$ and $\delta = .001$ for 80-bit security, and obtain $\log K \geq 2032$, meaning we can indeed fix 16 bits of the domain to zeros for ν-regularity. The concrete value of these of this savings is significant. For example, with $k = 2048$ we can support 274-bit messages for 80-bit security rather than 160-bits as obtained by the general bound of [25, Part 1, Theorem 4.2].

Acknowledgements. We thank the Eurocrypt 2013 anonymous reviewers for their helpful comments. A.O. is supported in part by NSF grants CNS-1012910 and CNS-0546614; additionally, this work was done in part while at the University of Texas at Austin, supported by NSF grants CNS-0915361 and CNS-0952692. M.L. is supported by a NSF Postdoctoral Fellowship, DMS-1204206.

References

1. Bellare, M., Brakerski, Z., Naor, M., Ristenpart, T., Segev, G., Shacham, H., Yilek, S.: Hedged public-key encryption: How to protect against bad randomness. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 232–249. Springer, Heidelberg (2009)
2. Bellare, M., Hofheinz, D., Yilek, S.: Possibility and impossibility results for encryption and commitment secure under selective opening. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 1–35. Springer, Heidelberg (2009)
3. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Ashby, V. (ed.) ACM CCS 1993, pp. 62–73. ACM Press (November 1993)
4. Bellare, M., Rogaway, P.: Optimal asymmetric encryption. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 92–111. Springer, Heidelberg (1995)
5. Bleichenbacher, D.: Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 1–12. Springer, Heidelberg (1998)
6. Boldyreva, A., Fehr, S., O'Neill, A.: On notions of security for deterministic encryption, and efficient constructions without random oracles. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 335–359. Springer, Heidelberg (2008)

7. Bourgain, J.: Multilinear exponential sums in prime fields under optimal entropy condition on the sources. Geom. Funct. Anal. 18(5), 1477–1502 (2009)
8. Bourgain, J., Glibichuk, A.A., Konyagin, S.V.: Estimates for the number of sums and products and for exponential sums in fields of prime order. J. London Math. Soc. (2) 73(2), 380–398 (2006)
9. Cachin, C.: Efficient private bidding and auctions with an oblivious third party. In: ACM CCS 1999, pp. 120–127. ACM Press (November 1999)
10. Cachin, C., Micali, S., Stadler, M.A.: Computationally private information retrieval with polylogarithmic communication. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 402–414. Springer, Heidelberg (1999)
11. Cochrane, T., Pinner, C.: Explicit bounds on monomial and binomial exponential sums. Q. J. Math. 62(2), 323–349 (2011)
12. Coppersmith, D.: Small solutions to polynomial equations, and low exponent RSA vulnerabilities. Journal of Cryptology 10(4), 233–260 (1997)
13. Coron, J.-S., Joye, M., Naccache, D., Paillier, P.: New attacks on PKCS#1 v1.5 encryption. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 369–381. Springer, Heidelberg (2000)
14. Gentry, C., Mackenzie, P.D., Ramzan, Z.: Password authenticated key exchange using hidden smooth subgroups. In: Atluri, V., Meadows, C., Juels, A. (eds.) ACM CCS 2005, pp. 299–309. ACM Press (November 2005)
15. Goldreich, O.: Foundations of Cryptography: Basic Applications, vol. 2. Cambridge University Press, Cambridge (2004)
16. Goldwasser, S., Micali, S.: Probabilistic encryption. Journal of Computer and System Sciences 28(2), 270–299 (1984)
17. Heath-Brown, D.R., Konyagin, S.: New bounds for Gauss sums derived from kth powers, and for Heilbronn's exponential sum. Q. J. Math. 51(2), 221–235 (2000)
18. Hemenway, B., Ostrovsky, R.: Public-key locally-decodable codes. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 126–143. Springer, Heidelberg (2008)
19. Hofheinz, D., Kiltz, E.: Practical chosen ciphertext secure encryption from factoring. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 313–332. Springer, Heidelberg (2009)
20. Hohenberger, S., Waters, B.: Short and stateless signatures from the RSA assumption. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 654–670. Springer, Heidelberg (2009)
21. Jager, T., Kohlar, F., Schäge, S., Schwenk, J.: On the security of TLS-DHE in the standard model. In: Safavi-Naini, R. (ed.) CRYPTO 2012. LNCS, vol. 7417, pp. 273–293. Springer, Heidelberg (2012)
22. Jager, T., Schinzel, S., Somorovsky, J.: Bleichenbacher's attack strikes again: Breaking PKCS#1 v1.5 in XML encryption. In: Foresti, S., Yung, M., Martinelli, F. (eds.) ESORICS 2012. LNCS, vol. 7459, pp. 752–769. Springer, Heidelberg (2012)
23. Kakvi, S.A., Kiltz, E.: Optimal security proofs for full domain hash, revisited. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 537–553. Springer, Heidelberg (2012)
24. Katz, J., Lindell, Y.: Introduction to Modern Cryptography. Chapman and Hall/CRC Press (2007)
25. Kiltz, E., O'Neill, A., Smith, A.: Instantiability of RSA-OAEP under chosen-plaintext attack. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 295–313. Springer, Heidelberg (2010)
26. Kohel, D.R., Shparlinski, I.: On exponential sums and group generators for elliptic curves over finite fields. In: Bosma, W. (ed.) ANTS 2000. LNCS, vol. 1838, pp. 395–404. Springer, Heidelberg (2000)

27. Konyagin, S.V.: Estimates for trigonometric sums over subgroups and for Gauss sums. In: IV International Conference "Modern Problems of Number Theory and its Applications": Current Problems, Part III (Russian) (Tula, 2001), pp. 86–114. Mosk. Gos. Univ. im. Lomonosova, Mekh.-Mat. Fak., Moscow (2002)
28. Lewko, M., O'Neill, A., Smith, A.: Regularity of lossy RSA on subdomains and its applications. Cryptology ePrint Archive (2013), `http://eprint.iacr.org/`
29. May, A.: Using lll-reduction for solving rsa and factorization problems: A survey. In: LLL+25 Conference in Honour of the 25th Birthday of the LLL Algorithm (2007), `http://www.cits.rub.de/imperia/md/content/may/paper/lll.ps`
30. Montgomery, H.L., Vaughan, R.C., Wooley, T.D.: Some remarks on Gauss sums associated with kth powers. Math. Proc. Cambridge Philos. Soc. 118(1), 21–33 (1995)
31. Nishimaki, R., Fujisaki, E., Tanaka, K.: Efficient non-interactive universally composable string-commitment schemes. In: Pieprzyk, J., Zhang, F. (eds.) ProvSec 2009. LNCS, vol. 5848, pp. 3–18. Springer, Heidelberg (2009)
32. Peikert, C., Waters, B.: Lossy trapdoor functions and their applications. In: Ladner, R.E., Dwork, C. (eds.) 40th ACM STOC, pp. 187–196. ACM Press (May 2008)
33. Pointcheval, D.: How to encrypt properly with rsa. RSA Laboratories Crypto Bytes 5(1), 9–19 (2002)
34. Rivest, R.L., Shamir, A., Adleman, L.M.: A method for obtaining digital signature and public-key cryptosystems. Communications of the Association for Computing Machinery 21(2), 120–126 (1978)
35. Schridde, C., Freisleben, B.: On the validity of the Φ-hiding assumption in cryptographic protocols. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 344–354. Springer, Heidelberg (2008)
36. Shoup, V.: A proposal for an ISO standard for public-key encryption. ISO/IEC JTC (2001), `http://www.shoup.net/papers/iso-2_1.pdf`
37. Shparlinski, I.: On gaussian sums for finite fields and elliptic curves. In: Lobstein, A., Litsyn, S.N., Zémor, G., Cohen, G. (eds.) Algebraic Coding 1991. LNCS, vol. 573, pp. 5–15. Springer, Heidelberg (1992)
38. Shparlinski, I.: Bilinear character sums over elliptic curves. Finite Fields and Their Applications 14(1), 132–141 (2008)
39. Steinfeld, R., Pieprzyk, J., Wang, H.: On the provable security of an efficient RSA-based pseudorandom generator. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 194–209. Springer, Heidelberg (2006)

A Public-key Encryption and Its Security

A *public-key encryption scheme* with message-space *MsgSp* is a triple of algorithms $\mathit{\Pi} = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$. The key-generation algorithm Kg returns a public key *pk* and matching secret key *sk*. The encryption algorithm Enc takes *pk* and a plaintext m to return a ciphertext. The deterministic decryption algorithm Dec takes *sk* and a ciphertext c to return a plaintext. We require that for all messages $m \in \mathit{MsgSp}$

$$\Pr[\mathsf{Dec}(sk, \mathsf{Enc}(pk, m)) \neq m \ : \ (pk, sk) \leftarrow_{\$} \mathsf{Kg}]$$

is negligible.

To an encryption scheme $\mathit{\Pi} = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$ and an adversary $A = (A_1, A_2)$ we associate a chosen-plaintext attack experiment,

```
**Experiment** $\mathbf{Exp}^{\text{ind-cpa}}_{\mathit{\Pi},A}(k)$
    $b \leftarrow_{\$} \{0,1\}$ ; $(pk, sk) \leftarrow_{\$} \mathsf{Kg}(1^k)$
    $(m_0, m_1, St) \leftarrow_{\$} A_1(pk)$
    $c \leftarrow_{\$} \mathsf{Enc}(pk, m_b)$
    $d \leftarrow_{\$} A_2(pk, c, St)$
    If $d = b$ then return 1 else return 0
```

where we require A's output to satisfy $|m_0| = |m_1|$. Define the *ind-cpa advantage* of A against $\mathit{\Pi}$ as

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathit{\Pi},A}(k) = 2 \cdot \Pr\left[\mathbf{Exp}^{\text{ind-cpa}}_{\mathit{\Pi},A}(k) \text{ outputs } 1\right] - 1 \, .$$

Efficient Cryptosystems from 2^k-th Power Residue Symbols[⋆]

Marc Joye and Benoît Libert

Technicolor 975 avenue des Champs Blancs, 35576 Cesson-Sévigné Cedex, France
{marc.joye,benoit.libert}@technicolor.com

Abstract. Goldwasser and Micali (1984) highlighted the importance of randomizing the plaintext for public-key encryption and introduced the notion of semantic security. They also realized a cryptosystem meeting this security notion under the standard complexity assumption of deciding quadratic residuosity modulo a composite number. The Goldwasser-Micali cryptosystem is simple and elegant but is quite wasteful in bandwidth when encrypting large messages. A number of works followed to address this issue and proposed various modifications.

This paper revisits the original Goldwasser-Micali cryptosystem using 2^k-th power residue symbols. The so-obtained cryptosystems appear as a very natural generalization for $k \geq 2$ (the case $k = 1$ corresponds exactly to the Goldwasser-Micali cryptosystem). Advantageously, they are efficient in both bandwidth and speed; in particular, they allow for fast decryption. Further, the cryptosystems described in this paper inherit the useful features of the original cryptosystem (like its homomorphic property) and are shown to be secure under a similar complexity assumption. As a prominent application, this paper describes the most efficient lossy trapdoor function based on quadratic residuosity.

Keywords: Public-key encryption, quadratic residuosity, Goldwasser-Micali cryptosystem, homomorphic encryption, standard model.

1 Introduction

Encryption is arguably one of the most fundamental cryptographic primitives. Although it seems an easy task to identify properties that a good encryption scheme must fulfill, it turns out that rigorously defining the right security notion is not trivial at all. Security is context sensitive. Merely requiring that the plaintext cannot be recovered from the ciphertext is not enough in most applications. One may require that the knowledge of some *a priori* information on the plaintext does not help the adversary to obtain any new information, that is, beyond what can be obtained from the *a priori* information. This intuition is formally captured by the notion of *semantic security*, introduced by Goldwasser and Micali in their seminal paper [20]. They also introduced the equivalent notion of *indistinguishability of encryptions*, which is usually easier to work with.

[⋆] Full version available at http://eprint.iacr.org/

Given the encryption of any two equal-length (distinct) plaintexts, an adversary should not be able to distinguish the corresponding ciphertexts.

Clearly, the latter notion is only achievable by probabilistic public-key encryption schemes. One such cryptosystem was also presented in [20]. It achieves ciphertext indistinguishability under the *Quadratic Residuosity* (QR) assumption. Informally, this assumption says that it is infeasible to distinguish squares from non-squares in $\mathbb{J}_N$ (*i.e.*, the set of elements in $\mathbb{Z}_N^*$ whose Jacobi symbol is 1) where $N = pq$ is an RSA-type modulus of unknown factorization.

The Goldwasser-Micali cryptosystem is simple and elegant. The public key comprises an RSA modulus $N = pq$ and a non-square $y \in \mathbb{J}_N$ while the private key is the secret factor p. The encryption of a bit $m \in \{0, 1\}$ is given by $c = y^m x^2 \bmod N$ for a random $x \in \mathbb{Z}_N^*$. The message m is recovered using p, by checking whether c is a square: $m = 0$ if so, and $m = 1$ otherwise —observe that a non-square $y \in \mathbb{J}_N$ is also a non-square modulo p. The encryption of a string $m = (m_{k-1}, \ldots, m_0)_2$, with $m_i \in \{0, 1\}$, proceeds by forming the ciphertexts $c_i = y^{m_i} x^2 \bmod N$, for $0 \le i \le k - 1$. The scheme is computationally efficient but somewhat wasteful in bandwidth as $k \cdot \log_2 N$ bits are needed to encrypt a k-bit message. Several proposals were made to address this issue.

A first attempt is due to Blum and Goldwasser [8]. They achieve a better ciphertext expansion: the ciphertext has the same length as the plaintext plus an integer of the size of modulus. The scheme is proved semantically secure assuming the unpredictability of the output of the Blum-Blum-Shub's pseudorandom generator [6,7] which resides on the factorisation hardness assumption. Details about this scheme can be found in [21].

Another direction, put forward by Benaloh and Fischer [12,5], is to use a k-bit prime r such that $r \mid p - 1$, $r^2 \nmid p - 1$ and $r \nmid q - 1$. The scheme also requires $y \in \mathbb{Z}_N^*$ such that $y^{\phi(N)/r} \not\equiv 1 \pmod{N}$, where $\phi(N) = (p - 1)(q - 1)$ denotes Euler's totient function. A k-bit message m (with $m < r$) is encrypted as $c = y^m x^r \bmod N$, where $x \in_R \mathbb{Z}_N^*$. It is recovered by searching over the entire message space, $[0, r) \subseteq \{0, 1\}^k$, for the element m satisfying $(y^{\phi(N)/r})^m \equiv c^{\phi(N)/r} \pmod{N}$. The scheme is shown to be secure under the *prime-residuosity assumption* (which generalizes the quadratic residuosity assumption). With the Benaloh-Fischer cryptosystem, the ciphertext corresponding to a k-bit message is short but the decryption process is now demanding. In practice, the scheme is therefore limited to small values of k, say $k < 40$.

The Benaloh-Fischer cryptosystem was subsequently extended by Naccache and Stern [39]. They observe that the decryption can be sped up by rather considering a product of small (odd) primes $R = \prod_i r_i$ such that $r_i \mid \phi(N)$ but ${r_i}^2 \nmid \phi(N)$ for each prime r_i. Given a ciphertext, the plaintext m is reconstructed from $m_i := m \bmod r_i$ through Chinese remaindering. The advantage is that each m_i is searched in the subspace $[0, r_i)$ instead of the entire message space. A variant of this technique was used by Groth [22].

Other generalizations and extensions of the Goldwasser-Micali cryptosystem but without formal security analysis can be found in [53,32,44]. More recently, Monnerat and Vaudenay developed applications using the more general theory of characters [38,37], specifically with characters of order ≤ 4. Related cryptosystems are described in [49,48]. Yet another, different approach was proposed by Okamoto and Uchiyama [42], who suggested to use moduli of the form $N = p^2q$. This allows encrypting messages of size up to $\log_2 p$ bits. This was later extended by Paillier [43] to the setting $N = p^2q^2$. In 2005, Boneh, Goh and Nissim [10] showed an additively homomorphic system also supporting one multiplication.

A useful application of additive homomorphic encryption schemes resides in the construction of *lossy trapdoor functions* (or LTDFs in short). These functions, as introduced by Peikert and Waters [45], are function families wherein injective functions are computationally indistinguishable from *lossy* functions, which lose many bits of information about their input. LTDFs have proved to be very powerful and versatile in the cryptographer's toolbox. They notably imply chosen-ciphertext-secure public-key encryption [45], deterministic encryption [2,9] as well as cryptosystems that retain some security in the absence of reliable randomness [3] or in the presence of selective-opening adversaries [4].

Our Contributions

New Homomorphic Cryptosystem. We suggest an improvement of the original Goldwasser-Micali cryptosystem. It can be seen as a follow-up of the earlier works due to Benaloh and Fischer [12] and Naccache and Stern [39]. Before discussing it, we quote from [39]:

> *"Although the question of devising new public-key cryptosystems appears much more difficult [...] we feel that research in this direction is still in order: simple yet efficient constructions may have been overlooked."*

It is striking that the generalized cryptosystem in this paper was not already proposed because, as will become apparent (cf. Section 3), it turns out to be a very natural generalization. Our approach consists in considering n^{th}-power residues modulo N with $n = 2^k$ (the Goldwasser-Micali system corresponds to the case $k = 1$). This presents certain advantages. First, the resulting cryptosystem is bandwidth-efficient. Only $\log_2 N$ bits are needed for encrypting a k-bit message in typical applications (e.g., using the KEM/DEM paradigm). Second, the decryption process is very fast, even faster than in the Naccache-Stern cryptosystem. Searches are no longer needed (not even in smaller subspaces) in the decryption algorithm as plaintext messages can be recovered bit by bit. Third, the underlying complexity assumption is similar. The proposed cryptosystem is shown to be secure under the quadratic residuosity assumption for RSA moduli $N = pq$ such that $p, q \equiv 1 \pmod{2^k}$.

We also note that, similarly to the Goldwasser-Micali cryptosystem, our generalized cryptosystem enjoys an additive property known as *homomorphic encryption*. If c_1 and c_2 denote two ciphertexts corresponding to k-bit plaintexts m_1 and m_2, respectively, then $c_1 \cdot c_2 \pmod N$ is an encryption of the message $m_1 + m_2 \pmod{2^k}$. This reveals useful in several applications like voting schemes. An interesting extension would be to thresholdize it as was done in [29].

As another useful property, the new scheme also inherits the selective opening security[1] [16,4] of the Goldwasser-Micali system (in the sense of a simulation-based definition given in [4]). We actually prove its semantic security by showing that its public key is indistinguishable from a so-called *lossy* key for which encryptions reveal nothing about the encrypted message.

We thus believe our system to provide an interesting competitor to Paillier's cryptosystem for certain applications. As a salient example, we show that it provides a dramatically improved lossy trapdoor function based on a quadratic residuosity assumption.

New Efficient Lossy Trapdoor Functions. The initial LTDF realizations [45] were based on the Decision Diffie-Hellman and Learning-with-Error [47] assumptions. More efficient examples based on the Composite Residuosity assumption were given in [9,17,18] while Kiltz *et al.* [30] showed that the RSA permutation provides a lossy function. Under the quadratic residuosity assumption, three distinct constructions were put forth in [23,17,18,51]. Those of Freeman *et al.* [17,18] and of Wee [51] must be used in combination with the results of Mol and Yilek [36] as they only lose single bits of information about the input. Hemenway and Ostrovsky [23] suggested a more efficient realization, of which Wee's framework [51] is a generalization. While their QR-based LTDF has found applications in the design of deterministic encryption schemes [11], it is conceptually very similar to the Peikert-Waters matrix-based schemes and suffers from similarly large outputs and descriptions.

We show that our variant of the Goldwasser-Micali cryptosystem drastically improves the efficiency of the Hemenway-Ostrovsky LTDF. Specifically, it reduces the length of the output (resp. the description of the function) by a factor of $O(\kappa)$ (resp. $O(\kappa^2)$), where κ is the security parameter. By appropriately selecting the parameters, we obtain evaluation keys and outputs consisting of a constant number of $\mathbb{Z}_N^*$ elements (and thus $O(\kappa)$ bits, instead of $O(\kappa^2)$ or $O(\kappa^3)$ as in the previous constructions). We thus obtain a QR-based LTDF, whose efficiency is competitive with Paillier-based realizations [9,17,18]. These improvements carry over to the deterministic encryption setting, when the Hemenway-Ostrovsky LTDF is used as a building block of the Brakerski-Segev system [11].

2 Background

We review some useful background and fix the notation. In particular, we define the n-th power residue symbol. We refer the reader to [25,50,52] for further details on (quadratic) residuosity. More information about encryption schemes can be found in textbooks in cryptography; e.g. [21,28].

[1] This notion refers to an attack scenario where the adversary is given t encryptions of possibly correlated messages, opens $t/2$ out of these (and thereby obtains the messages *and* encryption coins) before attempting to harm the security of remaining ciphertexts.

2.1 n^{th}-Power Residues

Let $N \in \mathbb{N}$. For each integer $n \geq 2$, we define $(\mathbb{Z}_N^*)^n = \{x^n \mid x \in \mathbb{Z}_N^*\}$ the set of n^{th}*-power residues modulo* N. If the relation $a = x^n$ has no solution in $\mathbb{Z}_N^*$ then a is called a n^{th}*-power non-residue modulo* N. Suppose that p is an odd prime. For any integer a with $\gcd(a,p) = 1$, it is easily verified that a is a n^{th}-power residue modulo p if and only if

$$a^{\frac{p-1}{\gcd(n,p-1)}} \equiv 1 \pmod{p} \ .$$

When $n = 2$ (and so $\gcd(n, p-1) = 2$), this is known as Euler's criterion. It allows one to distinguish quadratic residues from quadratic non-residues. This defines the *Legendre symbol.* There are several ways to generalize the Legendre symbol (see [33]). In this paper, we consider the n-th power residue symbol for a divisor n of $(p-1)$, as presented in [52, Definition 1.6.21].

Definition 1. *Let p be an odd prime and let $n \geq 2$ such that $n \mid p-1$. Then the symbol*

$$\left(\frac{a}{p}\right)_n = a^{\frac{p-1}{n}} \operatorname{mods} p$$

*is called the n-*th power residue symbol modulo p*, where $a^{\frac{p-1}{n}} \operatorname{mods} p$ represents the absolute smallest residue of $a^{\frac{p-1}{n}}$ modulo p (namely, the complete set of absolute smallest residues are: $-(p-1)/2, \ldots, -1, 0, 1, \ldots, (p-1)/2$).*

2.2 Quadratic Residuosity

Let $N = pq$ be the product of two (odd) primes p and q. For an integer a co-prime to N, the *Jacobi symbol* is the product of the corresponding Legendre symbols, namely $\left(\frac{a}{N}\right) = \left(\frac{a}{p}\right)\left(\frac{a}{q}\right)$. This gives rise to the multiplicative group $\mathbb{J}_N$ of integers whose Jacobi symbol is 1, $\mathbb{J}_N = \{a \in \mathbb{Z}_N^* \mid \left(\frac{a}{N}\right)_2 = 1\}$. A relevant subset of $\mathbb{J}_N$ is the set of quadratic residues modulo N, $\mathbb{QR}_N = \{a \in \mathbb{Z}_N^* \mid \left(\frac{a}{p}\right) = \left(\frac{a}{q}\right)_2 = 1\}$.

The *Quadratic Residuosity* (QR) assumption says that, given a random element $a \in \mathbb{J}_N$, it is hard to decide whether $a \in \mathbb{QR}_N$ if the prime factors of N are unknown. To emphasize that this should hold for moduli $N = pq$ with $p, q \equiv 1 \pmod{2^k}$, we will refer to it as the k-QR *assumption.* Formally, we have:

Definition 2 (Quadratic Residuosity Assumption). *Let* RSAGen *be a probabilistic algorithm which, given a security parameter κ, outputs primes p, q such that $p \equiv q \equiv 1 \pmod{2^k}$, and their product $N = pq$. The* Quadratic Residuosity (QR) *assumption asserts that the function $\mathbf{Adv}_{\mathcal{D}}^{\mathsf{QR}}(1^\kappa)$, defined as the distance*

$$\left|\Pr[\mathcal{D}(x, N) = 1 \mid x \xleftarrow{R} \mathbb{QR}_N] - \Pr[\mathcal{D}(x, N) = 1 \mid x \xleftarrow{R} \mathbb{J}_N \setminus \mathbb{QR}_N]\right|$$

is negligible for any probabilistic polynomial-time distinguisher $\mathcal{D}$; the probabilities are taken over the experiment of running $(N, p, q) \leftarrow \mathsf{RSAGen}(1^\kappa)$ and choosing at random $x \in \mathbb{QR}_N$ and $x \in \mathbb{J}_N \setminus \mathbb{QR}_N$.

3 A New Public-Key Encryption Scheme

We generalize the Goldwasser-Micali cryptosystem so that it can efficiently support the encryption of larger messages while remaining additively homomorphic.

3.1 Description

The setting is basically the same as for the Goldwasser-Micali cryptosystem. The only additional requirement is that primes p and q are chosen congruent to 1 modulo 2^k where k denotes the bit-size of the messages being encrypted.

In more detail, our encryption scheme is the tuple ($\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt}$) defined as follows.

$\mathsf{KeyGen}(1^\kappa)$ Given a security parameter κ, KeyGen defines an integer $k \geq 1$, randomly generates primes $p, q \equiv 1 \pmod{2^k}$, and sets $N = pq$. It also picks $y \in \mathbb{J}_N \setminus \mathbb{QR}_N$. The public and private keys are $pk = \{N, y, k\}$ and $sk = \{p\}$.

$\mathsf{Encrypt}(pk, m)$ Let $\mathcal{M} = \{0,1\}^k$. To encrypt a message $m \in \mathcal{M}$ (seen as an integer in $\{0, \ldots, 2^k - 1\}$), $\mathsf{Encrypt}$ picks a random $x \in \mathbb{Z}_N^*$ and returns the ciphertext $c = y^m\, x^{2^k} \bmod N$.

$\mathsf{Decrypt}(sk, c)$ Given $c \in \mathbb{Z}_N^*$ and the private key $sk = \{p\}$, the algorithm first computes $z = \left(\frac{c}{p}\right)_{2^k}$ and then finds $m \in \{0, \ldots, 2^k - 1\}$ such that the relation

$$\left[\left(\frac{y}{p}\right)_{2^k}\right]^m = z \pmod{\text{s}\ p}$$

holds. An efficient method to recover message m in a bit-by-bit fashion is detailed in the next section (§ 3.2).

The correctness is easily verified by observing that $\alpha := \left(\frac{y}{p}\right)_{2^k}$ has order 2^k as an element in $\mathbb{Z}_p^*$. Indeed, letting $n = \mathrm{ord}_p(\alpha)$ the order of α, we have $n \mid 2^k$ since, by definition, $\alpha \equiv y^{\frac{p-1}{2^k}} \pmod{p}$. But n cannot be equal to $2^{k'}$ for some $k' < k$ because $\alpha^{2^{k'}} \equiv 1 \pmod{p}$ would imply $y^{\frac{p-1}{2}} \equiv 1 \pmod{p}$, which contradicts the assumption that $y \in \mathbb{J}_N \setminus \mathbb{QR}_N \iff \left(\frac{y}{p}\right) = \left(\frac{y}{q}\right) = -1$. The decryption algorithm recovers the unique $m \in \{0, \ldots, 2^k - 1\}$ such that $\alpha^m \equiv z \pmod{p}$.

Remark 1. We notice that the case $k = 1$ corresponds to the Goldwasser-Micali cryptosystem. Indeed, the 2^k-th power residue symbol is then the classical Legendre symbol and the assumption $p, q \equiv 1 \pmod{2^k}$ is trivially verified.

3.2 Fast Decryption

At first glance, from the above description, it seems that the decryption process amounts to a search through the entire message space $\{0,1\}^k$, similarly to some earlier cryptosystems. But we can do better. One of the main advantages of the proposed cryptosystem is that it provides an efficient way to recover the message.

Hence, it remains practical, even for large values of k. The decryption algorithm proceeds similarly to the Pohlig-Hellman algorithm [46] and is detailed below.

Algorithm 1. Decryption algorithm

Input: Ciphertext c, private key p (and public-key elements y and k)
Output: Plaintext $m = (m_{k-1}, \ldots, m_0)_2$

```
1: $m \leftarrow 0$; $B \leftarrow 1$
2: for $i = 1$ to $k$ do
3:     $z \leftarrow \left(\frac{c}{p}\right)_{2^i}$; $t \leftarrow \left(\frac{y}{p}\right)_{2^i}^m \operatorname{mods} p$
4:     if $(t \neq z)$ then $m \leftarrow m + B$
5:     $B \leftarrow 2B$
6: end for
7: return $m$
```

The message $m \in \{0,1\}^k$ is viewed as a k-bit integer given by its binary expansion $m = \sum_{i=0}^{k-1} m_i \, 2^i$, with $m_i \in \{0,1\}$. Given $c = y^m x^{2^k} \bmod N$, we have

$$\left(\frac{c}{p}\right)_{2^i} = \left(\frac{y^m x^{2^k}}{p}\right)_{2^i} = \left(\frac{y^{\sum_{j=0}^{i-1} m_j \, 2^j}}{p}\right)_{2^i} = \left(\frac{y}{p}\right)_{2^i}^{\sum_{j=0}^{i-1} m_j \, 2^j} \qquad (\operatorname{mods} p)$$

since $y^m x^{2^k} = y^{\sum_{j=0}^{i-1} m_j \, 2^j} \cdot \left(y^{\sum_{j=i}^{k-1} m_j \, 2^{j-i}} x^{2^{k-i}}\right)^{2^i}$, for $1 \leq i \leq k$. As a result, m can be recovered bit by bit using p, starting from the rightmost bit. The algorithm uses an accumulator B which contains the successive powers of 2.

3.3 Security Analysis

We prove that the scheme provides indistinguishable encryptions under the k-QR assumption. The case $k = 1$ corresponds to the Goldwasser-Micali cryptosystem and the standard Quadratic Residuosity assumption. So, we henceforth assume $k \geq 2$. In this case, since $p, q \equiv 1 \pmod{2^k}$, we know that $p, q \equiv 1 \pmod 4$ and $\left(\frac{-1}{p}\right) = \left(\frac{-1}{q}\right) = 1$. This implies that the square roots of an element in $\mathbb{QR}_N$ all have the same Jacobi symbol.

The k-QR assumption states that, without knowing the factorization of N, random elements of $\mathbb{QR}_N$ are computationally indistinguishable from random elements of $\mathbb{J}_N \setminus \mathbb{QR}_N$. Here, it will be convenient to consider a *gap* variant of the k-QR assumption. We chose the terminology "gap" (not to be confused with computational problems which have an easy decisional counterpart [41]) by analogy with certain lattice problems, where not every instance is a YES or NO instance since a gap exists between these.

Definition 3 (Gap 2^k-Residuosity Assumption). *Let $N = pq$ be the product of two large primes p and q with $p, q \equiv 1 \pmod{2^k}$. The* Gap 2^k-Residuosity ($\mathsf{Gap}\text{-}2^k\text{-}\mathsf{Res}$) *problem in $\mathbb{Z}_N^*$ is to distinguish the distribution of the following two sets given only $N = pq$:*

$$V_0 = \{x \in \mathbb{J}_N \setminus \mathbb{QR}_N\} \quad \text{and} \quad V_1 = \{y^{2^k} \bmod N \mid y \in \mathbb{Z}_N^*\} \ .$$

The Gap 2^k-Residuosity *assumption posits that the advantage $\mathbf{Adv}_{\mathcal{D}}^{\mathsf{Gap}\text{-}2^k\text{-}\mathsf{Res}}(1^\kappa)$ of any PPT distinguisher $\mathcal{D}$, defined as the distance*

$$\left|\Pr[\mathcal{D}(x, k, N) = 1 \mid x \overset{R}{\leftarrow} V_0] \ - \Pr[\mathcal{D}(x, k, N) = 1 \mid x \overset{R}{\leftarrow} V_1]\right|$$

where probabilities are taken over all coin tosses, is negligible.

The latter assumption was independently considered by Abdalla, Ben Hamouda and Pointcheval [1] who used it to provide tighter security proofs for forward-secure signatures. Our result thus implies that their tighter reduction holds under the more standard k-QR assumption.

In the above definition, we explicitly give k to the distinguisher and remark that this information should be of little help considering that it can always be guessed with non-negligible probability. Also observe that from $p, q \equiv 1 \pmod{2^k}$, it follows that $2^k \mid N - 1$.

Theorem 1 (k-$\mathsf{QR} \implies \mathsf{Gap}\text{-}2^k\text{-}\mathsf{Res}$). *The Quadratic Residuosity assumption implies the Gap 2^k-Residuosity assumption. More precisely, for any PPT distinguisher $\mathcal{B}_0$ against the former, there exists a* QR *distinguisher $\mathcal{B}_1$ with comparable running time and for which $\mathbf{Adv}_{\mathcal{B}_0}^{\mathsf{Gap}\text{-}2^k\text{-}\mathsf{Res}}(1^\kappa) \leq 4 \cdot k \cdot \mathbf{Adv}_{\mathcal{B}_1}^{k\text{-}\mathsf{QR}}(1^\kappa)$.*

Proof. The proof is given in the full version of the paper. □

It is not hard to see that the semantic security of the scheme is equivalent to the $\mathsf{Gap}\text{-}2^k\text{-}\mathsf{Res}$ assumption. We thus obtain the following theorem as a corollary.

Theorem 2. *The scheme is semantically secure under the k-*QR *assumption. More precisely, for any* $\mathsf{IND\text{-}CPA}$ *adversary $\mathcal{A}$, we have a k-*QR *distinguisher $\mathcal{B}$ such that $\mathbf{Adv}_{\mathcal{A}}^{\mathsf{ind\text{-}cpa}}(1^\kappa) \leq 4 \cdot k \cdot \mathbf{Adv}^{k\text{-}\mathsf{QR}}(\mathcal{B})$.*

Proof. The proof proceeds by simply changing the distribution of the public key. Under the $\mathsf{Gap}\text{-}2^k\text{-}\mathsf{Res}$ assumption, instead of picking y uniformly in $\mathbb{J}_N \setminus \mathbb{QR}_N$, we can choose it in the subgroup of 2^k-th residue without the adversary noticing. However, in this case, the ciphertext carries no information about the message and the $\mathsf{IND\text{-}CPA}$ adversary has no advantage. □

Interestingly, the proof of Theorem 2 implicitly shows that, like the original Goldwasser-Micali system, our scheme is a *lossy* encryption scheme [4] (*i.e.*, it admits an alternative distribution of public keys for which encryptions statistically hide the plaintext), which provides security guarantees against selective-opening attacks [16]. Moreover, for a lossy key (y, N), there exists an efficient algorithm that opens a given ciphertext c to any arbitrary plaintext m (by finding random coins that explain c as an encryption of m). It implies that our scheme satisfies the simulation-based definition [4] of selective-opening security.

4 Implementation and Performance

We detail here some implementation aspects. We explain how to select the parameters involved in the system set-up and key generation. Finally, we discuss the ciphertext expansion and give a comparison with previous schemes.

4.1 Parameter Selection

The key generation (cf. § 3.1) requires two primes p and q such that $p, q \equiv 1 \pmod{2^k}$ and an element $y \in \mathbb{J}_N \setminus \mathbb{QR}_N$, where $N = pq$. The condition $y \in \mathbb{J}_N \setminus \mathbb{QR}_N$ is equivalent to $\left(\frac{y}{p}\right) = \left(\frac{y}{q}\right) = -1$. So, we need to generate an element $y \in \mathbb{Z}_N^*$ such that *(i)* $y \bmod p$ is primitive in $\mathbb{Z}_p^*$, and *(ii)* $y \bmod q$ is primitive in $\mathbb{Z}_q^*$. Finding a primitive element modulo a prime number p is not difficult when the factorization of $p-1$ is known. Therefore, we suggest to select prime p as a *k-quasi-safe prime*, that is, $p = 2^k\, p' + 1$ for some prime p' (likewise for prime q, we take $q = 2^k\, q' + 1$ for some prime q'). An efficient algorithm for generating k-quasi-safe primes is discussed in [27, Section 4.2].

Consider now the primitive 2^k-th root of unity $\zeta_{2^k} = e^{2i\pi/2^k}$ with $i = \sqrt{-1}$. It generates a cyclic group of order 2^k under multiplication. In our case, the key observation is that, when p is 2^k-quasi-safe prime, if y is a square modulo p then $\zeta_{2^k}\, y$ is not. Indeed, we have

$$\left(\frac{\zeta_{2^k}\, y}{p}\right) = \left(\frac{\zeta_{2^k}}{p}\right)\left(\frac{y}{p}\right) \equiv {\zeta_{2^k}}^{\frac{p-1}{2}}\left(\frac{y}{p}\right) \equiv (e^{i\pi})^{p'}\left(\frac{y}{p}\right) = -\left(\frac{y}{p}\right) \pmod{p}$$

since p' is odd. This leads to the following algorithm.

Algorithm 2. Generation of y

Input: Modulus $N = pq$ (with $p = 2^k\, p' + 1$ and $q = 2^k\, q' + 1$), primes p, q, p', q', and integer $k \geq 1$
Output: $y \in \mathbb{J}_N \setminus \mathbb{QR}_N$

```
1: Pick at random $y_p \in \mathbb{Z}_p^*$ and $y_q \in \mathbb{Z}_q^*$
2: if $\left(\frac{y_p}{p}\right) = 1$ then $y_p \leftarrow \zeta_{2^k}\, y_p \bmod p$
3: if $\left(\frac{y_q}{q}\right) = 1$ then $y_q \leftarrow \zeta_{2^k}\, y_q \bmod q$
4: Set $y \leftarrow y_p + p\big(p^{-1}(y_q - y_p) \bmod q\big)$
5: return $y$
```

The primes p and q are chosen so that $p, q \equiv 1 \pmod{2^k}$. Sharing common factors for $(p-1)$ and $(q-1)$ was used already in several other systems; see e.g. [19,34]. Letting r denote a common factor of $(p-1)$ and $(q-1)$, a baby-step giant-step approach developed by McKee and Pinch [35] can factor RSA modulus $N = pq$ in essentially $O(N^{1/4}/r)$ operations. In our case, we have $r = 2^k$. For security it is therefore necessary that $\frac{1}{4}\log_2 N - k > \kappa$, or equivalently,

$$k < \tfrac{1}{4}\log_2 N - \kappa$$

where κ is the security parameter.

A powerful LLL-based technique due to Coppersmith [13,14] also bounds the size of k to at most $\frac{1}{2}\min(\log_2 p, \log_2 q)$ bits as, otherwise, the factors of N would be revealed. Going beyond polynomial-time attacks, one should add an extra security margin to take into account exhaustive searches [40]. RSA moduli being balanced (*i.e.*, $\frac{1}{2}\min(\log_2 p, \log_2 q) = \frac{1}{4}\log_2 N$), we so end up with the same upper bound as for the McKee-Pinch's approach: $k < \frac{1}{4}\log_2 N - \kappa$.

In practice, this restriction on k is not a limitation because, as described in the next section, long messages can be encrypted using the KEM/DEM paradigm. For example, a specific parameter choice is $k = 128$ and $\log_2 N = 2048$.

4.2 Ciphertext Expansion

Hybrid encryption allows designing efficient asymmetric schemes, as suggested by Shoup in the ISO 18033-2 standard for public-key encryption [26]. An asymmetric cryptosystem is used to encrypt a secret key that is then used to encrypt the actual message. This is the so-called *KEM/DEM paradigm.*

The next table compares the ciphertext expansion in the encryption of k-bit messages for different generalized Goldwasser-Micali cryptosystems. Only cryptosystems with a formal security analysis are considered. Further, the value of k is assumed to be relatively small (e.g., 128 or 256) as the "message" being encrypted is typically a symmetric key (for example a 128- or 256-bit AES key) in a KEM/DEM construction.

Table 1. Ciphertext expansion in a typical encryption

Encryption scheme	Assumption	Ciphertext size
Goldwasser-Micali [20]	Quadratic Residuosity (QR)	$k \cdot \log_2 N$
Benaloh-Fisher [12]	Prime residuosity (PR)	$\lceil \frac{k}{\log_2 r} \rceil \cdot \log_2 N$
Naccache-Stern [39]	Prime residuosity (PR)	$\log_2 N$
Okamoto-Uchiyama [42]	p-subgroup	$\log_2 N$
Paillier [43]	N-th residuosity	$2\log_2 N$
This paper	Quadratic residuosity (k-QR)	$\log_2 N$

It appears that the Goldwasser-Micali cryptosystem has the higher ciphertext expansion but its semantic security relies on the standard quadratic residuosity assumption. The ciphertext expansion of Benaloh-Fischer cryptosystem is similar to that of Naccache-Stern cryptosystem for *small* messages; *i.e.*, when $k \leq \log_2 r$. For larger messages, the Naccache-Stern cryptosystem should be preferred. It also offers the further advantage of providing a faster decryption procedure. The same is true for the Okamoto-Uchiyama cryptosystem. The Paillier cryptosystem produces twice larger ciphertexts.

The encryption scheme proposed in this paper has the same ciphertext expansion as in the Naccache-Stern cryptosystem. Moreover, its decryption algorithm is fast (it is even faster than in the Naccache-Stern cryptosystem), requires less memory, and the security relies on a quadratic residuosity assumption.

5 More Efficient Lossy Trapdoor Functions from the *k*-Quadratic Residuosity Assumption

In this section, we show that our homomorphic cryptosystem allows constructing a lossy trapdoor function based on the k-QR assumption with much shorter outputs and keys than in previous QR-based examples.

In comparison with the function of Hemenway and Ostrovsky [23], it compresses function values by a factor of k when we work with a modulus $N = pq$ such that $p \equiv q \equiv 1 \pmod{2^k}$. Moreover, the size of the evaluation key is decreased by a factor of $O(k^2)$ while increasing the lossiness by $2k$ more bits. Finally, our inversion trapdoor has constant size, whereas [23] uses a trapdoor of size $O(n)$ to recover n-bit inputs. Our function also compares favorably with the QR-based function of Freeman *et al.* [17,18], which only loses a single bit.

In fact, by appropriately tuning our construction, we obtain the first QR-based lossy trapdoor function with short outputs and keys that loses many input bits. Among known lossy trapdoor functions based on traditional number-theoretic assumptions [45,9,17,18,30,23,36], this appears as a rare efficiency tradeoff. To the best of our knowledge, it has only been achieved under the Composite Residuosity assumption [9,17,18] so far.

Interestingly, our LTDF provides similar efficiency improvements to the QR-based deterministic encryption scheme of Brakerski and Segev [11], which also builds on the Hemenway-Ostrovsky LTDF. Note that the scheme of [11] is important in the deterministic encryption literature since it is one of the only known schemes providing security in the auxiliary input setting in the standard model.

5.1 Description and Security Analysis

We start by recalling the following definition.

Definition 4 ([45]). *Let $\kappa \in \mathbb{N}$ be a security parameter and $n : \mathbb{N} \to \mathbb{N}$, $\ell : \mathbb{N} \to \mathbb{R}$ be non-negative functions of κ. A collection of (n, ℓ)*-lossy trapdoor functions *(LTDF) is a tuple of efficient algorithms* $(\mathsf{InjGen}, \mathsf{LossyGen}, \mathsf{Eval}, \mathsf{Invert})$ *with the following specifications.*

- Sampling an injective function: *Given a security parameter κ, the randomized algorithm* $\mathsf{InjGen}(1^\kappa)$ *outputs the index ek of an injective function of the family and an inversion trapdoor t.*
- Sampling a lossy function: *Given a security parameter κ, the probabilistic algorithm* $\mathsf{LossyGen}(1^\kappa)$ *outputs the index ek of a lossy function.*
- Evaluation: *Given the index of a function ek —produced by either* InjGen *or* $\mathsf{LossyGen}$— *and an input $x \in \{0,1\}^n$, the evaluation algorithm* Eval *outputs $F_{ek}(x)$ such that:*
  - *If ek is an output of* InjGen*, then $F_{ek}(\cdot)$ is an injective function.*
  - *If ek was produced by* $\mathsf{LossyGen}$*, then $F_{ek}(\cdot)$ has image size $2^{n-\ell}$. In this case, the value $n - \ell$ is called* residual leakage.

– Inversion: *For any pair* (ek, t) *produced by* InjGen *and any input* $x \in \{0,1\}^n$, *the inversion algorithm* Invert *returns* $F_{ek}^{-1}(t, F_{ek}(x)) = x$.
– Security: *The two ensembles* $\{ek \mid (ek,t) \leftarrow \mathsf{InjGen}(1^\kappa)\}_{\kappa\in\mathbb{N}}$ *and* $\{ek \mid ek \leftarrow \mathsf{LossyGen}(1^\kappa)\}_{\kappa\in\mathbb{N}}$ *are computationally indistinguishable.*

Our construction goes as follows.

Sampling an injective function. Given a security parameter κ, let $\ell_N(\kappa)$ and $k(\kappa)$ be security parameters determined by κ. Let also $n(\kappa)$ be the desired input length. Algorithm InjGen defines $m = n/k$ (we assume that k divides n for simplicity) and conducts the following steps.

1. Generate a modulus $N = pq > 2^{\ell_N}$ such that $p = 2^k p' + 1$ and $q = 2^k q' + 1$ for primes p, q and odd co-prime integers p', q'. Choose $y \xleftarrow{R} \mathbb{J}_N \setminus \mathbb{QR}_N$.
2. For each $i \in \{1, \ldots, m\}$, pick h_i in the subgroup of order $p'q'$, by setting $h_i = g_i^{2^k} \bmod N$ for a randomly chosen $g_i \xleftarrow{R} \mathbb{Z}_N^*$.
3. Choose $r_1, \ldots, r_m \xleftarrow{R} \mathbb{Z}_{p'q'}$ and compute a matrix $Z = \big(Z_{i,j}\big)_{i,j\in\{1,\ldots,m\}}$ given by

$$Z = \begin{pmatrix} y^{z_{1,1}} \cdot h_1^{r_1} \bmod N & \ldots\ldots & y^{z_{1,m}} \cdot h_m^{r_1} \bmod N \\ \vdots & & \vdots \\ y^{z_{m,1}} \cdot h_1^{r_m} \bmod N & \ldots\ldots & y^{z_{m,m}} \cdot h_m^{r_m} \bmod N \end{pmatrix},$$

where $(z_{i,j})_{i,j\in\{1,\ldots,m\}}$ denotes the identity matrix.

The evaluation key is $ek := \big(N, (Z_{i,j})_{i,j\in\{1,\ldots,m\}}\big)$ and the trapdoor is $t := p$.

Sampling a lossy function. The process followed by $\mathsf{LossyGen}$ is identical to the above one but the matrix $(z_{i,j})_{i,j\in\{1,\ldots,m\}}$ is replaced by the all-zeroes $m \times m$ matrix.

Evaluation. Given $ek = \big(N, (Z_{i,j})_{i,j\in\{1,\ldots,m\}}\big)$, algorithm Eval parses the input $x \in \{0,1\}^n$ as a vector of k-bit blocks $\tilde{x} = (x_1, \ldots, x_m)$, with $x_i \in \mathbb{Z}_{2^k}$ for each i. Then, it computes and returns $\tilde{y} = (y_1, \ldots, y_m)$, with $y_j \in \mathbb{Z}_N^*$, where

$$\begin{aligned} \tilde{y} &= \Big(\prod_{i=1}^m Z_{i,1}{}^{x_i} \bmod N, \ldots, \prod_{i=1}^m Z_{i,m}{}^{x_i} \bmod N\Big) \\ &= \Big(y^{\sum_{i=1}^m z_{i,1}x_i} \cdot h_1{}^{\sum_{i=1}^m r_i x_i} \bmod N, \ldots, y^{\sum_{i=1}^m z_{i,m}x_i} \cdot h_m{}^{\sum_{i=1}^m r_i x_i} \bmod N\Big) \,. \end{aligned}$$

Inversion. Given $t = p$ and $\tilde{y} = (y_1, \ldots, y_m) \in \mathbb{Z}_N^m$, Invert applies the decryption algorithm of § 3.2 to each y_j, for $j = 1$ to m. Observe that when $(z_{ij})_{i,j\in\{1,\ldots,m\}}$ is the identity matrix, $\big(\frac{y_j}{p}\big)_{2^k} \equiv \Big[\big(\frac{y}{p}\big)_{2^k}\Big]^{x_j} \pmod p$. From the resulting vector of plaintexts $\tilde{x} = (x_1, \ldots, x_m) \in \mathbb{Z}_{2^k}^m$, it recovers the input $x \in \{0,1\}^n$.

The Hemenway-Ostrovsky construction of [23] is slightly different in that, as in the DDH-based construction of Peikert and Waters [45], the evaluation key includes a vector of the form $G = (g^{r_1}, \ldots, g^{r_m})^T$, where $g \in \mathbb{QR}_N$, and the trapdoor is $t = (\log_g(h_1), \ldots, \log_g(h_m))$. In their scheme, the evaluation algorithm additionally computes $\prod_{i=1}^{m} (g^{r_i})^{x_i}$ while the inversion algorithm does not use the factorization of N but rather performs a coordinate-wise ElGamal decryption. Here, explicitly using the factorization of N in the inversion algorithm makes it possible to process k-bit blocks at once.

Theorem 3. *The above construction is a $(n(\kappa), n(\kappa) - \log_2(p'q'))$-LTDF if the k-*QR *assumption holds.*

Proof. The proof is given in the full version of the paper. □

It is worth noting that, with $N = pq$ such that $p \equiv q \equiv 1 \pmod{2^k}$, a side effect of working in the subgroup of odd order is an improved lossiness. Indeed, we lose $n - \log_2(p'q')$ bits in comparison with $n - \log_2 \phi(N)$ in [23].

Using the techniques of Peikert and Waters [45], it is easy to construct an equally efficient all-but-one trapdoor function providing the same amount of lossiness under the QR assumption. A difference is that, in order to enable inversion, the resulting all-but-one function handles $k/2$ bits (instead of k) in each chunk. The details are given in the full version of the paper.

More importantly, the dimension m of the matrix and the output vector can be reduced to a fairly small constant, as illustrated below.

5.2 Efficiency

Here, we consider chosen-ciphertext security as the targeted application.

By combining the lossy and all-but-one trapdoor function, a CCA-secure encryption scheme can be obtained using the construction of [45]. We argue that $m = O(1)$ suffices for this purpose. Recall that the scheme of [45] combines a pair-wise independent hash function $H : \{0,1\}^n \to \{0,1\}^\tau$, an (n, ℓ)-lossy function and an (n, ℓ')-all-but-one function such that $\ell + \ell' \geq n + \nu$ and $\tau \geq \nu - 2\log_2(1/\varepsilon)$, for some $\nu \in \omega(\log n)$ and where ε is the statistical distance in the modified Leftover Hash Lemma used in [15]. If we choose $\varepsilon \approx 2^{-\kappa}$ and $\tau = k$ in order to encrypt k-bit messages, we can set $\nu = k + 2\kappa$. Setting $\ell = \ell' = n - \log_2(p'q')$, the constraint $\ell + \ell' \geq n + \nu$ translates into $n - 2\log_2(p'q') \geq \nu$. If we set $k = \frac{1}{4}\log_2 N - \kappa$, we have $\log_2(p'q') = \log_2 \phi(N) - 2k \approx 4(k+\kappa) - 2k = 2k + 4\kappa$, which yields $n \geq 3k + 6\kappa$. If $k > \kappa$, it is sufficient to set $n \geq 9k$. If we take into account the fact that our all-but-one function processes blocks of $k/2$ bits, we find that $m = 2n/k = 18$ suffices here.

As it turns out, when the Peikert-Waters construction [45, § 4.3] of CCA-secure encryption is instantiated with our lossy and all-but-one trapdoor functions, it only requires a constant number of exponentiations while retaining constant-size public keys and ciphertexts.

With the exception of [24] (which relies on a weaker assumption), to the best of our knowledge, it yields the only known CCA-secure QR-based cryptosystem combining the aforementioned efficiency properties. Up to now, the most efficient chosen-ciphertext-secure cryptosystem strictly based on the QR assumption was the one of Kiltz *et al.*[31], where $O(\kappa)$ exponentiations are needed to encrypt and the public key contains $O(\kappa)$ group elements. On the other hand, our construction requires more specific moduli than [31].

6 Conclusion

This paper introduced a new generalization of the Goldwasser-Micali cryptosystem. The so-obtained cryptosystems are shown to be secure under the quadratic residuosity assumption. Further, they enjoy a number of useful features including fast decryption, optimal ciphertext expansion, and homomorphic property. We believe that our proposal is the most natural yet efficient generalization of the Goldwasser-Micali cryptosystem. It keeps the nice attributes and properties of the original scheme while improving the overall performance.

When applied to the Peikert-Waters framework for building lossy trapdoor functions, it yields a practical construction based on quadratic residuosity, with companion deterministic encryption scheme and CCA-secure cryptosystem.

Acknowledgments. We are grateful to the anonymous reviewers for EUROCRYPT 2013 for their useful comments.

References

1. Abdalla, M., Ben Hamouda, F., Pointcheval, D.: Tighter reductions for forward-secure signature schemes. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 292–311. Springer, Heidelberg (2013)
2. Bellare, M., Boldyreva, A., O'Neill, A.: Deterministic and efficiently searchable encryption. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 535–552. Springer, Heidelberg (2007)
3. Bellare, M., Brakerski, Z., Naor, M., Ristenpart, T., Segev, G., Shacham, H., Yilek, S.: Hedged public-key encryption: How to protect against bad randomness. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 232–249. Springer, Heidelberg (2009)
4. Bellare, M., Hofheinz, D., Yilek, S.: Possibility and impossibility results for encryption and commitment secure under selective opening. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 1–35. Springer, Heidelberg (2009)
5. Benaloh, J.D.C.: Verifiable Secret-Ballot Elections. PhD thesis, Yale University, New Haven, CT, USA (1987)
6. Blum, L., Blum, M., Shub, M.: Comparison of two pseudo-random number generators. In: Chaum, D., Rivest, R.L., Sherman, A.T. (eds.) Advances in Cryptology: Proceedings of CRYPTO 1982, pp. 61–78. Plenum Press (1983)
7. Blum, L., Blum, M., Shub, M.: A simple unpredictable pseudo-random number generator. SIAM J. Comput. 15(2), 363–383 (1986)
8. Blum, M., Goldwasser, S.: An efficient probabilistic public-key encryption scheme which hides all partial information. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 289–299. Springer, Heidelberg (1985)
9. Boldyreva, A., Fehr, S., O'Neill, A.: On notions of security for deterministic encryption, and efficient constructions without random oracles. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 335–359. Springer, Heidelberg (2008)

10. Boneh, D., Goh, E.-J., Nissim, K.: Evaluating 2-DNF formulas on ciphertexts. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 325–341. Springer, Heidelberg (2005)
11. Brakerski, Z., Segev, G.: Better security for deterministic public-key encryption: The auxiliary-input setting. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 543–560. Springer, Heidelberg (2011)
12. Cohen, J.D., Fischer, M.J.: A robust and verifiable cryptographically secure election scheme. In: 26th Annual Symposium on Foundations of Computer Science (FOCS 1985), pp. 372–382. IEEE Computer Society (1985)
13. Coppersmith, D.: Finding a small root of a bivariate integer equation; factoring with high bits known. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 178–189. Springer, Heidelberg (1996)
14. Coppersmith, D.: Small solutions to polynomial equations, and low exponent RSA vulnerabilities. J. Cryptology 10(4), 233–260 (1997)
15. Dodis, Y., Reyzin, L., Smith, A.: Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 523–540. Springer, Heidelberg (2004)
16. Dwork, C., Naor, M., Reingold, O., Stockmeyer, L.: Magic functions. Journal of the ACM 50(6), 852–921 (2003)
17. Freeman, D.M., Goldreich, O., Kiltz, E., Rosen, A., Segev, G.: More constructions of lossy and correlation-secure trapdoor functions. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 279–295. Springer, Heidelberg (2010)
18. Freeman, D.M., Goldreich, O., Kiltz, E., Rosen, A., Segev, G.: More constructions of lossy and correlation-secure trapdoor functions. J. Cryptology (2012)
19. Girault, M.: An identity-based identification scheme based on discrete logarithms modulo a composite number. In: Damgård, I.B. (ed.) EUROCRYPT 1990. LNCS, vol. 473, pp. 481–486. Springer, Heidelberg (1991)
20. Goldwasser, S., Micali, S.: Probabilistic encryption. J. Comput. Syst. Sci. 28(2), 270–299 (1984)
21. Golreich, O.: Foundations of Cryptography, vol. II. Cambridge University Press (2004)
22. Groth, J.: Cryptography in subgroups of Z_n^*. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 50–65. Springer, Heidelberg (2005)
23. Hemenway, B., Ostrovsky, R.: Lossy trapdoor functions from smooth homomorphic hash proof systems. In: Electronic Colloquium on Computational Complexity, ECCC (2009)
24. Hofheinz, D., Kiltz, E.: Practical chosen ciphertext secure encryption from factoring. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 313–332. Springer, Heidelberg (2009)
25. Ireland, K., Rosen, M.: *A Classical Introduction to Modern Number Theory*, 2nd edn. Graduate Texts in Mathematics, vol. 84. Springer (1990)
26. ISO/IEC 18033-2. Information technology – Security techniques – Encryption algorithms – Part 2: Asymmetric ciphers. International Organization for Standardization (May 2006)
27. Joye, M., Paillier, P.: Fast generation of prime numbers on portable devices: An update. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 160–173. Springer, Heidelberg (2006)
28. Katz, J., Lindell, Y.: Introduction to Modern Cryptography. CRC Press (2007)
29. Katz, J., Yung, M.: Threshold cryptosystems based on factoring. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 192–205. Springer, Heidelberg (2002)

30. Kiltz, E., O'Neill, A., Smith, A.: Instantiability of RSA-OAEP under chosen-plaintext attack. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 295–313. Springer, Heidelberg (2010)
31. Kiltz, E., Pietrzak, K., Stam, M., Yung, M.: A new randomness extraction paradigm for hybrid encryption. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 590–609. Springer, Heidelberg (2009)
32. Kurosawa, K., Katayama, Y., Ogata, W., Tsujii, S.: General public key residue cryptosystems and mental poker protocols. In: Damgård, I.B. (ed.) EUROCRYPT 1990. LNCS, vol. 473, pp. 374–388. Springer, Heidelberg (1991)
33. Lemmermeyer, F.: Reciprocity Laws. Springer Monographs in Mathematics. Springer (2000)
34. Lim, C.H., Lee, P.J.: Security and performance of server-aided RSA computation protocols. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 70–83. Springer, Heidelberg (1995)
35. McKee, J., Pinch, R.: Further attacks on server-aided RSA cryptosystems (1998) (unpublished manuscript)
36. Mol, P., Yilek, S.: Chosen-ciphertext security from slightly lossy trapdoor functions. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 296–311. Springer, Heidelberg (2010)
37. Monnerat, J., Vaudenay, S.: Generic homomorphic undeniable signatures. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 354–371. Springer, Heidelberg (2004)
38. Monnerat, J., Vaudenay, S.: Undeniable signatures based on characters: How to sign with one bit. In: Bao, F., Deng, R., Zhou, J. (eds.) PKC 2004. LNCS, vol. 2947, pp. 69–85. Springer, Heidelberg (2004)
39. Naccache, D., Stern, J.: A new public key cryptosystem based on higher residues. In: ACM Conference on Computer and Communications Security (CCS 1998), pp. 59–66. ACM Press (1998)
40. Nguyen, P.Q.: Public-key cryptanalysis. In: Luengo, I. (ed.) Recent Trends in Cryptography, Contemporary Mathematics. AMS–RSME (2009)
41. Okamoto, T., Pointcheval, D.: The gap-problems: A new class of problems for the security of cryptographic schemes. In: Kim, K.-C. (ed.) PKC 2001. LNCS, vol. 1992, pp. 308–318. Springer, Heidelberg (2001)
42. Okamoto, T., Uchiyama, S.: A new public-key cryptosystem as secure as factoring. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 308–318. Springer, Heidelberg (1998)
43. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999)
44. Park, S.J., Lee, B.Y., Won, D.H.: A probabilistic encryption using very high residuosity and its applications. In: Global Telecommunications Conference (GLOBECOM 1995), pp. 1179–1182. IEEE Press (1995)
45. Peikert, C., Waters, B.: Lossy trapdoor functions and their applications. In: Dwork, C. (ed.) 40th Annual ACM Symposium on Theory of Computing (STOC 2008), pp. 187–196. ACM Press (2008)
46. Pohlig, S.H., Hellman, M.E.: An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance. IEEE Tran. Inf. Theory 24(1), 106–110 (1978)
47. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: 37th Annual ACM Symposium on Theory of Computing (STOC 2005), pp. 84–93. ACM Press (2005)

48. Scheidler, R.: A public-key cryptosystem using purely cubic fields. J. Cryptology 11(2), 109–124 (1998)
49. Scheidler, R., Williams, H.C.: A public-key cryptosystem utilizing cyclotomic fields. Des. Codes Cryptography 6(2), 117–131 (1995)
50. Shoup, V.: A Computational Introduction to Number Theory and Algebra, 2nd edn. Cambridge University Press (2010)
51. Wee, H.: Dual projective hashing and its applications — lossy trapdoor functions and more. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 246–262. Springer, Heidelberg (2012)
52. Yan, S.Y.: Number Theory for Computing, 2nd edn. Springer (2002)
53. Zheng, Y., Matsumoto, T., Imai, H.: Residuosity problem and its applications to cryptography. Trans. IEICE E-71(8), 759–767 (1988)

Deterministic Public-Key Encryption for Adaptively Chosen Plaintext Distributions[⋆]

Ananth Raghunathan[1,⋆⋆], Gil Segev[1,⋆⋆⋆], and Salil Vadhan[2,†]

[1] Computer Science Department
Stanford University, Stanford, CA 94305, USA
{ananthr,segev}@stanford.edu

[2] School of Engineering and Applied Sciences
& Center for Research on Computation and Society,
Harvard University, Cambridge, MA 02138, USA
salil@seas.harvard.edu

Abstract. Bellare, Boldyreva, and O'Neill (CRYPTO '07) initiated the study of deterministic public-key encryption as an alternative in scenarios where randomized encryption has inherent drawbacks. The resulting line of research has so far guaranteed security only for adversarially-chosen plaintext distributions that are *independent* of the public key used by the scheme. In most scenarios, however, it is typically not realistic to assume that adversaries do not take the public key into account when attacking a scheme.

We show that it is possible to guarantee meaningful security even for plaintext distributions that depend on the public key. We extend the previously proposed notions of security, allowing adversaries to *adaptively* choose plaintext distributions *after* seeing the public key, in an *interactive* manner. The only restrictions we make are that: (1) plaintext distributions are unpredictable (as is essential in deterministic public-key encryption), and (2) the number of plaintext distributions from which each adversary is allowed to adaptively choose is upper bounded by 2^p, where p can be any predetermined polynomial in the security parameter. For example, with $p = 0$ we capture plaintext distributions that are independent of the public key, and with $p = O(s \log s)$ we capture, in particular, all plaintext distributions that are samplable by circuits of size s.

Within our framework we present both constructions in the random-oracle model based on any public-key encryption scheme, and constructions in the standard model based on lossy trapdoor functions (thus, based on a variety of number-theoretic assumptions). Previously known constructions heavily relied on the independence between the plaintext

[⋆] Due to space limitations the reader is referred to the full version [19].

[⋆⋆] Part of the work was done at Microsoft Research Silicon Valley.

[⋆⋆⋆] Part of the work was done while the author was a Postdoctoral Researcher at Microsoft Research Silicon Valley.

[†] Supported in part by NSF grant CCF-1116616. Work done primarily while on leave as a Visiting Researcher at Microsoft Research Silicon Valley and as a Visiting Scholar at Stanford University.

distributions and the public key for the purposes of randomness extraction. In our setting, however, randomness extraction becomes significantly more challenging once the plaintext distributions and the public key are no longer independent. Our approach is inspired by research on randomness extraction from seed-dependent distributions. Underlying our approach is a new generalization of a method for such randomness extraction, originally introduced by Trevisan and Vadhan (FOCS '00) and Dodis (PhD Thesis, MIT, '00).

1 Introduction

Deterministic public-key encryption was introduced by Bellare, Boldyreva, and O'Neill [1] as an alternative in scenarios where randomized encryption has inherent drawbacks. For example, ciphertexts that are produced by a randomized encryption algorithm are not length preserving (i.e., may be longer than their corresponding plaintexts), and are in general not efficient searchable – two properties that are problematic in many applications involving massive amounts of data. In addition, the security guarantees provided by randomized public-key encryption schemes are typically highly dependent on the assumption that fresh and essentially uniform random bits are available – which may not always be a valid assumption.

When using a deterministic encryption algorithm, however, the full-fledged notion of semantic security [14] is out of reach. In this light, Bellare et al. initiated the study of formalizing other strong and meaningful notions of security for deterministic public-key encryption, and quite a significant amount of work has been devoted to proposing various such notions and constructing schemes satisfying them [1,3,4,2,7,13,17,23]. Aiming to obtain as-strong-as-possible notions of security, this recent line of research has successfully shown that a natural variant of the notion of semantic security can be guaranteed even when using a deterministic encryption algorithm, as long as plaintexts are: (1) somewhat *unpredictable*, and (2) *independent* of the public key used by the scheme.

Plaintext Unpredictability. When using a deterministic encryption algorithm, essentially no meaningful notion of security can be satisfied when plaintexts are distributed over a small (e.g. polynomial-sized) set. In such a case, an adversary who is given a public key pk and an encryption c of some plaintext m under the public key pk can simply encrypt all possible plaintexts,[1] compare each of them to the given ciphertext c, and thus recover the plaintext m. Therefore, when formalizing a notion of security for deterministic public-key encryption, it is indeed essential to focus on security for unpredictable plaintext distributions.[2]

[1] More generally, an adversary can encrypt all plaintexts that occurs with at least some non-negligible probability.

[2] Unpredictable plaintext distributions do occur in some natural roles. A prime example is when using a public-key encryption scheme as a key-encapsulation mechanism that encrypts a uniformly distributed key k for a symmetric-key primitive.

Key-independent Plaintext Distributions. Even when dealing with highly unpredictable plaintext distributions, some restrictions should be made on their relation to the public key. Consider, for example, the uniform distribution over plaintexts m subject to the restriction that the first bit of m and the first bit of $c = \mathsf{Enc}_{pk}(m)$ are equal.[3] More generally, by constructing plaintext distributions that depends on the public key, adversaries can use any *deterministic* encryption algorithm as a *subliminal channel* that leaks much more information on the plaintexts than what any meaningful notion of security should allow.

This Paper. For preventing adversaries from exploiting deterministic encryption algorithms as subliminal channels, research on deterministic public-key encryption has so far guaranteed security only for plaintexts distributions that are independent of the public key used by the scheme (which is not realistic, as an adversary can often influence the plaintext distribution after seeing the public key). In this paper, we ask whether or not this is essential. Namely, is it possible to formalize a meaningful notion of security that allows dependencies between plaintext distributions and keys?

1.1 Our Contributions

In this paper, we show that it is *not* essential to focus only on plaintexts distributions that are independent of the keys used by the scheme. We formalize and realize a new notion of security for deterministic public-key encryption, allowing adversaries to *adaptively* choose plaintext distributions *after* seeing the public key of the scheme, in an *interactive* manner. The only restriction we make is that the number of plaintext distributions from which each adversary is allowed to adaptively choose is upper bounded by $2^{p(\lambda)}$, where $p(\lambda)$ can be any predetermined polynomial in the security parameter λ. We stress that the set of $2^{p(\lambda)}$ plaintext distributions can be different for each adversary. Intuitively, this bound says that the entire plaintext distribution (not just a single sample) contains at most $p(\lambda)$ bits of information about the public key. We view this as a natural first model for adaptively chosen plaintext distributions, particularly in light of the impossibility of handling arbitrary dependencies (as sketched earlier), and hope that it will pave the way for more realistic models.

Our approach is a generalization of the security notions that have been proposed so far. For example, with $p(\lambda) \equiv 0$ we obtain the notion of security introduced by Bellare, Boldyreva, and O'Neill [1], where the plaintext distribution chosen by the adversary is independent of the public key. As an additional example, with $p(\lambda) = O(s(\lambda) \log s(\lambda))$ we capture, in particular, all plaintext distributions that are samplable by boolean circuits of size at most $s(\lambda)$.

Within our framework we present both generic constructions in the random-oracle model based on any public-key encryption scheme, and generic constructions in the standard model based on lossy trapdoor functions. Our constructions are inspired by the constructions of Bellare, Boldyreva, and O'Neill [1]

[3] Note that the support of this distribution will contain nearly half of all plaintexts with high probability.

and of Boldyreva, Fehr, and O'Neill [4]. These constructions rely on the independence between the plaintext distributions and the keys for the purposes of extracting randomness from the plaintext distributions. Randomness extraction becomes significantly more difficult once the plaintexts distributions and the public keys are no longer independent. Challenges along somewhat similar lines arise in the context of deterministic randomness extraction, where one would like to construct seedless randomness extractors, or seeded randomness extractors for seed-dependent distributions. Indeed, underlying our approach is a new generalization of a method for deterministic extraction, originally introduced by Trevisan and Vadhan [21] and Dodis [9].

Finally, our approach naturally extends to the setting of "hedged" public-key encryption schemes, introduced by Bellare et al. [2]. In this setting, one would like to construct randomized schemes that are semantically secure in the standard sense, and maintain a meaningful and realistic notion of security even when "corrupt" randomness is used by the encryption algorithm. Our notions of adaptive security for deterministic public-key encryption give rise to analogous notions for hedged public-key encryption, and our constructions (when used within the framework of Bellare et al. [2][4]) yield the first adaptively-secure hedged public-key encryption schemes.

1.2 Related Work

The formal study of deterministic public-key encryption was initiated by Bellare, Boldyreva, and O'Neill [1], following research on symmetric-key encryption of high-entropy messages by Russell and Wang [20] and Dodis and Smith [11]. Bellare et al. formalized several notions of security, which were later refined and extended by Bellare, Fischlin, O'Neill, and Ristenpart [3], and by Boldyreva, Fehr, and O'Neill [4]. Bellare, Boldyreva, and O'Neill presented constructions in the random oracle model, and constructions in the standard model were first presented by Bellare, Boldyreva, and O'Neill, and additionally by Boldyreva, Fehr, and O'Neill. Brakerski and Segev [7] showed that the min-entropy requirement considered in all previous works on deterministic public-key encryption can be relaxed to consider hard-to-invert auxiliary inputs. Based on specific number-theoretic assumptions, they designed schemes that are secure in the more general auxiliary-input model, and their constructions were later unified by Wee [23]. Progress along similar lines was made by Fuller, O'Neill and Reyzin [13], who presented a scheme that can securely encrypt a small predetermined number of plaintexts with arbitrary dependencies as long as each has high min-entropy. Additional progress in studying deterministic public-key encryption schemes was recently made by Mironov, Pandey, Reingold, and Segev [17] who constructed such schemes with optimal incrementality.

[4] For example, as part of their generic "pad-then-deterministic" scheme, which deterministically encrypts the concatenation of the plaintext and the randomness.

A step towards obtaining adaptive security for deterministic public-key encryption was made by Bellare et al. [2] who defined and constructed "hedged" public-key encryption schemes (discussed in Section 1.1). Whereas the notions of security considered in [1,3,4,7,23,13,17] capture only "single-shot" adversaries (i.e., adversaries that challenge the given scheme with only one plaintext distribution), Bellare et al. [2] showed that it is possible to guarantee security even against "multi-shot" adversaries (i.e., adversaries that interactively challenge the scheme with plaintext distributions depending on previous ciphertexts that they received). In their notion of security, however, adversaries are not given access to the public key that is being attacked. In our work we consider the more general, and more typical, scenario where adversaries are given *direct access* to the public key being attacked (and are allowed to adaptively and interactively choose plaintext distributions depending on previous ciphertexts that they received).[5] As discussed in Section 1.1, our constructions yield the first adaptively-secure hedged public-key encryption schemes.

1.3 Overview of Our Approach

In this section we provide a high-level overview of our notions of security and of the main ideas underlying our constructions. We focus here on our constructions in the standard model (i.e., without random oracles), as these emphasize more clearly the main challenges in designing encryption schemes satisfying our notions of security.

Our Notions of Security. As discussed above, our notions of security for deterministic public-key encryption differ from the previously proposed ones by providing adversaries with *direct* access to the public key. Specifically, we formalize security via a game between an adversary and a "real-or-random" encryption oracle. First, a pair containing a public key and a secret key is produced using the key-generation algorithm of the scheme under consideration, and the adversary is given the public key. Then, the adversary adaptively interacts with the encryption oracle, where each query consists of a description of a plaintext distribution M. For simplicity, here we consider distributions over plaintexts, but in fact our notion allows distributions over blocks of plaintexts. The encryption oracle operates in one of two modes, "real" or "random", which is chosen uniformly at random at the beginning of the game. In the "real" mode, the encryption oracle samples a plaintext according to M, and the adversary is given its encryption under the public key. In the "random" mode, the encryption

[5] In fact, the approach of Bellare et al. [2] relies on encryption schemes in which ciphertexts reveal essentially no information on the corresponding public key. Therefore, even multi-shot adversaries learn essentially no information on the public key being attacked, and thus their "adaptive" choices of plaintext distributions are still independent of the public key. This approach does not seem to extend to our setting, where adversaries are given direct access to the public key.

oracle samples a plaintext from the uniform distribution over the plaintext space, and the adversary is again given its encryption under the public key.[6]

The goal of the adversary in this game is to distinguish between the "real" mode and "random" mode with a non-negligible probability, subject only to the requirement that for any such adversary there exists a set $\mathcal{X} = \mathcal{X}_\lambda$ of plaintext distributions such that:

1. $|\mathcal{X}| \leq 2^p$, where $p = p(\lambda)$ is any predetermined polynomial in the security parameter (the construction of the scheme can depend on the polynomial p).
2. The adversary queries the encryption oracle only with plaintext distributions in $\mathcal{X}$.
3. Each plaintext distribution in $\mathcal{X}$ has min-entropy at least k, where $k = k(\lambda)$ is a predetermined function of the security parameter.

In addition, we naturally extend the above game to capture chosen-ciphertext attacks, by allowing adversaries adaptive access to a decryption oracle (subject to the standard requirement of not querying the decryption oracle with any ciphertext that was produced by the encryption oracle).

We note that our security game is in fact almost identical to the standard "real-or-random" one for randomized public-key encryption. Specifically, unlike the previously proposed notions of security for deterministic public-key encryption, we provide the adversary with direct access to the public key, and allow the adversary to adaptively interact with the encryption and decryption oracles *in any order.*[7]

Chosen-plaintext Security in the Standard Model. The starting point for our construction is the one of Boldyreva, Fehr, and O'Neill, which we now briefly describe. In their construction, the public key consists of a function f that is sampled from the injective mode of a collection of lossy trapdoor functions, and a permutation π sampled from a pairwise-independent collection of permutations. (We refer the reader to Section 2 for the relevant definitions.) The secret key consists of the trapdoor for inverting f. (We require that π is efficiently invertible.) The encryption of a message m is defined as $\mathsf{Enc}_{pk}(m) = f(\pi(m))$, and decryption is naturally defined.

The proof of security consists of two steps. First, the security of the collection of lossy trapdoor functions allows one to replace the injective function f with a lossy function $\widetilde{f}$ (where lossy means that the size of $\widetilde{f}$'s image is significantly smaller than the size of its domain). Then, the Crooked Leftover Hash Lemma of Dodis and Smith [10] states that for any plaintext distribution M that has a certain amount of min-entropy, for a uniformly and independently chosen

[6] We note that the resulting notion of security is polynomially equivalent (via a standard hybrid argument) to an analogous "left" or "right" formulation in which the adversary specifies two plaintext distributions, and the encryption oracle uses either the left one of the right one.

[7] In contrast, due to requiring key-independent plaintext distributions, Bellare et al. [1] and Boldyreva et al. [4] allow chosen-ciphertext adversaries to query the decryption oracle *only after* they have queried the encryption oracle.

pairwise-independent permutation π it holds that the distributions $\widetilde{f}(\pi(M))$ and $\widetilde{f}(U)$ are statistically close (even given $\widetilde{f}$ and π), where U is the uniform distribution over plaintexts. That is, essentially no information on the plaintext is revealed.

This construction, however, becomes insecure when adversaries can choose the plaintext distribution M after receiving the description of π. Specifically, the Crooked Leftover Hash Lemma no longer holds when M may depend on π, and adversaries may easily use the encryption algorithm as a subliminal channel for leaking information about the plaintext, as discussed above.

The main idea underlying our basic construction is to sample the permutation π from a collection of highly-independent permutations. We prove that this modification results in a scheme that is secure according to our new notion of security by proving a *High-Moment* Crooked Leftover Hash Lemma.[8] Informally, we prove that for any lossy function $\widetilde{f}$, and for any set $\mathcal{X}$ of sources with a certain amount of min-entropy, with an overwhelming probability over the choice of a permutation π from a t-wise almost-independent collection of permutations (where t depends only logarithmically on the size of $\mathcal{X}$), for *every* $M \in \mathcal{X}$ it holds that $\widetilde{f}(\pi(M))$ and $\widetilde{f}(U)$ are statistically close. In particular, in such a setting the specific choice of $M \in \mathcal{X}$ can adaptively depend on the permutation π, and still the statistical distance is negligible.

Chosen-ciphertext Security in the Standard Model. While in the setting of chosen-plaintext security our construction is a natural generalization of that of Boldyreva et al. [4] (given our high-moment generalization of the crooked leftover hash), this is not the case in the setting of chosen-ciphertext security. In this setting, the CCA-secure scheme of Boldyreva et al. relies more strongly on the assumption that the challenge plaintext distribution is independent of the public key of the scheme (not just in the context of the Crooked Leftover Hash Lemma as above) – an assumption that we do not make. Nevertheless, we show that some of the ideas underlying their approach can still be utilized to construct a scheme that is secure according to our notion of security.

The scheme of Boldyreva et al. follows the "all-but-one" simulation paradigm of Peikert and Waters [18] using all-but-one lossy trapdoor functions. These are tag-based functions, where one of the tags corresponds to a lossy function, and all other tags correspond to injective functions. As in the work of Peikert and Waters [18], the approach of Boldyreva et al. makes sure that the challenge

[8] As already noted, a high-moment generalization of the (standard) Leftover Hash Lemma was given by Trevisan and Vadhan [21] and Dodis [9]. In addition, an analogous generalization of the Crooked Leftover Hash Lemma for collections of *functions* was implicitly given in the work of Kiltz, O'Neill and Smith [16, Proof of Theorem 2]. Their generalization, however, does not seem to admit a direct translation to collections of *permutations*. A different high-moment generalization of the Crooked Leftover Hash Lemma was proved by Fuller et al. [13] for the purpose of extracting randomness from a small number of possibly correlated sources. This generalization does not allow seed-dependent sources, and therefore allows only non-adaptive adversaries.

plaintext corresponds to a lossy tag (and thus the challenge ciphertext reveals no information), while all other plaintexts corresponds to injective tags (and a suitable simulator is able to properly simulate the decryption oracle). When dealing with a deterministic encryption algorithm, note that tags must be derived deterministically from the plaintext and the public key. The approach of Boldyreva et al. is based on first sampling the challenge plaintext m^*, and only then generating a public key for which m^* corresponds to a lossy tag, but all other plaintexts correspond to injective tags.

This approach fails in our setting, where adversaries specify the distribution of the challenge plaintext in an adaptive manner as a function of the public key. Thus, in our setting we must be able to generate a public key before the challenge plaintext is known. We note that a somewhat similar issue arises in the setting of identity-based encryption (IBE): "selective security" considers adversaries that specify the challenge identity in advance, whereas "full security" considers adversaries that can adaptively choose the challenge identity. One simple solution that was proposed in the IBE setting is to a-priori guess the challenge identity, and this solution naturally extends to our setting by guessing the tag corresponds to the challenge plaintext. This, however, requires sub-exponential hardness assumptions, which we aim to avoid.

Our approach is based on the one of Boneh and Boyen [5] (and on its refinement by Cash, Hofheinz, Kiltz, and Peikert [8] for converting a large class of selectively-secure IBE schemes to fully-secure ones,[9] combined with the idea of $\mathcal{R}$-lossiness due to Boyle, Segev, and Wichs [6]. Specifically, we derive tags from plaintexts using an admissible hash functions [5,8], and instead of using all-but-one lossy trapdoor functions, we introduce the notion of $\mathcal{R}$-lossy trapdoor functions (which we generically construct based on lossy trapdoor functions).[10] This is a generalization of the notion of all-but-one lossy trapdoor functions, where the set of tags is partitioned into lossy tags and injective tags according to the relation $\mathcal{R}$. (In particular, there may be more than one lossy tag.) Combined with an admissible hash function, we are able to ensure that even with an adaptive adversary, with some non-negligible probability, the challenge plaintext corresponds to a lossy tag (and thus the challenge ciphertext reveals no information), while all other plaintexts corresponds to injective tags (and a suitable simulator is able to properly simulate the decryption oracle). We show that such a guarantee enables us to prove the security of our scheme with respect to adaptive adversaries.

2 Preliminaries

For an integer $n \in \mathbb{N}$ we denote by $[n]$ the set $\{1, \ldots, n\}$, and by U_n the uniform distribution over the set $\{0,1\}^n$. For a random variable X we denote by $x \leftarrow X$

[9] We note that the work of Cash et al. [8] is based on ideas introduced by Boneh and Boyen [5] and Waters [22].

[10] Boyle, Segev and Wichs [6] introduced the notion of $\mathcal{R}$-lossy public-key encryption, which can be viewed as a randomized variant of our notion of $\mathcal{R}$-lossy trapdoor functions.

the process of sampling a value x according to the distribution of X and by $\mathbb{E}[X]$ the expectation of the random variable X. Similarly, for a finite set S we denote by $x \leftarrow S$ the process of sampling a value x according to the uniform distribution over S. We denote by $\boldsymbol{X} = (X_1, \ldots, X_T)$ a joint distribution of T random variables, and by $\boldsymbol{x} = (x_1, \ldots, x_T)$ a sample drawn from $\boldsymbol{X}$. For two bit-strings x and y we denote by $x \| y$ their concatenation. A non-negative function $f : \mathbb{N} \to \mathbb{R}$ is *negligible* if it vanishes faster than any inverse polynomial.

In this paper we consider the uniform adversarial model (i.e. consider uniform probabilistic polynomial-time adversaries). We note that all of our results also apply to the nonuniform adversarial model (under nonuniform complexity assumptions).

The *min-entropy* of a random variable X is $\mathbf{H}_\infty(X) = -\log(\max_x \Pr[X = x])$. A *$k$-source* is a random variable X with $\mathbf{H}_\infty(X) \geq k$. A *$(T, k)$-source* is a random variable $\boldsymbol{X} = (X_1, \ldots, X_T)$ where each X_i is a k-source for every $i \in [T]$. A *(T, k)-block source* is a random variable $\boldsymbol{X} = (X_1, \ldots, X_T)$ where for every $i \in [T]$ and $x_1, \ldots, x_{i-1}$ it holds that $\mathbf{H}_\infty(X_i | X_1 = x_1, \ldots, X_{i-1} = x_{i-1}) \geq k$.

The *statistical distance* between two random variables X and Y over a finite domain Ω is $\mathbf{SD}(X, Y) = \frac{1}{2} \sum_{\omega \in \Omega} |\Pr[X = \omega] - \Pr[Y = \omega]|$. Two random variables X and Y are *δ-close* if $\mathbf{SD}(X, Y) \leq \delta$. Two distribution ensembles $\{X_\lambda\}_{\lambda \in \mathbb{N}}$ and $\{Y_\lambda\}_{\lambda \in \mathbb{N}}$ are *statistically indistinguishable* if it holds that $\mathbf{SD}(X_\lambda, Y_\lambda)$ is negligible in λ. They are *computationally indistinguishable* if for every probabilistic polynomial-time algorithm $\mathcal{A}$ it holds that $\left|\Pr\left[\mathcal{A}(1^\lambda, x) = 1\right] - \Pr\left[\mathcal{A}(1^\lambda, y) = 1\right]\right|$ is negligible in λ, where $x \leftarrow X_\lambda$ and $y \leftarrow Y_\lambda$.

2.1 t-Wise δ-Dependent Permutations

A collection Π of permutations over $\{0,1\}^n$ is *t-wise δ-dependent* if for any distinct $x_1, \ldots, x_t \in \{0,1\}^n$ the distribution $(\pi(x_1), \ldots, \pi(x_t))$ where π is sampled from Π is δ-close in statistical distance to the distribution $(\pi^*(x_1), \ldots, \pi^*(x_t))$ where π^* is a truly random permutation. For our construction in the standard model we rely on an explicit construction of such a collection due to Kaplan, Naor, and Reingold [15] that enjoys an asymptotically optimal description length (although we note that in fact any other construction can be used):

Theorem 2.1 ([15]). *For any integers n and $t \leq 2^n$, and for any $0 < \delta < 1$, there exists an explicit t-wise δ-dependent collection Π of permutations over $\{0,1\}^n$ where each permutation $\pi \in \Pi$ can be described using $O(nt + \log(1/\delta))$ bits, and is computable and invertible in time polynomial in n, t and $\log(1/\delta)$.*

2.2 Admissible Hash Functions

The concept of an *admissible hash function* was first defined by Boneh and Boyen [5] to convert a large class of selectively-secure identity-based encryption scheme into a fully-secure ones.[11] In this paper we use such hash functions in a somewhat

[11] The work of Boneh and Boyen [5] shows how to construct admissible hash functions from collision-resistant hash functions.

similar way as part of our construction of a CCA-secure deterministic public-key encryption scheme. The main idea of an admissible hash function is that it allows the reduction in the proof of security to secretly partition the message space into two subsets, which we will label as "lossy tags" and "injective tags," such that there is a noticeable probability that all of the messages in the adversary's decryption queries will correspond to injective tags, but the challenge ciphertext will correspond to a lossy tag. This is useful if the simulator can efficiently answer decryption queries with injective tags, while a challenge ciphertext with a lossy tag reveals essentially no information on the encrypted message. Our exposition and definition of admissible hash function follows that of Cash, Hofheinz, Kiltz, and Peikert [8].

For $K \in \{0,1,\perp\}^{v(\lambda)}$, we define the "partitioning" function $P_K : \{0,1\}^{v(\lambda)} \to \{\mathsf{Lossy}, \mathsf{Inj}\}$ which partitions the space $\{0,1\}^{v(\lambda)}$ of tags in the following way:

$$P_K(y) := \begin{cases} \mathsf{Lossy} & \text{if } \forall\, i \in \{1,\dots,v(\lambda)\} \quad : \quad K_i = y_i \text{ or } K_i = \perp \\ \mathsf{Inj} & \text{otherwise} \end{cases}$$

For any $u = u(\lambda) < v(\lambda)$, we let $\mathcal{K}_{u,\lambda}$ denote the uniform distribution over $\{0,1,\perp\}^{v(\lambda)}$ *conditioned on* exactly u positions having $\perp$ values. (Note, if K is chosen from $\mathcal{K}_{u,\lambda}$, then the map $P_K(\cdot)$ defines exactly 2^u values as Lossy.) We would like to pick a distribution $\mathcal{K}_{u,\lambda}$ for choosing K so that, there is a noticeable probability for every set of tags $y_0, \dots, y_q$, of y_0 being classified as "lossy" and all other tags "injective." Unfortunately, this cannot happen if we allow all tags. Instead, we will need to rely on a special hash function the maps messages x to tags y.

Definition 2.2 (Admissible hash functions [5,8]). *Let $\mathcal{H} = \{\mathcal{H}_\lambda\}_{\lambda\in\mathbb{N}}$ be a hash-function ensemble, where each $h \in \mathcal{H}_\lambda$ is a polynomial-time computable function $h : \{0,1\}^{n(\lambda)} \to \{0,1\}^{v(\lambda)}$. We say that $\mathcal{H}$ is an* admissible hash-function ensemble *if for every $h \in \mathcal{H}$ there exists a efficiently recognizable set $\mathsf{Unlikely}_h \subseteq \bigcup_{q\in\mathbb{N}} \left(\{0,1\}^{n(\lambda)}\right)^q$ of string-tuples such that the following two properties hold:*

- *For every probabilistic polynomial-time algorithm $\mathcal{A}$ there exists a negligible function $\nu(\lambda)$ satisfying $\Pr[(x_0,\dots,x_q) \in \mathsf{Unlikely}_h] \leq \nu(\lambda)$, where $h \leftarrow \mathcal{H}_\lambda$ and $(x_0,\dots,x_q) \leftarrow \mathcal{A}(1^\lambda, h)$.*
- *For every polynomial $q = q(\lambda)$ there is a polynomial $\Delta = \Delta(\lambda)$ and an efficiently computable $u = u(\lambda)$ such that, for every $h \in \mathcal{H}_\lambda$ and $(x_0,\dots,x_q) \notin \mathsf{Unlikely}_h$ with $x_0 \notin \{x_1,\dots,x_q\}$ we have:*

$$\Pr_{K \leftarrow \mathcal{K}_{u,\lambda}} \left[P_K(h(x_0)) = \mathsf{Lossy} \wedge P_K(h(x_1)) = \cdots = P_K(h(x_q)) = \mathsf{Inj}\right] \geq \frac{1}{\Delta(\lambda)}.$$

2.3 Lossy Trapdoor Functions

A collection of lossy trapdoor functions [18] consists of two families of functions. Functions in one family are injective and can be efficiently inverted using a trapdoor. Functions in the other family are "lossy," which means that the size of their image is significantly smaller than the size of their domain. The only security requirement is that a description of a randomly chosen function from the family of injective functions is computationally indistinguishable from a description of a randomly chosen function from the family of lossy functions.

Definition 2.3 (Lossy trapdoor functions [18,12]). *Let $n : \mathbb{N} \to \mathbb{N}$ and $\ell : \mathbb{N} \to \mathbb{N}$ be non-negative functions, and for any $\lambda \in \mathbb{N}$ let $n = n(\lambda)$ and $\ell = \ell(\lambda)$. A* collection of (n, ℓ)-lossy trapdoor functions *is a 4-tuple of probabilistic polynomial-time algorithms* $(\mathsf{Gen}_0, \mathsf{Gen}_1, \mathsf{F}, \mathsf{F}^{-1})$ *such that:*

1. **Sampling a lossy function:** $\mathsf{Gen}_0(1^\lambda)$ *outputs a function index* $\sigma \in \{0,1\}^*$.
2. **Sampling an injective function:** $\mathsf{Gen}_1(1^\lambda)$ *outputs a pair* $(\sigma, \tau) \in \{0,1\}^* \times \{0,1\}^*$, *where σ is a function index and τ is a trapdoor.*
3. **Evaluation:** *Let $n = n(\lambda)$ and $\ell = \ell(\lambda)$. Then, for every function index σ produced by either* Gen_0 *or* Gen_1, *the algorithm* $\mathsf{F}(\sigma, \cdot)$ *computes a function* $f_\sigma : \{0,1\}^n \to \{0,1\}^*$ *with one of the two following properties:*
   - *Lossy: If σ is produced by* Gen_0, *then the image of f_σ has size at most* $2^{n-\ell}$.
   - *Injective: If σ is produced by* Gen_1, *then the function f_σ is injective.*
4. **Inversion of injective functions:** *For every pair (σ, τ) produced by* Gen_1 *and every $x \in \{0,1\}^n$, we have* $\mathsf{F}^{-1}(\tau, \mathsf{F}(\sigma, x)) = x$.
5. **Indistinguishability of indices:** *The ensembles* $\left\{\sigma : \sigma \leftarrow \mathsf{Gen}_0(1^\lambda)\right\}_{\lambda \in \mathbb{N}}$ *and* $\left\{\sigma : (\sigma, \tau) \leftarrow \mathsf{Gen}_1(1^\lambda)\right\}_{\lambda \in \mathbb{N}}$ *are computationally indistinguishable.*

Constructions of lossy trapdoor functions were proposed based on a wide variety of number-theoretic assumptions and for a large range of parameters (see, for example, [12,18] and the references therein). In particular, in terms of parameters, several constructions are known to offer $\ell = n - n^\epsilon$ for any fixed constant $0 < \epsilon < 1$ with $n = \mathrm{poly}(\lambda)$.

2.4 Deterministic Public-Key Encryption

A deterministic public-key encryption scheme is a triplet $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ of polynomial-time algorithms with the following properties:

- The key-generation algorithm KeyGen is a randomized algorithm that takes as input the security parameter 1^λ and outputs a key pair (sk, pk) consisting of a secret key sk and a public key pk.
- The encryption algorithm Enc is a *deterministic* algorithm that takes as input a public key pk and a message $m \in \{0,1\}^{n(\lambda)}$, and outputs a ciphertext $c = \mathsf{Enc}_{pk}(m)$.
- The decryption algorithm is a possibly randomized algorithm that takes as input a secret key sk and a ciphertext c and outputs a message $m \leftarrow \mathsf{Dec}_{sk}(c)$ such that $m \in \{0,1\}^{n(\lambda)} \cup \{\perp\}$.

3 Formalizing Adaptive Security for Deterministic Public-Key Encryption

In this section we present a framework for modeling the security of deterministic public-key encryption schemes in an *adaptive* setting. As discussed in Section 1.3, we consider adversaries that *adaptively* choose plaintext distributions *after* seeing the public key of the scheme, in an *interactive* manner. The only restriction we make is that the *number* of plaintext distributions from which each adversary is allowed to choose is upper bounded by $2^{p(\lambda)}$, where $p(\lambda)$ can be any a-priori given polynomial in the security parameter λ. The security definitions that follow are parameterized by three parameters:

– $p = p(\lambda)$ denoting the 2^p bound on the number of allowed plaintext distributions.
– $T = T(\lambda)$ denoting the number of blocks in each plaintext distribution.
– $k = k(\lambda)$ denoting the min-entropy requirement.

Additionally, they are implicitly parameterized by bit-length $n = n(\lambda)$ of plaintexts. We begin by defining the "real-or-random" encryption oracle which we use formalize security.

Definition 3.1 (Real-or-random encryption oracle). *The real-or-random oracle* RoR *takes as input triplets of the form* $(\mathsf{mode}, pk, \boldsymbol{M})$*, where* $\mathsf{mode} \in \{\mathsf{real}, \mathsf{rand}\}$*, pk is a public key, and $\boldsymbol{M} = (M_1, \dots, M_T)$ is a circuit representing a joint distribution over T messages. If* $\mathsf{mode} = \mathsf{real}$ *then the oracle samples $(m_1, \dots, m_T) \leftarrow \boldsymbol{M}$, and if* $\mathsf{mode} = \mathsf{rand}$ *then the oracle samples $(m_1, \dots, m_T) \leftarrow U^T$ where U is the uniform distribution over the appropriate message space. It then outputs* $(\mathsf{Enc}_{pk}(m_1), \dots, \mathsf{Enc}_{pk}(m_T))$.

Following [1,4] we consider two classes of adversarially-chosen message distributions $\boldsymbol{M} = (M_1, \dots, M_T)$: The class of (T, k)-sources, where each M_i is assumed to be a k-source, and the more restrictive class of (T, k)-block-sources, where each M_i is assumed to be a k-source even given $M_1, \dots, M_{i-1}$. (See Section 2 for formal definitions.) Our constructions in the random oracle model are secure with respect to (T, k)-sources, and our constructions in the standard model are secure with respect to (T, k)-block-sources. This gap was recently shown by Wichs [24] to be inherent to our techniques, and in fact to all the techniques that were so far used for designing deterministic public-key encryption schemes without random oracles [3,4,2,7,13,17,23]. Specifically, Wichs showed that no deterministic public-key encryption scheme can be proven secure for all (T, k)-sources using a black-box reduction to a "falsifiable" hardness assumption. (We refer the reader to [24] for more details on his notion of falsifiability.)

The following two definitions capture the class of chosen-plaintext adversaries and security game that we consider in this paper. We refer the reader to the full version [19] for their natural generalization to chosen-ciphertext attacks.

Definition 3.2 (2^p-bounded (T, k)-source adversary). *Let $\mathcal{A}$ be a probabilistic polynomial-time algorithm that is given as input a pair $(1^\lambda, pk)$ and* *oracle access to* $\mathsf{RoR}(\mathsf{mode}, pk, \cdot)$ *for some* $\mathsf{mode} \in \{\mathsf{real}, \mathsf{rand}\}$. *Then,* $\mathcal{A}$ *is a* 2^p-bounded (T, k)-source adversary *if for every* $\lambda \in \mathbb{N}$ *there exists a set* $\mathcal{X} = \mathcal{X}_\lambda$ *of polynomial-time samplable* (T, k)*-sources such that:*

1. $|\mathcal{X}| \leq 2^p$.
2. *For each of* $\mathcal{A}$*'s* RoR *queries* $\boldsymbol{M}$ *it holds that:*
   - $\boldsymbol{M} \in \mathcal{X}$.
   - *For all* $(m_1, \ldots, m_T)$ *in the support of* $\boldsymbol{M}$ *and for all distinct* $i, j \in [T]$ *it holds that* $m_i \neq m_j$.

In addition, $\mathcal{A}$ *is a* block-source *adversary if* $\mathcal{X}$ *is a set of* (T, k)*-block-sources.*

Definition 3.3 (Adaptive chosen-distribution attacks (ACD-CPA)). *A deterministic public-key encryption scheme* $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *is* (p, T, k)-ACD-CPA-secure *(resp.* block-wise (p, T, k)-ACD-CPA-secure*) if for any probabilistic polynomial-time* 2^p*-bounded* (T, k)*-source (resp. block-source) adversary* $\mathcal{A}$*, there exists a negligible function* $\nu(k)$ *such that*

$$\mathbf{Adv}^{\text{ACD-CPA}}_{\Pi,\mathcal{A}}(\lambda) \stackrel{\mathsf{def}}{=} \left| \Pr\left[\mathsf{Expt}^{\mathsf{real}}_{\Pi,\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{Expt}^{\mathsf{rand}}_{\Pi,\mathcal{A}}(\lambda) = 1\right]\right| \leq \nu(\lambda),$$

where for each $\mathsf{mode} \in \{\mathsf{real}, \mathsf{rand}\}$ *and* $\lambda \in \mathbb{N}$ *the experiment* $\mathsf{Expt}^{\mathsf{mode}}_{\Pi,\mathcal{A}}(\lambda)$ *is defined as follows:*

1. $(sk, pk) \leftarrow \mathsf{KeyGen}(1^\lambda)$.
2. $b \leftarrow \mathcal{A}^{\mathsf{RoR}(\mathsf{mode}, pk, \cdot)}(1^\lambda, pk)$.
3. *Output* b.

In addition, such a scheme is (p, T, k)-ACD1-CPA-secure *(resp.* block-wise (p, T, k)-ACD1-CPA-secure*) if the above holds for any probabilistic polynomial-time* 2^p*-bounded* (T, k)*-source (resp. block-source) adversary* $\mathcal{A}$ *that queries the* RoR *oracle at most once.*

Our adaptive notion of security enables an immediate reduction of "multi-shot" adversaries to "single-shot" ones, as in the case of randomized public-key encryption. The following theorem follows via a standard hybrid argument.

Theorem 3.4. *For any polynomials* p*,* T*, and* k*, a deterministic public-key encryption scheme* Π *is* (p, T, k)-ACD-CPA*-secure (resp. block-wise* (p, T, k)-ACD-CPA*-secure) if and only if it is* (p, T, k)-ACD1-CPA*-secure (resp. block-wise* (p, T, k)-ACD1-CPA*-secure).*

4 Chosen-Plaintext Security Based on Lossy Trapdoor Functions

In this section we present our basic construction of a public-key deterministic encryption scheme that is secure according to our notion of adaptive security. We refer the reader to Section 1.3 for a high-level description of the scheme, and of the main challenges and ideas underlying our approach. In what follows we formally describe the scheme and discuss the parameters that we obtain using known instantiations of its building blocks.

Let $n = n(\lambda)$, $\ell = \ell(\lambda)$, $t = t(\lambda)$ and $\delta = \delta(\lambda)$ be functions of the security parameter $\lambda \in \mathbb{N}$. Let $(\mathsf{Gen}_0, \mathsf{Gen}_1, \mathsf{F}, \mathsf{F}^{-1})$ be a collection of (n, ℓ)-lossy trapdoor functions, and for every $\lambda \in \mathbb{N}$ let Π_λ be a t-wise δ-dependent collection of permutations over $\{0,1\}^n$. Our scheme $\mathcal{DE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is defined as follows:

- **Key generation.** The key-generation algorithm KeyGen on input 1^λ samples $(\sigma, \tau) \leftarrow \mathsf{Gen}_1(1^\lambda)$ and $\pi \leftarrow \Pi_\lambda$. It then outputs $pk = (\sigma, \pi)$ and $sk = \tau$.
- **Encryption.** The encryption algorithm Enc on input a public key $pk = (\sigma, \pi)$ and a message $m \in \{0,1\}^n$ outputs $c = \mathsf{F}(\sigma, \pi(m))$.
- **Decryption.** The decryption algorithm Dec on input a secret key $sk = \tau$ and a ciphertext c outputs $m = \pi^{-1}\left(\mathsf{F}^{-1}(\tau, c)\right)$.

Theorem 4.1. *The scheme $\mathcal{DE}$ is block-wise (p, T, k)-ACD-CPA-secure for any $n = n(\lambda)$, $\ell = \ell(\lambda)$, $p = p(\lambda)$, and $T = T(\lambda)$ by setting $t = p + n - \ell + \log T + \omega(\log \lambda)$, $k = n - \ell + 2\log T + 2\log t + \omega(\log \lambda)$, and $\delta = 2^{-nt}$.*

Parameters. Using existing constructions of lossy trapdoor functions (see Section 2.3), for any $n = n(\lambda)$ and for any constant $0 < \epsilon < 1$ we can instantiate our scheme with $\ell = n - n^\epsilon$. Therefore, for any $n = n(\lambda)$, $p = p(\lambda)$, and $T = T(\lambda)$, we obtain schemes with $t = p + n^\epsilon + \omega(\log \lambda)$, $k = n^\epsilon + \omega(\log \lambda)$, and $\delta = 2^{-nt}$.

5 $\mathcal{R}$-Lossy Trapdoor Functions

The notion of $\mathcal{R}$-lossy *public-key encryption schemes* was put forward by Boyle, Segev, and Wichs [6], and here we define an analogous notion for *trapdoor functions*. Informally, an $\mathcal{R}$-lossy trapdoor function family is a collection of tagged functions where the set of possible tags is partitioned into two subsets: *injective* tags, and *lossy* tags. Functions evaluated with an injective tag can be efficiently inverted with a trapdoor (where all injective tags share the same trapdoor information). On the other hand, functions evaluated with a lossy tag lose information – the size of their image is significantly smaller than the size of their domain. The partitioning of the tags is defined by a binary relation $\mathcal{R} \subseteq \mathcal{K} \times \mathcal{T}$: the key-generation algorithm receives as input an *initialization value* $K \in \mathcal{K}$ and this partitions the set tags $\mathcal{T}$ so that $t \in \mathcal{T}$ is lossy if and only if $(K, t) \in \mathcal{R}$. More, formally, we require that the relation $\mathcal{R} \subseteq \mathcal{K} \times \mathcal{T}$ consists of a sequence of efficiently (in λ) recognizable sub-relations $\mathcal{R}_\lambda \subseteq \mathcal{K}_\lambda \times \mathcal{T}_\lambda$. The only computational requirement of an $\mathcal{R}$-lossy trapdoor function family is that its description hides the initialization value K.

Definition 5.1 ($\mathcal{R}$-lossy trapdoor functions). *Let $n : \mathbb{N} \to \mathbb{R}$ and $\ell : \mathbb{N} \to \mathbb{R}$ be non-negative functions, and for any $\lambda \in \mathbb{N}$ let $n = n(\lambda)$ and $\ell = \ell(\lambda)$. Also, let $\mathcal{R} \subseteq \mathcal{K} \times \mathcal{T}$ be an efficiently computable binary relation. An* $\mathcal{R}$-(n, ℓ)-lossy trapdoor function family *is a triplet of probabilistic polynomial-time algorithms $\Pi = (\mathsf{Gen}_\mathcal{R}, \mathsf{G}, \mathsf{G}^{-1})$ such that:*

1. **Key generation:** *For any initialization value $K \in \mathcal{K}_\lambda$, the key-generation algorithm $\mathsf{Gen}_{\mathcal{R}}(1^\lambda, K)$ outputs a public index σ and a trapdoor τ.*
2. **Evaluation:** *For any $K \in \mathcal{K}$, $(\sigma, \tau) \leftarrow \mathsf{Gen}_{\mathcal{R}}(1^\lambda, K)$, and any $t \in \mathcal{T}$, the algorithm $\mathsf{G}(\sigma, t, \cdot)$ computes a function $f_{\sigma,t} : \{0,1\}^n \to \{0,1\}^*$ with one of the two following properties:*
   - *Lossy tags: If $(K, t) \in \mathcal{R}$, then the image of $f_{\sigma,t}$ has size at most $2^{n-\ell}$.*
   - *Injective tags: If $(K, t) \notin \mathcal{R}$, then the function $f_{\sigma,t}$ is injective.*
3. **Inversion under injective tags:** *For any $K \in \mathcal{K}$ and $t \in \mathcal{T}$ such that $(K, t) \notin \mathcal{R}$, and for any input $x \in \{0,1\}^n$, we have $\mathsf{G}^{-1}(\tau, t, \mathsf{G}(\sigma, t, x)) = x$.*
4. **Indistinguishability of initialization values:** *For every probabilistic polynomial-time adversary $\mathcal{A}$, there exists a negligible function $\nu(\lambda)$ such that*

$$\mathbf{Adv}^{\mathcal{R}\text{-}lossy}_{\Pi,\mathcal{A}}(\lambda) \overset{\mathsf{def}}{=} \left| \Pr\left[\mathsf{Expt}^{(0)}_{\Pi,\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{Expt}^{(1)}_{\Pi,\mathcal{A}}(\lambda) = 1\right] \right| \leq \nu(\lambda),$$

where for each $b \in \{0,1\}$ and $\lambda \in \mathbb{N}$ the experiment $\mathsf{Expt}^{(b)}_{\Pi,\mathcal{A}}(\lambda)$ is defined as follows:
   (a) $(K_0, K_1, \mathsf{state}) \leftarrow \mathcal{A}(1^\lambda)$.
   (b) $(\sigma, \tau) \leftarrow \mathsf{Gen}_{\mathcal{R}}(1^\lambda, K_b)$.
   (c) $b' \leftarrow \mathcal{A}(1^\lambda, \sigma, \mathsf{state})$.
   (d) Output b'.

We are interested mainly in the bit-matching relation $\mathcal{R}^{\mathsf{BM}}$, as defined by Boyle, Segev, and Wichs [6]. For every $\lambda \in \mathbb{N}$ let $\mathcal{K}_\lambda = \{0, 1, \perp\}^{v(\lambda)}$ and $\mathcal{T}_\lambda = \{0,1\}^{v(\lambda)}$, and define $(K, t) \in \mathcal{R}^{\mathsf{BM}}_\lambda \subseteq \mathcal{K}_\lambda \times \mathcal{T}_\lambda$ if for every $i \in \{1, \ldots, v(\lambda)\}$ it holds that $K_i = t_i$ or $K_i = \perp$. That is, given some fixed initialization value K, the set of lossy tags t are exactly those whose bits match K in all positions i for which $K_i \neq \perp$. In our construction of CCA-secure deterministic encryption schemes, the $\mathcal{R}^{\mathsf{BM}}$-lossy trapdoor functions will be used in combination with an *admissible hash function* (discussed in Section 2.2). An admissible hash function enables us to map messages to encryption tags such that, with high probability over an appropriate distribution of K, all decryption queries map to injective tags while the challenge query maps to a lossy tag which loses information about the plaintext. We refer the reader to the full version for a generic construction of $\mathcal{R}^{\mathsf{BM}}$-lossy trapdoor functions based on any collection of lossy trapdoor functions. In turn, this implies that $\mathcal{R}^{\mathsf{BM}}$-lossy trapdoor functions can be based on a variety of number-theoretic assumptions.

6 Chosen-Ciphertext Security Based on $\mathcal{R}$-Lossy Trapdoor Functions

In this section we present a construction of a public-key deterministic encryption scheme that is secure according to our notion of adaptive security even when adversaries can access a decryption oracle. As discussed in Section 1.3, our construction is inspired by that of Boldyreva et al. [4] combined with the approach of Boneh and Boyen [5] (and its refinement by Cash, Hofheinz, Kiltz, and Peikert [8]) for converting a large class of selectively-secure IBE schemes to fully-secure ones, and the notion of $\mathcal{R}$-lossy trapdoor functions that we introduced in Section 5 following Boyle, Segev, and Wichs [6]. In what follows we formally describe the scheme and discuss the parameters that we obtain using known instantiations of its building blocks.

Let $n = n(\lambda)$, $\ell = \ell(\lambda)$, $v = v(\lambda)$, $t_1 = t_1(\lambda)$, $t_2 = t_2(\lambda)$, $\delta_1 = \delta_1(\lambda)$, and $\delta_2 = \delta_2(\lambda)$ be functions of the security parameter $\lambda \in \mathbb{N}$. Our construction relies on the following building blocks: a collection $\{\mathcal{H}_\lambda\}_{\lambda\in\mathbb{N}}$ of admissible hash functions $h : \{0,1\}^n \to \{0,1\}^v$; a collection $(\mathsf{Gen}_0, \mathsf{Gen}_1, \mathsf{F}, \mathsf{F}^{-1})$ of (n,ℓ)-lossy trapdoor functions; acollection $(\mathsf{Gen}_{\mathtt{BM}}, \mathsf{G}, \mathsf{G}^{-1})$ of $\mathcal{R}^{\mathtt{BM}}$-$(n,\ell)$-lossy trapdoor functions; a t_1-wise δ_1-dependent collection $\{\varPi_\lambda^{(1)}\}_{\lambda\in\mathbb{N}}$ of permutations over $\{0,1\}^n$; a t_2-wise δ_2-dependent collection $\{\varPi_\lambda^{(2)}\}_{\lambda\in\mathbb{N}}$ of permutations over $\{0,1\}^n$. Our scheme $\mathcal{DE}_{\mathrm{CCA}} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is defined as follows:

– **Key generation.** The key-generation algorithm KeyGen on input 1^λ samples $h \leftarrow \mathcal{H}_\lambda$, $(\sigma_f, \tau_f) \leftarrow \mathsf{Gen}_1(1^\lambda)$, $K \leftarrow \mathcal{K}_\lambda$, $(\sigma_g, \tau_g) \leftarrow \mathsf{Gen}_{\mathtt{BM}}(1^\lambda, K)$, $\pi_1 \leftarrow \varPi_\lambda^{(1)}$, and $\pi_2 \leftarrow \varPi_\lambda^{(2)}$. Then, it outputs $pk = (h, \sigma_f, \sigma_g, \pi_1, \pi_2)$ and $sk = (\tau_f, \tau_g)$.

– **Encryption.** The encryption algorithm Enc on input a public key $pk = (h, \sigma_f, \sigma_g, \pi_1, \pi_2)$ and a message $m \in \{0,1\}^n$ outputs

$$c = \Big(h(\pi_1(m)),\ \ \mathsf{F}\big(\sigma_f, \pi_2(m)\big),\ \ \mathsf{G}\big(\sigma_g, h(\pi_1(m)), \pi_2(m)\big)\Big).$$

– **Decryption.** The decryption algorithm Dec on input a secret key $sk = (\tau_f, \tau_g)$ and a ciphertext (c_h, c_f, c_g) first computes $m = \pi_2^{-1}\left(\mathsf{F}^{-1}(\tau_f, c_f)\right)$. Then, if $\mathsf{Enc}_{pk}(m) = (c_h, c_f, c_g)$ it outputs m, and otherwise it outputs $\perp$. In other words, the decryption algorithm inverts c_f using the trapdoor τ_f, and outputs m if the ciphertext is well-formed.

Theorem 6.1. *The scheme $\mathcal{DE}_{\mathrm{CCA}}$ is block-wise (p,T,k)-ACD-CCA-secure for any $n = n(\lambda)$, $\ell = \ell(\lambda)$, $v = v(\lambda)$, $p = p(\lambda)$, and $T = T(\lambda)$ by setting*

$$\begin{aligned} t_1 &= p + v + \log T + \omega(\log\lambda), & \delta_1 &= 2^{-nt_1}, \\ t_2 &= p + v + \log T + n - (2\ell - n) + \omega(\log\lambda), & \delta_2 &= 2^{-nt_2}, \\ k &= \max\big(n - (2\ell - n), v\big) + 2\log t_2 + \omega(\log\lambda). \end{aligned}$$

Parameters. Using existing constructions of admissible hash functions and lossy trapdoor functions (see Sections 2.2 and 2.3, respectively), and using our construction of $\mathcal{R}^{\mathtt{BM}}$-lossy trapdoor functions (see Section 5), for any $n = n(\lambda)$ and for any constant $0 < \epsilon < 1$ we can instantiate our scheme with $v = n^\epsilon$ and $\ell = n - n^\epsilon$. Therefore, for any $n = n(\lambda)$, $p = p(\lambda)$, and $T = T(\lambda)$, we obtain schemes with $t_1 = p + n^\epsilon + \omega(\log\lambda)$, $\delta_1 = 2^{-nt_1}$, $t_2 = p + 2n^{2\epsilon} + \omega(\log\lambda)$, $\delta_2 = 2^{-nt_2}$, and $k = 2n^{2\epsilon} + \omega(\log\lambda)$.

References

1. Bellare, M., Boldyreva, A., O'Neill, A.: Deterministic and efficiently searchable encryption. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 535–552. Springer, Heidelberg (2007)

2. Bellare, M., Brakerski, Z., Naor, M., Ristenpart, T., Segev, G., Shacham, H., Yilek, S.: Hedged public-key encryption: How to protect against bad randomness. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 232–249. Springer, Heidelberg (2009)
3. Bellare, M., Fischlin, M., O'Neill, A., Ristenpart, T.: Deterministic encryption: Definitional equivalences and constructions without random oracles. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 360–378. Springer, Heidelberg (2008)
4. Boldyreva, A., Fehr, S., O'Neill, A.: On notions of security for deterministic encryption, and efficient constructions without random oracles. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 335–359. Springer, Heidelberg (2008)
5. Boneh, D., Boyen, X.: Secure identity based encryption without random oracles. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 443–459. Springer, Heidelberg (2004)
6. Boyle, E., Segev, G., Wichs, D.: Fully leakage-resilient signatures. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 89–108. Springer, Heidelberg (2011)
7. Brakerski, Z., Segev, G.: Better security for deterministic public-key encryption: The auxiliary-input setting. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 543–560. Springer, Heidelberg (2011)
8. Cash, D., Hofheinz, D., Kiltz, E., Peikert, C.: Bonsai trees, or how to delegate a lattice basis. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 523–552. Springer, Heidelberg (2010)
9. Dodis, Y.: Exposure-Resilient Cryptography. PhD thesis, MIT (2000)
10. Dodis, Y., Smith, A.: Correcting errors without leaking partial information. In: STOC, pp. 654–663 (2005)
11. Dodis, Y., Smith, A.: Entropic security and the encryption of high entropy messages. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 556–577. Springer, Heidelberg (2005)
12. Freeman, D., Goldreich, O., Kiltz, E., Rosen, A., Segev, G.: More constructions of lossy and correlation-secure trapdoor functions. J. Cryptology 26(1), 39–74 (2013)
13. Fuller, B., O'Neill, A., Reyzin, L.: A unified approach to deterministic encryption: New constructions and a connection to computational entropy. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 582–599. Springer, Heidelberg (2012)
14. Goldwasser, S., Micali, S.: Probabilistic encryption. Journal of Computer and System Sciences 28(2), 270–299 (1984)
15. Kaplan, E., Naor, M., Reingold, O.: Derandomized constructions of k-wise (almost) independent permutations. Algorithmica 55(1), 113–133 (2009)
16. Kiltz, E., O'Neill, A., Smith, A.: Instantiability of RSA-OAEP under chosen-plaintext attack. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 295–313. Springer, Heidelberg (2010)
17. Mironov, I., Pandey, O., Reingold, O., Segev, G.: Incremental deterministic public-key encryption. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 628–644. Springer, Heidelberg (2012)
18. Peikert, C., Waters, B.: Lossy trapdoor functions and their applications. SIAM Journal on Computing 40(6), 1803–1844 (2011)
19. Raghunathan, A., Segev, G., Vadhan, S.: Deterministic public-key encryption for adaptively chosen plaintext distributions. Cryptology ePrint Archive, Report 2013/125 (2013)

20. Russell, A., Wang, H.: How to fool an unbounded adversary with a short key. IEEE Transactions on Information Theory 52(3), 1130–1140 (2006)
21. Trevisan, L., Vadhan, S.P.: Extracting randomness from samplable distributions. In: Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science, pp. 32–42 (2000)
22. Waters, B.: Efficient identity-based encryption without random oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)
23. Wee, H.: Dual projective hashing and its applications — lossy trapdoor functions and more. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 246–262. Springer, Heidelberg (2012)
24. Wichs, D.: Barriers in cryptography with weak, correlated and leaky sources. In: Proceedings of the 4th ITCS (2013)

How to Watermark Cryptographic Functions

Ryo Nishimaki

NTT Secure Platform Laboratories
nishimaki.ryo@lab.ntt.co.jp

Abstract. We propose a scheme for watermarking cryptographic functions. Informally speaking, a digital watermarking scheme for cryptographic functions embeds information, called a *mark*, into functions such as (trapdoor) one-way functions and decryption functions of public-key encryption. It is required that a mark-embedded function is functionally equivalent to the original function and it is difficult for adversaries to remove the embedded mark without damaging the function. In spite of its importance and usefulness, there have only been a few theoretical studies on watermarking for functions (or program), and we do not have rigorous and meaningful definitions of watermarking for cryptographic functions and concrete constructions.

To solve the above problem, we introduce a notion of watermarking for cryptographic functions and define its security. We present a lossy trapdoor function (LTF) based on the decisional linear (DLIN) problem and a watermarking scheme for the LTF. Our watermarking scheme is secure under the DLIN assumption in the standard model. We use the techniques of dual system encryption and dual pairing vector spaces (DPVS) to construct our watermarking scheme. This is a new application of DPVS.

Keywords: digital watermarking, dual pairing vector space, dual system encryption, vector decomposition problem.

1 Introduction

1.1 Background

Digital watermarking is a method of embedding information, called a “mark”, in digital objects such as images, movies, and audio files. Marked objects look similar to the original objects and it is difficult to remove embedded marks without destroying the object. One of the applications of watermarking is protecting copyright, i.e., we can prevent unauthorized copying of digital content by detecting watermarks. Another application is tracing and identifying owners of digital content, that is, if we find illegally copied digital content, we can detect a watermark and identify the owner who distributed the illegal copy. Most watermarking methods have been designed for perceptual objects, such as images, and only a few studies have focused on watermarking for non-perceptual objects (e.g., software, program). Software is digital content, so it can be easily copied. Software piracy is a serious problem today. Watermarking for programs is one of tools to solve the problem and has very useful, attractive, and practical applications, but they are little understood.

We briefly explain related studies on program watermarking below. Naccache, Shamir, and Stern introduced the notion of copyrighted functions and proposed a method for tracking different copies of functionally equivalent algorithms containing "marks" [14]. This is related to watermarking schemes for program (functions), but their security definition is a bit weak and not sufficient for program watermarking. Barak, Goldreich, Impagliazzo, Rudich, Sahai, Vadhan, and Yang considered the notion of software watermarking (program watermarking) from a cryptographic point of view in their seminal work [1]. They proposed a formalization of software watermarking and its security definition, but the definition is simulation based and too strong. They gave an impossibility result for general-purpose program watermarking by using impossibility results of general-purpose program obfuscation [1]. "General-purpose" means that program watermarking/obfuscation can be applied to *any* program. Their security requirements cannot be achieved, so they leave positive theoretical results about watermarking (achieving concrete constructions for specific function families by using a game-based security definition) as an open problem. Yoshida and Fujiwara introduced the notion of *watermarking for cryptographic data* and a concrete scheme for signatures [23]. Their idea is very exciting, but they did not propose a formal security definition of watermarking for cryptographic data and their scheme is not provably secure. They claim that the security of their scheme is based on the *vector decomposition (VD)* problem, which was introduced by Yoshida, Mitsunari, and Fujiwara [24], but their proof is heuristic and they showed no reduction.

Hopper, Molnar, and Wagner proposed a rigorous complexity-theoretic (game-based) definition of security for watermarking schemes, but they focused on watermarking for only *perceptual* objects [8]. They gave no concrete construction that satisfies their security definition.

1.2 Motivations and Applications

As explained in the previous section, there is no watermarking scheme for (cryptographic) functions that is provably secure in a complexity-theoretic definition of security. We consider functions as a kind of program. Copyrighted functions by Naccache et al. are provably secure based on the factoring assumption, but their definition of security is weaker than that of watermarking, and their construction can embed a *bounded* number of marks [14].

Traceable Cryptographic Primitives. One application of watermarking for cryptographic functions (we sometimes call it cryptographic watermarking) is constructing various *traceable cryptographic primitives*. If we have a watermarking scheme for cryptographic functions, for example, trapdoor one-way functions, collision-resistant hash functions (CRHF), and decryption functions, we can construct a variety of traceable primitives or copyrighted cryptographic primitives since private-key encryption, public-key encryption (PKE), digital signatures, and so on are constructed from trapdoor one-way functions and often use CRHFs in their algorithms.

As pointed out by Naccache et al., watermarked functions have the following applications [14]:

- If we consider software or program that generates ciphertexts of the Feistel cipher based on a one-way function [13], signatures of Rompel's signature scheme [21], or decrypted values of ciphertexts under PKE schemes based on a trapdoor one-way function, and a malicious user illegally makes copies of such software and distributes them, then a company that sold the software can trace them and identify the guilty users.
- If a company sells MAC-functions based on watermarked one-way functions to users and records user IDs and marked functions in a database and the users use them to log-in a member web site, then they do not need to disclose their identity since all marked functions are functionally equivalent. However, if a malicious user distributes an illegal copy and it is discovered, then the company can identify the guilty identity by detecting an embedded mark.

Black-box Traitor Tracing. Kiayias and Yung proposed a method of constructing black-box traitors tracing schemes from copyrighted PKE functions [10]. When we want to broadcast digital content to a set of legitimate subscribers, we can use broadcast encryption schemes. If some of the subscribers leak partial information about their decryption keys to a pirate, who is a malicious user in broadcast encryption systems, then the pirate may be able to construct a pirate-decoder. Traitor tracing enables us to identify such malicious subscribers called traitor [3]. Our cryptographic watermarking scheme can be seen as a generalized notion of copyrighted functions and our construction is based on identity-based encryption (IBE) schemes whose private keys for identities are marked (these are copyrighted decryption functions of PKE), so our construction technique can be used to construct *black-box traitor tracing* schemes and it has a quite powerful application.

Theoretical Treatment of Watermarking. There are many heuristic methods for software watermarking [4], but there have only been a few studies that theoretically and rigorously treat the problem in spite of its importance. Functions can be seen as a kind of software (and program) and a large amount of software uses cryptographic functions, especially in a broadcast system, users must use software with decryption functions to view content. We believe that our scheme is an important step toward constructing practical software watermarking.

1.3 Our Contributions and Construction Ideas

We introduce the notion of watermarking for cryptographic functions, a game-based security definition of them, and a concrete construction. Our watermarking scheme is provably secure under the decisional linear (DLIN) assumption. To the best of our knowledge, this is the first provably secure watermarking scheme for functions (program) in terms of theoretical cryptography.

Our security notion is based on the notion of strong watermarking introduced by Hopper et al. [8]. Their definition takes into account only perceptual objects and they modeled the notion of similarity by a perceptual metric space on objects that measures the distance between objects. Therefore, to construct watermarking schemes for cryptographic functions, we should modify their definition. We define the similarity by preserving functionality, that is, if a marked function is functionally equivalent to an original function, then we assume the marked function is similar to the original function. Watermarking schemes should guarantee that no adversary can generate a function which is functionally equivalent to a marked function but unmarked, that is, no adversary can remove embedded marks without destroying functions.

We propose a watermarking scheme for lossy trapdoor functions (LTFs) [20]. LTFs are quite powerful cryptographic functions. They imply standard trapdoor one-way functions, oblivious transfers, CRHFs, and secure PKE schemes against adaptive chosen ciphertext attacks (CCA) [20]. The watermarking scheme consists of four algorithms, key generation, mark, detect, and remove algorithms. Marked function indices are functionally equivalent to the original ones, that is, for any input, outputs of marked functions are the same as those of the original function. The construction can be used to construct an IBE scheme that can generate marked private keys for identities and marked signatures since our LTFs are based on IBE schemes, as explained in the next paragraph. That is, we can construct decryption algorithms in which watermarks can be embedded.

Key Techniques and Ideas Behind Our Construction. Our construction is based on the dual pairing vector space (DPVS) proposed by Okamoto and Takashima [16,17,19]. We use the IBE scheme of Okamoto and Takashima [19] (which is a special case of their inner-product predicate encryption (IPE) scheme) and that of Lewko [11] to construct LTFs. Loosely speaking, LTFs are constructed from homomorphic encryption schemes, and the IBE schemes of Okamoto-Takashima and Lewko are homomorphic. There are many other homomorphic encryption schemes but we selected Okamoto-Takashima and Lewko IBE schemes because they are constructed by DPVS and the dual system encryption methodology introduced by Waters [22]. The methodology is a key technique to achieve a watermarking scheme.

We apply the dual system encryption technique to not only security proofs but also *constructions of cryptographic primitives*. In the dual system encryption, we can use *semi-functional ciphertexts* and *semi-functional keys*. Semi-functional ciphertexts can be decrypted using normal keys and normal ciphertext can be decrypted using semi-functional keys, but semi-functional ciphertexts cannot be decrypted using semi-functional keys. Normal ciphertexts/keys are computationally indistinguishable from semi-functional ciphertexts/keys. In most cases, function indices of LTFs consist of *ciphertexts of homomorphic encryption* [5,7,20], so, intuitively speaking, if we can construct a function index by using not only (normal) ciphertexts but also semi-functional keys, then the function index is functionally equivalent to a function index generated by (normal ciphertexts and) normal keys as long as normal ciphertexts are used. Moreover, if we use semi-functional ciphertexts, we can determine whether a function index is generated by semi-functional keys or not since semi-functional ciphertexts cannot be decrypted using semi-functional key. Thus, a function index that consists of semi-functional keys can be seen as a marked index and semi-functional ciphertexts can be used in a detection algorithm of a watermarking scheme. This is the main idea. Note that our construction technique can be used to construct an IBE scheme whose private keys can be marked because our LTFs are based on such an IBE scheme.

Our watermarking scheme is based on DPVS. We can set a hidden linear subspace by concealing the basis of a subspace from public parameters due to a nice property of DPVS. A pair of dual orthonormal bases, $\mathbb{B}$ and $\mathbb{B}^*$, are generated using a random linear transformation matrix. We use a hidden linear subspace spanned by a subset of $\mathbb{B}$/$\mathbb{B}^*$ for semi-functional ciphertexts/keys (We denote the subset by $\widehat{\mathbb{B}} \subset \mathbb{B}$, $\widehat{\mathbb{B}}^* \subset \mathbb{B}^*$, respectively). A hidden linear subspace for semi-functional ciphertexts and keys can be used as a detect key and a mark key of our watermarking scheme, respectively. Thus, we can embed "marks" into the hidden linear subspace and they are indistinguishable from non-marked objects because the decisional subspace problem is believed to be hard [15, 17]. Informally speaking, the decisional subspace problem is determining whether a given vector is spanned by $\mathbb{B}$ (resp, $\mathbb{B}^*$) or $\mathbb{B} \setminus \widehat{\mathbb{B}}$ (resp, $\mathbb{B}^* \setminus \widehat{\mathbb{B}}^*$).

Okamoto and Takashima introduced complexity problems based on the DLIN problem to prove the security of their scheme [17, 19] and these problems are deeply related to the VD problem [24] and the decisional subspace problem. The VD problem says that it is difficult to decompose a vector in DPVS into a vector spanned by bases of a subspace. Lewko also introduced the subspace assumption [11], which is implied by the DLIN assumption and highly related to the decisional subspace assumption introduced by Okamoto and Takashima [15] and the VD problem. All assumptions introduced by Okamoto-Takashima [17, 19] and Lewko [11] are implied by the standard DLIN assumption.

If we can decompose a vector in DPVS into each linearly independent vector, then we can convert semi-functional ciphertexts/keys into normal ciphertexts/keys by eliminating elements in hidden linear subspaces, that is, we can remove an embedded mark from a marked function index. Galbraith and Verheul and Yoshida, Mitsunari, and Fujiwara argued that the VD problem is related to computational Diffie-Hellman problem [6, 24]. It is believed that the VD problem is hard. Therefore, no adversary can remove marks of our watermarking scheme (this is a just intuition). However, we do not directly use the VD problem but the DLIN problem to prove the security of our scheme. On the other hand, if we have a linear transformation matrix behind dual orthonormal bases of DPVS, then we can easily solve the VD problem [15, 17], that is, we can remove a mark if we have the matrix. Such an algorithm was proposed by Okamoto and Takashima [15].

Our construction is a new application of DPVS. DPVS has been used to construct fully secure functional encryption, IPE, IBE and attribute-based signatures [11, 12, 16–19], but to the best of our knowledge, a linear transformation matrix for dual orthonormal bases in DPVS has never been explicitly used for algorithms of cryptographic schemes. This is of independent interest.

Remark. In this extended abstract, we do not have enough space to give complete proofs and all definitions, so we omitted some of them.

2 Preliminaries

Notations. For any $n \in \mathbb{N} \setminus \{0\}$, let $[n]$ be the set $\{1, \ldots, n\}$. When D is a random variable or distribution, $y \xleftarrow{\mathsf{R}} D$ means that y is randomly selected from D according to its distribution. If S is a set, then $x \xleftarrow{\mathsf{U}} S$ means that x is uniformly selected from S. We denote y is set, defined or substituted by z by $y := z$. When b is a fixed value, $A(x) \to b$ (e.g., $A(x) \to 1$) denotes the event that probabilistic polynomial-time (PPT) machine (or algorithm) A outputs a on input x. We say that function $f : \mathbb{N} \to \mathbb{R}$ is negligible in $\lambda \in \mathbb{N}$ if $f(\lambda) = \lambda^{-\omega(1)}$ (We write $f < \mathsf{negl}(\lambda)$). We denote the finite field of order q by $\mathbb{F}_q$, and $\mathbb{F}_q \setminus \{0\}$ by $\mathbb{F}_q^\times$. A bold face small letter denotes an element of vector space $\mathbb{V}$, e.g., $\boldsymbol{x} \in \mathbb{V}$. Set $GL(n, \mathbb{F}_q)$ denotes the general linear group of degree n over $\mathbb{F}_q$. Let $\mathcal{G}_{\mathsf{bm}}$ be a parameter generation algorithm that takes as input security parameter λ and outputs $(q, \mathbb{G}, \mathbb{G}_T, e, g)$. If we use g/G to denote a generator in $\mathbb{G}$, then we use multiplicative/additive notation, respectively.

2.1 Function Family of Lossy Trapdoor Functions

Definition 1 (Lossy Trapdoor Functions [20]). *A lossy trapdoor function* LTF *with domain* D *consists of four efficient algorithms satisfying three properties*

Injective Key Generation: $\mathsf{LTF.IGen}$ *outputs* (ek, ik) *where* ek/ik *is an evaluation/inversion key.*

Evaluation: $\mathsf{LTF.Eval}_{ek}(X)$ *(*$X \in \mathsf{D}$*) outputs an image* $Y = f_{ek}(X)$.

Inversion: $\mathsf{LTF.Invert}_{ik}(Y)$ *outputs a pre-image* $X = f_{ik}^{-1}(Y)$.

Lossy Key Generation: $\mathsf{LTF.LGen}$ *outputs* $(ek', \bot)$.

Correctness: $\forall (ek, ik) \xleftarrow{\mathsf{R}} \mathsf{LTF.IGen}(1^\lambda)$ *and* $\forall X \in \mathsf{D}$, $f_{ik}^{-1}(f_{ek}(X)) = X$.

Indistinguishability: *Let* λ *be a security parameter. For all PPT* $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{IND}}_{\mathsf{LTF},\mathcal{A}}(\lambda) := \left|\Pr[\mathcal{A}(1^\lambda, [\mathsf{LTF.IGen}(1^\lambda)]_1)] - \Pr[\mathcal{A}(1^\lambda, [\mathsf{LTF.LGen}(1^\lambda)]_1)]\right| < \mathsf{negl}(\lambda)$.

Lossiness: *We say that* LTF *is* ℓ*-lossy if for all* ek' *generated by using* $\mathsf{LTF.LGen}(1^\lambda)$, *the image set* $f_{ek'}(\mathsf{D})$ *is of size at most* $|\mathsf{D}|/2^\ell$.

We define a function family of LTF, $\mathsf{LTF}_\lambda := \{\mathsf{LTF.Eval}_{ek}(,\cdot) | (ek, ik) \xleftarrow{\mathsf{R}} \mathsf{LTF.Gen}(1^\lambda, b), b \in \{0,1\}\}$ where $\mathsf{LTF.Gen}(1^\lambda, 0) := \mathsf{LTF.IGen}(1^\lambda)$ and $\mathsf{LTF.Gen}(1^\lambda, 1) := \mathsf{LTF.LGen}(1^\lambda)$.

2.2 Dual Pairing Vector Space [12,16,17]

Definition 2. *"Dual pairing vector space (DPVS)"* $(q, \mathbb{V}, \mathbb{G}_T, \mathbb{A}, e)$ *by a direct product of symmetric pairing groups* $(q, \mathbb{G}, \mathbb{G}_T, e, G)$ *are a tuple of prime* q, N*-dimensional vector space* $\mathbb{V} := \mathbb{G}^N$ *over* $\mathbb{F}_q$, *cyclic group* $\mathbb{G}_T$ *of order* q, *canonical basis* $\mathbb{A} := (\boldsymbol{a}_1, \ldots, \boldsymbol{a}_N)$ *of* $\mathbb{V}$, *where* $\boldsymbol{a}_i := (0, \ldots, 0, G, 0, \ldots, 0)$ *(only the* i*-th coordinate is* G*), and pairing* $\boldsymbol{e} : \mathbb{V} \times \mathbb{V} \to \mathbb{G}_T$. *The pairing is defined as* $\boldsymbol{e}(\boldsymbol{x}, \boldsymbol{y}) := \prod_{i=1}^N e(G_i, H_i) \in \mathbb{G}_T$ *where* $\boldsymbol{x} := (G_1, \ldots, G_N) \in \mathbb{V}$ *and* $\boldsymbol{y} := (H_1, \ldots, H_N) \in \mathbb{V}$. *This is non-degenerate bilinear, i.e.,* $\boldsymbol{e}(s\boldsymbol{x}, t\boldsymbol{y}) = e(\boldsymbol{x}, \boldsymbol{y})^{st}$ *and if* $\boldsymbol{e}(\boldsymbol{x}, \boldsymbol{y}) = 1$ *for all* $\boldsymbol{y} \in \mathbb{V}$, *then* $\boldsymbol{x} = \boldsymbol{0}$. *For all* i *and* j, $\boldsymbol{e}(\boldsymbol{a}_i, \boldsymbol{a}_j) = e(G, G)^{\delta_{i,j}}$ *where* $\delta_{i,j} = 1$ *if* $i = j$, *and* 0 *otherwise, and* $e(G, G) \neq 1$. *DPVS also has linear transformations* $\phi_{i,j}$ *on* $\mathbb{V}$ *s.t.* $\phi_{i,j}(\boldsymbol{a}_j) = \boldsymbol{a}_i$ *and* $\phi_{i,j}(\boldsymbol{a}_k) = \boldsymbol{0}$ *if* $k \neq j$, *which can be easily achieved by* $\phi_{i,j}(\boldsymbol{x}) := (0, \ldots, 0, G_j, 0, \ldots, 0)$ *(only the* i*-th coordinate is* G*) where* $\boldsymbol{x} := (G_1, \ldots, G_N)$. *We call* $\phi_{i,j}$ *canonical maps. DPVS generation algorithm* $\mathcal{G}_{\mathsf{dpvs}}$ *takes input* 1^λ *and* $N \in \mathbb{N}$ *and outputs a description of* $\mathsf{pp}'_{\mathbb{V}} := (q, \mathbb{V}, \mathbb{G}_T, \mathbb{A}, e)$ *with security parameter* λ *and* N*-dimensional* $\mathbb{V}$. *It can be constructed using* $\mathcal{G}_{\mathsf{bm}}$.

Canonical basis $\mathbb{A}$ is changed to *dual orthonormal bases* $\mathbb{B} := (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_N)$ and $\mathbb{B}^* := (\boldsymbol{b}_1^*, \ldots, \boldsymbol{b}_N^*)$ of $\mathbb{V}$. We describe random dual orthonormal bases generator $\mathcal{G}_{\mathsf{ob}}(1^\lambda, N)$: Generate $\mathsf{pp}'_{\mathbb{V}} := (q, \mathbb{V}, \mathbb{G}_T, \mathbb{A}, e) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathsf{dpvs}}(1^\lambda, N)$, $\boldsymbol{X} := (\chi_{i,j}) \xleftarrow{\mathsf{U}} GL(N, \mathbb{F}_q)$, $\psi \xleftarrow{\mathsf{U}} \mathbb{F}_q^\times$, $(\vartheta_{i,j}) := \psi(\boldsymbol{X}^\top)^{-1}$, $g_T := e(G,G)^\psi$, $\mathsf{pp}_{\mathbb{V}} := (\mathsf{pp}'_{\mathbb{V}}, g_T)$, $\boldsymbol{b}_i := \sum_{j=1}^N \chi_{i,j}\boldsymbol{a}_j$, $\mathbb{B} := (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_N)$, $\boldsymbol{b}_i^* := \sum_{j=1}^N \vartheta_{i,j}\boldsymbol{a}_j$, $\mathbb{B}^* := (\boldsymbol{b}_1^*, \ldots, \boldsymbol{b}_N^*)$, return $(\mathsf{pp}_{\mathbb{V}}, \mathbb{B}, \mathbb{B}^*)$. It holds that $\boldsymbol{e}(\boldsymbol{b}_i, \boldsymbol{b}_j^*) = (g_T)^{\delta_{i,j}}$.

Vector Decomposition Problem. The VD problem was originally introduced by Yoshida, Mitsunari, and Fujiwara [24]. We present the definition of a higher dimensional version by Okamoto and Takashima [15] to fit the VD problem into DPVS. Note that a specific class of the CVDP instances that are specified over canonical basis $\mathbb{A}$ are tractable.

Definition 3 (CVDP: (ℓ_1, ℓ_2)-Computational Vector Decomposition Problem [15]). *For $\ell_1 > \ell_2$ and all $\lambda \in \mathbb{N}$, we define the advantage*

$$\mathsf{Adv}^{\mathsf{CVDP}}_{\mathcal{A},(\ell_1,\ell_2)}(\lambda) := \Pr\left[\omega = \sum_{i=1}^{\ell_2} x_i \boldsymbol{b}_i \;\middle|\; \begin{array}{l} (\mathsf{pp}_{\mathbb{V}}, \mathbb{B}, \mathbb{B}^*) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathsf{ob}}(1^\lambda, \ell_1), \\ (x_1, \ldots, x_{\ell_1}) \xleftarrow{\mathsf{U}} (\mathbb{F}_q)^{\ell_1}, \\ \boldsymbol{v} := \sum_{i=1}^{\ell_1} x_i \boldsymbol{b}_i, \omega \xleftarrow{\mathsf{R}} \mathcal{A}(1^\lambda, \mathsf{pp}_{\mathbb{V}}, \mathbb{B}, \boldsymbol{v}) \end{array}\right].$$

The $\mathsf{CVDP}_{(\ell_1,\ell_2)}$ *assumption : For any PPT $\mathcal{A}$,* $\mathsf{Adv}^{\mathsf{CVDP}}_{\mathcal{A},(\ell_1,\ell_2)}(\lambda) < \mathsf{negl}(\lambda)$.

Trapdoor. If we have a trapdoor, matrix $\boldsymbol{X}$ behind $\mathbb{B}$, then we can efficiently decompose vectors in DPVS, i.e., solve $\mathsf{CVDP}_{(\ell_1,\ell_2)}$ by using the efficient algorithm Decomp given by Okamoto and Takashima [15]. The input is $(\boldsymbol{v}, (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{\ell_2}), \boldsymbol{X}, \mathbb{B})$ such that $\boldsymbol{v} := \sum_{i=1}^{\ell_1} y_i \boldsymbol{b}_i$ is a target vector for decomposition, $(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{\ell_2})$ is a subspace to be decomposed into, $\boldsymbol{X}$ is a trapdoor, and $\mathbb{B} := (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{\ell_1})$ is a basis generated by using $\boldsymbol{X}$. Algorithm $\mathsf{Decomp}(\boldsymbol{v}, (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{\ell_2}), \boldsymbol{X}, \mathbb{B})$: computes $\boldsymbol{u} := \sum_{i=1}^{\ell_1} \sum_{j=1}^{\ell_2} \sum_{\kappa=1}^{\ell_1} \tau_{i,j}\chi_{j,\kappa}\phi_{\kappa,i}(\boldsymbol{v})$ where ϕ is the canonical map in Definition 2, $(\chi_{i,j}) = \boldsymbol{X}$ and $(\tau_{i,j}) := (\boldsymbol{X})^{-1}$.

Lemma 1 (Okamoto-Takashima [15]). *Algorithm* Decomp *efficiently solves* $\mathsf{CVDP}_{(\ell_1,\ell_2)}$ *by using $\boldsymbol{X} := (\chi_{i,j})$ such that $\boldsymbol{b}_i := \sum_{j=1}^{\ell_1} \chi_{i,j}\boldsymbol{a}_j$.*

Multiplicative Notation of DPVS by Lewko [11]. We introduce a multiplicative notation by Lewko [11]. For $\vec{v}, \vec{w} \in \mathbb{F}_q^n$, $a \in \mathbb{F}_q$, and $g \in \mathbb{G}$, we define $g^{\vec{v}} := (g^{v_1}, \ldots, g^{v_n})$, $g^{a\vec{v}} := (g^{av_1}, \ldots, g^{av_n})$, $g^{\vec{v}+\vec{w}} := (g^{v_1+w_n}, \ldots, g^{v_n+w_n})$, and $\boldsymbol{e}(g^{\vec{v}}, g^{\vec{w}}) := \prod_{i=1}^n e(g^{v_i}, g^{w_i})$. Lewko defined algorithm $\mathsf{Dual}(\mathbb{F}_q^n)$ as follows: It chooses $\vec{b}_i, \vec{b}_j^* \in \mathbb{F}_q^n$ and $\psi \xleftarrow{\mathsf{U}} \mathbb{F}_q$ such that $\vec{b}_i \cdot \vec{b}_j^* = 0 \bmod q$ for $i \neq j$, $\vec{b}_i \cdot \vec{b}_i^* = \psi \bmod q$ for all $i \in [n]$ and outputs $(\mathfrak{B}, \mathfrak{B}^*)$ where $\mathfrak{B} := (\vec{b}_1, \ldots, \vec{b}_n)$ and $\mathfrak{B}^* := (\vec{b}_1^*, \ldots, \vec{b}_n^*)$. We use the notation $(\mathfrak{B}, \mathfrak{B}^*)$ to express dual orthonormal bases in $\mathbb{F}_q$ to distinguish from bases $(\mathbb{B}, \mathbb{B}^*)$ in $\mathbb{V}$. We can consider $\boldsymbol{b}_i = g^{\vec{b}_i}$, $\boldsymbol{b}_j^* = g^{\vec{b}_j^*}$, $\vec{b}_i = (\chi_{i,1}, \ldots, \chi_{i,n})$, $\vec{b}_i^* = (\vartheta_{i,1}, \ldots, \vartheta_{i,n})$ where $\boldsymbol{X} = (\chi_{i,j})$ and $\psi(\boldsymbol{X}^{-1})^\top = (\vartheta_{i,j})$.

2.3 Complexity Assumptions

Definition 4 (DLIN Assumption). *Let $\mathcal{G}_{\mathsf{b}}^{\mathsf{dlin}}(1^\lambda)$ be an algorithm that generates $\Gamma := (q, \mathbb{G}, \mathbb{G}_T, e, g) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathsf{bm}}(1^\lambda)$, chooses $\xi, \kappa, \delta, \sigma, \zeta \xleftarrow{\mathsf{U}} \mathbb{F}_q$, lets $Q_0 := g^{\delta+\sigma}$, $Q_1 := g^\zeta$,* *and outputs* $\mathcal{I} := (\Gamma, f, h, f^{\delta}, h^{\sigma}, Q_{\mathsf{b}})$*. The advantage is* $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{dlin}}(\lambda) := |\Pr[\mathcal{A}(\mathcal{I}) \to 1 | \mathcal{I} \xleftarrow{\mathsf{R}} \mathcal{G}_0^{\mathsf{dlin}}(1^{\lambda})] - \Pr[\mathcal{A}(\mathcal{I}) \to 1 | \mathcal{I} \xleftarrow{\mathsf{R}} \mathcal{G}_1^{\mathsf{dlin}}(1^{\lambda})]|$*. We say that the DLIN assumption holds if for all PPT* $\mathcal{A}$*,* $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{dlin}}(\lambda) < \mathsf{negl}(\lambda)$*.*

Definition 5 (Subspace Assumption). *Let* $\mathcal{G}_{\mathsf{b}}^{\mathsf{dss}}(1^{\lambda})$ *be an algorithm that generates* $\Gamma \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathsf{bm}}(1^{\lambda})$*, chooses* $\eta, \beta, \tau_1, \tau_2, \tau_3, \mu_1, \mu_2, \mu_3 \xleftarrow{\mathsf{U}} \mathbb{F}_q$*,* $(\mathfrak{B}, \mathfrak{B}^*) \xleftarrow{\mathsf{R}} \mathsf{Dual}(\mathbb{F}_q^n)$*, for* $i \in [k]$ *where* $3k \leq n$ *lets* $U_i := g^{\mu_1 \vec{b}_i + \mu_2 \vec{b}_{k+i} + \mu_3 \vec{b}_{2k+i}}$*,* $V_i := g^{\tau_1 \eta \vec{b}_i^* + \tau_2 \beta \vec{b}_{k+i}^*}$*,* $W_i := g^{\tau_1 \eta \vec{b}_i^* + \tau_2 \beta \vec{b}_{k+i}^* + \tau_3 \vec{b}_{2k+i}^*}$*,* $D := (g^{\vec{b}_1}, \dots, g^{\vec{b}_{2k}}, g^{\vec{b}_{3k+1}}, \dots, g^{\vec{b}_n}, g^{\eta \vec{b}_1^*}, \dots, g^{\eta \vec{b}_k^*}, g^{\beta \vec{b}_{k+1}^*}, \dots, g^{\beta \vec{b}_{2k}^*}, g^{\vec{b}_{2k+1}^*}, \dots, g^{\vec{b}_n^*}, U_1, \dots, U_k, \mu_3)$*,* $Q_0 := (V_1, \dots, V_k)$*,* $Q_1 := (W_1, \dots, W_k)$*, and outputs* $\mathcal{I} := (\Gamma, D, Q_{\mathsf{b}})$*. The advantage is* $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{dss}}(\lambda) := |\Pr[\mathcal{A}(\mathcal{I}) \to 1 | \mathcal{I} \xleftarrow{\mathsf{R}} \mathcal{G}_0^{\mathsf{dss}}(1^{\lambda})] - \Pr[\mathcal{A}(\mathcal{I}) \to 1 | \mathcal{I} \xleftarrow{\mathsf{R}} \mathcal{G}_1^{\mathsf{dss}}(1^{\lambda})]|$*. We say that the subspace assumption holds if for all PPT* $\mathcal{A}$*,* $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{dss}}(\lambda) < \mathsf{negl}(\lambda)$*.*

Theorem 1 (Lewko [11]). *The DLIN assumption implies the subspace assumption.*

3 Definitions of Cryptographic Watermarking

We define watermarking schemes for cryptographic functions (one-way functions, hash functions, etc.). Our definition of watermarking schemes can be extended to treat cryptographic data introduced by Yoshida and Fujiwara [23]. We consider a family of functions $\mathcal{F} := \{\mathcal{F}_{\lambda}\}_{\lambda}$. For example, LTFs are cryptographic functions and function F is sampled from family $\mathsf{LTF}_{\lambda} := \{f_{ek}(\cdot) | (ek, ik) \xleftarrow{\mathsf{R}} \mathsf{LTF.IGen}(1^{\lambda})\}$. A watermarking key generation algorithm takes as inputs security parameter λ and family $\mathcal{F}$ and outputs public parameter pk, secret key sk, mark key mk, detect key dk, and remove key rk. That is, our watermarking schemes is an *asymmetric key* watermarking scheme. Public parameter pk includes sampling algorithm $\mathsf{Samp}_{\mathcal{F}}$, which outputs a function $F \xleftarrow{\mathsf{R}} \mathcal{F}_{\lambda}$ (note that we include the case in which the sampling algorithm takes sk as an input). Note that the description of $\mathsf{Samp}_{\mathcal{F}}$ does not include sk. Our cryptographic watermarking scheme for cryptographic functions $\mathcal{F}$ uses public parameter pk and secret key sk to choose a function $F \xleftarrow{\mathsf{R}} \mathcal{F}_{\lambda}$ from the function family. A mark key allows us to embed a mark in function F. A marked function F' should be similar to original function F. A detect/remove key allows us to detect/remove a mark in marked function F'.

Definition 6. *A watermarking scheme for family* $\mathcal{F}$ *is a tuple of algorithms* $\mathsf{CWM}_{\mathcal{F}} := \{\mathsf{WMGen}, \mathsf{Mark}, \mathsf{Detect}, \mathsf{Remove}\}$ *as follows:*

WMGen: *The key generation algorithm takes as input security parameter* λ *and function family* $\mathcal{F}$*, outputs public parameter* pk *(including sampling algorithm* $\mathsf{Samp}_{\mathcal{F}}$*), secret key* sk*, mark key* mk*, detect key* dk*, and remove key* rk*, that is,* $(pk, sk, mk, dk, rk) \xleftarrow{\mathsf{R}} \mathsf{WMGen}(1^{\lambda}, \mathcal{F})$*.*

Mark: *The mark algorithm takes as inputs* mk *and unmarked function* F *and outputs marked function* $\widetilde{F}$*, that is,* $\widetilde{F} \xleftarrow{\mathsf{R}} \mathsf{Mark}(pk, mk, F)$ *(hereafter, we often omit* pk *from inputs).*

Detect: *The detect algorithm takes as inputs* dk *and function* F' *and outputs* marked *(detect a mark) or* $\mathsf{unmarked}$ *(no mark), that is,* $\mathsf{Detect}(pk, dk, F') \to \mathsf{marked}/\mathsf{unmarked}$.

Remove**:** *The remove algorithm takes as inputs rk and marked function $\widetilde{F}$ and outputs unmarked function $F :=$ Remove$(pk, rk, \widetilde{F})$, that is functionally equivalent to the original one.*

As Hopper et al. noted [8], we do not allow any online communication between the Detect and Mark procedures. We sometimes use notation $\mathsf{WM}(F)$ to denote a marked function of F.

We define the security of cryptographic watermarking based on the definition of strong watermarking with respect to the metric space proposed by Hopper et al. [8] and software watermarking proposed by Barak et al. [1]. We borrow some terms from these studies. Hopper et al. defined a metric space equipped with distance function d and say that object O_1 and O_2 are similar if $d(O_1, O_2) \leq \delta$ for some δ, but we do not use it since we do not consider perceptual objects.

Basically, the following properties should be satisfied: Most objects $F \in \mathcal{F}_\lambda$ must be unmarked. We define similarity by a functional preserving property, that is, for all input x, output distributions $F(x)$ and $F'(x)$ are identical. Given marked function F', an adversary should not be able to construct a new function $\widetilde{F}$, which is functionally equivalent to F' but unmarked without remove key rk.

Our definitions of the non-removability and unforgeability are game-based definitions and based on the notion of strong watermarking by Hopper et al. [8]. Our definitions are specialized to focus on cryptographic functions (do not consider metric spaces). The non-removability states that even if the adversary is given marked functions, it cannot find a function that is similar to a marked function but does not contain any mark. This is based on the security against removal introduced by Hopper et al. [8]. The unforgeability states that the adversary cannot find a new marked function. This is based on the security against insertion introduced by Hopper et al. [8].

```
Experiment WMark_{F,A}(λ)
  $keys \xleftarrow{R} \mathsf{WMGen}(1^\lambda, \mathcal{F})$;
  $kesy = (pk, sk, mk, dk, rk)$;
  $\mathsf{MList} := \emptyset$; $\mathsf{CList} := \emptyset$;
  $(\beta, F) \xleftarrow{R} \mathcal{A}^{\mathcal{MO},\mathcal{CO},\mathcal{DO}}(1^\lambda, pk)$;
  If $\beta = 0$, then $\mathsf{Detect}(dk, F) \to b$;
  $\mathsf{IdealDtc}(F) \to B'$;
  if $b = \mathsf{unmarked}$ and $B' = \{\mathsf{marked}\}$;
  then return $(0, \mathsf{win})$   else return $\mathsf{lose}$
  If $\beta = 1$, then $\mathsf{Detect}(dk, F) \to b$;
  $\mathsf{IdealDtc}(F) \to B'$;
  if $b = \mathsf{marked}$, and $B' = \{\mathsf{unmarked}\}$;
  then return $(1, \mathsf{win})$   else return $\mathsf{lose}$
```

```
Oracle $\mathcal{MO}(F)$
  $F' \xleftarrow{R} \mathsf{Mark}(mk, F)$;
  $\mathsf{MList} := \mathsf{MList} \cup \{F'\}$;
  return $F'$;

Oracle $\mathcal{CO}_{\mathcal{F}_\lambda}()$
  $F \xleftarrow{R} \mathcal{F}_\lambda$;
  $F' \xleftarrow{R} \mathsf{Mark}(mk, F)$;
  $\mathsf{CList} := \mathsf{CList} \cup \{F'\}$;
  $\mathsf{MList} := \mathsf{MList} \cup \{F'\}$;
  return $F'$
```

```
Oracle $\mathcal{DO}(F)$
  $\mathsf{Detect}(dk, F) \to b$;
  return $b$

Procedure $\mathsf{IdealDtc}(F)$
  if
  $(\exists F' \in \mathsf{CList} : F \equiv F')$;
  then return $\{\mathsf{marked}\}$
  else if
  $(\exists F' \in \mathsf{MList} : F \equiv F')$
  then return
  $\{\mathsf{marked}, \mathsf{unmarked}\}$
  else return $\{\mathsf{unmarked}\}$
```

Fig. 1. Experiment for non-removability and unforgeability

Definition 7 (Secure Watermarking for Functions). *A watermarking scheme for function family $\mathcal{F}$ is secure if it satisfies the following properties.*

Meaningfulness: *It holds that for any $F \in \mathcal{F}_\lambda$,* $\mathsf{Detect}(dk, F) \to \mathsf{unmarked}$.

Correctness: *For any $F \in \mathcal{F}_\lambda$, $(pk, sk, mk, dk, rk) \overset{R}{\leftarrow} \mathsf{WMGen}(1^\lambda, \mathcal{F})$ and $\mathsf{WM}(F) \overset{R}{\leftarrow} \mathsf{Mark}(mk, F)$, it holds that $\mathsf{Detect}(dk, \mathsf{WM}(F)) \to \mathsf{marked}$ and* $\mathsf{Detect}(dk, \mathsf{Remove}(rk, \mathsf{WM}(F))) \to \mathsf{unmarked}$.

Preserving Functionality: *For any input $x \in \{0,1\}^n$ and $F \in \mathcal{F}_\lambda$, it holds that $\mathsf{WM}(F)(x) = F(x)$. If function F' preserves the functionality of function F, then we write $F \equiv F'$.*

Polynomial Slowdown: *There exists a polynomial p such that for any $F \in \mathcal{F}_\lambda$,* $|\mathsf{WM}(F)| \le p(|F| + |mk|)$.

Non-Removability: *We say that a watermarking scheme is non-removable if it holds that $\mathsf{Adv}^{\mathsf{Remove}}_{\mathcal{F},\mathcal{A}}(1^\lambda) := \Pr[\mathsf{WMark}_{\mathcal{F},\mathcal{A}}(\lambda) \to (0, \mathsf{win})] < \mathsf{negl}(\lambda)$. Experiment $\mathsf{WMark}_{\mathcal{F},\mathcal{A}}(\lambda)$ is shown in Figure 1.*

Unforgeability: *We say that a watermarking scheme is unforgeable if it holds that* $\mathsf{Adv}^{\mathsf{Forge}}_{\mathcal{F},\mathcal{A}}(1^\lambda) := \Pr[\mathsf{WMark}_{\mathcal{F},\mathcal{A}}(\lambda) \to (1, \mathsf{win})] < \mathsf{negl}(\lambda)$.

The adversary tries to find a function such that the outputs of the actual detection algorithm and the *ideal detection procedure* are different. The ideal detection procedure searches a database and outputs a decision by using online communication to the marking algorithm. The adversary has access to oracles, i.e., the mark, detect, and challenge oracles. The mark oracle returns a marked function for a queried non-marked function. The detect oracle determines whether a queried function is marked or not. The challenge oracle generates a new (non-marked) function, embeds a mark in the new function, and returns the marked function (the original non-marked function is hidden). Eventually, the adversary outputs function F and bit β. When $\beta = 0$, it means that the adversary claim that it succeeded in removing a mark from some marked function F' without the remove key. This case is for security against removal. When $\beta = 1$, it means that the adversary claim that it succeeded in embedding a mark in some original function F' without the mark key. This case is for security against forgery.

As Hopper et al. explained [8], we must introduce the challenge oracle because if it does not exist, then a trivial attack exists. If the adversary samples an unmarked function $F \in \mathcal{F}_\lambda$, queries it to the mark oracle, and finally outputs them as solutions for $\beta = 0$. The actual detect algorithm returns $\mathsf{unmarked}$ but the ideal detect procedure returns $\{\mathsf{marked}, \mathsf{unmarked}\}$ since an equivalent function is recorded in MList.

4 Proposed Watermarking Scheme

We present LTFs and a watermarking scheme for LTFs that are secure under the DLIN assumption. Generally speaking, LTFs can be constructed from homomorphic encryption schemes as discussed in many papers [5, 20]. Lewko/Okamoto-Takashima proposed an IBE/IPE scheme based on DPVS, which is homomorphic and secure under the DLIN assumption. We can easily construct an LTF from the IBE scheme by applying the matrix encryption technique introduced by Peikert and Waters [20]. In this extended abstract, we present a scheme based on only the Lewko IBE scheme due to page limitations (since cryptographers are used to multiplicative notation). See a full version of this paper for a scheme based on the Okamoto-Takashima IPE scheme.

Basically, we use homomorphic *PKE* schemes to construct LTFs, but we use homomorphic *IBE* schemes to achieve watermarking scheme since we want to use identities as tags for function indices and dual system encryption. To construct LTFs based on IBE schemes, we use not only ciphertexts under some identity but also a private key for the identity. If there is no private key (we call it conversion key in the scheme), then we cannot obtain valid outputs that can be inverted by an inversion key of the LTF. Note that the conversion key is *not a trapdoor inversion key for the LTF*. Our LTF $\mathsf{LTF}_{\mathsf{mult}}$ based on the Lewko IBE is as follows:

$\mathsf{LTF.IGen}(1^\lambda)$ **:** It generates $(\mathfrak{D}, \mathfrak{D}^*) \xleftarrow{\mathsf{U}} \mathsf{Dual}(\mathbb{F}_q^8)$, chooses $\alpha, \theta, \sigma \xleftarrow{\mathsf{U}} \mathbb{F}_q$, $\boldsymbol{\psi} := (\psi_1, \ldots, \psi_\ell) \xleftarrow{\mathsf{U}} \mathbb{F}_q^\ell$, and sets $g_T := e(g,g)^{\alpha\theta \vec{d}_1 \cdot \vec{d}_1^*}$ and $g_{T_j} := g_T^{\psi_j}$ for all $j \in [\ell]$. It chooses $s_{i,1}, s_{i,2} \xleftarrow{\mathsf{U}} \mathbb{F}_q$ for all $i \in [\ell]$ and generates $u_{i,j} := g_{T_j}^{s_{i,1}} \cdot g_T^{m_{i,j}}$ and $\boldsymbol{v}_i := g^{s_{i,1}\vec{d}_1 + s_{i,1} ID \vec{d}_2 + s_{i,2}\vec{d}_3 + s_{i,2} ID \vec{d}_4}$ for all $i, j \in [\ell]$ where $m_{i,i} = 1$ and $m_{i,j} = 0$ (if $i \neq j$). For a conversion key, it chooses $r_1, r_2 \xleftarrow{\mathsf{U}} \mathbb{F}_q$ and generates $\boldsymbol{k}_{ID} := g^{(\alpha + r_1 ID)\theta \vec{d}_1^* - r_1\theta \vec{d}_2^* + r_2 ID \sigma \vec{d}_3^* - r_2 \sigma \vec{d}_4^*}$. It returns $ek := (\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{k}_{ID}) := (\{u_{i,j}\}_{i,j}, \{\boldsymbol{v}_i\}_i, \boldsymbol{k}_{ID})$ $(i, j \in [\ell])$, $ik := \boldsymbol{\psi}$. Note ek includes ID, but we omit it for simplicity.

$\mathsf{LTF.LGen}(1^\lambda)$ **:** This is the same as $\mathsf{LTF.IGen}$ except that for all $i, j \in [\ell]$, $m_{i,j} = 0$ and $ik := \perp$.

$\mathsf{LTF.Eval}(ek, \vec{x})$**:** For input $\vec{x} \in \{0,1\}^\ell$, it computes $y_j := \prod_i u_{i,j}^{x_i} = g_{T_j}^{\vec{x}\cdot\vec{s}_1} g_T^{x_j}$, $y_{\ell+1} := \prod_i \boldsymbol{v}_i^{x_i} = g^{\vec{x}\cdot\vec{s}_1 \vec{d}_1 + \vec{x}\cdot\vec{s}_1 ID \vec{d}_2 + \vec{x}\cdot\vec{s}_2 \vec{d}_3 + \vec{x}\cdot\vec{s}_2 ID \vec{d}_4}$ where $\vec{s}_1 := (s_{1,1}, \ldots, s_{1,\ell})$, $\vec{s}_2 := (s_{2,1}, \ldots, s_{2,\ell})$, and $y'_{\ell+1} := \boldsymbol{e}(y_{\ell+1}, \boldsymbol{k}_0) = e(g,g)^{\alpha\theta \vec{d}_1 \cdot \vec{d}_1^* \vec{x}\cdot\vec{s}_1}$ and returns output $\boldsymbol{y} := (y_1, \ldots, y_\ell, y'_{\ell+1})$.

$\mathsf{LTF.Invert}(ik, \boldsymbol{y})$**:** For input $\boldsymbol{y}$, it computes $x'_j := y_j / (y'_{\ell+1})^{\psi_j} = g_{T_j}^{\vec{x}\cdot\vec{s}_1} g_T^{x_j} / g_T^{\vec{x}\cdot\vec{s}_1 \cdot \psi_j}$ and let $x_j \in \{0,1\}$ be such that $x'_j = g_T^{x_j}$. It returns $\vec{x} = (x_1, \ldots, x_\ell)$.

Theorem 2. $\mathsf{LTF}_{\mathsf{mult}}$ *is a lossy trapdoor function if the DBDH assumption holds.*

We omit the proof and the definition of the DBDH assumption (this assumption is implied by the DLIN assumption).

Next, we present our watermarking scheme. We added extra two dimensions of DPVS to the original Lewko IBE scheme since we use the extra dimensions to embed watermarks. Even if we add a vector spanned by $\vec{d}_7^*$ and $\vec{d}_8^*$ to $\boldsymbol{k}_{ID}$, which is spanned by $\vec{d}_1^*, \ldots, \vec{d}_4^*$, it is indistinguishable from the original since vectors $\vec{d}_7, \vec{d}_8, \vec{d}_7^*, \vec{d}_8^*$ are hidden. Moreover, the marked index works as the original non-marked index since $\boldsymbol{V}$ is spanned by $\vec{d}_1, \ldots, \vec{d}_4$ and components $\vec{d}_7^*, \vec{d}_8^*$ are canceled. However, if we have a vector which is spanned by $\vec{d}_7, \vec{d}_8$, then we can detect the mark generated by $\vec{d}_7^*, \vec{d}_8^*$. If we have complete dual orthonormal bases $(\mathfrak{D}, \mathfrak{D}^*)$, then we can use the decomposition algorithm introduced in Section 2.2 and eliminate the vector spanned by $\vec{d}_7^*, \vec{d}_8^*$, i.e., watermarks. Our watermarking scheme $\mathsf{CWM}_{\mathsf{mult}}$ for $\mathsf{LTF}_{\mathsf{mult}}$ is as follows:

$\mathsf{WMGen}(\mathsf{LTF}_{\mathsf{mult}})$**:** It generates $(\mathfrak{D}, \mathfrak{D}^*) \xleftarrow{\mathsf{U}} \mathsf{Dual}(\mathbb{F}_q^8)$, chooses $\alpha, \theta, \sigma \xleftarrow{\mathsf{U}} \mathbb{F}_q$, $g_T := e(g,g)^{\alpha\theta \vec{d}_1 \cdot \vec{d}_1^*}$, and sets $\widehat{\mathfrak{D}} := (g_T, g^{\vec{d}_1}, \ldots, g^{\vec{d}_4})$ $pk := (\widehat{\mathfrak{D}}, \mathsf{Samp})$, $sk := (g^{\alpha\theta\vec{d}_1^*}, g^{\theta\vec{d}_1^*}, g^{\sigma\vec{d}_2^*}, g^{\sigma\vec{d}_3^*}, g^{\vec{d}_4^*})$, $mk := (g^{\vec{d}_7^*}, g^{\vec{d}_8^*})$, $dk := (g^{\vec{d}_7}, g^{\vec{d}_8})$, and $rk := (\mathfrak{D}, \mathfrak{D}^*)$. The sampling algorithm $\mathsf{Samp}(\mathsf{pp}_{\mathbb{V}}, \widehat{\mathfrak{D}}, sk)$ chooses $\boldsymbol{\psi} \xleftarrow{\mathsf{U}} \mathbb{F}_q^{\ell}$, $\vec{s}_1, \vec{s}_2 \xleftarrow{\mathsf{U}} \mathbb{F}_q^{\ell}$, and generates $(ek, ik) := ((\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{k}_{ID}), \boldsymbol{\psi})$ as $\mathsf{LTF.IGen}$. It computes $\boldsymbol{k}_{ID} := g^{(\alpha + r_1 ID)\theta\vec{d}_1^* - r_1\theta\vec{d}_2^* + r_2 ID\sigma\vec{d}_3^* - r_2\sigma\vec{d}_4^*}$. Note that sk is *not* included in the description of Samp. Keys sk, mk, and rk are secret. Key dk can be disclosed.

$\mathsf{Mark}(mk, ek)$**:** It parses $ek = (\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{k}_{ID})$, chooses $t_1, t_2 \xleftarrow{\mathsf{U}} \mathbb{F}_q$, and computes $\widetilde{\boldsymbol{k}}_{ID} := \boldsymbol{k}_{ID} \cdot g^{t_1\vec{d}_7^* + t_2\vec{d}_8^*}$ by using $g^{\vec{d}_7^*}$ and $g^{\vec{d}_8^*}$. It outputs marked function index $\mathsf{WM}(ek) = (\boldsymbol{U}, \boldsymbol{V}, \widetilde{\boldsymbol{k}}_{ID})$.

$\mathsf{Detect}(dk, \widetilde{ek})$**:** It parses $\widetilde{ek} = (\boldsymbol{U}, \boldsymbol{V}, \widetilde{\boldsymbol{k}}_{ID})$, chooses $z_1, z_2 \xleftarrow{\mathsf{U}} \mathbb{F}_q^{\times}$, and computes $\boldsymbol{c} := g^{z_1\vec{d}_7 + z_2\vec{d}_8}$. Next, it computes $\Delta := \boldsymbol{e}(\boldsymbol{c}, \widetilde{\boldsymbol{k}}_{ID})$. If it holds that $\Delta = \boldsymbol{e}(\boldsymbol{c}, \widetilde{\boldsymbol{k}}_{ID}) \neq 1$, then it outputs marked. Else if, it holds that $\Delta = 1$, then it outputs $\mathsf{unmarked}$.

$\mathsf{Remove}(rk, \widetilde{ek})$**:** It parses $\widetilde{ek} = (\boldsymbol{U}, \boldsymbol{V}, \widetilde{\boldsymbol{k}}_{ID})$, runs algorithm $\mathsf{Decomp}(\widetilde{\boldsymbol{v}}_i, (g^{\vec{d}_1^*}, \ldots, g^{\vec{d}_m^*}), \mathfrak{D}^*, (g^{\vec{d}_1^*}, \ldots, g^{\vec{d}_8^*}))$ for all $m < 8$, and obtains $g^{z_j\vec{d}_j^*}$ for all $j = 1, \ldots, 8$ where $z_j \in \mathbb{F}_q$. It holds $\widetilde{\boldsymbol{k}}_{ID} = g^{z_1\vec{d}_1^* + \cdots + z_8\vec{d}_8^*}$. It computes $\boldsymbol{k}'_{ID} := \widetilde{\boldsymbol{k}}_{ID}/g^{z_7\vec{d}_7^* + z_8\vec{d}_8^*}$ and outputs $(\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{k}'_{ID})$ as an unmarked index.

Correctness, preserving functionality, and polynomial slowdown are easily followed. Meaningfulness follows since $(g^{\vec{d}_1^*}, \ldots, g^{\vec{d}_8^*})$ are hidden. Note that if we do not have secret key $(g^{\vec{d}_1^*}, \ldots, g^{\vec{d}_4^*})$, then we cannot compute a complete function index, that is, we cannot compute conversion key $\boldsymbol{k}_{ID}$. This seems to be a restriction, but in the scenario of watermarking schemes, this is acceptable. We use watermarking schemes to authorize objects, and such objects are privately generated by authors. For example, movies, music files, and software are generated by some companies and they do not distribute unauthorized (unmarked) objects. Moreover, in the experiment on security, the adversary is given a oracle which gives marked function indices. Thus, it is reasonable that unauthorized parties cannot efficiently sample functions by themselves.

4.1 Security Proofs for $\mathsf{CWM}_{\mathsf{mult}}$

Our watermarking scheme $\mathsf{CWM}_{\mathsf{mult}}$ is secure under the DLIN assumption. We prove this by proving Theorems 3 and 4.

Theorem 3. $\mathsf{CWM}_{\mathsf{mult}}$ *is non-removable under the subspace assumption.*

Proof. If $\mathcal{A}$ outputs $(0, ek^*)$, where $\mathsf{Detect}(dk, ek) \to \mathsf{unmarked}$ and $\mathsf{IdealDtc}(ek^*) \to \mathsf{marked}$, then we construct algorithm $\mathcal{B}$, which solves the subspace problem with $k = 1$ and $n = 8$. $\mathcal{B}$ is given Γ, $D = (g^{\vec{b}_1}, g^{\vec{b}_2}, g^{\vec{b}_4}, \ldots, g^{\vec{b}_8}, g^{\eta\vec{b}_1^*}, g^{\beta\vec{b}_2^*}, g^{\vec{b}_3^*}, \ldots, g^{\vec{b}_8^*}, U_1)$, and Q_{b} for $\mathsf{b} \in \{0,1\}$. We set $Q_0 := V_1 = g^{\tau_1\eta\vec{b}_1^* + \tau_2\beta\vec{b}_2^*}$ and $Q_1 := W_1 = g^{\tau_1\eta\vec{b}_1^* + \tau_2\beta\vec{b}_2^* + \tau_3\vec{b}_3^*}$. $\mathcal{B}$ sets

$$\vec{d}_1 := \vec{b}_3^* \quad \vec{d}_2 := \vec{b}_4^* \quad \vec{d}_3 := \vec{b}_5^* \quad \vec{d}_4 := \vec{b}_6^* \quad \vec{d}_5 := \vec{b}_7^* \quad \vec{d}_6 := \vec{b}_8^* \quad \vec{d}_7 := \vec{b}_1^* \quad \vec{d}_8 := \vec{b}_2^*$$

$$\vec{d}_1^* := \vec{b}_3 \quad \vec{d}_2^* := \vec{b}_4 \quad \vec{d}_3^* := \vec{b}_5 \quad \vec{d}_4^* := \vec{b}_6 \quad \vec{d}_5^* := \vec{b}_7 \quad \vec{d}_6^* := \vec{b}_8 \quad \vec{d}_7^* := \vec{b}_1 \quad \vec{d}_8^* := \vec{b}_2$$

$\mathcal{B}$ chooses $\theta, \alpha', \sigma \overset{\mathsf{U}}{\leftarrow} \mathbb{Z}_p$ and can generate public key $pk = (e(g,g)^{\alpha\theta\vec{d}_1\cdot\vec{d}_1^*}, g^{\vec{d}_1}, \dots, g^{\vec{d}_4}) := (e(g^{\vec{b}_2^*}, g^{\vec{b}_2})^{\alpha'\mu_3\theta}, g^{\vec{b}_3^*}, g^{\vec{b}_4^*}, g^{\vec{b}_5^*}, g^{\vec{b}_6^*})$ and mark key $mk = (g^{\vec{d}_7^*}, g^{\vec{d}_8^*}) := (g^{\vec{b}_1}, g^{\vec{b}_2})$. $\mathcal{B}$ has a detect key, which is essentially the same as $g^{\vec{d}_7}$ and $g^{\vec{d}_8}$ since $g^{\eta\vec{b}_1^*}, g^{\beta\vec{b}_2^*}$ are given. Coefficients β and η do not affect the detect algorithm. $\mathcal{B}$ can have $(g^{\vec{d}_2^*}, \dots, g^{\vec{d}_8^*})$ but does not have $g^{\vec{d}_1^*}$ since $g^{\vec{b}_3}$ is not given. That is, $\mathcal{B}$ has the mark key and perfectly simulates the mark oracle but the secret key is incomplete as follows: $sk = (\perp, \perp, g^{\theta\vec{d}_2^*}, g^{\sigma\vec{d}_3^*}, g^{\sigma\vec{d}_4^*}) := (\perp, \perp, g^{\theta\vec{b}_4}, g^{\sigma\vec{b}_5}, g^{\sigma\vec{b}_6})$.

It implicitly holds $\alpha = \alpha'\mu_3$. To simulate the challenge oracle without the complete sk, for ID, $\mathcal{B}$ chooses $r_1', r_2, t_7, t_8 \overset{\mathsf{U}}{\leftarrow} \mathbb{Z}_p$ and computes

$$\begin{aligned}\widetilde{\boldsymbol{k}}_{ID} &:= (U_1)^{(\alpha'+r_1'ID)\theta} g^{-r_1'\mu_3\theta\vec{d}_2^*+r_2ID\sigma\vec{d}_3^*-r_2\sigma\vec{d}_4^*+t_7\vec{d}_7^*+t_8\vec{d}_8^*} \\ &= g^{(\alpha+r_1ID)\theta\vec{d}_1^*-r_1\theta\vec{d}_2^*+r_2ID\sigma\vec{d}_3^*-r_2\sigma\vec{d}_4^*+t_7'\vec{d}_7^*+t_8'\vec{d}_8^*}\end{aligned}$$

where $t_7' = t_7 - \theta(\alpha' + r_1'ID)\mu_1$ and $t_8' = t_8 - \theta(\alpha' + r_1'ID)\mu_2$ We set $r_1 := \mu_3 r_1'$. This is a valid marked index. If $\mathcal{A}$ outputs valid unmarked index $ek^* = (\boldsymbol{U}^*, \boldsymbol{V}^*, \boldsymbol{k}_{ID^*}^*)$ where $\boldsymbol{k}_{ID^*}^* = g^{(\alpha+r_1^*ID^*)\theta\vec{d}_1^*-r_1^*\theta\vec{d}_2^*+r_2^*ID^*\sigma\vec{d}_3^*-r_2^*\sigma\vec{d}_4^*}$, then $\mathcal{B}$ computes $\Delta := \boldsymbol{e}(Q_{\mathsf{b}}, \boldsymbol{k}_{ID^*}^*)$. If $\Delta = 1$, then $\mathcal{B}$ outputs 0 ($\mathsf{b} = 0$), otherwise, it outputs 1. If $Q_{\mathsf{b}} = g^{\tau_1\eta\vec{b}_1^*+\tau_2\beta\vec{b}_2^*} = g^{\tau_1\eta\vec{d}_7+\tau_2\beta\vec{d}_8}$, then $\Delta = 1$. If $Q_{\mathsf{b}} = g^{\tau_1\eta\vec{b}_1^*+\tau_2\beta\vec{b}_2^*+\tau_3\vec{b}_3^*} = g^{\tau_1\eta\vec{d}_7+\tau_2\beta\vec{d}_8+\tau_3\vec{d}_1}$, then $\Delta = e(g,g)^{(\alpha+r_1ID)\theta\tau_3\vec{d}_1\cdot\vec{d}_1^*} \neq 1$. That is, $\mathcal{B}$ breaks the problem. □

Next, we prove unforgeability. Note that the adversary is not allowed to output a function index whose identity is equal to those of indices generated by the challenge oracle or are queried to the mark oracle. This is justified by the following fact. If it is allowed, then it means the adversary has *already had a (functionally equivalent) marked index* for the given or queried identity, that is, an IBE private key for the same identity. This is unavoidable and in the experiment on unforgeability, $\mathsf{IdealDtc}$ always returns marked for identities that oracles used. For simplicity, we prove the unforgeability explained above, but we can extend it to stronger ones by using known techniques that convert standard unforgeable signature schemes into *strongly unforgeable* signature schemes. We now define algorithm $\mathsf{Xtr}(pk, sk, ID)$. It chooses $r_1, r_2 \overset{\mathsf{U}}{\leftarrow} \mathbb{F}_q$ and outputs $\boldsymbol{k}_{ID} := g^{(\alpha+r_1ID)\theta\vec{d}_1^*-r_1\theta\vec{d}_2^*+r_2ID\sigma\vec{d}_3^*-r_2\sigma\vec{d}_4^*}$. We can consider $\boldsymbol{k}_{ID}$ be a signature for ID. In fact, Naor pointed out that signature schemes can be derived from IBE schemes [2]. Thus, we can prove the unforgeability of our watermarking scheme by using the unforgeability of signature schemes derived from IBE schemes of Okamoto-Takashima and Lewko.

Huang, Wong, and Zhao proposed a generic transformation technique for converting unforgeable signature schemes into strongly unforgeable ones [9]. We can achieve strong unforgeability of watermarking schemes by using their technique and the strongly unforgeably property.

Theorem 4. $\mathsf{CWM}_{\mathsf{mult}}$ *is unforgeable under the subspace assumption.*

Proof. Let q_{M} and q_{C} be the number of queries to the mark oracle and the challenge oracle, respectively. There are two types of conversion keys $\boldsymbol{k}_{ID}$.

Normal: $g^{(\alpha+r_1 ID)\theta\vec{d}_1^*-r_1\theta\vec{d}_2^*+r_2 ID\sigma\vec{d}_3^*-r_2\sigma\vec{d}_4^*+t_7\vec{d}_7^*+t_8\vec{d}_8^*}$
Semi-functional: $g^{(\alpha+r_1 ID)\theta\vec{d}_1^*-r_1\theta\vec{d}_2^*+r_2 ID\sigma\vec{d}_3^*-r_2\sigma\vec{d}_4^*+t_5\vec{d}_5^*+t_6\vec{d}_6^*+t_7\vec{d}_7^*+t_8\vec{d}_8^*}$.
We can generate them if we have the secret key, mark key, and $(g^{\vec{d}_5^*}, g^{\vec{d}_6^*})$.

Both types of conversion keys give a correct output (we can check this by simple calculation). To show that our scheme satisfies unforgeability, we introduce the following games: We consider game $\mathsf{Game}\text{-}i$ where the challenge oracle generates semi-functional conversion keys for the first $i \in [q_{\mathsf{C}}]$ queries and semi-functional conversion keys for the remaining $q_{\mathsf{C}} - i$ queries. Note that the mark oracle does not generate function indices. It only embeds marks for queried indices. Let $\mathsf{Adv}_i^{\mathsf{forge\text{-}N}}$ (resp. $\mathsf{Adv}_i^{\mathsf{forge\text{-}S}}$) denote the advantage of the adversary in $\mathsf{Game}\text{-}(i)$ for outputting a normal (resp. semi-functional) conversion key for a non-given or non-queried ID.

Lemma 2. *If $\mathcal{A}$ outputs a semi-functional marked index in* $\mathsf{Game}\text{-}(0)$*, then we can break the subspace assumption with $k = 2$ and $n = 8$.*

Lemma 3. *If there exists $\mathcal{A}$, that distinguishes* $\mathsf{Game}\text{-}(i-1)$ *from* $\mathsf{Game}\text{-}(i)$*, then we can break the subspace assumption with $k = 2$ and $n = 8$.*

Lemma 4. *If $\mathcal{A}$ outputs a normal marked index in* $\mathsf{Game}\text{-}(q_{\mathsf{C}})$*, then we can break the subspace assumption with $k = 1$ and $n = 8$.*

By Lemmas 2, 3, and 4, we can show the following:

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Forge}}(\lambda) = \mathsf{Adv}_0^{\mathsf{forge\text{-}N}} + \mathsf{Adv}_0^{\mathsf{forge\text{-}S}} < (q_{\mathsf{C}} + 2)\mathsf{Adv}_{\mathcal{B}}^{\mathsf{dss}}.$$

The theorem follows from the lemmas and Theorem 1. □

Acknowledgements. The author would like to thank Keita Xagawa for illuminating discussions and insightful comments about constructions of watermarking schemes and Kazumaro Aoki and Atsushi Fujioka for helpful comments about applications of watermarking and the presentation of the introduction. The author also thanks Maki Yoshida and Akinori Kawachi for helpful comments.

References

1. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S.P., Yang, K.: On the (im)possibility of obfuscating programs. J. ACM 59(2), 6 (2012)
2. Boneh, D., Franklin, M.K.: Identity-based encryption from the Weil pairing. SIAM J. Comput. 32(3), 586–615 (2003)
3. Chor, B., Fiat, A., Naor, M.: Tracing traitors. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 257–270. Springer, Heidelberg (1994)
4. Collberg, C.S., Thomborson, C.D.: Watermarking, tamper-proofing, and obfuscation-tools for software protection. IEEE Trans. Software Eng. 28(8), 735–746 (2002)
5. Freeman, D.M., Goldreich, O., Kiltz, E., Rosen, A., Segev, G.: More constructions of lossy and correlation-secure trapdoor functions. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 279–295. Springer, Heidelberg (2010)
6. Galbraith, S.D., Verheul, E.R.: An analysis of the vector decomposition problem. In: Cramer, R. (ed.) PKC 2008. LNCS, vol. 4939, pp. 308–327. Springer, Heidelberg (2008)

7. Hemenway, B., Ostrovsky, R.: Extended-DDH and lossy trapdoor functions. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 627–643. Springer, Heidelberg (2012)
8. Hopper, N.J., Molnar, D., Wagner, D.: From weak to strong watermarking. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 362–382. Springer, Heidelberg (2007)
9. Huang, Q., Wong, D.S., Zhao, Y.: Generic transformation to strongly unforgeable signatures. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 1–17. Springer, Heidelberg (2007)
10. Kiayias, A., Yung, M.: Traitor tracing with constant transmission rate. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 450–465. Springer, Heidelberg (2002)
11. Lewko, A.: Tools for simulating features of composite order bilinear groups in the prime order setting. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 318–335. Springer, Heidelberg (2012)
12. Lewko, A., Okamoto, T., Sahai, A., Takashima, K., Waters, B.: Fully secure functional encryption: Attribute-based encryption and (Hierarchical) inner product encryption. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 62–91. Springer, Heidelberg (2010)
13. Luby, M., Rackoff, C.: How to construct pseudorandom permutations from pseudorandom functions. SIAM J. Computing 17(2), 373–386 (1988)
14. Naccache, D., Shamir, A., Stern, J.P.: How to copyright a function? In: Imai, H., Zheng, Y. (eds.) PKC 1999. LNCS, vol. 1560, pp. 188–196. Springer, Heidelberg (1999)
15. Okamoto, T., Takashima, K.: Homomorphic encryption and signatures from vector decomposition. In: Galbraith, S.D., Paterson, K.G. (eds.) Pairing 2008. LNCS, vol. 5209, pp. 57–74. Springer, Heidelberg (2008)
16. Okamoto, T., Takashima, K.: Hierarchical predicate encryption for inner-products. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 214–231. Springer, Heidelberg (2009)
17. Okamoto, T., Takashima, K.: Fully secure functional encryption with general relations from the decisional linear assumption. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 191–208. Springer, Heidelberg (2010)
18. Okamoto, T., Takashima, K.: Efficient attribute-based signatures for non-monotone predicates in the standard model. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 35–52. Springer, Heidelberg (2011)
19. Okamoto, T., Takashima, K.: Adaptively attribute-hiding (Hierarchical) inner product encryption. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 591–608. Springer, Heidelberg (2012)
20. Peikert, C., Waters, B.: Lossy trapdoor functions and their applications. SIAM J. Comput. 40(6), 1803–1844 (2011)
21. Rompel, J.: One-way functions are necessary and sufficient for secure signatures. In: STOC, pp. 387–394. ACM (1990)
22. Waters, B.: Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 619–636. Springer, Heidelberg (2009)
23. Yoshida, M., Fujiwara, T.: Toward digital watermarking for cryptographic data. IEICE Transactions 94-A(1), 270–272 (2011)
24. Yoshida, M., Mitsunari, S., Fujiwara, T.: The vector decomposition problem. IEICE Transactions 93A-(1), 188–193 (2010)

Security Evaluations beyond Computing Power
How to Analyze Side-Channel Attacks You Cannot Mount?

Nicolas Veyrat-Charvillon[1], Benoît Gérard[2], and François-Xavier Standaert[1]

[1] UCL Crypto Group, Université catholique de Louvain
Place du Levant 3, B-1348, Louvain-la-Neuve, Belgium
[2] Direction Générale de l'Armement–Maîtrise de l'information, France

Abstract. Current key sizes for symmetric cryptography are usually required to be at least 80-bit long for short-term protection, and 128-bit long for long-term protection. However, current tools for security evaluations against side-channel attacks do not provide a precise estimation of the remaining key strength after some leakage has been observed, e.g. in terms of number of candidates to test. This leads to an uncomfortable situation, where the security of an implementation can be anywhere between enumerable values (i.e. $2^{10} - 2^{50}$ key candidates to test) and the full key size (i.e. $2^{60} - 2^{128}$ key candidates to test). In this paper, we propose a solution to this issue, and describe a key rank estimation algorithm that provides tight bounds for the security level of leaking cryptographic devices. As a result and for the first time, we are able to analyze the full complexity of "standard" (i.e. divide-and-conquer) side-channel attacks, in terms of their tradeoff between time, data and memory complexity.

1 Introduction

Concrete security evaluations are at the core of cryptographic research. Taking the example of symmetric cryptography, they are at the same time central in formal definitions of security (e.g. as introduced by Bellare et al. [2]) and in the evaluation of attacks such as linear and differential cryptanalysis [4,22]. Their goal is to provide bounds on the success probability of an adversary as a function of the resources she expends, typically measured in time, data and memory. But somewhat surprisingly, while such concrete (and complete) evaluations are usual in the context of mathematical cryptanalysis (e.g. [3], Table 5), they appear much harder to obtain in the context of physical cryptanalysis, even for "classical" attacks such as Kocher et al.'s Differential Power Analysis (DPA) [20].

The challenging nature of physical security evaluations mainly relates to the difficulty of capturing the "device-specificity" of the attacks. For example, statistical models used to evaluate the complexity of linear and differential cryptanalysis have been intensively studied for more than 20 years. They generally reflect the peculiarities of actual block ciphers to a good extent [10,19,30]. Under reasonable assumptions and using design tools such as the wide-trail strategy, one can even guarantee security against large classes of statistical attacks [9]. By contrast, there is no general theory explaining how to build secure implementations, and most countermeasures used by device manufacturers highly depend on the chosen technology. Therefore, *present security evaluations against side-channel attacks need to rely on experimental validation.* For example, certification reports emitted by national authorities such as the ANSSI in France [1], or the BSI in Germany [6], are based on extensive analysis from evaluation laboratories.

From a cryptographic point of view, a purely empirical approach is hardly satisfying, as it only determines whether a given laboratory (with given equipment, time, data and memory) is able to recover some secret information contained in a leaking device. Hence, a fundamental question is to determine which parts of the physical security evaluations actually need experiments. Since the leakage in cryptographic implementations is technology-dependent, it is clear that some characterization through measurements is unavoidable, in order to determine its informativeness. Yet, given a certain amount of information leakage, it remains to analyze the impact of the time and memory complexities on the success probability of actual side-channel attacks. Answering these two questions (i.e. information extraction and exploitation) is the main goal of evaluation frameworks such as [31]. In the context of block ciphers (that will be our running example), most distinguishers published in the literature are based on a divide-and-conquer strategy[1]. As a result, the usual solution to exploit computational power is to perform enumeration [25,32]. But *this implies that present security evaluations are limited to the computational power of the evaluator.* That is, the only leaking devices for which we can evaluate the security are the ones that are "practically insecure" (i.e. for which the leakage allows key enumeration). It leaves the (most interesting) evaluation of "practically secure" devices as an open problem. For example, an evaluation laboratory could claim that he could consistently not recover an AES key within time complexity 2^{40} (i.e. after repeated experiments and with good statistical confidence). But this does not give clear hints whether the security level of the corresponding target leaking device is 2^{41} or 2^{100}.

Main Result. We show that in the (realistic) scenario where the evaluator of a leaking device knows its secret key, it is possible to estimate the probability of success of "standard" side-channel attacks (e.g. the ones listed in footnote 1) that he is unable to perform (e.g. attacks of time complexities beyond 2^{80}). For this purpose, we provide a *rank estimation algorithm* solving the following problem: "given a set of discrete probability distributions for independent parts of a key and a correct key k^* with posterior likelihood p^*, provide tight bounds for the rank of this key among the set of all possible ones" (i.e. for the order statistic of p^* in the set of all key probabilities). The set of distributions is typically the outcome of a template attack [8], but can also be derived from most non-profiled side-channel attacks such as listed in Footnote 1. In fact, rank estimation requires essentially the same inputs as key enumeration, excepted the correct key and its likelihood (unknown during enumeration). Based on several experiments, we further show that our algorithm features small ratios between the lower and

[1] Including Kocher et al.'s DPA, Brier et al.'s Correlation Power Analysis (CPA) [5], Chari et al.'s Template Attacks (TA) [8], Gierlichs et al.'s Mutual Information Analysis (MIA) [14], Schindler et al.'s stochastic approach [28], and many variations.

higher bounds on the key rank, and small running times (e.g. ratios between 2^2 and 2^{10} for a 128-bit key are obtained in a couple of seconds on a PC).

Consequences. Besides being a tool of choice for side-channel evaluation laboratories, our algorithm has a number of important consequences for the theory and practice of side-channel attacks. First, and for the first time, it allows the estimation of all the metrics put forward in [31] (namely, the success rates of all orders and guessing entropy). For example, the estimation of the guessing entropy for large master keys was previously impossible, as illustrated by the reports of the DPA contests `v1` and `v2` [26]. Our rank estimation algorithm could be directly used in further versions of such contests. Second, it provides a method to connect actual side-channel attacks with the need of "limited information leakage" in certain formal works aiming to prove security against physical attacks. For example, it allows quantifying the hardness assumption in Dodis et al.'s cryptography with auxiliary input [11], or the seed-preserving assumption used in [34]. Although less directly connected, it also provides a lower bound on the λ-bit leakage required in leakage-resilient cryptography [12]. Third, rank estimation yields an exact solution to evaluate the complexity of a number of "non-standard" side-channel attacks. For example, it is perfectly suited to estimate the workload of collision attacks such as [24,29] (see [13], Section 5). It would also be handy for analyzing the key-dependent algorithmic noise in the CHES 2012 leakage-resilient PRF [23], where most subkeys cannot be highly rated by the adversary. Fourth, our experiments suggest a cautionary note for the use of lightweight ciphers with small key sizes in leaking devices, as a few measurements can be enough to degrade their security within adversarial reach. Finally, we note that the proposed algorithm is not limited to physical security evaluations, and is potentially useful in any mathematical cryptanalysis setting where experiments are still needed to validate an hypothetical model.

2 Rank Estimation: An Overview

This section describes the approach that allows us to perform efficient and accurate rank estimations for standard key spaces (i.e. from 2^{80} to 2^{256}, typically). Let us denote the independent parts of the key for which information has been obtained as subkeys. Our general idea is to organize the key space by sorting each of these subkeys according to their posterior likelihood, in decreasing order. As a result, the full key space is partitioned into 2 main volumes. The first one is defined by all key candidates with probability higher than the correct key. The second one is defined by all key candidates with probability smaller than the correct key[2]. Given this geometrical representation, our rank estimation problem can be stated as the one of finding bounds for the "higher" and "lower" volumes.

[2] Between these volumes, we may find other volumes where all key candidates have the same probability as the correct one - which will be considered by our algorithm. Yet, in practice this "middle zone" usually contains the correct key only.

Organizing the key space with such volumes has one main advantage. Namely, the “higher” (resp. “lower”) set of key candidates is delimited by a concave (resp. convex) surface within the key space. This means that if we pick a key candidate with a probability higher (resp. lower) than the target key, all keys with index lower (resp. higher) will also have the same property. This fact is illustrated in Figure 1 for a simplified 2-dimension case, with small subkey spaces. In this example, the correct key is the blue circle, and the equipotential surface splits the key space into candidates with higher (green, light) and lower (red, dark) probabilities. If one picks a key candidate within the lower probability set (e.g. the black circle), we notice that all candidates with higher indexes (inside the gray rectangle) will be on the same side of the surface. Hence, they will have a lower probability than that of the correct key. Taking advantage of this fact, our method for rank estimation essentially consists in “carving” such boxes of key candidates on each side of the probability surface, and use the volumes of these boxes to progressively refine the (lower and higher) bounds on the key rank.

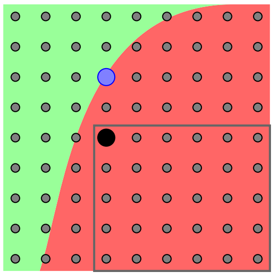


Fig. 1. Sorted key space in a simplified two-subkeys case

Note that for small key spaces, standard quadrature tools such as Monte-Carlo integration could be used to bound the key rank. But as soon as typical cryptographic parameters are considered, e.g. 16 dimensions supported by 2^8 discrete points for the case of AES-128, these tools do not work anymore. Even when partially merging some dimensions[3] (e.g. in order to obtain 4 dimensions with support 2^{32}), they fail to provide an answer in reasonable time. To the best of our knowledge, the algorithm carefully described here is the first one to provide an efficient solution that fits cryptographic evaluation purposes.

3 Algorithms for Efficient Rank Estimation

This section aims at presenting the rank estimation algorithm. First, a high-level description of the algorithm is provided. For this algorithm to work efficiently in practice, several refinements are needed, detailed in the subsequent sections.

[3] Which will generally be beneficial to improve the performances of our approach too.

3.1 High-Level Algorithm Representation

Algorithm 1 is an high-level view of the rank estimation algorithm. For the sake of simplicity, we assume that the target cipher is AES-128 (that is we are provided 16 subkey probability mass functions of support 2^8 each). The algorithm remains valid for other ciphers by simply modifying the corresponding entries.

Algorithm 1. Rank estimation algorithm

```
Input: Subkey distributions $\mathcal{D} = (D_i)_{1 \le i \le 16}$ and the key probability $p^*$.
Output: An interval $I = [I_0; I_1]$ containing the key rank.
$\mathcal{L} \leftarrow \{[0; 255]^{16}\}$;
$I \leftarrow [0; 2^{128}]$;
PreProcess($\mathcal{D}$);
while $\mathcal{L} \neq \emptyset$ do
    $(V, \mathcal{L}) \leftarrow$ PickVolume($\mathcal{L}$);
    if IsCarved($V$) = false then
        $(V', I) \leftarrow$CarveBox($V, I, p^*, \mathcal{D}$);
        $\mathcal{L} \leftarrow$ InsertVolume($\mathcal{L}, V'$);
    else
        $\{V'_i\} \leftarrow$ SplitVolume($V$);
        $\mathcal{L} \leftarrow$ InsertVolume($\mathcal{L}, \{V'_i\}$);
return $I$;
```

As stated in Section 2, our algorithm is based on the fact that when distributions are sorted by decreasing probability, the frontier between the "lower" and "higher" key spaces is convex. Thus, a `PreProcess()` procedure is first used to sort distributions. It also performs other treatments in order to improve the algorithm efficiency (discussed in Section 3.2). After initializing a list $\mathcal{L}$ with a volume containing the whole key space, we iterate over volumes in this list until it is depleted or the target accuracy is reached. The volume to process is chosen by the `PickVolume()` procedure that removes the largest volume from $\mathcal{L}$. The extracted volume is then processed. For reasons that will be clarified later in the section, we store volumes as either simple boxes (i.e. a Cartesian product of intervals), or the set difference of two such boxes that we will denote as *carved* volumes. Depending on the case, two alternative procedures are possible.

If the extracted volume is a full box, the `CarveBox()` procedure is called, that chooses a point on one side of the equipotential surface and carves a key set from this point. The result is a carved volume V', which is actually a difference between two key boxes. The rank estimate I is then updated with the carved set. Afterwards, the remaining carved volume is inserted back into list $\mathcal{L}$ using the `InsertVolume()` procedure. Else the volume extracted from list $\mathcal{L}$ is a carved volume, in which case the `SplitVolume()` procedure is used first, that splits it into smaller volumes having simpler geometries. As for the first case, these are then inserted back into the list using the `InsertVolume()` procedure.

A run of the algorithm is illustrated in Figure 2. First, a box is carved from the key space and subtracted from the higher bound. The resulting carved box is then split in two. In the third step, a new box is carved on the green (light) side of the top box, and added to the lower bound. In the fourth step, another box is carved from the green (light) side of the bottom box. After several steps, an exact bound can be given for the correct key in Figure 1. Note that during the rank estimation for an actual cipher, the limiting surface has too many details to be exactly computed, hence we are limited to an estimation of the rank.

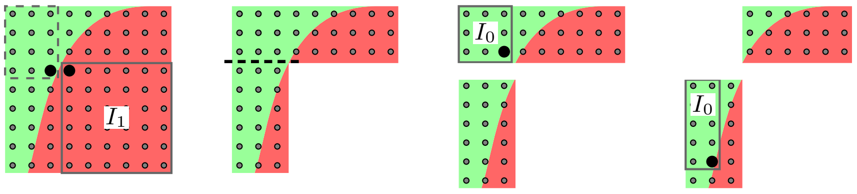


Fig. 2. Example run of Algorithm 1

While this algorithm is seemingly simple, any direct implementation results in either intractable computation time or too large memory requirements. We only managed to attain tractability (and efficiency) through several specific choices and refinements for the different procedures, described below.

3.2 The `PreProcess` Procedure

In addition to sorting the distributions by decreasing probability, the `PreProcess` procedure also lowers the number of dimensions in the key space by merging some subkey lists. Instead of treating a 16-dimension cube of side length 2^8, we will work, for instance, on a 6-dimension space with sides up to 2^{24}. This merging step leads to significant improvements in the algorithm efficiency and should be applied as far as memory allows it. The impact of such a merging on the algorithm performances will be illustrated in the experiments of Section 4.

3.3 The `PickVolume` Procedure

When extracting a volume from the list, one can either pick the largest one or the smallest. Extracting the smallest volume leads to small memory requirements, as we basically perform a depth-first exploration of the key space. But this strategy has high computation time and becomes intractable as the algorithm gets lost in the details of the equipotential surface. By contrast, picking the largest volume first is equivalent to performing a breadth-first search, and allows bounds to converge faster. Hence, we choose to maintain a list of largest volumes. Such a list is efficiently implemented using an heap-based priority queue. Yet, its memory cost can become critical (as many volumes have to be stored at any given time). Improvements were needed to minimize this memory, as detailed next.

3.4 The `InsertVolume` Procedure

When the number of volumes stored in the queue increases beyond what we can efficiently store on a computer, we have two solutions: either switch to a depth-first search (which does stop the storage increase but has the side-effect of slowing down the convergence of bounds almost to a stop), or we can truncate the smallest volumes in the heap (since we use a heap and not a binary tree, we actually truncate volumes *among* the smallest ones, not exactly the smallest ones). The second approach naturally leads to accuracy losses in the estimation. Fortunately, we are not interested in the exact rank of the key but on "good enough" bounds. Hence, we opted for this second strategy. In practice, truncation is acceptable if the accuracy loss is small compared to the key rank, which depends on the storage limit set in the algorithm. We will show in the Section 4 that current computer memories allow very satisfactory results in this respect.

3.5 The `IsCarved` Procedure

Iterations of Algorithm 1 essentially take boxes from the key space and carve other boxes out of them. If we were only able to represent plain boxes, we would need to perform splits along each dimension each time a box is carved. Essentially, carving a piece out of a box would lead to an increase in memory requirements (due to the storage of the resulting pieces): a naive split generates up to $2^d - 1$ new boxes with d the number of dimensions. The storage technique suggested in Section 3.1 allows a significant reduction of this cost. By allowing the representation of *differences* between key boxes, we can store carved volumes within as much memory as "plain ones". As a result, the `IsCarved` procedure is used each time a volume is picked in Algorithm 1, in order to determine whether the volume passed as an argument is a plain box or a carved one.

3.6 The `SplitVolume` Procedure

Whenever the volume we extract from the list is not a box, but rather a difference of two boxes, we have to simplify it and insert the resulting volumes back into the list. As stated above, the naive approach of splitting along each dimension is very inefficient and can generate up to $2^d - 1$ new boxes. Instead, we propose another way to perform this task. That is, given a volume consisting of a difference between two boxes, we split it along *a single* axis. This results in two volumes, one of which is a box, the other being either a box or a carved volume. The axis used to split is chosen in order to maximize the volume of the resulting simple box. This solution is illustrated in Figure 3. The carved box (left) can either be split into seven smaller boxes (middle), or into a larger box and another carved volume (right). We note that this alternative approach, on top of minimizing the size of the volume list, also preserves larger volumes (which is positive since it improves the refinement of bounds during the subsequent carving steps).

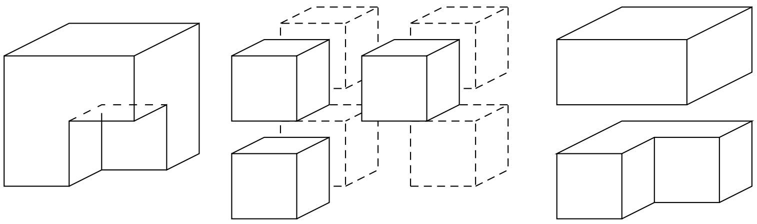


Fig. 3. Possible approaches to the volume split

3.7 The `CarveBox` Procedure

(a) The Carving Point. In order to refine rank estimation, we classify key candidates inside a volume as more or less probable than the correct key, and carve pieces of this volume. The easiest way to do this is to pick the central point in the box, estimate its probability and carve the corresponding box. This approach is inefficient as it only classifies one 2^{-d}-th of the box volume where d is the number of dimensions. Instead, we perform an heuristic optimization on the number of keys contained in the carved part, repeated on both sides of the equipotential surface. In practice, the surface is sufficiently "well-behaved" so that simple heuristics such as hill climbing (with some random restarts) was sufficient. More computationally intensive approaches that we tested (such as simulated annealing) only offer a marginal improvement while hampering speed.

(b) The Carving Side. Iterations of Algorithm 1 provide two boxes for updating either the inner or outer bound of the key rank. In order to choose between them, we need a criteria. The naive proposal trying to minimize the *difference* between the updated bounds is not efficient, as it will almost always choose the largest carved volume of the two. Indeed, the rank of the key in a side-channel attack usually tends to be small with respect to the size of the key space, with most key candidates having probability smaller than the correct one (otherwise the attack was ineffective). As a result, such a criteria will almost always update the higher bound of the key rank and refine the lower one very late (i.e. when almost all key candidates have been classified). In order to avoid such shortcomings, a better criteria is to minimize the *ratio* (or log difference) between higher and lower bounds. This way, the carving results in fast refinements of the bounds from a computational point of view - with typical bounds within a few binary orders.

4 Experimental Results

The goal of this section is two-fold. In a first part we evaluate the speed and accuracy of our rank estimation algorithm. In a second part we apply this tool in the context of a security evaluation and discuss its usefulness. The experiments are based on a C++ implementation of the rank estimation algorithm.

Its functional correctness was tested by comparing the results to those of our enumeration algorithm [32] for computationally reachable key ranks.

4.1 Performances of the Rank Estimation Algorithm

We first tried to gauge the efficiency of the rank estimation algorithm in the previous section. To this end, we used the results of simulated template attacks against an unprotected AES-128 implementation for inputs. That is, we considered 16 attacks targeting the 16 S-boxes in the first AES round and taking advantage of leakage samples of the shape: $l_i = \mathsf{HW}(\mathsf{S}[x_i \oplus k_i]) + n_i$, with HW the Hamming weight function, x_i (resp. k_i) the ith byte of the plaintext (resp. key), and n_i a Gaussian-distributed noise variable. As a result, we obtained attack outcomes in the form of 16 lists of 256 posterior likelihoods. Note that our following analysis is quite independent of the exact type of side-channel attack implemented. As discussed in [32], key enumeration (hence, rank estimation) apply to any profiled or non-profiled DPA. The only important parameter for our performance evaluations is the rank of the correct key candidate suggested by the attack. In practice, we played with the noise variable and number of measurements in order to make this rank vary. The main criteria used to measure the efficiency of our algorithm are speed, memory and the tightness of the bounds.

In this context, a first task is the study of the impact of merging subkey lists during the `PreProcess` procedure. We analyzed convergence of the bounds obtained by our rank estimation algorithm as a function of the execution time in the following contexts: (*i*) No merging has been done on the subkey posterior distributions, i.e. the algorithm considers 16 dimensions of support 2^8, leading to 4 KB of memory requirements for the tables; (*ii*) Subkey lists were merged two by two, i.e. the algorithm considers 8 dimensions of support 2^{16}, leading to 524 KB of memory requirements for the tables; (*iii*) Subkey lists were been merged by three, i.e. the algorithm considers 5 dimensions of support 2^{24} and one of 2^8, leading to 83 MB of memory requirements for the tables. Merging further would require more than 4GB of memory and was not necessary in our context. As can be noticed in the example run of Figure 4, merging has a strongly positive impact on the performances of the algorithm. It should always be performed up to the memory limit. The different experiments presented in the rest of this section were always obtained by merging the lists as per (*iii*).

The next point of interest is the rate of convergence of the proposed algorithm. In Figure 5, the ratios between higher and lower bound on the estimated rank are plotted against the actual rank. More precisely, if the algorithm returns an estimated interval $[i_0; i_1]$, then we plot $\log_2(i_1/i_0)$ on the Y-axis to show the relative accuracy of the current estimation, against the geometrical mean of the best estimated bounds on the X-axis. By repeating the experiment and removing outliers[4], we obtain the banana-shaped envelopes illustrated on the figure. Each envelope corresponds to a given running-time between 5 and 900 seconds.

[4] Corresponding to cases where the equipotential surface is so simple that the algorithm returns a significantly more accurate interval compared to other experiments.

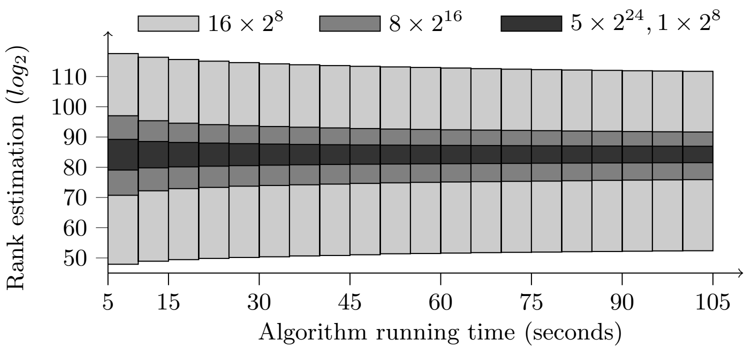


Fig. 4. Convergence of bounds versus running time depending on merge options

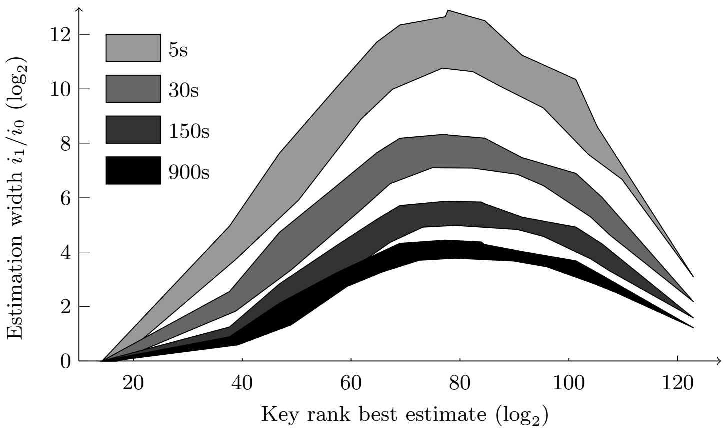


Fig. 5. Estimation accuracy versus rank (best estimate) given running time

These results highlight that in our setting, most of the estimation work is done after 150 seconds. We also note that in 5 seconds, the ratio between higher and lower bounds is at most of 14 binary orders of magnitude, which is actually a very good indicator when evaluating the security of a cryptographic component. In this respect, ranks from 2^{60} to 2^{100} are the most difficult to estimate, while ranks smaller than 2^{30} can be accurately determined within a few seconds.

Eventually, another natural question is to determine how much our rank estimation algorithm allows improving crude lower bounds obtained by simply multiplying the subkey ranks together. Figure 6 shows the ratio between this approximation and the bounds returned by our algorithm after only 30s. Estimated intervals are plotted as gray vertical segments, with bound values divided by the rank product estimate. It can be observed that the product bound underestimates the actual rank by 20 to 40 binary orders of magnitude.

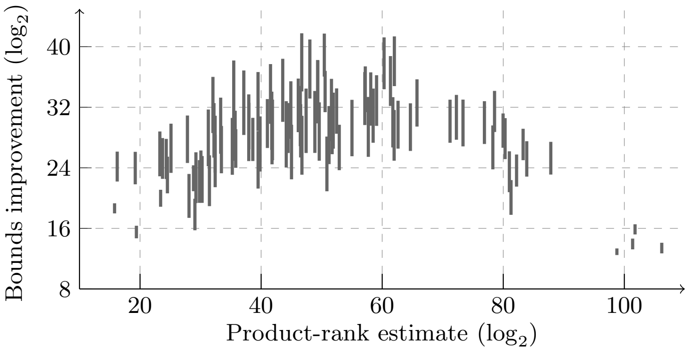


Fig. 6. Improvement obtained by rank estimation over rank-product lower bound

4.2 Using Rank Estimation in Physical Security Evaluations

In general, the main objective of a physical security evaluation is to determine how many encryption traces allow an adversary to recover the secret key with non-negligible probability, given some reasonable restrictions of its computing power and memory requirements. For this purpose, a natural solution is to estimate the metrics introduced in [31], namely the o-th order success rate (which gives the probability that the correct key stands among the o first ones provided by the attack), and the guessing entropy (which is the expected rank of the correct key after the attack). However, and as already mentioned in introduction, the estimation of these metrics was so far limited to subkeys, or to success rates of enumerable orders for master keys. In the remainder of this section, we show how efficient rank estimation can be used to mitigate this limitation, and provide the complete picture for side-channel attack security evaluations.

For this purpose, we define an "all-order" success rate graph, which provides the probability of a successful key recovery (on the Z axis or color map), depending on both the number of traces (on the X-axis) and the enumeration effort (on the Y-axis). That is, any point in this graph corresponds to the success rate for a given number of queries $q = x$ and order $o = y$. In order to relate this graph with the evaluation framework proposed in [31], we first remark that any of its "slices" obtained for a fixed y corresponds to a y-th order success rate. Furthermore, any slice for a fixed x is a rank distribution graph for a given number of measurements, the mean value of which equals the guessing entropy.

Building such a graph simply requires to perform several rank estimations, for different values of the number of encryption traces measured. The outputs provide a set of intervals that can then be used as input to a kernel density estimation (e.g. with uniform kernels). What is actually of interest to an evaluator is the cumulant of the density function resulting from this estimation: it indicates the probability that a key will be found, depending of the amount of keys that the adversary can enumerate. Interestingly, such density estimations converge quickly. For example, the security graphs in Figures 7, 8 were obtained by performing 100 attacks with random keys and estimating the key rank during 5 seconds, which amounts to less than 10 minutes of computation for each possible number of traces. The continuous black (resp. white) lines indicate the minimum (resp. maximum) ranks observed. Basically, any data/enumeration point below the black line seems safe, while points above the white line lead to certain key recovery. The medium zone indicates a non-negligible probability of key recovery.

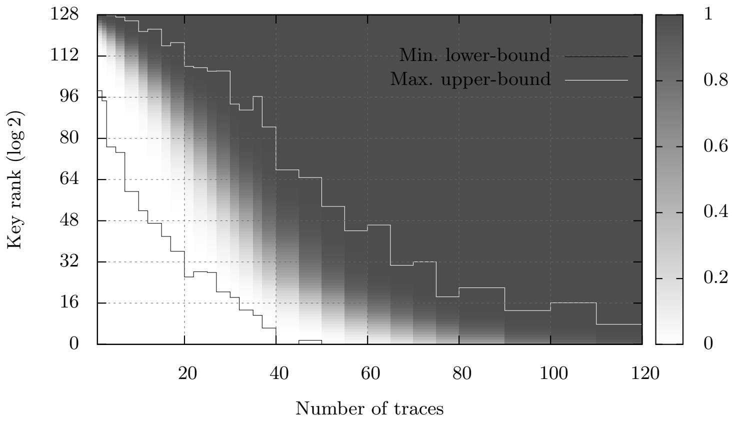


Fig. 7. Example of a security graph for a template attack against unprotected AES

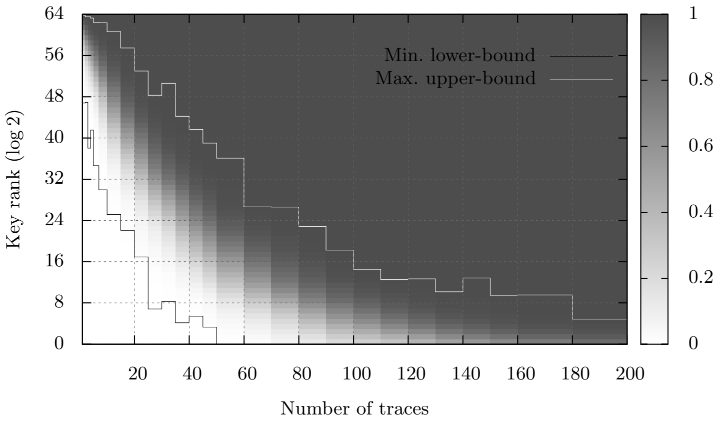


Fig. 8. Example of a security graph for a template attack against unprotected LED

For illustration, we produced the security graphs for implementations of the block ciphers AES and LED [16], with respective key lengths 128 and 64 bits. Our experiments exploited exactly the same leakage models as in the previous section, with the same noise variance for both ciphers, and the number of target subkey bytes as only difference. In the case of the AES implementation shown in Figure 7, an adversary with a personal computer spending two weeks of computation (i.e. enumerating $\approx 2^{40}$ keys) will have a small chance of recovering the master key when she has less than 15 traces at her disposal, and will almost certainly succeed if she has more than 60 traces. Comparing this result with the first-order success rate curve, we observe that this success rate is still stuck at zero for 60 traces, hence suggesting the necessity of key ranking in security evaluations. Besides, it is also interesting to observe that the impact of side-channel leakage is more critical for LED, as a small number of traces (e.g. 5 in the example of Figure 8) already allows decreasing the computing power required for possible key recovery down to approximately 2^{40} (whereas it is still around 2^{80} for the AES given the same number of traces). This vulnerability is simply due to the smaller key space. In general, the graphs of Figures 7 and 8 can be used directly to determine the re-keying rate in a leakage-resilient construction. The designer just has to choose an enumeration threshold corresponding to the computing power of the adversary considered, then extract the number of measurements that can be tolerated without risking a key recovery (using the figures black line minus a small security margin). Note that the rank and related enumeration effort can be translated into both time and memory costs thanks to the enumeration algorithm presented in [32]. Exemplary performances are reported in Table 1 (where enumeration times do not include the key testing as it depends on the cipher under attack). Figures 7 and 8 thus represent the security of a target device for different data/time/memory trade-offs.

Table 1. Time and memory requirements for key enumeration using [32]

# of keys enumerated	2^{20}	2^{30}	2^{40}	2^{50}	2^{60}
Time	0.25 s	6 m	2 w	165 y	77×10^5 y
Memory	<3 MB	100 MB	11.5 GB	1.35 TB	157 TB

Eventually, and in order to confirm that the tools introduced in this paper apply to actual implementations as easily as to simulated experiments, we built the security graph corresponding the best attack submitted to the DPA contest v2 [26], depicted in Figure 9. Producing this graph required approximately 20 minutes of computation on an 8-core computer, and did not imply any modification of the rank estimation algorithm. Note that this evaluation was performed on the public database of the contest, which is easier to attack than the private one. This explains the small discrepancy between our graph and the "Hall of Fame" available online. Namely, the best attack reported in the contest needs 1173 traces to reach an 80% key recovery success rate when no enumeration is done. By adding enumeration up to rank 2^{32}, the data complexity requirement falls down to only 439 traces! In the "easier" context of the public database (described in Figure 9), the security graph shows that only 350 traces are needed with a 2^{32}-key enumeration, while a first-order success rate reaches 80% for 935 traces. To conclude, we mention that just as our rank estimation applies to profiled and non-profiled DPA, it also applies to protected implementations (e.g. with masking [7,15] or shuffling [17,21]). The main reason is that standard side-channel attacks against such implementations would produce lists of subkey scores or probabilities, as described beforehand and exploited in this paper.

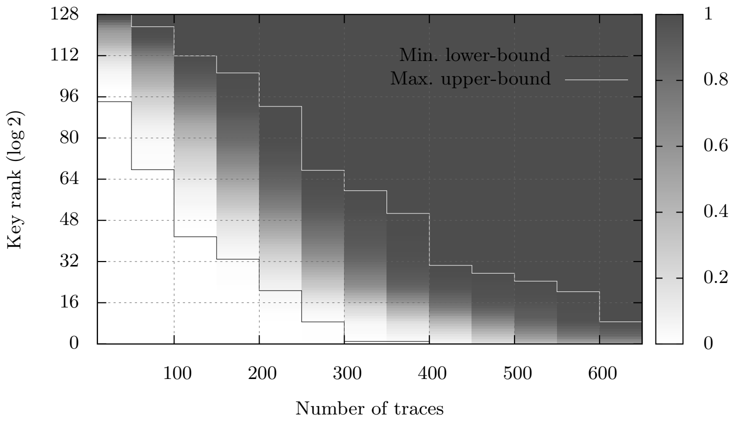


Fig. 9. Security graph for the latest attack of DPA contest v2 (public traces)

5 Conclusion

In this paper, we presented an algorithm for key rank estimation. It can be seen as the evaluator's counterpart to the attacker's key enumeration algorithm described in [32]. The rank estimation algorithm allows an evaluator to predict the workload of an attacker trying to recover a cryptographic key using the outcome of a side-channel attack. It typically returns accurate interval estimates $[2^x; 2^{x+e}]$ of the key rank in a few seconds. By contrast, the enumeration algorithm returns an exact rank for keys of which the rank is small enough (e.g. below 2^{40}), but will fail for keys of which the rank is beyond computing power.

Besides their direct application to single attacks, these two algorithms naturally gain additional interest in an evaluation setting where multiple attacks can be launched to analyze the security of a leaking device with statistical confidence. In particular, the combination of key enumeration and rank estimation enables the efficient computation of the metrics defined in [31]. Taking advantage of these tools, we are now able to produce security graphs that summarize the success probabilities of side-channel attacks, according to both the number of traces and the computational power available to an adversary.

We hope that these tools will be beneficial to researchers and evaluators and allow more thorough physical security evaluations. To that purpose, open-source C++ implementations (and `matlab` hooks) for both the enumeration and rank estimation algorithms are distributed via the authors' home pages.

Acknowledgements. This work has been funded in parts by the ERC project 280141 (acronym CRASH), the 7th framework European project PREMISE and the Walloon Region project MIPSs. F.-X. Standaert is a Research Associate of the Belgian Fund for Scientific Research (FNRS-F.R.S).

References

1. ANSSI. Agence nationale de la securite des systemes d'information, `http://www.ssi.gouv.fr/en/products/certified-products/` (retrieved on August 1, 2012)
2. Bellare, M., Desai, A., Jokipii, E., Rogaway, P.: A concrete security treatment of symmetric encryption. In: FOCS, pp. 394–403. IEEE Computer Society (1997)
3. Biham, E., Dunkelman, O., Keller, N.: New results on boomerang and rectangle attacks. In: Daemen, J., Rijmen, V. (eds.) FSE 2002. LNCS, vol. 2365, pp. 1–16. Springer, Heidelberg (2002)
4. Biham, E., Shamir, A.: Differential cryptanalysis of DES-like cryptosystems. In: Menezes, A., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 2–21. Springer, Heidelberg (1991)
5. Brier, E., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. In: Joye and Quisquater [18], pp. 16–29
6. BSI. Federal office for information security, `https://www.bsi.bund.de/en/topics/certification/certification_node.html` (retrieved on August 1, 2012)
7. Chari, S., Jutla, C.S., Rao, J.R., Rohatgi, P.: Towards sound approaches to counteract power-analysis attacks. In: Wiener [33], pp. 398–412
8. Chari, S., Rao, J.R., Rohatgi, P.: Template attacks. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 13–28. Springer, Heidelberg (2003)
9. Daemen, J., Rijmen, V.: The wide trail design strategy. In: Honary, B. (ed.) Cryptography and Coding 2001. LNCS, vol. 2260, pp. 222–238. Springer, Heidelberg (2001)
10. Daemen, J., Rijmen, V.: Probability distributions of correlation and differentials in block ciphers. Journal of Mathematical Cryptology 1(3), 221–242 (2007)
11. Dodis, Y., Kalai, Y.T., Lovett, S.: On cryptography with auxiliary input. In: Mitzenmacher, M. (ed.) STOC, pp. 621–630. ACM (2009)
12. Dziembowski, S., Pietrzak, K.: Leakage-resilient cryptography. In: FOCS, pp. 293–302. IEEE Computer Society (2008)
13. Gérard, B., Standaert, F.-X.: Unified and optimized linear collision attacks and their application in a non-profiled setting. In: Prouff and Schaumont [27], pp. 175–192
14. Gierlichs, B., Batina, L., Tuyls, P., Preneel, B.: Mutual information analysis. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 426–442. Springer, Heidelberg (2008)
15. Goubin, L., Patarin, J.: Des and differential power analysis the "duplication" method. In: Koç, Ç.K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, pp. 158–172. Springer, Heidelberg (1999)
16. Guo, J., Peyrin, T., Poschmann, A., Robshaw, M.: The LED block cipher. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 326–341. Springer, Heidelberg (2011)

17. Herbst, C., Oswald, E., Mangard, S.: An aes smart card implementation resistant to power analysis attacks. In: Zhou, J., Yung, M., Bao, F. (eds.) ACNS 2006. LNCS, vol. 3989, pp. 239–252. Springer, Heidelberg (2006)
18. Joye, M., Quisquater, J.-J. (eds.): CHES 2004. LNCS, vol. 3156. Springer, Heidelberg (2004)
19. Junod, P.: On the complexity of Matsui's attack. In: Vaudenay, S., Youssef, A.M. (eds.) SAC 2001. LNCS, vol. 2259, pp. 199–211. Springer, Heidelberg (2001)
20. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener [33], pp. 388–397
21. Mangard, S., Oswald, E., Popp, T.: Power analysis attacks - revealing the secrets of smart cards. Springer (2007)
22. Matsui, M.: Linear cryptoanalysis method for DES cipher. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 386–397. Springer, Heidelberg (1994)
23. Medwed, M., Standaert, F.-X., Joux, A.: Towards super-exponential side-channel security with efficient leakage-resilient prfs. In: Prouff and Schaumont [27], pp. 193–212
24. Moradi, A., Mischke, O., Eisenbarth, T.: Correlation-enhanced power analysis collision attack. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 125–139. Springer, Heidelberg (2010)
25. Pan, J., van Woudenberg, J.G.J., den Hartog, J.I., Witteman, M.F.: Improving DPA by peak distribution analysis. In: Biryukov, A., Gong, G., Stinson, D.R. (eds.) SAC 2010. LNCS, vol. 6544, pp. 241–261. Springer, Heidelberg (2011)
26. Telecom ParisTech., `http://www.dpacontest.org/` (retrieved on August 1, 2012)
27. Prouff, E., Schaumont, P. (eds.): CHES 2012. LNCS, vol. 7428. Springer, Heidelberg (2012)
28. Schindler, W., Lemke, K., Paar, C.: A stochastic model for differential side channel cryptanalysis. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 30–46. Springer, Heidelberg (2005)
29. Schramm, K., Leander, G., Felke, P., Paar, C.: A collision-attack on AES: Combining side channel- and differential-attack. In: Joye and Quisquater [18], pp. 163–175
30. Selçuk, A.A.: On probability of success in linear and differential cryptanalysis. J. Cryptology 21(1), 131–147 (2008)
31. Standaert, F.-X., Malkin, T.G., Yung, M.: A unified framework for the analysis of side-channel key recovery attacks. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 443–461. Springer, Heidelberg (2009)
32. Veyrat-Charvillon, N., Gérard, B., Renauld, M., Standaert, F.-X.: An optimal key enumeration algorithm and its application to side-channel attacks. In: Knudsen, L.R., Wu, H. (eds.) SAC 2012. LNCS, vol. 7707, pp. 390–406. Springer, Heidelberg (2013)
33. Wiener, M. (ed.): CRYPTO 1999. LNCS, vol. 1666. Springer, Heidelberg (1999)
34. Yu, Y., Standaert, F.-X., Pereira, O., Yung, M.: Practical leakage-resilient pseudorandom generators. In: Al-Shaer, E., Keromytis, A.D., Shmatikov, V. (eds.) ACM Conference on Computer and Communications Security, pp. 141–151. ACM (2010)

Masking against Side-Channel Attacks: A Formal Security Proof

Emmanuel Prouff[1] and Matthieu Rivain[2]

[1] ANSSI
emmanuel.prouff@ssi.gouv.fr
[2] CryptoExperts
matthieu.rivain@cryptoexperts.com

Abstract. Masking is a well-known countermeasure to protect block cipher implementations against side-channel attacks. The principle is to randomly split every sensitive intermediate variable occurring in the computation into $d+1$ shares, where d is called the masking order and plays the role of a security parameter. Although widely used in practice, masking is often considered as an empirical solution and its effectiveness is rarely proved. In this paper, we provide a formal security proof for masked implementations of block ciphers. Specifically, we prove that the information gained by observing the leakage from one execution can be made negligible (in the masking order). To obtain this bound, we assume that every elementary calculation in the implementation leaks a noisy function of its input, where the amount of noise can be chosen by the designer (yet linearly bounded). We further assume the existence of a leak-free component that can refresh the masks of shared variables. Our work can be viewed as an extension of the seminal work of Chari *et al.* published at CRYPTO in 1999 on the soundness of combining masking with noise to thwart side-channel attacks.

1 Introduction

Side-channel analysis is a class of cryptanalytic attacks that exploit the physical environment of a cryptosystem to recover some *leakage* about its secrets. It is often more efficient than a cryptanalysis in the so-called *black-box model* in which no leakage occurs. Two attack categories are usually considered: the *bounded side-channel attacks* and the *continuous side-channel attacks*. In a bounded side-channel attack [9], the total amount of leakage accessible to the adversary is bounded. In a continuous side-channel attack, the adversary gets some information at each invocation of the cryptosystem, and the amount of leakage can thus be arbitrarily large. Attacks where the adversary measures the running-time [24], the power consumption [25] or the electromagnetic radiations [15] of a cryptographic implementation fall into this category.

Continuous side-channel attacks have proved to be especially effective to break unprotected cryptographic implementations. And although many ingenious countermeasures have been developed during past years, very few of them gave rise to concrete security guaranties. This has raised the need for models and methods to formally prove the security of cryptographic implementations against continuous side-channel attacks. A pioneering work in this direction is the *physically observable cryptography* framework introduced by Micali and Reyzin in [29] which puts forward a general theory of side-channel attacks. In particular, they formalize the assumptions that a cryptographic device can at least keep some secrets and that *only computation leaks information* [29]. A few years later, Dziembowski and Pietrzak introduced the *leakage resilient cryptography* model [12], which is a generalization of the *bounded retrieval model* [9] where every step of the computation leaks information on the processed part of the device state trough a function whose range is bounded (*i.e.* taking values in $\{0,1\}^\lambda$ for some parameter λ). Under this assumption, the authors were able to design secure pseudo-random number generators [12,32]. Further leakage resilient cryptographic primitives were then constructed under the same – or sometimes stronger – assumptions (see for instance [10, 13, 23, 42]). The issue of designing generic compilers that can transform any cryptographic algorithm into a leakage resilient implementation was also recently addressed [11, 17, 18, 22]. These works are nice proofs of concept but the proposed constructions are not suited for practical implementation, especially in constrained environments such as embedded systems. Moreover the practical meaning of the underlying bounded range leakage model with respect to power or electromagnetic side channels is questionable [40].

A more practical and traditionally used approach to secure implementations against side-channel attacks is *secret sharing* [1, 37] also called *masking* in this context. The idea is to randomly split a secret into several shares such that the adversary needs all of them to reconstruct the secret. Masking was soon identified as a sound countermeasure when side-channel attacks appeared in the literature [4, 19]. Since then, many works have been published to address the practical implementation and/or the security of masking for various ciphers. However a formal security proof is still missing at this day. Our goal is to fill this gap.

1.1 Related Works

Soundness of Masking. In [4], Chari *et al.* conduct a formal study of masking in the presence of noisy leakage. More precisely, the authors investigate the soundness of randomly sharing a secret bit into d shares when the adversary has only access to a noisy version of those shares, the noise having a Gaussian distribution with variance σ^2. They prove that the number of observations required to distinguish, with success probability α, the leakage distribution when the secret equals 0 from the leakage distribution when the secret equals 1 is lower bounded by $\sigma^{d+4\log\alpha/\log\sigma}$. This bound is frequently recalled to argue for the soundness of masking when combined with noise. Despite its great interest and impact, Chari *et al.*'s analysis has an important limitation: no solution is provided to apply masking to the whole implementation of a cryptographic algorithm and to analyze the global security of such an implementation.

Private Circuits and Extensions. In [21], Ishai *et al.* show that any circuit with n logical gates can be transformed into a circuit of size $O(nd^2)$ which is secure against *probing attacks* spying up to d wires. The main contribution of [21] is a method to compute an AND gate between shared inputs while ensuring the security against a d-probing adversary. However, a security proof against probing attacks does not give full satisfaction since it does not encompass an adversary exploiting the entire leakage produced during the processing.

In [14], Faust *et al.* propose an extension of Ishai *et al.*'s scheme. Their scheme requires a leak-free hardware component but it is provably secure under two different and more general leakage models. In the first model, the leakage at each cycle is any function of the circuit internal state (*i.e.* the logical values carried by all the wires) which is computationally bounded: it is assumed to be in the complexity class AC^0 (*i.e.* it must be computable by a circuit of constant depth). In the second model, the leakage reveals the value of each internal state bit, flipped with a probability $p < 1/2$ (*i.e.* xor-ed with a p-Bernoulli noise). In a recent paper [35], Rothblum further showed how to remove the requirement of leak-free component in the AC^0 leakage model. These works achieve an important progress towards provable security against side-channel attacks since they show that masking can bring security even in the presence of a global leakage on the entire state. However, the practicability of the considered models is questionable. In particular it is unclear whether the AC^0 leakage or the full leakage with Benouilli noise really fit the physical reality of power and/or electromagnetic leakages.

Masking Schemes. On the other hand, several works have shown how to apply masking to various algorithms in practice. They however often omit to prove the security of the resulting implementations. The first masking scheme was proposed by Goubin and Patarin in [19] for the DES cipher. Further schemes were subsequently published in which masking is applied at hardware or software level at the cost of different area-time-memory trade-offs (see for instance [2,28,30]). Originally, most of these schemes deal with *first-order masking* which splits each sensitive variable in two shares (a mask and a masked variable). Then *higher-order masking schemes* were defined to get security against side-channel attacks exploiting the leakage of several, say d, intermediate computations [3, 16, 33, 34]. The purpose of these schemes is analogous to the d-probing secure circuit of Ishai *et al.* : the computation is performed such that any d intermediate variables occurring in the algorithm reveal no sensitive information. Most of these schemes are actually based on the method of Ishai *et al.* to securely process a multiplication between two shared variables. Consequently, they suffer the same limitation as Ishai *et al.* 's scheme: they only thwart a limited adversary that does not exploit the overall leakage.

1.2 Our Contribution

In this paper we formally prove the security of masked implementations of block ciphers in the *only computation leaks information* model [29]. In this model, every *step* of the processing reveals a leakage function of the touched part of the device state. This function is chosen adaptively by the adversary in some predetermined class. For our security proof, we split the computation into several *elementary calculations* (in practice, a sequence of few CPU instructions) that each accesses a subpart x of the device state and leaks a function of x. Starting from the observation that masking is sound when combined with noise [4] and that many practical solutions exist to amplify leakage noise (see for instance [6–8, 20, 27, 39, 41]), we assume the leakage functions to be *noisy*. The noisy feature of a leakage function f is captured by assuming that an observation of $f(x)$ only implies a *bounded bias* in the probability distribution of x. Namely the statistical distance between the distributions $\mathrm{P}[x]$ and $\mathrm{P}[x|f(x)]$ is bounded. We further assume that this bound depends on a noise parameter ω that can be chosen by the designer according to the required security level. Our security proof has a natural limitation which is the requirement of a leak-free component, an elementary calculation refreshing the masks of a shared variable. Under these assumptions, we achieve an information theoretic security proof: we show that the mutual information between the cipher input (plaintext and secret key) and the overall leakage on the block cipher processing is upper bounded by $\omega^{-(d+1)}$, where d is the masking order.

This bound can be seen as an extension of the seminal work of Chari *et al.* [4] as it is derived from the combination of masking with noise. We extend their analysis in two ways. First we consider a more general leakage model which no longer requires particular assumptions (single-bit target variable, Gaussian noise). More importantly, we provide a security bound for a full masked block cipher implementation whereas Chari *et al.* analysis focus on leaking shares independently of any computation. Our work can also be viewed as an alternative to previous works on program or circuit compilers with formal security proofs against side-channel attacks [11,14,17,18,22,35]. Whereas the practical meaning of the leakage models considered in these works is questionable, our leakage model aims to be compliant with practical investigations about side-channel leakage (see for instance [27,31,36,38]).

2 Preliminaries

Calligraphic letters, like $\mathcal{X}$, are used to denote finite sets. The corresponding large letter X is used to denote a random variable over $\mathcal{X}$, while the lower-case letter x denotes a particular element from $\mathcal{X}$. To every discrete random variable X, one associates a probability mass function P_X defined by $\mathrm{P}_X(x) = \mathrm{P}[X = x]$. Let Y be a random variable defined over some set $\mathcal{Y}$ and let $y \in \mathcal{Y}$. Then $(X|Y = y)$ denotes the random variable with probability mass function $x \mapsto \mathrm{P}[X = x|Y = y]$. The *entropy* (or *Shannon entropy*) $\mathrm{H}[X]$ of a discrete random variable X is defined by $\mathrm{H}[X] = -\sum_{x\in\mathcal{X}} \mathrm{P}_X(x) \log_2(\mathrm{P}_X(x))$. The *mutual information* between two random variables X and Y is then defined by $\mathrm{I}(X; Y) = \mathrm{H}[X] - \mathrm{H}[X|Y]$, where $\mathrm{H}[X|Y]$ is called the *conditional entropy of X given Y* and is defined by $\mathrm{H}[X|Y] = \sum_{y\in\mathcal{Y}} \mathrm{P}_Y(y)\ \mathrm{H}[(X|Y = y)]$. The *statistical distance* between two random variables X_0 and X_1 is denoted by $\mathrm{d}(X_0; X_1)$ and is defined by $\mathrm{d}(X_0; X_1) = \|\mathrm{P}_{X_0} - \mathrm{P}_{X_1}\|$ where $\|\cdot\|$ denotes the Euclidean norm and P_{X_i} denotes the vector $(\mathrm{P}_{X_i}(x))_{x\in\mathcal{X}}$. We recall that for any N-dimensional vector $\boldsymbol{y} = (y_1, y_2, \dots, y_N)$ we have

$$\|\boldsymbol{y}\| \leq L_1(\boldsymbol{y}) \leq \sqrt{N}\|\boldsymbol{y}\| \ , \tag{1}$$

where $L_1(\boldsymbol{y})$ denotes the *Manhattan norm* $\sum_i |y_i|$.

We now introduce the notion of *bias* that is extensively used in our security proof.

Definition 1. *Let X and Y be two random variables. The* bias of X given $Y = y$*, denoted $\beta(X|Y = y)$, is defined as*

$$\beta(X|Y = y) = \mathrm{d}\big(X; (X|Y = y)\big) \ .$$

The bias of X given Y*, $\beta(X|Y)$, is then defined as the expected bias of X given $Y = y$ over $\mathcal{Y}$, that is*

$$\beta(X|Y) = \sum_{y\in\mathcal{Y}} \mathrm{P}_Y(y)\beta(X|Y = y) \ .$$

The bias of X given Y is an information metric between X and Y. If X and Y are independent then $\beta(X|Y)$ equals zero. Moreover, as shown in the next proposition, it is related to the mutual information between X and Y (the proof is provided in the full version).

Proposition 1. *Let X and Y be two random variables, with X uniformly distributed over a set $\mathcal{X}$ of cardinality N. The mutual information between X and Y satisfies $\mathrm{I}(X; Y) \leq \frac{N}{\ln 2}\beta(X|Y)$.*

3 Model of Leaking Computation

We describe hereafter our model of leaking computation.

An algorithm is modelled by a sequence of *elementary calculations* $(\mathsf{C}_i)_i$ that are Turing machines augmented with a common random access memory called *the state.* Each elementary calculation reads its input and writes its output on the state. When an elementary calculation C_i is invoked, its input is written from the state to its input tape, then C_i is executed, afterwards its output is written back to the state.

A *physical implementation* of an algorithm is modelled by a sequence of *physical elementary calculations.* A physical elementary calculation $(\mathsf{C}_i, f_i)_i$ is composed of an elementary calculation C_i and a *leakage function* f_i. A leakage function is defined as a function that takes two parameters: the value held by the accessed part of the state and a random string long enough to model the leakage noise. Let $\mathcal{I} = (\mathsf{C}_i, f_i)_i$ be a physical implementation of an algorithm $\mathcal{A}$ as defined above. Each execution of $\mathcal{I}$ leaks the values $\big(f_i(x_i, r_i)\big)_i$ where x_i denotes the input of C_i and r_i is a fresh random string. In particular all the r_i involved in successive executions of $\mathcal{I}$ are uniformly and independently drawn.

For the sake of simplicity, we shall omit the random string parameter, which leads to the notation $f_i(x)$ where x is the accessed value. Note that $f_i(x)$ is not the result of a function but it can be seen as the output of a probabilistic Turing machine. In particular, $f_i(x)$ can take several values with a given probability distribution, and is therefore considered as a random variable in the following. Let $\mathcal{X}$ be the definition set of the accessed part of the state. We shall then say that f_i is defined over $\mathcal{X}$ (or equivalently that $\mathcal{X}$ is the domain of f_i).

For our security proof, we will consider special classes of leakage functions that we shall call *noisy leakage functions*. Let f be a leakage function defined over some set $\mathcal{X}$ and let X denote a uniform random variable over $\mathcal{X}$. The noisy property of f is captured by assuming that the bias introduced in the distribution of X given the leakage $f(X)$ is bounded.[1] For any positive real number ε, we define the class of *noisy leakage functions w.r.t. bias* ε, and we denote by $\mathcal{N}(\varepsilon)$, the set of noisy functions such that $\beta(X|f(X)) \leq \varepsilon$. In this paper, we shall assume that the designer can constrain as willing the set of noisy leakage functions related to any elementary calculation by linearly increasing the amount of noise in the leakage. More precisely, we assume that the designer controls a noise parameter ω such that an elementary calculation C can yield a physical elementary calculation (C, f) with $f \in \mathcal{N}(1/\omega)$, where ω is linear in the security parameter.

3.1 Discussion

Our model can be seen as a specialization of the physically observable cryptography framework [29] with leakage functions belonging to the class of noisy functions as defined above (the similarities between our model and this framework are discussed in the full version). Our model is also comparable to the leakage resilient cryptography model [12] with two important differences relating to the computation granularity and the class of leakage functions.

Computation Granularity. As nicely explained in [17], a computation in the *only computation leaks* model is divided into several *sub-computations* and a leakage function applies to the input of each sub-computation. In [12,32] the authors construct stream ciphers that output an unbounded number of key-stream blocks from a secret key block. A sub-computation is then naturally identified as the computation of one block. In contrast, we consider a finer granularity: the computation of one block (*i.e.* a single block cipher computation) is divided into several elementary calculations, each one leaking a function of its input. In other words, [12, 32] address the issue of constructing secure protocols from

[1] For the above definition of noisy leakage functions to be sound, we need to precise the distribution of X while bounding $\beta(X|f(X))$, and a natural choice is the uniform distribution. Of course, this does not constrain the leakage function to only apply on uniformly distributed inputs.

a cryptographic primitive with limited leakage whereas we address the issue of constructing secure cryptographic primitives from elementary calculations with noisy leakages.

Class of Leakage Functions. The most noticeable difference between our work and the previous leakage-resilient constructions resides in the considered class of leakage functions. Most previous works consider the class of bounded-range functions taking values in $\{0,1\}^\lambda$ for some parameter λ [10–13,17,18,22,23,32]. This hypothesis is conservative in terms of security since it encompasses leakage functions with complex structures. However its practical meaning is unclear with regard to power and electromagnetic side channels for which the leakage is usually substantially larger than the secret state (but hopefully does not contain its overall entropy).

In contrast, several techniques are known to add some noise in the side-channel leakage in practice [6–8,20,26,27,39,41]. By noise addition we mean that the relevant signal in the leakage is lowered compared to random variations, although this may not literally result from noise addition (the terminology of *hiding* is sometimes used). This motivates our definition of noisy leakage functions. Note that in practice, power and electromagnetic leakages can realistically be modeled by a multivariate deterministic function g of the processed data with an additional multivariate Gaussian noise N [5,27,36,38]. The class $\mathcal{N}(\varepsilon)$ corresponding to such a leakage function can then be determined from the description of g and N (see the full version for further details).

4 Masked Implementation of Block Ciphers

Several schemes have been proposed to securely process a block cipher composed of linear layers and non-linear s-boxes. In this paper, we prove the security of the scheme described in [3] with a secure multiplication processing close to those of [14,21], and with additional mask-refreshing computations. Before presenting the masking scheme, we start by formalizing the considered block cipher processing.

4.1 Block Cipher Processing

A block cipher is a cryptographic algorithm which, from a secret key, transforms a plaintext block into a ciphertext block. In this paper, we focus on block ciphers designed as a succession of linear functions and substitution boxes (s-boxes). S-boxes are nonlinear functions from $\{0,1\}^n$ to $\{0,1\}^m$ with $m \leq n$ and n small (typically $n \in \{4,6,8\}$). We assume that the addition law is the bitwise addition, denoted $\oplus$, and that the s-boxes are *balanced*; namely every $y \in \{0,1\}^m$ has the same number of preimages in $\{0,1\}^n$ under the s-box. In the computation model introduced in Section 3, the processing of such a block cipher is represented as a sequence of elementary calculations, each of them implementing either a linear function or an s-box. The input of this processing is a pair composed of the plaintext and the secret key and the output is the corresponding ciphertext.

Uniformity Property. We shall assume that the block cipher satisfies the following uniformity property: a uniform distribution of the cipher input (plaintext-key pair) induces a uniform distribution of the input of every of its elementary calculation. The uniformity property is satisfied by classical block cipher designs such as DES and AES.

4.2 Securing the Block Cipher Processing

We start with the following definition that formalizes the notion of d^{th}*-order encoding.*

Definition 2 ($d^{\mathbf{th}}$-order encoding). *Let d be a positive integer and let I denote the integer interval $\{0, 1, \ldots, d\}$. The* d^{th}-order encoding *of $x \in \mathcal{X}$ is a $(d+1)$-tuple $(x_i)_{i \in I}$ satisfying $\bigoplus_i x_i = x$ and such that $(x_i)_{i \in I \setminus \{i_0\}}$ is uniformly distributed over $\mathcal{X}^d$ for every $i_0 \in I$.*

Masking a block cipher implementation consists in choosing a security parameter d (called *masking order*) and in performing the computation on a d^{th}-order encoding of the state. Namely, the plaintext and the secret key are split into $d+1$ shares. Then, a scheme is defined that specifies how each elementary calculation is replaced by a sequence of new elementary calculations operating only on the shares. At the end, the new sequence must return an encoding of the ciphertext (from which the actual ciphertext can be straightforwardly recovered).

According to the block cipher model of Section 4.1, we describe hereafter how to process a linear function and an s-box computation on shared inputs as proposed in [3]. We first introduce the mask-refreshing component used at several steps in the masking scheme and which is assumed to be *leak-free* in our security proof.

Mask Refreshing. Our scheme requires a special kind of elementary calculation to refresh the masks of an encoding *without leaking information.* This mask-refreshing oracle is denoted by $\mathcal{O}^{\$}$. From an encoding of any value x, it computes a new encoding of x with fresh random values. For the sake of simplicity, we assume that when invoked the input and output of our leak-free component do not leak. However this assumption could be relaxed since the input comes from a previous elementary calculation and the output is used in the subsequent elementary calculations (otherwise its masks would not need to be refreshed), therefore they both leak at some point in the computation.

Secure Linear Function Processing. To secure the processing of any linear function g, the following process is applied:

1. For every $i \in \{0, 1, \ldots, d\}$, process $z_i \leftarrow g(x_i)$.
2. Output $(z_i)_i \leftarrow \mathcal{O}^{\$}\big((z_i)_i\big)$.
3. If the encoding $(x_i)_i$ is used subsequently in the block cipher processing, process $(x_i)_i \leftarrow \mathcal{O}^{\$}\big((x_i)_i\big)$.

Secure S-Box Processing. Let S be an s-box with input dimension n and output dimension $m \leq n$. Then S can be represented by a polynomial function $x \in \mathbb{F}_{2^n} \mapsto \bigoplus_{i=0}^{2^n-1} \alpha_i x^i$ where the α_i are constant coefficients in $\mathbb{F}_{2^n}$. The α_i can be computed from the s-box look-up table by applying Lagrange's Theorem.[2] Thanks to this representation, the s-box calculation can be done by processing four kinds of elementary calculations over $\mathbb{F}_{2^n}$: addition, multiplication by a constant, square, and regular multiplication (*i.e.* of two different elements). The three former kinds of calculations are linear (or affine including the addition by a non-zero constant) and their processing can hence be done exactly as for linear transformations. When the calculation is a regular multiplication, the following scheme is applied.

Secure Regular Multiplication Processing. To secure the processing of a regular multiplication, we use a method similar to that of [14, 21]. The input is a pair of encodings of the multiplication operands a and b. The definition of the sequence of elementary calculations computing the encoding of $a \times b$ is deduced from the following relation: $a \times b = \left(\bigoplus_i a_i\right) \times \left(\bigoplus_i b_i\right) = \bigoplus_{i,j} a_i \times b_j$. It is described hereafter:

1. For every $(i, j) \in \{0, 1, \ldots, d\}^2$, process $v_{i,j} \leftarrow a_i \times b_j$.
2. For every $j \in \{0, 1, \ldots, d\}$, process $(v_{0j}, v_{1j}, \ldots, v_{dj}) \leftarrow \mathcal{O}^\$(v_{0j}, v_{1j}, \ldots, v_{dj})$.
3. Process $(v_{00}, v_{01}, \ldots, v_{0d}) \leftarrow \mathcal{O}^\$(v_{00}, v_{01}, \ldots, v_{0d})$.
4. For every $i \in \{0, 1, \ldots, d\}$, process $z_i \leftarrow \bigoplus_{j=0}^{d} v_{i,j}$
5. Output $(z_i)_i \leftarrow \mathcal{O}^\$\big((z_i)_i\big)$.
6. If one of the input encodings $(a)_i$ and $(b_i)_i$ is involved in a subsequent elementary calculation, then process $(a_i)_i \leftarrow \mathcal{O}^\$\big((a_i)_i\big)$ and/or $(b_i)_i \leftarrow \mathcal{O}^\$\big((b_i)_i\big)$.

Note that Steps 2 and 3 intend to refresh the masks between the subsequences of elementary calculations in Steps 1 and 4. Namely, these steps render the probability distributions $\big((a_i)_i, (b_j)_j | (a, b)\big)$ (in Step 1) and $\big((v_{i,j})_{i,j} | (a, b)\big)$ (in Step 4) mutually independent. Note that Step 3 is mandatory to this aim as Step 2 only makes $\big((v_{i,j})_i | (a, b)\big)$ independent of $\big((a_i)_i, (b_j)_j | (a, b)\big)$ for every column j separately. From this point, Step 3 ensures the global independence.

For our security proof, we shall consider each sum $z_i \leftarrow \bigoplus_{j=0}^{d} v_{i,j}$ in Step 4 as d successive elementary calculations $t_{i,j} \leftarrow t_{i,j-1} \oplus v_{i,j}$ for $1 \leq j \leq d$ with $t_{i,0} = v_{i,0}$ and giving $z_i = t_{i,d}$.

It is clear from the above description that securing a regular multiplication is expensive compared to securing a linear function. The complexity of a secure multiplication is quadratic in d whereas the complexity of a secure linear function is linear in d. Moreover the secure multiplication requires several calls to the mask refreshing oracle. For efficiency purpose, one should hence try to minimize the number of multiplications in the polynomial representation of the s-box. We

[2] When m is strictly lower than n, the m-bit outputs can be embedded in $\mathbb{F}_{2^n}$ by padding them to n-bit outputs (*e.g.* by setting most significant bits to 0). The padding is then removed after the polynomial evaluation.

refer to [3] where efficient heuristics of polynomial evaluation are proposed with respect to this criterion.

5 Main Theorems

It is well-known that any subset of at most d shares X_i gives no information on a secret X encoded at order d, while the whole $d+1$ shares enable to fully reconstruct the secret. In [4], Chari *et al.* consider an adversary which has access to the noisy version of all the shares, i.e. $X_i + N_i$ where N_i is a Gaussian noise with standard deviation σ. They restrict themselves to the case of a single secret bit and show that distinguishing the distribution of the noisy shares associated to a secret bit at 0 and that associated to a secret bit at 1 with a probability $\alpha \in [0,1)$ requires at least $\sigma^{d+4\log\alpha/\log\sigma}$ samples. In other words, the required number of leakage observations increases exponentially with the masking order, the underlying base being the noise standard deviation.

Chari *et al.* 's result demonstrates the soundness of using masking under a practically relevant leakage model. However, they only focus on a static leakage of the shares and not on the leakage occurring while computing on the shares. In this paper, we fill this gap by providing security bounds for masked implementations that process shared variables. As explained in Section 4, a block cipher may be decomposed into linear operations and multiplications in a finite field. We derive hereafter upper bounds on the amount of sensitive information leakage for both operations when protected by masking. The proofs of the corresponding theorems are given in the full version. Then in Section 6, we derive an upper bound on the information leakage for the full masked implementation of the cipher.

5.1 Sequential Processing of the Shares

Our first context deals with the case where the shares are processed sequentially *e.g.* by applying the same linear function to them. For such a processing, we provide hereafter an upper bound on the bias of the secret value distribution given noisy leakages on its shares.

Theorem 1. *Let X be a uniform random variable over some set $\mathcal{X}$ of cardinality N, let d be a positive integer and let $(X_i)_i$ be a d^{th}-order encoding of X. Let $\varepsilon \in [0,1)$ and let f_0, f_1, ..., f_d be noisy leakage functions defined over $\mathcal{X}$ and belonging to $\mathcal{N}(\varepsilon)$. We have:*

$$\beta\big(X \big| f_0(X_0), f_1(X_1), \dots, f_d(X_d)\big) \leq N^{\frac{d}{2}} \varepsilon^{d+1} \ .$$

Theorem 1 shows that the bias of X given the leakages on its shares decreases exponentially with the order d, provided that the initial bias is sufficiently low, namely lower than $\frac{1}{\sqrt{N}}$. Seeing the *amount of noise* as the inverse of the bias, we hence obtained an upper-bound that decreases exponentially with the encoding order, the base of the exponent being the amount of noise. This result is in accordance with [4] while being more general since it encompasses any (univariate or multivariate) leakage distribution and any data dimension.

In some contexts, the shared variable is not uniformly distributed, but it is a deterministic function of a uniform secret variable. This case is addressed in the following corollary of Theorem 1.

Corollary 1. *Let X be a uniform random variable over some set $\mathcal{X}$ of cardinality N and let g be a deterministic function from $\mathcal{X}$ to $\mathcal{X}$. Let d be a positive integer and let $(G_i)_i$ be a d^{th}-order encoding of $g(X)$. Let $\varepsilon \in [0,1)$ and let f_0, f_1, …, f_d be noisy leakage functions defined over $\mathcal{X}$ and belonging to $\mathcal{N}(\varepsilon)$. We have:*

$$\beta\big(X \big| f_0(G_0), f_1(G_1), \ldots, f_d(G_d)\big) \leq N^{\frac{d+3}{2}} \varepsilon^{d+1} \ .$$

5.2 Multiplication of the Shares

The previous theorem deals with a situation where all the shares leak separately which matches the context of the secure linear functions processing. However it is not sufficient alone to deduce an upper bound for the secure multiplication processing given in Section 4. In the latter case, the secure multiplication of two variables A and B from their respective encodings $(A_i)_i$ and $(B_j)_j$ requires to compute the cross-terms $A_i \times B_j$. Hence each share A_i of A appears in $d+1$ different multiplications, one per share B_j, and *vice versa*. Our strategy is first to bound the bias on each share A_i and B_j given the multiple leakages on each share. We can then bound the bias of A and B using a similar approach as in Theorem 1, and we finally derive a bound for the bias of the pair (A, B).

We give hereafter our result for the bias given multiple leakages, and then provide our result for the bias given the cross-term leakages.

Bias Given Multiple Leakages. The next theorem deals with the case of repeated leakages on a variable X. We will then apply it in the secure multiplication context.

Theorem 2. *Let X be a uniform random variable defined over a finite set $\mathcal{X}$ of cardinality N. Let $\varepsilon \in [0,1)$ and let f_1, f_2, …, f_t be t noisy leakage functions defined over $\mathcal{X}$ and belonging to $\mathcal{N}(\varepsilon)$. For any real number $\alpha \in (0,1]$, if $\varepsilon \leq \frac{\alpha}{tN}$, then we have*

$$\beta(X | f_1(X), f_2(X), \ldots, f_t(X)) \ \leq \ \left(\left(\frac{\mathrm{e}^{\alpha} - 1}{\alpha}\right) t + \mathrm{e}^{\alpha}\right) \varepsilon \ .$$

The bound in Theorem 2 shows that the bias of X given t leakages increases linearly with t. A requirement is that the bias given a single leakage, namely ε, is t times lower than $\frac{1}{N}$ or less, namely $\varepsilon \leq \frac{\alpha}{tN}$ for some $\alpha \in (0,1]$. Then the bias of X given t leakages is smaller that $\lambda(t)\,\varepsilon$ where λ is an affine function. The value α provides a trade-off between the constraint on ε and the coefficients of λ. If $\alpha = 1$ then $\lambda(t) = (\mathrm{e}-1)t + \mathrm{e} \approx 1.72\,t + 2.72$, and when α approaches 0, then $\lambda(t)$ tends towards $t+1$.

Bias Given Cross-Term Leakages. The next theorem gives an upper bound on the bias of a uniform pair (A, B) given the leakage $(f_{i,j}(A_i, B_j))_{i,j}$.

Theorem 3. *Let A and B be two random variables uniformly distributed over some finite set $\mathcal{X}$ of cardinality N. Let d be a positive integer, and let $(A_i)_i$ and $(B_j)_j$ be two d^{th}-order encodings of A and B respectively. Let ε be a real number such that $\varepsilon \leq \frac{\alpha}{(d+1)N^2}$ for some $\alpha \in (0, 1]$ and let $(f_{i,j})_{i,j}$ be noisy leakage functions defined over $\mathcal{X} \times \mathcal{X}$ and belonging to $\mathcal{N}(\varepsilon)$. We have:*

$$\beta\big((A, B)|(f_{i,j}(A_i, B_j))_{i,j}\big) \leq 2N^{\frac{3d+2}{2}} \big((\lambda_1 d + \lambda_0)\varepsilon\big)^{d+1} ,$$

$\lambda_1 = \frac{\mathrm{e}^\alpha - 1}{\alpha}$ *and* $\lambda_0 = \lambda_1 + \mathrm{e}^\alpha$.

In our context, the pair (A, B) is not uniformly distributed but it is of the form $(A, B) = (g(X), h(X))$ where X is uniform, and g and h are polynomial functions. We will then use the following corollary of Theorem 3.

Corollary 2. *Let X be a random variable uniformly distributed over some set $\mathcal{X}$ and let g and h be deterministic functions from $\mathcal{X}$ to $\mathcal{X}$. Let d be a positive integer and let $(G_i)_i$ and $(H_j)_j$ be d^{th}-order encodings of $g(X)$ and $h(X)$ respectively. Let ε be a real number such that $\varepsilon \leq \frac{\alpha}{(d+1)N^2}$ for some $\alpha \in (0, 1]$. And let $(f_{i,j})_{i,j}$ be noisy leakage functions defined over $\mathcal{X} \times \mathcal{X}$ and belonging to $\mathcal{N}(\varepsilon)$. We have:*

$$\beta\big(X \mid (f_{i,j}(G_i, H_j))_{i,j}\big) \leq 2N^{\frac{3d+7}{2}} \big((\lambda_1 d + \lambda_0)\varepsilon\big)^{d+1} ,$$

$\lambda_1 = \frac{\mathrm{e}^\alpha - 1}{\alpha}$ *and* $\lambda_0 = \lambda_1 + \mathrm{e}^\alpha$.

The bound in Corollary 2 shows that the bias of X given the leakages on all the pairs of shares (A_i, B_j) decreases exponentially with d. A requirement is that the bias given a single leakage, namely ε, is $(d + 1)$ times lower than $\frac{1}{N^2}$ or less, namely $\varepsilon \leq \frac{\alpha}{(d+1)N^2}$ for some $\alpha \in (0, 1]$. Then the bias of X given the $(d + 1)^2$ leakages is smaller that $(\lambda(d)\ \varepsilon)^{d+1}$ where λ is an affine function. Once again, the value α provides a trade-off between the constraint on ε and the coefficients of λ.

In the next section, we will use the theorems and corollaries introduced above to deduce a security bound for a full masked implementation of block cipher.

6 Overall Security Proof

In this section, we formally prove the security of the physical implementation $\mathcal{I} = (\mathsf{C}_i, f_i)_i$ of a block cipher following the scheme described in Section 4 with masking order d. Our security proof is *information theoretic*: we show that the information about the cipher input (message and secret key) provided by the overall leakage within an execution of $\mathcal{I}$ is upper bounded by a *negligible function* of the masking order d.

The cipher is assumed to involve t_{lin} linear transformations, t_{nlm} nonlinear multiplications and t_{aff} affine functions (in the s-boxes evaluations). The plaintext and the secret key in input of the cipher are modeled as uniform random variables M and K respectively. The input of every elementary calculation C_i is modeled as a random variable X_i and C_i leaks a noisy function f_i of X_i. The designer is then allowed to choose a noise parameter ψ_i linear in the security parameter d, such that the leakage function f_i lies in the class of noisy functions $\mathcal{N}(1/\psi_i)$.

Theorem 4. *Let d be a positive integer and let $\mathcal{I} = (\mathsf{C}_i, f_i)_{1\leq i\leq q}$ be the physical implementation of a block cipher under the scheme described in Section 4 with masking order d. For any positive real number ω, there exists a family of real numbers $\psi_i = O(d)\omega$ such that if f_i lies in $\mathcal{N}(1/\psi_i)$ for every i, then the mutual information between the cipher input (M, K) and the overall leakage $(f_1(X_1), f_2(X_2), \ldots, f_q(X_q))$ satisfies*

$$\mathrm{I}\big((M,K);(f_1(X_1), f_2(X_2), \ldots, f_q(X_q))\big) \ \leq \ \frac{T}{\omega^{d+1}} \ , \tag{2}$$

where $T = 2t_{\mathit{nlm}} + t_{\mathit{aff}} + t_{\mathit{lin}}$.

The rest of this section provides a proof of Theorem 4 and exhibits the noise parameters ψ_i that must be chosen for every elementary calculation C_i.

From the description of the scheme in Section 4.2, the overall sequence of elementary calculations of the protected block cipher algorithm can be split into several subsequences separated by masks-refreshing calculations. These subsequences are of four types:

1. $\big(z_i \leftarrow g(x_i)\big)_i$ where g is a linear function of the block cipher,
2. $\big(z_i \leftarrow g(x_i)\big)_i$ where g is an affine function of an s-box evaluation,
3. $\big(v_{i,j} \leftarrow a_i \times b_j\big)_{i,j}$ (first step of a secure nonlinear multiplication),
4. $\big(t_{i,j} \leftarrow t_{i,j-1} \oplus v_{i,j}\big)_{i,j}$ (fourth step of a secure nonlinear multiplication).

All the remaining elementary calculations are masks-refreshing calculations which are leak-free by assumption.

Let us denote by $\mathcal{I}_1$, $\mathcal{I}_1$, ..., $\mathcal{I}_T$ the successive subsequences of elementary calculations and by L_1, L_2, ..., L_T their respective leakages. For every $t \in \{1, 2, \ldots, T\}$, the leakage L_t satisfies

$$L_t = \begin{cases} \big(f_i(X_i)\big)_i & \text{if } \mathcal{I}_t \text{ is of type 1 or 2,} \\ \big(f_{i,j}(A_i, B_j)\big)_{i,j} & \text{if } \mathcal{I}_t \text{ is of type 3,} \\ \big(f_{i,j}(T_{i,j-1}, V_{i,j})\big)_{i,j} & \text{if } \mathcal{I}_t \text{ is of type 4.} \end{cases}$$

where the f_i or $f_{i,j}$ are the leakage functions associated to the elementary calculations in $\mathcal{I}_t$ and where the $(X_i)_i$, $(A_i)_i$, $(B_j)_j$, $(V_{i,j})_{i,j}$, or $(T_{i,j})_{i,j}$ are random variables modeling the elementary calculations inputs in $\mathcal{I}_t$ (note that the indexing is intra-subsequence and it differs from that in Theorem 4).

A crucial point of our security proof is that every subsequence operates on shares with fresh random masks. As a result, given a cipher input $(M, K) = (m, k)$, the encodings processed in the different subsequences are mutually independent, which in turn implies that the leakages of the different subsequences are mutually independent (since by definition, the randomness introduced by a noisy leakage function is independent of the randomness introduced by the others). We deduce that the entropy of the overall leakage $(L_1, L_2, \ldots, L_T)$ knowing the cipher input (M, K) satisfies:

$$\mathrm{H}[(L_1, L_2, \ldots, L_T)|(M, K)] = \sum_{t=1}^{T} \mathrm{H}[L_t|(M, K)] \ .$$

The mutual information between the cipher input and the overall leakage therefore satisfies:

$$\begin{aligned}\mathrm{I}\big((M, K); (L_1, L_2, \ldots, L_T)\big) &= \mathrm{H}[(L_1, L_2, \ldots, L_T)] - \sum_{t=1}^{T} \mathrm{H}[L_t|(M, K)] \\ &\leq \sum_{i=1}^{T} \mathrm{I}((M, K); L_t) \ ,\end{aligned}$$

where the inequality holds since $\mathrm{H}[(L_t)_t] \leq \sum_t \mathrm{H}[L_t]$.

For every subsequence $\mathcal{I}_t$, there exists a uniform random variable[3] $X_t = g(M, K)$ such that $\mathrm{I}((M, K); L_t) = \mathrm{I}(X_t; L_t)$. To complete the proof of Theorem 4, we now demonstrate that the mutual information $\mathrm{I}(X_t; L_t)$ is upper bounded by $(1/\omega)^{d+1}$ for some noise parameter $\psi_t = O(d)\omega$, for every $t \in \{1, 2, \ldots, T\}$. Due to Proposition 1, this is equivalent to prove that the bias of X_t given L_t satisfies

$$\beta(X_t|L_t) \leq \frac{\ln 2}{N} \left(\frac{1}{\omega}\right)^{d+1} \ , \tag{3}$$

where N is the cardinal of the definition set of X_t. In the rest of the section, we demonstrate the claim for every type of subsequence.

Security of Type 1 Subsequences. In a type 1 subsequence every elementary calculation processes a share of the uniform input X_t of a linear function. The security of such a subsequence directly holds from Theorem 1. Indeed, according to Theorem 1 with $\varepsilon = \psi_t^{-1}$, choosing a noise parameter $\psi_t = c\omega$ with a constant c satisfying $c^{d+1} \geq (\ln 2)^{-1} N^{\frac{d+2}{2}}$ implies bound (3). In particular c can be taken equal to $(\ln 2)^{-\frac{1}{2}} N^{\frac{3}{4}} \approx 1.44 N^{\frac{3}{4}}$ for any $d \geq 1$ and equal to a value close to $1.44\sqrt{N}$ for a large d.

[3] For a subsequence of type 1, this variable is simply the (unmasked) input of the corresponding linear transformation (which is indeed uniform since the cipher input is uniform by assumption, and according to the uniformity property stated in Section 4.1). For a subsequence of type 2, 3 or 4, this variable is the (unmasked) input of the corresponding s-box (which is also uniformly distributed for the same reasons).

Security of Type 2 Subsequences. In a type 2 subsequence every elementary calculation processes a share G_i of an encoding of $g(X_t)$ where X_t is a uniform s-box input and g is some polynomial function.

According to Corollary 1, choosing a noise parameter $\psi_t = c\omega$ with a constant c satisfying $c^{d+1} \geq (\ln 2)^{-1} N^{\frac{d+5}{2}}$ implies bound (3). In particular c can be taken equal to $(\ln 2)^{-\frac{1}{2}} N^{\frac{3}{2}} \approx 1.44 N^{\frac{3}{2}}$ for any $d \geq 1$ and equal to a value close to $1.44\sqrt{N}$ for a large d.

Security of Type 3 Subsequences. In a type 3 subsequence every elementary calculation processes a multiplication from a pair of shares (A_i, B_j), where $(A_i)_i$ and $(B_j)_j$ are d^{th}-order encodings of two variables A and B to multiply. Both A and B rely on a uniform s-box input X_t by $A = g(X_t)$ and $B = h(X_t)$ for some polynomial functions g and h.

According to Corollary 2, choosing a noise parameter $\psi_t = \lambda(d)\,\omega$ for a subsequence $\mathcal{I}_t$ of type 3 implies bound (3), as long as there exists $\alpha \in (0;1]$ such that $\lambda(d)$ satisfies

$$\lambda(d) \geq \frac{N^2}{\alpha\,\omega}(d+1) \quad \text{and} \quad \lambda(d) \geq c(\lambda_{1,\alpha}\, d + \lambda_{0,\alpha}) \ ,$$

where $\lambda_{1,\alpha} = \frac{e^{\alpha}-1}{\alpha}$, $\lambda_{0,\alpha} = \lambda_{1,\alpha} + e^{\alpha}$ and c is a constant satisfying $c^{d+1} \geq 2(\ln 2)^{-1} N^{\frac{3d+9}{2}}$. In particular c can be taken to $(\ln 2)^{-\frac{1}{2}} N^3 \approx 1.44 N^3$ for any $d \geq 1$ and equal to a value close to $1.44 N^{\frac{3}{2}}$ for a large d.

The first constraint aims to meet the requirement on ε for Corollary 2 and the second constraint implies bound (3). To summarize, the best choice is to take λ as

$$\lambda(d) = \min_{\alpha \in (0;1]} \max\left(\frac{N^2}{\alpha\,\omega}(d+1)\ ,\ c(\lambda_{1,\alpha}\, d + \lambda_{0,\alpha})\right) \ .$$

Security of Type 4 Subsequences. In a type 4 subsequence every elementary calculation processes an addition $T_{i,j} = T_{i,j-1} \oplus V_{i,j}$ for $0 \leq i \leq d$ and $1 \leq j \leq d$, and where $T_{i,0} = V_{i,0}$. For every i, the sequence of additions results in a share $Z_i = T_{i,d}$ of the underlying multiplication output. That is $(Z_0, Z_1, \ldots, Z_d)$ is an encoding of $g(X_t)$ where X_t is a uniform s-box input and g is some polynomial function over $\mathcal{X}$. Each elementary calculation takes as input a pair $(T_{i,j-1}, V_{i,j})$ and leaks $f_{i,j}(T_{i,j-1}, V_{i,j})$ where $f_{i,j} \in \mathcal{N}(1/\psi_t)$. Our goal is to set ψ_t such that the bias $\beta\big(X_t | (F_0(Z_0), F_1(Z_1), \ldots, F_d(Z_d))\big)$ satisfies bound (3), where

$$F_i(Z_i) = \big(f_{i,1}(T_{i,0}, V_{i,1}), f_{i,2}(T_{i,1}, V_{i,2}) \ldots, f_{i,d}(T_{i,d-1}, V_{i,d})\big) \ .$$

Note that F_i can be seen as a noisy leakage function applied to Z_i (and where $V_{i,1}$, $V_{i,2}$, $\ldots$, $V_{i,d}$ are seen as the internal randomness of F_i).

We first analyse the bias on each Z_i given the leakage $F_i(Z_i)$. Let $f'_{i,j}$ be the noisy leakage functions defined by $f'_{i,j} : (X, Y) \mapsto f_{i,j}(X, X \oplus Y)$. We can then rewrite $F_i(Z_i)$ as

$$F_i(Z_i) = \big(f'_{i,1}(T_{i,0}, T_{i,1}), f'_{i,2}(T_{i,1}, T_{i,2}) \ldots, f'_{i,d}(T_{i,d-1}, T_{i,d})\big) \ ,$$

and we have $f'_{i,j} \in \mathcal{N}(1/\psi_t)$ for every (i,j) by definition of the bias.[4] Moreover $T_{i,0}$, $T_{i,1}$, ..., $T_{i,d}$ are uniformly distributed and mutually independent, and $T_{i,d} = Z_i$. We then apply the following lemma.

Lemma 1. *Let T_0, T_1, ..., T_d be $d+1$ independent random variables uniformly distributed over some set $\mathcal{X}$ of cardinality N. Let $\varepsilon \in [0,1)$ and let f_1, f_2, ..., f_d be noisy leakage functions defined over $\mathcal{X} \times \mathcal{X}$ and belonging to $\mathcal{N}(\varepsilon)$. We have:*

$$\beta\big(T_d \big| f_1(T_0,T_1), f_2(T_1,T_2), \ldots, f_d(T_{d-1},T_d)\big) \le 2N\varepsilon \ .$$

Lemma 1 implies $\beta(Z_i|F_i(Z_i)) \le 2N/\psi_t$ for every i. Then by Corollary 1, we get

$$\beta\big(X_t|(F_0(Z_0), F_1(Z_1), \ldots, F_d(Z_d))\big) \le N^{\frac{d+3}{2}}\left(\frac{2N}{\psi_t}\right)^{d+1} = N^{\frac{3d+5}{2}}\left(\frac{2}{\psi_t}\right)^{d+1} .$$

According to the above inequality, choosing a noise parameter $\psi_t = c\,\omega$ with a constant c satisfying $c^{d+1} \ge 2^{d+1}(\ln 2)^{-1}N^{\frac{3d+7}{2}}$ implies bound (3). In particular c can be taken equal to $2(\ln 2)^{-1}N^{\frac{5}{2}} \approx 2.89N^{\frac{5}{2}}$ for any $d \le 1$ and equal to a value close to $2.89N^{\frac{3}{2}}$ for a large d.

References

1. Blakely, G.: Safeguarding cryptographic keys. In: National Comp. Conf., vol. 48, pp. 313–317. AFIPS Press, New York(1979)
2. Blömer, J., Guajardo, J., Krummel, V.: Provably Secure Masking of AES. In: Handschuh, H., Hasan, M.A. (eds.) SAC 2004. LNCS, vol. 3357, pp. 69–83. Springer, Heidelberg (2004)
3. Carlet, C., Goubin, L., Prouff, E., Quisquater, M., Rivain, M.: Higher-order masking schemes for s-boxes. In: Canteaut, A. (ed.) FSE 2012. LNCS, vol. 7549, pp. 366–384. Springer, Heidelberg (2012)
4. Chari, S., Jutla, C., Rao, J., Rohatgi, P.: Towards Sound Approaches to Counteract Power-Analysis Attacks. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 398–412. Springer, Heidelberg (1999)
5. Chari, S., Rao, J., Rohatgi, P.: Template Attacks. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 13–28. Springer, Heidelberg (2003)
6. Clavier, C., Coron, J.-S., Dabbous, N.: Differential Power Analysis in the Presence of Hardware Countermeasures. In: Koç, Ç.K., Paar, C. (eds.) CHES 2000. LNCS, vol. 1965, pp. 252–263. Springer, Heidelberg (2000)
7. Coron, J.-S., Kizhvatov, I.: Analysis and Improvement of the Random Delay Countermeasure of CHES 2009. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 95–109. Springer, Heidelberg (2010)
8. Coron, J.-S., Kocher, P., Naccache, D.: Statistics and secret leakage. In: Frankel, Y. (ed.) FC 2000. LNCS, vol. 1962, pp. 157–173. Springer, Heidelberg (2001)
9. Di Crescenzo, G., Lipton, R.J., Walfish, S.: Perfectly Secure Password Protocols in the Bounded Retrieval Model. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 225–244. Springer, Heidelberg (2006)

[4] For every noisy leakage function f defined over $\mathcal{X} \times \mathcal{X}$, and for $f' : (X,Y) \mapsto f(X, X \oplus Y)$, we have $\beta\big((X,Y)\big|f(X,Y)\big) = \beta\big((X,Y')\big|f'(X,Y')\big)$ where $Y' = X \oplus Y$.

10. Dodis, Y., Pietrzak, K.: Leakage-Resilient Pseudorandom Functions and Side-Channel Attacks on Feistel Networks. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 21–40. Springer, Heidelberg (2010)
11. Dziembowski, S., Faust, S.: Leakage-Resilient Circuits without Computational Assumptions. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 230–247. Springer, Heidelberg (2012)
12. Dziembowski, S., Pietrzak, K.: Leakage-resilient cryptography. In: FOCS, pp. 293–302. IEEE Computer Society (2008)
13. Faust, S., Kiltz, E., Pietrzak, K., Rothblum, G.N.: Leakage-Resilient Signatures. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 343–360. Springer, Heidelberg (2010)
14. Faust, S., Rabin, T., Reyzin, L., Tromer, E., Vaikuntanathan, V.: Protecting Circuits from Leakage: the Computationally-Bounded and Noisy Cases. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 135–156. Springer, Heidelberg (2010)
15. Gandolfi, K., Mourtel, C., Olivier, F.: Electromagnetic Analysis: Concrete Results. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 251–261. Springer, Heidelberg (2001)
16. Genelle, L., Prouff, E., Quisquater, M.: Thwarting higher-order side channel analysis with additive and multiplicative maskings. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 240–255. Springer, Heidelberg (2011)
17. Goldwasser, S., Rothblum, G.N.: Securing computation against continuous leakage. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 59–79. Springer, Heidelberg (2010)
18. Goldwasser, S., Rothblum, G.N.: How to Compute in the Presence of Leakage. In: 53rd Annual IEEE Symposium on Foundations of Computer Science – FOCS 2012, pp. 31–40. IEEE Computer Society (2012)
19. Goubin, L., Patarin, J.: DES and Differential Power Analysis – The Duplication Method. In: Koç, Ç.K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, pp. 158–172. Springer, Heidelberg (1999)
20. Herbst, C., Oswald, E., Mangard, S.: An AES Smart Card Implementation Resistant to Power Analysis Attacks. In: Zhou, J., Yung, M., Bao, F. (eds.) ACNS 2006. LNCS, vol. 3989, pp. 239–252. Springer, Heidelberg (2006)
21. Ishai, Y., Sahai, A., Wagner, D.: Private Circuits: Securing Hardware against Probing Attacks. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 463–481. Springer, Heidelberg (2003)
22. Juma, A., Vahlis, Y.: Protecting Cryptographic Keys against Continual Leakage. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 41–58. Springer, Heidelberg (2010)
23. Kiltz, E., Pietrzak, K.: Leakage Resilient ElGamal Encryption. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 595–612. Springer, Heidelberg (2010)
24. Kocher, P.: Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996)
25. Kocher, P., Jaffe, J., Jun, B.: Differential Power Analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
26. Macé, F., Standaert, F.-X., Quisquater, J.-J.: Information Theoretic Evaluation of Side-Channel Resistant Logic Styles. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 427–442. Springer, Heidelberg (2007)
27. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks – Revealing the Secrets of Smartcards. Springer (2007)

28. Messerges, T.: Securing the AES Finalists against Power Analysis Attacks. In: Schneier, B. (ed.) FSE 2000. LNCS, vol. 1978, pp. 150–164. Springer, Heidelberg (2001)
29. Micali, S., Reyzin, L.: Physically Observable Cryptography (Extended Abstract). In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 278–296. Springer, Heidelberg (2004)
30. Oswald, E., Mangard, S., Pramstaller, N., Rijmen, V.: A Side-Channel Analysis Resistant Description of the AES S-box. In: Gilbert, H., Handschuh, H. (eds.) FSE 2005. LNCS, vol. 3557, pp. 413–423. Springer, Heidelberg (2005)
31. Peeters, E., Standaert, F.-X., Quisquater, J.-J.: Power and Electromagnetic Analysis: Improved Model, Consequences and Comparisons. Integration 40(1), 52–60 (2007)
32. Pietrzak, K.: A Leakage-Resilient Mode of Operation. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 462–482. Springer, Heidelberg (2009)
33. Prouff, E., Roche, T.: Higher-order glitches free implementation of the aes using secure multi-party computation protocols. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 63–78. Springer, Heidelberg (2011)
34. Rivain, M., Prouff, E.: Provably Secure Higher-Order Masking of AES. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 413–427. Springer, Heidelberg (2010)
35. Rothblum, G.N.: How to compute under$\mathcal{AC}^0$leakage without secure hardware. In: Safavi-Naini, R. (ed.) CRYPTO 2012. LNCS, vol. 7417, pp. 552–569. Springer, Heidelberg (2012)
36. Schindler, W., Lemke, K., Paar, C.: A Stochastic Model for Differential Side Channel Cryptanalysis. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 30–46. Springer, Heidelberg (2005)
37. Shamir, A.: How to Share a Secret. Commun. ACM 22(11), 612–613 (1979)
38. Standaert, F.-X., Archambeau, C.: Using Subspace-Based Template Attacks to Compare and Combine Power and Electromagnetic Information Leakages. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 411–425. Springer, Heidelberg (2008)
39. Standaert, F.-X., Örs, S.B., Preneel, B.: Power Analysis of an FPGA: Implementation of Rijndael: Is Pipelining a DPA Countermeasure? In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 30–44. Springer, Heidelberg (2004)
40. Standaert, F.-X., Pereira, O., Yu, Y., Quisquater, J.-J., Yung, M., Oswald, E.: Leakage resilient cryptography in practice. Cryptology ePrint Archive, Report 2009/341 (2009), `http://eprint.iacr.org/`
41. Tiri, K., Verbauwhede, I.: A Logic Level Design Methodology for a Secure DPA Resistant ASIC or FPGA Implementation. In: Design, Automation and Test in Europe Conference and Exposition – DATE 2004, pp. 246–251. IEEE Computer Society (2004)
42. Yu, Y., Standaert, F.-X., Pereira, O., Yung, M.: Practical leakage-resilient pseudorandom generators. In: Al-Shaer, E., Keromytis, A.D., Shmatikov, V. (eds.) ACM Conference on Computer and Communications Security – CCS 2010, pp. 141–151 (2010)

Leakage-Resilient Cryptography from Minimal Assumptions

Carmit Hazay[1,⋆], Adriana López-Alt[2,⋆⋆], Hoeteck Wee[3,⋆⋆⋆], and Daniel Wichs[4,†]

[1] Bar-Ilan University
[2] New York University
[3] George Washington University
[4] Northeastern University

Abstract. We present new constructions of leakage-resilient cryptosystems, which remain provably secure even if the attacker learns some arbitrary partial information about their internal secret key. For any polynomial ℓ, we can instantiate these schemes so as to tolerate up to ℓ bits of leakage. While there has been much prior work constructing such leakage-resilient cryptosystems under concrete number-theoretic and algebraic assumptions, we present the first schemes under general and minimal assumptions. In particular, we construct:

– Leakage-resilient *public-key encryption* from any standard public-key encryption.
– Leakage-resilient *weak pseudorandom functions*, *symmetric-key encryption*, and *message-authentication codes* from any one-way function.

These are the first constructions of leakage-resilient *symmetric-key* primitives that do not rely on *public-key* assumptions. We also get the first constructions of leakage-resilient public-key encryption from "search assumptions", such as the hardness of factoring or CDH. Although our schemes can tolerate arbitrarily large *amounts* of leakage, the tolerated *rate* of leakage (defined as the ratio of leakage-amount to key-size) is rather poor in comparison to prior results under specific assumptions.

As a building block of independent interest, we study a notion of *weak* hash-proof systems in the public-key and symmetric-key settings. While these inherit some of the interesting security properties of standard hash-proof systems, we can instantiate them under general assumptions.

1 Introduction

A central goal in cryptography is to base cryptosystems on intractability assumptions that are as weak and as general as possible; that way, if one

⋆ Research partially conducted while at Aarhus University.
⋆⋆ Supported by AT&T Labs Fellowship. Research partially conducted while at Aarhus University.
⋆⋆⋆ Supported by NSF CAREER Award CNS-1237429.
† Research conducted while at IBM Research, T.J. Watson.

problem turns out to be susceptible to a new attack or if another turns out to yield better performance, we may readily replace the underlying problem in our cryptosystem. Another goal is to design cryptosystems in strong security models that account for a wide range of possible attacks. Our work lies at the intersection of these two areas, by studying leakage-resilient security under general and minimal assumptions.

Leakage Resilience. Leakage-resilient cryptosystems maintain their security even if an attacker can learn some partial information about the internal secret key. Aside from being a basic question of theoretical interest, the study of leakage-resilience is motivated by several real-world scenarios where information leaks. One such scenario involves side-channel attacks, where the physical attributes of a computing device (e.g., its power consumption, electromagnetic radiation, timing, temperature, acoustics, etc.) can reveal information about its internal secret state. See e.g., [1, 5, 24, 38, 39, 46–48] for many examples of such attacks that completely break otherwise secure cryptosystems. Another source of leakage occurs through imperfect erasures (such as in the *cold-boot* attack [27]), where memory contents, including secret key information, aren't properly erased and some partial information becomes available to an attacker. Another source of leakage occurs if the secret key is stored on a compromised system to which the attacker has remote access. As suggested in prior work, we can impede an attacker from retrieving the secret key in its entirety by making it deliberately huge (e.g., many gigabytes in length), but the attacker can still obtain some partial leakage [4, 12, 16, 21]. As yet another example, we may need to use a cryptosystem within the context of a larger protocol that intentionally leaks some information about the secret key as a part of its design. Leakage-resilience provides a powerful tool, allowing us to easily analyze the security of such constructions. In summary, we believe that leakage-resilience is an interesting and fundamental property worth studying because of its relevance to many diverse problems including (but not limited to) side-channel attacks.

Bounded-Leakage Model. There are several security models of leakage-resilience in the literature, differing in their specification of what information can become available to the attacker. In this work we will focus on a simple yet general model, called the *bounded-leakage* (or sometimes *memory leakage*) model, which has received much attention in recent years [2–4,6–9,11–13,18,22,25,29,35,37,43]. In this model, the attacker can learn arbitrary information about the secret key, as long as the total number of bits learned is bounded by some parameter ℓ, called the *leakage bound*. We formalize this security notion by giving the attacker access to a *leakage oracle* that she can repeatedly and adaptively query; each query to the oracle consists of a *leakage function* f and the oracle responds with the "leakage" $f(\text{SK})$ computed on secret key SK. The leakage oracle is only restricted in the total number of bits that it outputs throughout its lifetime, which is bounded by ℓ. This model is particularly interesting because of its elegance and simplicity and its wide applicability to scenarios such as incomplete erasure, compromised systems, and information released by high-level protocols.

We note that several other models of leakage-resilience consider a more complex scenario, where information can leak continually over time, with no overall bound on the total amount of leakage. See [10,17,19,23,26,34,41,42,45] for some examples. These models may offer a more realistic view of side-channel attacks, where many measurements may be made by an attacker over time. Many of these works rely on results from the bounded-leakage model as basic building blocks. Therefore, we believe that a thorough understanding of the bounded-leakage model is a necessary, but perhaps not sufficient, prerequisite to understanding other more complex models. We mention that it remains debatable how accurately any of the above models reflects realistic side-channel attacks (see e.g., the discussion in [49]).

Prior Constructions. It turns out that essentially all cryptographic schemes are already resilient against small amounts of leakage. In particular, they can tolerate $\ell = O(\log(\lambda))$ bits of leakage, where λ is the security parameter, and schemes with stronger *exact security* can tolerate correspondingly larger amounts of leakage. Intuitively, this follows since we can correctly "guess" small leakage values with reasonable probability and hence they cannot help in an attack.[1]

Most prior research in leakage-resilient cryptography attempts to construct schemes that provably tolerate larger amounts of leakage, without making any strong exact-security assumptions on the underlying primitives. Ultimately, we aim to tolerate any polynomial leakage bound $\ell(\lambda)$ just by instantiating the scheme with a sufficiently large secret key. Prior to this work, we had such results for public-key encryption [2, 8, 43], under specific assumptions such as LWE, DDH, DCR, QR, or somewhat more generally, the existence of "hash-proof systems". We also had such results for signatures [4,18,37] assuming the existence of NIZKs and public-key encryption. Essentially nothing better was known for symmetric-key encryption or message-authentication codes, beyond simply using the corresponding public-key constructions in the symmetric setting.

Our Main Results. We present new constructions of several leakage-resilient cryptosystems under the *minimal assumption* that such cryptosystems exist in the standard setting, without any leakage. For any polynomial leakage-bound $\ell(\lambda)$ in the security parameter λ, we can instantiate these schemes so as to resit $\ell(\lambda)$ bits of leakage. In particular, we construct the following primitives:

- Leakage-resilient *public-key encryption* from any public-key encryption.
- Leakage-resilient *weak pseudorandom functions*, *symmetric-key encryption*, and *message-authentication codes* from any one-way function.

We only assume the underlying primitives satisfy the usual asymptotic notion of security, and do *not* require any stronger levels of exact security. These results give us the first constructions of leakage-resilient symmetric-key primitives

[1] This simple argument works for "unpredictability" applications such as signatures. A more subtle argument also works for many "indistinguishability" applications, including public-key encryption, weak-PRFs and symmetric-key CPA encryption (but not, e.g., one-time encryption). See [20] for a general treatment of this question.

that do not rely on public-key assumptions. They also give us the first constructions of leakage-resilient public-key encryption from several specific "search assumptions" such as the hardness of RSA, factoring, or CDH.

Leakage Amount vs. Rate. Although our schemes can tolerate an arbitrarily large polynomial *amount* of leakage ℓ, the tolerated *rate* of leakage (defined as the ratio of ℓ to the secret-key size) in these constructions is rather poor. In particular, the leakage rate in our schemes is $O(\log(\lambda)/s(\lambda))$ where $s(\lambda)$ is the secret-key size of the underlying non-leakage-resilient primitives. In contrast, the state-of-the-art constructions of leakage-resilient schemes from concrete number-theoretic assumptions such as DDH can usually achieve a $(1-o(1))$ leakage rate, meaning that almost the entire secret key can leak. Allowing higher leakage rates under general assumptions remains as an open problem.

Extensions of Our Results. We explore several extensions of our main results. Firstly, we show that all of the results also apply to an alternate notion of *entropy-bounded leakage* [17, 43], where we restrict the amount of entropy-loss caused by the leakage rather than restricting its length. Unlike length-bounded leakage, achieving resilience to even 1-bit of entropy-bounded leakage is non-trivial and does not follow from the standard security of a scheme. We also show that our public/symmetric key encryption schemes provide resilience to *"after-the-fact"* leakage as defined by Halevi and Lin [29]. In particular, if the attacker can choose to learn some arbitrary ℓ_{post} bits of leakage on the secret key adaptively *after* seing a challenge ciphertext, she learns no more than ℓ_{post} bits of information about the encrypted message (in contrast, if the leakage is independent of the challenge ciphertext, she learns nothing about the message). Lastly, we extend our results to the *bounded-retrieval model* [4, 12, 16, 21], where we want to have efficient schemes tolerating huge amounts (many gigabytes) of leakage, meaning that the efficiency of the scheme should not degrade even as the leakage-bound ℓ increases. Since the secret-key size of such schemes must exceed ℓ and therefore also be huge, these schemes cannot even read their entire secret key during each cryptographic operation. This model is motivated by the problem of system compromise, where an attacker can download large amounts of data from a compromised system. Due to space limitations, we defer the extensions of our results to entropy-bounded leakage, after-the-fact leakage, and the bounded-retrieval model to the full version of this paper [31].

1.1 Overview of Our Techniques

Our starting point is a result of Naor and Segev [43] (journal version [44]), which constructs leakage-resilient public-key encryption from any *hash-proof system* (HPS) [15]. As observed in [4, 43], this construction does not require the full security notion of HPS and it turns out that a weaker variant, which we will call a *weak* HPS (wHPS), actually suffices.[2] As our first result we show

[2] Although this weaker variant was used by [4, 43] to simplify the exposition of HPS, neither work seemed to considered the differences between wHPS and full HPS as very significant.

that, surprisingly, wHPS can be constructed generically from any public-key encryption scheme. This is in contrast to the full notion of HPS, which we only know how to construct from concrete number-theoretic assumptions such as DDH, DCR or QR. This gives us our results for *public-key encryption*. Next, we also define a new and meaningful notion of a *symmetric-key* wHPS, which allows us to construct leakage-resilient *weak pseudo-random functions* and *symmetric-key encryption*. We show how to construct symmetric-key wHPS generically from any pseudorandom function (PRF), and hence only under the assumption that one-way functions exist. Lastly, we employ several additional ideas to construct leakage-resilient *message authentication codes*.

We briefly define wHPS, how it relates to leakage resilience, and how to construct it. We focus on the public-key setting since it is conceptually simpler.

Weak Hash-Proof Systems (wHPS). A weak hash-proof system (wHPS) can be thought of as a special type of *key-encapsulation mechanism*. It consists of:

- A public-key *encapsulation* algorithm $(c, k) \leftarrow \mathsf{Encap}(\textsc{pk})$ that creates a ciphertext c encapsulating a random secret value k.
- A secret-key *decapsulation* algorithm $k = \mathsf{Decap}(\textsc{sk}, c)$ that recovers k from the ciphertext c.

Within the security definition of wHPS, we also require an additional *invalid encapsulation* algorithm $c^* \leftarrow \mathsf{Encap}^*(\textsc{pk})$, which is *not* used by honest parties. The scheme must satisfy the following:

- Ciphertext Indistinguishability: Valid ciphertexts $(c, \cdot) \leftarrow \mathsf{Encap}(\textsc{pk})$ are computationally indistinguishable from invalid ciphertexts $c^* \leftarrow \mathsf{Encap}^*(\textsc{pk})$, even given the secret key $\textsc{sk}$.
- Smoothness: Let $(\textsc{pk}, \textsc{sk})$ be a random wHPS key pair and $c^* \leftarrow \mathsf{Encap}^*(\textsc{pk})$ be a random *invalid* ciphertext. Given $\textsc{pk}$ and c^*, the output $k = \mathsf{Decap}(\textsc{sk}, c^*)$ is uniformly random and independent (information theoretically). The randomness of k comes from the choice of the secret key $\textsc{sk}$ consistent with $\textsc{pk}$, meaning that there must be multiple ones.

In other words, the secret key $\textsc{sk}$ maintains real entropy even conditioned on $\textsc{pk}$, and this entropy is transferred to the output $k = \mathsf{Decap}(\textsc{sk}, c^*)$ when we decapsulate a random invalid ciphertext c^*.

The above definition of wHPS departs from that of standard hash-proof systems in several ways, but most importantly, our "smoothness" property is defined for an *average-case* invalid ciphertext $c^* \leftarrow \mathsf{wHPS.Encap}^*(\textsc{pk})$ rather than a worst-case choice of c^* from some invalid set. Indeed, this makes our definition unsuitable for applications dealing with chosen-ciphertext (CCA or even CCA-1) security, for which hash-proof systems were originally intended.

Leakage-Resilience from wHPS. Weak hash-proof systems are particularly suited for leakage-resilience. Assume the attacker gets a wHPS public-key $\textsc{pk}$ and observes ℓ bits of leakage on the secret key $\textsc{sk}$. Later, the attacker sees a random valid ciphertext c computed via $(c, k) \leftarrow \mathsf{Encap}(\textsc{pk})$; what has she learned about the hidden value k? Firstly, we can switch c to an invalid ciphertext

$c^* \leftarrow \mathsf{Encap}^*(\text{PK})$ and define $k = \mathsf{Decap}(c^*, \text{SK})$. This change is indistinguishable even given the secret key SK in full, and therefore also when only given leakage on SK. Secondly, because $k = \mathsf{Decap}(c^*, \text{SK})$ is information-theoretically random even when given PK and c^*, the ℓ-bits of leakage that the attacker observes about SK can reduce the entropy of k by at most ℓ bits. Therefore, if k is sufficiently large, it still has high entropy given the view of the attacker, and we can easily convert it to a uniformly random value using a randomness extractor. The above argument closely follows that of [43].

Constructing wHPS. Our main result for public-key encryption is to construct wHPS from general assumptions. As a starting point, we give a very simple construction where the output $k \in \{0,1\}$ consists of a single bit. We do so given any standard public-key encryption (PKE) scheme, as follows:

- Choose two random PKE key-pairs $(\text{PK}_0, \text{SK}_0), (\text{PK}_1, \text{SK}_1)$ and define the wHPS public-key as $\text{PK} = (\text{PK}_0, \text{PK}_1)$ and the wHPS secret key as $\text{SK} = (b, \text{SK}_b)$ where $b \leftarrow \{0,1\}$ is a random bit. Notice that, given PK, there are at least two possible consistent secret keys: $(0, \text{SK}_0)$ and $(1, \text{SK}_1)$.
- The valid encapsulation algorithm $(c, k) \leftarrow \mathsf{Encap}(\text{PK})$ chooses a random bit $k \leftarrow \{0,1\}$ and sets $c = (c_0, c_1)$ where $c_0 \leftarrow \mathsf{PKE.Enc}(\text{PK}_0, k), c_1 \leftarrow \mathsf{PKE.Enc}(\text{PK}_1, k)$ both encrypt the *same* bit k.
- The invalid encapsulation algorithm $c^* \leftarrow \mathsf{Encap}^*(\text{PK})$ chooses a random bit $k \leftarrow \{0,1\}$ and sets $c^* = (c_0, c_1)$ where $c_0 \leftarrow \mathsf{PKE.Enc}(\text{PK}_0, k), c_1 \leftarrow \mathsf{PKE.Enc}(\text{PK}_1, 1-k)$ encrypt *opposite* bits.
- The decapsulation algorithm $\mathsf{Decap}(\text{SK}, c)$ takes $c = (c_0, c_1)$ and the secret key $\text{SK} = (b, \text{SK}_b)$, and outputs the decryption $\mathsf{PKE.Dec}(\text{SK}_b, c_b)$ of the ciphertext c_b using the key SK_b.

Input indistinguishability follows since, even given the secret key $\text{SK} = (b, \text{SK}_b)$, the attacker cannot distinguish if the ciphertext c_{1-b} encrypts the same bit k as contained in c_b or the opposite bit $1 - k$. The *smoothness* property follows since the decapsulation of a random invalid ciphertext $c^* = (c_0, c_1)$ is uniformly random over the choice of the secret-key bit b.

Amplifying wHPS. The above construction only gives us a wHPS with 1-bit output. However, we can easily amplify the output size of a wHPS to any arbitrary polynomial $n = n(\lambda)$, simply by taking n independent copies of the scheme in parallel. Notice that in the new scheme, there will be at least 2^n possible secret keys consistent with any public key, and the output of the wHPS on an invalid ciphertext will consist of n random and independent bits. Since the *amount* of tolerated leakage ℓ is roughly equal to the wHPS output-size n, we can set it to be arbitrarily high.

We note that the concept of amplifying leakage-resilience directly via parallel repetition has been suggested and explored in several works [3,4,9,36,40], with surprising counter-examples showing that it is not secure in general. In our special case, we only argue that parallel repetition amplifies the *output size* of a wHPS (which is trivial), and then use our connection between output size and leakage resilience to indirectly argue that the latter amplifies as well.

The above construction can tolerate roughly n bits of leakage by storing n decryption keys, meaning that the *rate* of leakage is roughly $1/s(\lambda)$, where $s(\lambda)$ is the size of the decryption key in the underlying PKE scheme. In our final construction, we show how to increase this to any $O(\log(\lambda)/s(\lambda))$ leakage rate. Getting an even higher rate remains as an open problem.

Symmetric-Key wHPS. In the second part of our work, we carry the above ideas over to the symmetric-key setting. To do so, we first define a notion of a symmetric-key wHPS analogously to our public-key wHPS. We can think of symmetric-key wHPS as a special type of pseudorandom function (PRF) $f_k(\cdot)$ with the following properties (simplified):

- INPUT INDISTINGUISHABILITY: There are two special distributions on the inputs x which we call *valid* and *invalid*, and which are indistinguishable from uniform even given the secret-key k.
- SMOOTHNESS: Given multiple inputs/outptus $\{(x, f_k(x))\}$ for various random *valid* x, and a random choice of an *invalid* input x^*, the output $f_k(x^*)$ is uniformly random and independent (information theoretically), where the randomness comes from the choice of a consistent key k.

In other words, the key k maintains real entropy even conditioned on seeing $f_k(x)$ for many random valid inputs x, but this entropy comes out when evaluating $f_k(x^*)$ at a random invalid input x^*.

We show how to use such symmetric-key wHPS schemes to construct leakage-resilient symmetric-key encryption and weak PRFs. We then construct symmetric-key wHPS generically from standard weak PRF, and therefore only assuming that one-way functions exist. Our construction of MACs departs somewhat from this abstraction and requires additional ideas.

2 Preliminaries

Notation. We let λ denote the security parameter. For an integer n, we let $[n]$ denote the set $\{1, \ldots, n\}$. For a randomized function f, we write $f(x; r)$ to denote the unique output of f on input x with random coins r. We write $f(x)$ to denote a random variable for the output of $f(x; r)$ over the random coins r. For a distribution or random variable X, we write $x \leftarrow X$ to denote the operation of sampling a random x according to X. For a set S, we write $s \leftarrow S$ to denote sampling s *uniformly at random* from S. For distributions X, Y, we write $X \equiv Y$ to mean that X, Y are identically distributed, $X \approx_s Y$ to mean that they are statistically close, and $X \approx_c Y$ to say that they are computationally indistinguishable. We let negl(λ) denote the set of all negligible function $\mu(\lambda) = \lambda^{-\omega(1)}$. We use calligraphic letters such as $\mathcal{X}$ to denote an *ensemble* of sets $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda \in \mathbb{N}}$. To simplify notation, we often exclude the subscript λ when clear from context, and write e.g. $x \leftarrow \mathcal{X}$ to denote $x \leftarrow \mathcal{X}_\lambda$. We say that an ensemble $\mathcal{X}$ is *efficient* if the operations of sampling a uniformly random $x \leftarrow \mathcal{X}_\lambda$ and testing $x \in \mathcal{X}_\lambda$ can be performed in $\mathsf{poly}(\lambda)$ time.

The Leakage Oracle. We model *leakage attacks* on a secret key $\textsc{sk}$ by giving the adversary access to a *leakage oracle*, which he can adaptively access to learn information about the secret key. The leakage oracle, denoted $\mathcal{O}^{\ell}_{\mathrm{SK}}(\cdot)$, is parameterized by a secret key SK and a leakage parameter ℓ. Each query to the leakage oracle consists of a function $f_i : \{0,1\}^{|\mathrm{SK}|} \to \{0,1\}^{\ell_i}$ (represented by a circuit), to which the oracle answers with $f_i(\mathrm{SK})$.[3] The oracle keeps track of the output sizes ℓ_i of all the leakage queries so far, and only responds to the qth leakage query if $\sum_{i=1}^{q} \ell_i \le \ell$.

3 Leakage-Resilient Public-Key Encryption

We begin with a definition of leakage-resilient public-key encryption (PKE). Our definition is equivalent to that used by prior works [2,43].

Definition 1 (Leakage-Resilient PKE). *An* $\ell(\lambda)$-leakage-resilient PKE *consists of the algorithms* $(\mathsf{LR.Gen}, \mathsf{LR.Enc}, \mathsf{LR.Dec})$ *and a message space* $\mathcal{M}$ *satisfying the following properties:*

Correctness: *For all* $(\mathrm{PK}, \mathrm{SK})$ *in the support of* $\mathsf{LR.Gen}(1^{\lambda})$ *and all messages* $\mathrm{M} \in \mathcal{M}$*,* $\mathsf{LR.Dec}(\mathrm{SK}, \mathsf{LR.Enc}(\mathrm{PK}, \mathrm{M})) = \mathrm{M}$.

Semantic Security with ℓ-Leakage: *For all* PPT *adversaries* $\mathcal{A}$*, the advantage of* $\mathcal{A}$ *in the following game is negligible in* λ*:*

Key Generation: *The challenger runs* $(\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{LR.Gen}(1^{\lambda})$ *and gives* PK *to* $\mathcal{A}$.

Leakage Queries: $\mathcal{A}$ *is given access to the leakage oracle* $\mathcal{O}^{\ell}_{\mathrm{SK}}(\cdot)$*. Without loss of generality, we can assume that* $\mathcal{A}$ *queries* $\mathcal{O}^{\ell}_{\mathrm{SK}}(\cdot)$ *only once with a function* f *whose output is* ℓ *bits.*

Challenge: $\mathcal{A}$ *chooses two plaintexts* $\mathrm{M}_0, \mathrm{M}_1 \in \mathcal{M}$ *and gives these to the challenger. The challenger chooses a random bit* $b \leftarrow \{0,1\}$*, and sends* $c^* \leftarrow \mathsf{LR.Enc}(\mathrm{PK}, \mathrm{M}_b)$ *to* $\mathcal{A}$*. The attacker* $\mathcal{A}$ *outputs a bit* b'.

We define the advantage of $\mathcal{A}$ *as* $Adv_{\mathcal{A}}(\lambda) = \left|\Pr[b' = b] - \frac{1}{2}\right|$.

If an encryption scheme is *0-leakage-resilient* we refer to it as *semantically secure.*

3.1 Leakage-Resilience from Weak Hash-Proof Systems

We specify our notion of weak hash-proof systems (wHPS). Our definition essentially follows an informal description given in [43] and a formal definition of [3], who considered a similar notion in the "identity based" setting.

Definition 2. *A* weak hash-proof system (wHPS) *with output space* $\mathcal{K}$ *consists of four algorithms* $(\mathsf{Gen}, \mathsf{Encap}, \mathsf{Encap}^*, \mathsf{Decap})$ *with the following syntax:*

- $(\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{Gen}(1^{\lambda})$*: Given security parameter* λ*, creates a key pair.*
- $(c, k) \leftarrow \mathsf{Encap}(\mathrm{PK})$*: Given a public key* PK*, creates a "valid" ciphertext* c *encapsulating* $k \in \mathcal{K}$.
- $c^* \leftarrow \mathsf{Encap}^*(\mathrm{PK})$*: Given a public key* PK*, creates an "invalid" ciphertext* c^*.
- $k = \mathsf{Decap}(c, \mathrm{SK})$*: Given a ciphertext* c *and secret key* SK*, deterministically recovers* $k \in \mathcal{K}$.

[3] We insist on a circuit representation to ensure that a poly-time attacker can only query poly-sized circuits, meaning that the leakage is poly-time computable.

We require a weak hash-proof system to satisfy the following properties:

Correctness: *For all* (PK, SK) *in the range of* $\mathsf{Gen}(1^\lambda)$*, and for* $(c, k) \leftarrow \mathsf{Encap}(\text{PK})$*, we have* $k = \mathsf{Decap}(c, \text{SK})$*.*

Ciphertext Indistinguishability: *If we sample* $(\text{PK}, \text{SK}) \leftarrow \mathsf{Gen}(1^\lambda)$, $(c, k) \leftarrow \mathsf{Encap}(\text{PK})$, $c^* \leftarrow \mathsf{Encap}^*(\text{PK})$*, we have the computational indistinguishability:* $(\text{PK}, \text{SK}, c) \approx_c (\text{PK}, \text{SK}, c^*)$*. In other words, avalid ciphertext c created with* Encap *is indistinguishable from an invalid ciphertext* c^* *created with* Encap^**,* even given the secret key SK.

Smoothness: *If we sample* $(\text{PK}, \text{SK}) \leftarrow \mathsf{Gen}(1^\lambda)$, $c^* \leftarrow \mathsf{Encap}^*(\text{PK})$, $k \leftarrow \mathcal{K}$*, and set* $k^* = \mathsf{Decap}(c^*, \text{SK})$*, we have the distributional equivalence:* $(\text{PK}, c^*, k^*) \equiv (\text{PK}, c^*, k)$*. In other words, the decapsulated value* $k^* = \mathsf{Decap}(c^*, \text{SK})$ *is uniformly random over* $\mathcal{K}$ *and independent of* c^* *and* PK*. Since all of the randomness of* k^* *must therefore come from the choice of* SK*, this implicitly requires that there are many possible choices of* SK *for a fixed* PK*.*

Constructing LR-PKE from wHPS. In the full version [31], we show how to construct leakage-resilient PKE from wHPS by following the construction of Naor and Segev [43], while formalizing that the the weaker security of wHPS is sufficient. We apply an extractor to the output k of the wHPS, and then use the extracted randomness as a one-time-pad to encrypt a message of our choice.

3.2 Constructing Weak Hash-Proof Systems from Any PKE

We present a weak hash proof system starting from any semantically secure PKE. We begin by constructing a wHPS with a very small output-space $\mathcal{K} = \mathbb{Z}_m$ for some polynomial $m = m(\lambda)$. In other words, the entropy of the output is only $\log(m) = O(\log(\lambda))$ bits. We will then amplify this via parallel repetition. The construction below generalizes the scheme we described in the introduction, which corresponds to the special case of $m = 2$ and the output is only 1 bit. By increasing m, we get an improvement in the *leakage rate* of our scheme.

Basic Construction. Let $m = m(\lambda)$ be some polynomial parameter and let $\Pi = (\mathsf{PKE.Gen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ be a public-key encryption scheme with message-space $\mathcal{M} \supseteq \mathbb{Z}_m$.[4] We construct a wHPS with output space $\mathcal{K} = \mathbb{Z}_m$ as follows:

- $\mathsf{wHPS.Gen}(1^\lambda)$: Generate m key pairs: $\{(\text{PK}_i, \text{SK}_i) \leftarrow \mathsf{PKE.Gen}(1^\lambda)\}_{i \in [m]}$. Sample a random $t \leftarrow [m]$. Output $\text{SK} = (t, \text{SK}_t), \text{PK} = (\text{PK}_1, \ldots, \text{PK}_m)$.
- $\mathsf{wHPS.Encap}(\text{PK})$: Choose $k \leftarrow \mathbb{Z}_m$, and set $c := \{c_i \leftarrow \mathsf{PKE.Enc}(\text{PK}_i, k)\}_{i \in [m]}$. Output (c, k).
- $\mathsf{wHPS.Encap}^*(\text{PK})$: Choose $k \leftarrow \mathbb{Z}_m$. Output $c^* = \{c_i^* \leftarrow \mathsf{PKE.Enc}(\text{PK}_i, k + i)\}_{i \in [m]}$, where the addition $k + i$ is performed in the group $\mathbb{Z}_m$.
- $\mathsf{wHPS.Decap}(\text{SK}, c)$: Parse $\text{SK} = (t, \text{SK}_t)$ and $c = \{c_i\}$. Output $k = \mathsf{PKE.Dec}(\text{SK}_t, c_t)$.

Theorem 1. *If* $(\mathsf{PKE.Gen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ *is a semantically secure public-key encryption scheme, then the construction above is a weak hash-proof system with output space* $\mathcal{K} = \mathbb{Z}_m$*.*

[4] We can set $\mathcal{M} = \{0,1\}^{\lceil \log(m) \rceil}$ and naturally interpret it as containing $\mathbb{Z}_m$.

Output Amplification Via Parallel Repetition. The above construction gives us a public-key wHPS with a polynomial-sized output domain $\mathcal{K} = \mathbb{Z}_m$, so that the entropy of the output is only logarithmic. Unfortunately, we cannot use this scheme directly to get a meaningful LR-PKE, since we don't even have enough entropy to extract a single bit! However, it turns out to be very simple to increase the output-length of a wHPS just by taking several independent copies.

Theorem 2. *Let Π be any wHPS with output-domain $\mathcal{K}$. Let $n = n(\lambda)$ be a polynomial and Π^n be the n-wise parallel-repetition of Π. Then Π^n is a wHPS with output-domain $\mathcal{K}^n$.*

By taking our basic construction of wHPS with parameter m and applying n-wise parallel-repetition, we get a scheme with leakage resilience $\ell \approx n \cdot \log(m)$ and secret-key size $\approx n \cdot s$, meaning that we get a leakage-rate $\alpha \approx \log(m)/s$. By taking a sufficiently large n and m, the following theorem.

Theorem 3. *Assume the existence of semantically-secure PKE with secret-key size $s = s(\lambda)$. Then, for any arbitrarily large polynomial $\ell = \ell(\lambda)$ and any $\alpha = \alpha(\lambda) = O\left(\frac{\log \lambda}{s(\lambda)}\right)$ there exists an ℓ-leakage-resilient PKE where the* leakage rate *(ratio of ℓ to secret key size) is α.*

4 Leakage-Resilient wPRF and Symmetric-Key Encryption

Defining LR-wPRF. We begin with the definition of a *Leakage-Resilient weak PRF (wPRF).* Recall that the standard notion of a wPRF tells us that, given arbitrarily many *uniformly random* inputs $x_1, \ldots, x_q$, the outputs of the wPRF $y_1 = f_k(x_1), \ldots, y_q = f_k(x_q)$ look pseudo-random. This is in contrast with standard PRFs where the above holds for a *worst-case (adversarial)* choice of inputs $\{x_i\}$. Our definition of leakage-resilient wPRF requires that wPRF security holds even if the attacker can leak some information about the secret key. In particular, any future output of the wPRF on a fresh random input will still look random. Note that, since the attacker can always leak a few bits of $f_k(x)$ for some x of his choice, we cannot hope to achieve full PRF security in the presence of leakage, and hence settling for wPRF security is a natural choice.

Definition 3 (Leakage-Resilient weak PRF (LR-wPRF)). *Let $\mathcal{X}, \mathcal{Y}, \mathcal{K}$ be efficient ensembles and let $\mathcal{F} = \{ F_K : \mathcal{X} \to \mathcal{Y}\}_{K \in \mathcal{K}}$ be some efficient function family. We say that $\mathcal{F}$ is an $\ell(\lambda)$-*leakage-resilient weak PRF (LR-wPRF) *if, for all PPT attackers $\mathcal{A}$ the advantage of $\mathcal{A}$ is negligible in the following game:*

Initialization: *The challenger chooses a random $K \leftarrow \mathcal{K}_\lambda$.*

Learning Stage: *The attacker $\mathcal{A}^{\mathcal{O}^\ell_K(\cdot), F_K(\$)}(1^\lambda)$ gets access to the* leakage oracle $\mathcal{O}^\ell_K(\cdot)$ *(allowing him to learn up to ℓ bits of information about K) and also the* wPRF oracle $F_K(\$)$ *which does not take any input and, on each invocation, chooses a freshly random $X \leftarrow \mathcal{X}$ and outputs $(X, F_K(X))$.*[5]

[5] Without loss of generality, we can also assume that the attacker only makes a single call to the leakage oracle $\mathcal{O}^\ell_K(\cdot)$ after making all of its calls to the wPRF oracle $F_K(\$)$.

Challenge Stage: *The challenger chooses a challenge bit $b \leftarrow \{0,1\}$ and a random input $X^* \leftarrow \mathcal{X}$. If $b = 0$, it sets $Y^* := F_K(X^*)$ and if $b = 1$ it chooses $Y^* \leftarrow \mathcal{Y}$. The challenger gives (X^*, Y^*) to $\mathcal{A}$ who outputs a bit b'. We define the advantage of the attacker $\mathcal{A}$ as $Adv_{\mathcal{A}}(\lambda) = \left|\Pr[b' = b] - \frac{1}{2}\right|$.*

In the setting of no leakage we call a function $\mathcal{F}$ satisfying the above definition a standard wPRF.

From wPRF to CPA Encryption. In the full version [31] we show how to construct leakage-resilient CPA-secure (LR-CPA) symmetric-key encryption from LR-wPRF.

4.1 Leakage-Resilience Via Symmetric-Key wHPS

Toward the goal of constructing a LR-wPRF, we define a new notion of a *symmetric-key weak hash-proof system* (SwHPS), which can be thought of as a symmetric-key version of wHPS. In particular, we define a symmetric-key wHPS as a type of wPRF family $\mathcal{F} = \{F_K : \mathcal{X} \to \mathcal{Y}\}_{K \in \mathcal{K}}$ with some special properties.

Other than being able to choose inputs $X \leftarrow \mathcal{X}$ uniformly at random from their domain (which we refer to as the distribution Dist_0), we can also define two additional distributions Dist_1 (valid), and Dist_2 (invalid) over the input-domain $\mathcal{X}$. We require that samples from these various distributions are indistinguishable even when given the secret key K. Furthermore, conditioned on seeing many pairs $\{(X_i, F_K(X_i))\}$ for many different $X_i \leftarrow \mathsf{Dist}_1$ and a random choice of $X^* \leftarrow \mathsf{Dist}_2$, the output of $F_K(X^*)$ will be *truly* random and independent, where the randomness comes from the choice of a consistent secret key K.

Definition 4 (Symmetric-Key wHPS). *Let $\mathcal{X}, \mathcal{Y}, \mathcal{K}$ be some efficient ensembles and let $\mathcal{F} = \{ F_K : \mathcal{X} \to \mathcal{Y}\}_{K \in \mathcal{K}}$ be some efficient function family with the following PPT algorithms:*

- $\mathsf{samK} \leftarrow \mathsf{SamGen}(K)$ *takes an input $K \in \mathcal{K}$, outputs a* sampling key samK.
- $X \leftarrow \mathsf{Dist}_1(\mathsf{samK}), X \leftarrow \mathsf{Dist}_2(\mathsf{samK})$ *are two distributions that sample $X \in \mathcal{X}$ using the sampling key* samK. *For convenience, we also define the distribution $X \leftarrow \mathsf{Dist}_0(\mathsf{samK})$ which just samples a uniformly random $X \leftarrow \mathcal{X}$ and ignores the sampling key* samK.

We say that $\mathcal{F}$ is a symmetric-key wHPS (SwHPS) *if it satisfies the following two properties:*

Input Indistinguishability. *For any polynomial $q = q(\lambda)$ and any choice of $(b_1, \ldots, b_q), (b'_1, \ldots, b'_q) \in \{0,1,2\}^q$, the following distributions are computationally indistinguishable: $(K, X_1, \ldots, X_q) \approx_c (K, X'_1, \ldots, X'_q)$, where $K \leftarrow \mathcal{K}_\lambda, \mathsf{samK} \leftarrow \mathsf{SamGen}(K), \{X_i \leftarrow \mathsf{Dist}_{b_i}(\mathsf{samK})\}, \{X'_i \leftarrow \mathsf{Dist}_{b'_i}(\mathsf{samK})\}$.*

Smoothness. *For any polynomial $q = q(\lambda)$ the following distributions are equivalent: $(X_1, \ldots, X_q, Y_1, \ldots, Y_q, X^*, Y^*) \equiv (X_1, \ldots, X_q, Y_1, \ldots, Y_q, X^*, U)$, where $K \leftarrow \mathcal{K}_\lambda, \mathsf{samK} \leftarrow \mathsf{SamGen}(K), \{X_i \leftarrow \mathsf{Dist}_1(\mathsf{samK}), Y_i := F_K(X_i)\}_{i \in [q]}$, $X^* \leftarrow \mathsf{Dist}_2(\mathsf{samK})$, $Y^* = F_K(X^*)$, and $U \leftarrow \mathcal{Y}$. In other words, Y^* is uniformly random and independent of the other elements, where the randomness comes from the choice of a key K.*

Constructing LR-wPRF from SwHPS. We now construct a leakage-resilient wPRF from any symmetric-key wHPS. As in the public-key setting, we simply apply an extractor to the output of the symmetric-key wHPS.

Theorem 4. *Assume that* $\mathcal{X}, \mathcal{Y}, \mathcal{S}, \mathcal{Z}$ *are efficient ensembles such that* $\mathcal{F} = \{ F_K : \mathcal{X} \to \mathcal{Y}\}_{K \in \mathcal{K}}$ *is a symmetric-key wHPS and* $\mathsf{Ext} : \mathcal{Y} \times \mathcal{S} \to \mathcal{Z}$ *is a* $(\log(|\mathcal{Y}|) - \ell(\lambda), \varepsilon(\lambda))$*-extractor for some negligible* $\varepsilon(\lambda)$*. Define the function family* $\mathcal{F}' = \{F'_K : (\mathcal{X} \times \mathcal{S}) \to \mathcal{Z}\}_{K \in \mathcal{K}}$ *via* $F'_K((X, S)) := \mathsf{Ext}(F_K(X); S)$*. Then* $\mathcal{F}'$ *is an* $\ell(\lambda)$*-LR wPRF.*

4.2 Constructing Symmetric-Key wHPS

We now construct symmetric-key wHPS (SwHPS) from any weak PRF, and therefore also from the mere existence of one-way functions.

Basic Construction. Let $m = m(\lambda)$ be some polynomial and let $\mathcal{F}_{weak} = \{f_k : \mathcal{X} \to \mathbb{Z}_m\}_{k \in \mathcal{K}}$ be a standard (0-LR) wPRF family.[6] Let $(\mathsf{Enc}, \mathsf{Dec})$ be a standard symmetric-key encryption scheme constructed from $\mathcal{F}_{weak}$ as follows:

- $\mathsf{Enc}_k(\textsc{m})$: Choose $x \leftarrow \mathcal{X}$ and output $c = (x, f_k(x) + \textsc{m})$, where the addition is performed in $\mathbb{Z}_m$.
- $\mathsf{Dec}_k(c = (x, z))$: Output $\textsc{m} := z - f_k(x)$.

Notice that this encryption scheme has message space $\mathcal{M} = \mathbb{Z}_m$, ciphertext space $\mathcal{C} = (\mathcal{X} \times \mathbb{Z}_m)$ and key-space $\mathcal{K}$. A useful property of this encryption scheme is that we can obliviously sample $c \leftarrow \mathcal{C}$ without knowing the key k, and this induces the same distribution as encrypting a random $\textsc{m} \leftarrow \mathbb{Z}_m$. Given the wPRF $\mathcal{F}_{weak}$ and the resulting encryption scheme $(\mathsf{Enc}, \mathsf{Dec})$ as above, we define the symmetric-key wHPS system: $\mathcal{F}_{SwHPS} = \{F_K : \mathcal{C}^m \to \mathbb{Z}_m\}_{K \in ([m] \times \mathcal{K})}$ where

$$F_{(K=(t,k))}(X = (c_1, \ldots, c_m)) := \mathsf{Dec}_k(c_t).$$

Notice that we can efficiently sample random inputs $X \leftarrow \mathcal{C}^m$ without knowing the key K. We define the additional algorithms needed for the definition of SwHPS as follows:

- $\mathsf{samK} \leftarrow \mathsf{SamGen}(K)$. Parse $K = (t, k)$. Choose $m - 1$ values $\{k_i \leftarrow \mathcal{K} : i \in [m], i \neq t\}$ and define $k_t := k$. Set $\mathsf{samK} := (k_1, \ldots, k_m)$.
- $X \leftarrow \mathsf{Dist}_1(\mathsf{samK})$ (*Valid*). Choose $r \leftarrow \mathbb{Z}_m$ and $\{c_i \leftarrow \mathsf{Enc}_{k_i}(r)\}_{i \in [m]}$. Output $X = (c_1, \ldots, c_m)$.
- $X \leftarrow \mathsf{Dist}_2(\mathsf{samK})$ (*Invalid*). Choose $r \leftarrow \mathbb{Z}_m$ and $\{c_i \leftarrow \mathsf{Enc}_{k_i}(r + i)\}_{i \in [m]}$ where the addition is performed in $\mathbb{Z}_m$. Output $X = (c_1, \ldots, c_m)$.

For a *valid* X all of the ciphertexts c_i decrypt to the *same* value r, and for an *invalid* X they all decrypt to *different* values $r + i$. It is easy to see that the distributions $\mathsf{Dist}_1, \mathsf{Dist}_2$ are indistinguishable from uniform (Dist_0) even given $K = (t, k)$ since the ciphertext c_t always *is* uniform on its own, and we cannot distinguish the ciphertexts $c_i : i \neq t$ from uniform by the security of the

[6] If m is a power of 2, then we can just identify the elements of $\mathbb{Z}_m$ with those of $\{0, 1\}^{\log(m)}$ in a natural way. Therefore, the existence of such wPRFs does not require any special assumptions.

wPRF. Furthermore, given many values $\{X_i, F_K(X_i)\}$ where X_i is *valid*, we learn nothing (information theoretically) about the secret index t contained in $K = (t, k)$. Therefore, for a random *invalid* $X^* \leftarrow \mathsf{Dist}_2(\mathsf{samK})$, the output $F_K(X^*) = \mathsf{Dec}_{k_t}(c_t) = r + t$ is truly random and independent.

Theorem 5. *Assuming $\mathcal{F}_{weak}$ is a standard wPRF, the function family $\mathcal{F}_{SwHPS}$ as defined above is a symmetric-key wHPS.*

Output Amplification Via Parallel Repetition. The above construction only obtains a polynomial-sized output domain, which means that the outputs only have $O(\log(\lambda))$ entropy. We amplify the output domain and entropy by using "parallel repetition". Formally,

Theorem 6. *Assume that $\mathcal{F} = \{f_k : \mathcal{X} \to \mathcal{Y}\}_{k \in \mathcal{K}}$ is a symmetric-key wHPS and let $n = n(\lambda)$ be an arbitrary polynomial. Define $\mathcal{F}^n = \{F_K : \mathcal{X}^n \to \mathcal{Y}^n\}_{K \in \mathcal{K}^n}$ via $F_{(k_1,\ldots,k_n)}(x_1,\ldots,x_n) \stackrel{\text{def}}{=} (f_{k_1}(x_1),\ldots,f_{k_n}(x_n))$. Then $\mathcal{F}^n$ is also a symmetric-key wHPS, whose output is amplified by a factor of n.*

We summarize our final results by combining all the developed ingredients.

Theorem 7. *Assuming the existence of one-way functions, there exist $\ell(\lambda)$-LR-wPRFs and $\ell(\lambda)$-LR-CPA symmetric-key encryption schemes for any polynomial $\ell(\lambda)$. Furthermore, assuming the existence of standard wPRFs with key-size $s(\lambda)$, the above schemes exist for any* leakage rate $\alpha(\lambda) = O(\log(\lambda)/s(\lambda))$.

5 Leakage-Resilient Message Authentication

In this section, we construct leakage-resilient message-authentication codes (LR-MACs) from the minimal assumption that one-way functions exit.

Definition 5 (Leakage-Resilient MAC). *A MAC consists of the algorithms* $(\mathsf{Tag}, \mathsf{Ver})$ *and an efficient ensemble $\mathcal{K}$ of secret-keys. For* correctness*, we require that for every message* $\text{M} \in \{0,1\}^*$*, and every key $K \in \mathcal{K}$, and every correctly generated tag* $\sigma \leftarrow \mathsf{Tag}_K(\text{M})$*, we have* $\mathsf{Ver}_K(\text{M}, \sigma) = 1$*. For* security*, we consider the following game between an attacker $\mathcal{A}$ and a challenger:*

Initialization: *The challenger chooses a random key $K \leftarrow \mathcal{K}_\lambda$.*
Learning Stage: *The attacker $\mathcal{A}^{\mathcal{O}^\ell_K(\cdot), \mathsf{Tag}_K(\cdot), \mathsf{Ver}_K(\cdot,\cdot)}$ can adaptively ask arbitrary* leakage*, tagging and* verification *queries to its oracles.*
Forgery: *The attacker provides a* forgery (M^*, σ^*) *and* wins *if* M^* *was never given as an input to the tagging oracle* $\mathsf{Tag}_K(\cdot)$ *during the leaning stage and* $\mathsf{Ver}_K(\text{M}^*, \sigma^*) = 1$.

We say that such a scheme is an $\ell(\lambda)$-leakage-resilient message authentication code (ℓ-LR-MAC) *if, for all PPT attackers $\mathcal{A}$, the probability that $\mathcal{A}$ wins in the above game is negligible.*

In addition to the above definition, we also define a weaker notion of security against *"no-verification-query attacks" (nvq-MAC)*, where the attacker does not get access to the verification oracle $\mathsf{Ver}_K(\cdot,\cdot)$ during the learning stage. See the full version [31] for additional discussion. As a starting step, we instantiate a leakage-resilient nvq-MAC and then upgrade it to achieve full security.

Constructing nvq-MACs. Let $\mathcal{F}_{\mathsf{prf}} = \{f_k : \{0,1\}^* \to \mathcal{Y}\}_{k \in \mathcal{K}}$ be a *pseudorandom function (PRF) family* with super-polynomial output domain $|\mathcal{Y}_\lambda| = \lambda^{\omega(1)}$. Let $n = n(\lambda)$, $m = m(\lambda)$ be arbitrary polynomials. We construct a MAC with key-space $\mathcal{K}^{MAC} = (\mathcal{K} \times [m])^n$, by parsing $K \in \mathcal{K}^{MAC}$ as $K = ((k_1, t_1), \ldots, (k_n, t_n))$ where $k_i \in \mathcal{K}, t_i \in [m]$. We define the algorithms $(\mathsf{Tag}, \mathsf{Ver})$ as follows:

- $\mathsf{Tag}_K(\textsc{m})$: Parse $K = ((k_1, t_1), \ldots, (k_n, t_n))$. Choose a random nonce $r \leftarrow \{0,1\}^\lambda$ and output the tag $\sigma = (r, \{\sigma_{i,j}\})$ where $\{\sigma_{i,j}\}$ is an $n \times m$ matrix defined by:
$$\sigma_{i,j} := \begin{cases} f_{k_i}(r||\textsc{m}) & \text{if } j = t_i \\ y \leftarrow \mathcal{Y} & \text{otherwise} \end{cases}$$
That is, each row $i \in [n]$ of the matrix $\{\sigma_{i,j}\}$ contains one pseudorandom value under key k_i in the column t_i, and the rest of the row is truly random.
- $\mathsf{Ver}_K(\textsc{m}, \sigma)$: Parse $\sigma = (r, \{\sigma_{i,j}\})$. For all $i \in [n]$, check that $f_{k_i}(r||\textsc{m}) = \sigma_{i,t_i}$.

Security Intuition. We explain the security of the above construction via several abstract properties. Firstly, we can define an *alternate tagging oracle*, which is computationally indistinguishable from the original one even given the secret key $K = ((k_1, t_1), \ldots, (k_n, t_n))$ in full. The alternate oracle initially chooses an entire $n \times m$ matrix of PRF keys $\{k_{i,j}\}$, where the keys $k_{i,t_i} := k_i$ are taken from K and the rest of the keys $k_{i,j}$ for $j \neq t_i$ are chosen randomly. When answering tagging queries, the alternate tagging oracle sets *all* of the values $\sigma_{i,j} := f_{k_{i,j}}(r||\textsc{m})$ to be pseudorandom under the appropriate keys. Once we define the alternate tagging oracle, we can also define two types of forgeries: valid and invalid. A forgery $\textsc{m}^*, \sigma^* = (r^*, \{\sigma^*_{i,j}\})$ is *valid* if there is some pair (i,j) with $j \neq t_i$ such that $\sigma^*_{i,j} = f_{k_{i,j}}(r^*||\textsc{m}^*)$, and is *invalid* otherwise. We have the following properties:

(1) Given access to the alternate tagging oracle and the key K in full, it is computationally hard to come up with a *valid* accepting forgery. *(Doing so requires guessing a PRF output at some fresh point* $(r^*||\textsc{m}^*)$*, for some PRF key* $k_{i,j}$ *which doesn't appear in* K*.)*

(2) Given access to the alternate tagging oracle but not the key K, the information-theoretic probability of outputting an *invalid* accepting forgery is $< 2^{-n\log(m)}$. *(Doing so requires guessing the indices* $T = (t_1, \ldots, t_n)$ *since the only pairs* (i,j) *for which* $\sigma^*_{i,j} = f_{k_{i,j}}(r^*||\textsc{m}^*)$ *are when* $j = t_i$*. But* T *has* $n\log(m)$ *bits of entropy and is independent of the above oracle.)*

The above properties ensure leakage-resilience for up to $\ell = n\log(m) - \omega(\log(\lambda))$ bits of leakage on the key K. Given such leakage, producing a valid forgery becomes no easier, since it is already hard given K in full. On the other hand, the probability of producing an invalid forgery can go up by a factor of at most 2^ℓ, which remains negligible. We formalize this in the following theorem.

Theorem 8. *If* $\mathcal{F}_{\mathsf{prf}}$ *is a PRF family with parameters as above, then the given construction is an* $\ell(\lambda)$*-leakage-resilient nvq-MAC for any* $\ell(\lambda) = n(\lambda) \log(m(\lambda)) - \omega(\log(\lambda))$*.*

In the full version [31], we show how to upgrade nvq-MAC security to full MAC security. As a consequence, we get the following theorem.

Theorem 9. *Assuming the existence of one-way functions, there exist $\ell(\lambda)$-leakage-resilient MACs for any polynomial $\ell(\lambda)$. Furthermore, assuming the existence of PRFs with variable-length input-size, output size λ, and key-size $s(\lambda)$, such MACs exist for any* leakage rate $\alpha(\lambda) = O(\log(\lambda)/s(\lambda))$.

6 Conclusions

We saw how to construct several leakage-resilient primitives under the minimal assumption that they exists in the standard setting without any leakage. Perhaps the main open question is to improve the *leakage rate* of such constructions (say, to some constant fraction of the secret key), or to provide black-box separations showing that this is not be possible. Another interesting open question is to construct leakage-resilient *signatures* under the minimal assumption that one-way functions exist. Lastly, it would be interesting to come up with other applications where *weak* hash-proof systems (wHPS) can replace standard HPS.

References

1. Agrawal, D., Archambeault, B., Rao, J.R., Rohatgi, P.: The EM side-channel(s). In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 29–45. Springer, Heidelberg (2003)
2. Akavia, A., Goldwasser, S., Vaikuntanathan, V.: Simultaneous hardcore bits and cryptography against memory attacks. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 474–495. Springer, Heidelberg (2009)
3. Alwen, J., Dodis, Y., Naor, M., Segev, G., Walfish, S., Wichs, D.: Public-key encryption in the bounded-retrieval model. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 113–134. Springer, Heidelberg (2010)
4. Alwen, J., Dodis, Y., Wichs, D.: Leakage-resilient public-key cryptography in the bounded-retrieval model. In: Halevi [28], pp. 36–54
5. Bar-El, H.: Known attacks against smartcards (2003), `http://www.hbarel.com/publications/Known_Attacks_Against_Smartcards.pdf` (last accessed: August 26, 2009)
6. Bitansky, N., Canetti, R., Halevi, S.: Leakage-tolerant interactive protocols. In: Cramer [14], pp. 266–284
7. Boyle, E., Segev, G., Wichs, D.: Fully leakage-resilient signatures. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 89–108. Springer, Heidelberg (2011)
8. Brakerski, Z., Goldwasser, S.: Circular and leakage resilient public-key encryption under subgroup indistinguishability (or: Quadratic residuosity strikes back). In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 1–20. Springer, Heidelberg (2010)
9. Brakerski, Z., Kalai, Y.T.: A parallel repetition theorem for leakage resilience. In: Cramer [14], pp. 248–265
10. Brakerski, Z., Katz, J., Kalai, Y., Vaikuntanathan, V.: Overcomeing the hole in the bucket: Public-key cryptography against resilient to continual memory leakage. In: FOCS [32], pp. 501–510
11. Braverman, M., Hassidim, A., Kalai, Y.T.: Leaky pseudo-entropy functions. In: Chazelle, B. (ed.) ICS, pp. 353–366. Tsinghua University Press (2011)

12. Cash, D., Ding, Y.Z., Dodis, Y., Lee, W., Lipton, R.J., Walfish, S.: Intrusion-resilient key exchange in the bounded retrieval model. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 479–498. Springer, Heidelberg (2007)
13. Chow, S.S.M., Dodis, Y., Rouselakis, Y., Waters, B.: Practical leakage-resilient identity-based encryption from simple assumptions. In: Al-Shaer, E., Keromytis, A.D., Shmatikov, V. (eds.) ACM Conference on Computer and Communications Security, pp. 152–161. ACM (2010)
14. Cramer, R. (ed.): TCC 2012. LNCS, vol. 7194. Springer, Heidelberg (2012)
15. Cramer, R., Shoup, V.: Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 45–64. Springer, Heidelberg (2002)
16. Crescenzo, G.D., Lipton, R.J., Walfish, S.: Perfectly secure password protocols in the bounded retrieval model. In: Halevi and Rabin [30], pp. 225–244
17. Dodis, Y., Haralambiev, K., López-Alt, A., Wichs, D.: Cryptography against continuous memory attacks. In: FOCS [32], pp. 511–520
18. Dodis, Y., Haralambiev, K., López-Alt, A., Wichs, D.: Efficient public-key cryptography in the presence of key leakage. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 613–631. Springer, Heidelberg (2010)
19. Dodis, Y., Lewko, A.B., Waters, B., Wichs, D.: Storing secrets on continually leaky devices. In: Ostrovsky, R. (ed.) FOCS, pp. 688–697. IEEE (2011)
20. Dodis, Y., Yu, Y.: Overcoming weak expectations. In: ITW (2012), `http://www.cs.nyu.edu/~dodis/ps/weak-expe.pdf`
21. Dziembowski, S.: Intrusion-resilience via the bounded-storage model. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 207–224. Springer, Heidelberg (2006)
22. Dziembowski, S.: Intrusion-resilience via the bounded-storage model. In: Halevi and Rabin [30], pp. 207–224
23. Dziembowski, S., Pietrzak, K.: Leakage-resilient cryptography. In: 49th Symposium on Foundations of Computer Science, October 25-28, pp. 293–302. IEEE Computer Society, Philadelphia (2008)
24. ECRYPT: Side channel cryptanalysis lounge, `http://www.emsec.rub.de/research/projects/sclounge/` (last accessed: May 1, 2011)
25. Goldwasser, S., Kalai, Y.T., Peikert, C., Vaikuntanathan, V.: Robustness of the learning with errors assumption. In: Yao, A.C.C. (ed.) ICS, pp. 230–240. Tsinghua University Press (2010)
26. Goldwasser, S., Rothblum, G.N.: How to compute in the presence of leakage. In: Electronic Colloquium on Computational Complexity (ECCC), vol. 19, p. 10 (2012)
27. Halderman, J.A., Schoen, S.D., Heninger, N., Clarkson, W., Paul, W., Calandrino, J.A., Feldman, A.J., Appelbaum, J., Felten, E.W.: Lest we remember: cold-boot attacks on encryption keys. Commun. ACM 52(5), 91–98 (2009)
28. Halevi, S. (ed.): CRYPTO 2009. LNCS, vol. 5677. Springer, Heidelberg (2009)
29. Halevi, S., Lin, H.: After-the-fact leakage in public-key encryption. In: Ishai [33], pp. 107–124
30. Halevi, S., Rabin, T. (eds.): TCC 2006. LNCS, vol. 3876. Springer, Heidelberg (2006)
31. Hazay, C., López-Alt, A., Wee, H., Wichs, D.: Leakage-resilient cryptography from minimal assumptions. Cryptology ePrint Archive, Report 2012/604 (2012), `http://eprint.iacr.org/`
32. IEEE: 51th Symposium on Foundations of Computer Science, October 23-26 (2010)

33. Ishai, Y. (ed.): TCC 2011. LNCS, vol. 6597. Springer, Heidelberg (2011)
34. Ishai, Y., Sahai, A., Wagner, D.: Private circuits: Securing hardware against probing attacks. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 463–481. Springer, Heidelberg (2003)
35. Garg, S., Jain, A., Sahai, A.: Leakage-resilient zero knowledge. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 297–315. Springer, Heidelberg (2011)
36. Jain, A., Pietrzak, K.: Parallel repetition for leakage resilience amplification revisited. In: Ishai [33], pp. 58–69
37. Katz, J., Vaikuntanathan, V.: Signature schemes with bounded leakage resilience. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 703–720. Springer, Heidelberg (2009)
38. Kocher, P.C.: Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996)
39. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
40. Lewko, A., Waters, B.: On the insecurity of parallel repetition for leakage resilience. In: FOCS [32], pp. 521–530
41. Lewko, A.B., Lewko, M., Waters, B.: How to leak on key updates. In: STOC (2011) (to appear)
42. Micali, S., Reyzin, L.: Physically observable cryptography (extended abstract). In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 278–296. Springer, Heidelberg (2004)
43. Naor, M., Segev, G.: Public-key cryptosystems resilient to key leakage. In: Halevi [28], pp. 18–35
44. Naor, M., Segev, G.: Public-key cryptosystems resilient to key leakage. SIAM Journal on Computing 41(4), 772–814 (2012); a preliminary version appeared in Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 18–35. Springer, Heidelberg (2009)
45. Pietrzak, K.: A leakage-resilient mode of operation. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 462–482. Springer, Heidelberg (2009)
46. Quisquater, J.-J., Samyde, D.: ElectroMagnetic analysis (EMA): Measures and counter-measures for smart cards. In: Attali, S., Jensen, T. (eds.) E-smart 2001. LNCS, vol. 2140, pp. 200–210. Springer, Heidelberg (2001)
47. Quisquater, J.J., Koene, F.: Side channel attacks: State of the art (October 2002), `http://www.ipa.go.jp/security/enc/CRYPTREC/fy15/doc/1047_Side_Channel_report.pdf` (last accessed: August 26, 2009)
48. Reliable Computing Laboratory, Boston University: Side channel attacks database, `http://www.sidechannelattacks.com` (last accessed: August 26, 2009)
49. Standaert, F.X.: How leaky is an extractor? In: Abdalla, M., Barreto, P.S.L.M. (eds.) LATINCRYPT 2010. LNCS, vol. 6212, pp. 294–304. Springer, Heidelberg (2010)

Faster Index Calculus for the Medium Prime Case Application to 1175-bit and 1425-bit Finite Fields

Antoine Joux

CryptoExperts and
Université de Versailles Saint-Quentin-en-Yvelines, Laboratoire PRISM,
45 avenue des États-Unis, F-78035 Versailles Cedex, France
antoine.joux@m4x.org

Abstract. Many index calculus algorithms generate multiplicative relations between smoothness basis elements by using a process called *Sieving*. This process allows us to quickly filter potential candidate relations, without spending too much time to consider bad candidates. However, from an asymptotic point of view, there is not much difference between sieving and straightforward testing of candidates. The reason is that even when sieving, some small amount of time is spent for each bad candidate. Thus, asymptotically, the total number of candidates contributes to the complexity.

In this paper, we introduce a new technique: *Pinpointing*, which allows us to construct multiplicative relations much faster, thus reducing the asymptotic complexity of relations' construction. Unfortunately, we only know how to implement this technique for finite fields which contain a medium-sized subfield. When applicable, this method improves the asymptotic complexity of the index calculus algorithm in the cases where the sieving phase dominates. In practice, it gives a very interesting boost to the performance of state-of-the-art algorithms. We illustrate the feasability of the method with discrete logarithm records in two medium prime finite fields, the first of size 1175 bits and the second of size 1425 bits.

1 Introduction

Index calculus algorithms form a large class of algorithms for solving hard number theoretic problems which are often used as a basis for public key cryptosystems. They can be used for factoring large integers [19] and for computing discrete logarithms in finite fields [2,11,1] and in some elliptic or hyperelliptic curve groups [7,9,6,10,8].

All index calculus algorithms have in common two main algorithmic phases. The first of these phases is the generation of multiplicative[1] relations, which are converted into linear or affine equalities involving the logarithms of the elements

[1] In the case of curves, the relation are denoted additively, but the principle remains.

which appear in the multiplicative relations. The second phase is the linear algebra phase, which solves the resulting system of equations. For factoring, the linear algebra is performed modulo 2. For discrete logarithms, it is done modulo the order of the relevant group. In addition to these two common phases, several other phases also appear: these extra phases heavily depend on the exact algorithm being considered. They can be further classified as preparatory or final phases. The preparatory phases search for a good representation of the structure being considered in order to speed-up the main phases. For example, polynomial selection is a typical preparatory phase which appears when factoring with the number field sieve [20]. The final phases transform the raw output of the linear algebra phase into a solution of the considered problem. Typically, this includes the so-called square root phase of factoring algorithms and the individual logarithm phase encountered in many discrete logarithm algorithms. It should be noted that the computational cost of these prepatory and final phases is usually much smaller than the cost of the main phases.

In most cases, the designers of index calculus algorithms aim at balancing the theoretical complexity of the two main phases, since this usually yields the best global effectiveness. However, this is not always possible as illustrated by the function field sieve for the medium prime case introduced in [18]. In this specific case, the exact asymptotic complexity varies depending on the relative contribution of the base field and of the extension degree to the total size of the finite field being considered (see Section 2). In practice, the two main phases are usually much less balanced. This is due to the fact that the generation of relations phase can, in general, be distributed among machines in a straightforward way. On the contrary, the linear algebra requires a tightly coordinated computation and is generally performed on a centralized super-computer (or sometimes on a few super-computers). Since centralized computations that require tight communications are more expensive than distributed computations, implementers usually generate an extremely large number of linear equations compared to the number of unknowns. We can various techniques such as filtering, rebalancing or structured Gaussian elimination, in order to reduce the size of the linear system which is eventually solved and thus the cost of the linear algebra phase. This may increase the total computing power used for the computation, but trading expensive centralized computations for cheaper distributed computations is usually worthwhile.

As a consequence of these considerations, we see that the generation of relations is a very important phase of index calculus algorithms. Up to now, two main techniques are usually used. The simplest approach is direct trial where one simply checks whether a potential candidate turns into an effective relation by testing whether an integer or a polynomial splits into a product of “small” elements. In theory, the parameters of index calculus are selected to make sure that the cost of testing a candidate has a negligible contribution to the overall complexity. However, in practice, factoring these objects has a non-neglibible cost. Thus, the other approach called *sieving* is usually prefered. The basic idea of sieving is to proceed backward and mark all multiples of small elements. Clearly, an object which receives many marks is much more likely to generate a useful multiplicative relation that an object which receives few marks. Note that, from a theoretic point of view, sieving does not change the complexity of the sieving phase. Indeed, all the potential candidates still need to be considered and even reducing the cost of considering a candidate to a unit cost would not be enough to lower the overall asymptotic complexity.

In this paper, we introduce a new technique to generate relations which is much faster that sieving. In some cases, the cost of relation generation becomes essentially optimal: we only require a small number of arithmetic operations per *generated relation.* To indicate that this technique sometimes allows to directly access the relations, we name it *Pinpointing.* Unfortunately, we only know how to achieve this for a limited number of index calculus algorithms. More precisely, we show how to use pinpointing for the medium prime case as described in [18].

2 A Refresher on the Medium Prime Case

The medium prime discrete logarithms proposed in [18] works as follows. In order to compute discrete logarithms in $\mathbb{F}_{q^n}$, a degree n extension of the base field $\mathbb{F}_q$, it starts by defining the extension field implicitly from two bivariate polynomials in X and Y:

$$f_1(X,Y) = X - g_1(Y), \qquad f_2(X,Y) = -g_2(X) + Y,$$

where g_1 and g_2 are univariate polynomials of degree d_1 and d_2. In order to define the expected extension, this requires that the polynomial $-g_2(g_1(Y)) + Y$ has an irreducible factor $F(Y)$ of degree n over $\mathbb{F}_q$. As explained in [18], it is easy to find polynomials g_1 and g_2 that satisfy this requirement.

The relative degrees of d_1 and d_2 in this case are controlled by an extra parameter D, whose choice is determined by the size of q compared to q^n. More precisely, we have $d_1 \approx \sqrt{Dn}$ and $d_2 \approx \sqrt{n/D}$.

Starting from this definition of the finite field, the medium prime field algorithms consider objects of the form $\mathcal{A}(Y)\,X + \mathcal{B}(Y)$, where $\mathcal{A}$ and $\mathcal{B}$ are univariate polynomials of degree D and $\mathcal{A}$ is unitary. Substituting $g_1(Y)$ for X on one side and $g_2(X)$ for Y on the other, we obtain an equation:

$$\mathcal{A}(Y)\,g_1(Y) + \mathcal{B}(Y) = \mathcal{A}(g_2(X))\,X + \mathcal{B}(g_2(X)).$$

This relates a polynomial of degree $d_1 + D$ in Y and a polynomial of degree $Dd_2 + 1$ in X.

To use the equations as index calculus relations, the algorithm of [18] selects the set of all unitary polynomials of degree at most D in X or Y, with coefficients in $\mathbb{F}_q$ as its smoothness basis and keeps pairs of polynomials (a, b) such that the two polynomials $a(Y)\,g_1(Y) + b(Y)$ and $a(g_2(X))\,X + b(g_2(X))$ both factor into terms of degree at most D. These good pairs are found using a classical sieving approach.

Writing $Q = q^n$, to analyze the complexity of the medium prime discrete logarithms, [18] chooses to write $q = L_Q(\frac{1}{3}, \alpha\, D)$, where as usual:

$$L_Q(\beta, c) = \exp((c + o(1))(\log Q)^{\beta}(\log\log Q)^{1-\beta}).$$

In this setting, the (heuristic) asymptotic complexity of the sieving phase is $L_Q(\frac{1}{3}, c_1)$ and the complexity of the linear algebra is $L_Q(\frac{1}{3}, c_2)$, with:

$$c_1 = \frac{2}{3\sqrt{\alpha D}} + \alpha D \quad \text{and} \quad c_2 = 2\alpha D.$$

Note that the algorithm with parameter D only works under the condition:

$$(D+1)\alpha \geq \frac{2}{3\sqrt{\alpha D}}. \tag{1}$$

Otherwise, the number of expected relations is too small to relate all elements of the smoothness basis. For a finite field $\mathbb{F}_{q^n}$, [18] indicates that the best complexity is obtained choosing the smallest acceptable value for the parameter D.

2.1 Individual Discrete Logarithms Phase

Another very important phase that appears in many index calculus based algorithms is the individual discrete logarithms phase which allows to compute the logarithm of an arbitrary field element by finding a multiplicative relation which relates this element to the elements of the smoothness basis whose logarithms have already been computed.

In [18], this is done by first expressing the desired element as a product of elements which can be represented as low degree polynomials in X or Y. These polynomials can in turn be related to polynomials of a lower degree and so on, until hitting degree one, i.e. elements of the smoothness basis. For this reason, the individual logarithm phase is also called the descent phase.

As analyzed in [18], the asymptotic complexity of the descent phase is

$$L_Q\left(\frac{1}{3}, \frac{1}{3\mu\sqrt{\alpha D}}\right),$$

where $\mu < 1$ is an arbitrary parameter. Moreover, any choice of μ in the interval $\left]\frac{1}{2}; 1\right[$ ensures that the complexity of the descent phase is asymptotically negligible compared to (at least one of) the main phases.

3 Pinpointing

3.1 Basic Framework

In order to improve the generation of relations, we first consider the simple case with parameter $D = 1$ and we construct our finite field extension using two polynomials that have the following restricted form:

$$\begin{aligned} X &= Y^{d_1} \quad \text{and} \\ Y &= g_2(X), \end{aligned}$$

where g_2 is a polynomial of degree d_2. To generate relations, since $D = 1$, we consider the space spanned by XY, X, Y and 1, i.e., after renormalization we are thus considering the following candidates:

$$Y^{d_1+1} + aY^{d_1} + bY + c = X\, g_2(X) + a\, X + b\, g_2(X) + c,$$

where a, b and c are arbitrary coefficients in $\mathbb{F}_q$. A candidate yields a valid multiplicative relation when both sides factor into linear polynomials.

We now use a simple trick and remark that the left-hand side $Y^{d_1+1} + aY^{d_1} + bY + c$ splits into linear terms, if and only if, $U^{d_1+1} + U^{d_1} + b\, a^{-d_1}\, U + c\, a^{-d_1-1}$ factors into linear terms. Indeed, the polynomial in U can be obtained from the polynomial in Y by performing the change of variable $Y = aU$ and dividing by a^{d_1+1}. As stated in the following theorem, this change does not affect the way the polynomial factors.

Theorem 1. *Let $f(Y)$ be a monic polynomial of degree D over $\mathbb{F}_q$ and let $g(U) = a^{-D} f(aU)$ with $a \in \mathbb{F}_q$. Write the factorization of f into monic irreducible polynomials as $f(Y) = \prod_{i=1}^{k} F_i(Y)^{e_i}$, then the factorization of g into monic irreducible factors is given by:*

$$g(U) = \prod_{i=1}^{k} \left(a^{-\deg F_i} F_i(aU)\right)^{e_i}.$$

Proof. It suffices to show that the image of an irreducible polynomial $I(Y)$ by the change of variable is also irreducible. Write $J(U) = a^{-\deg I} I(aU)$, if J is not irreducible, we have a non-trivial factorization $J(U) = J_1(U)J_2(U)$. Reversing the change of variable, we find that:

$$I(Y) = a^{\deg I} J(Y/a) = a^{\deg I} J_1(Y/a) J_2(Y/a).$$

Since I is irreducible, this would be a contradiction.
Thus J is irreducible and the theorem follows. □

3.2 One-Sided Pinpointing

Using the change of variable trick, we obtain a first form of pinpointing which only focuses on the Y side. This form searches for smooth polynomials in U of the form $U^{d_1+1} + U^{d_1} + B\, U + C$, with B and C in $\mathbb{F}_q$. This can be done either by directly testing candidates or by sieving. We need to consider approximately $(d_1 + 1)!$ candidates to find a good polynomial.

Once we have obtained one such smooth polynomial, we can amplify it (using a change of variable $U = Y/a$) into many polynomials $Y^{d_1+1} + aY^{d_1} + bY + c$, where a is an arbitrary non-zero element in $\mathbb{F}_q$, $b = Ba^{d_1}$ and $c = Ca^{d_1+1}$. This amortizes the cost of finding the initial polynomial, distributing this cost among many candidates. Indeed, we expect to obtain approximately $(q - 1)/(d_2 + 1)!$ relations by testing the right-hand sides corresponding to $q - 1$ different values of a. Adding to this the cost of finding the initial smooth polynomial, we find an amortized cost per relation close to:

$$\frac{(d_1+1)!+(q-1)}{(q-1)/(d_2+1)!} = \frac{(d_1+1)!\,(d_2+1)!}{q-1} + (d_2+1)!$$

This is clearly better than the cost of classical sieving which, in this case, amounts to $(d_2+1)!\,(d_1+1)!$ operations per relation. More precisely, this improves the cost of the relation by a factor of, at least, $\min(q-1,(d_1+1)!)/2$.

3.3 Kummer Extensions, Frobenius and Advanced Pinpointing

With some specific extension fields, it is possible to achieve an even better improvement over sieving, using a two-side approach to pinpointing. Moreover, this can be done while taking into account the action of Frobenius which allows us to reduce the size of the linear system.

We illustrate this using Kummer extensions of degree $n = d_1d_2 - 1$. We recall that a Kummer extension of degree n is defined over a finite field $\mathbb{F}_q$ which contains n-th roots of unity by a polynomial $P(X) = X^n - \kappa$, where κ has no root of prime order $m|n$ in $\mathbb{F}_q$. Let μ denote a primitive n-th root of unity in $\mathbb{F}_q$ and x denote an n-th root of κ in $\mathbb{F}_{q^n}$, then we have:

$$P(X) = \prod_{i=0}^{n-1}(X - \mu^i x).$$

As a consequence, there exists an i_0, prime to n such that $x^q = \mu^{i_0}x$. By changing our choice of primitive root μ, we can ensure that $i_0 = 1$. Thus, throughout the sequel, we have $x^q = \mu\, x$.

Such a Kummer extension can be obtained in our framework by defining:

$$\begin{aligned} X &= Y^{d_1}/\kappa \quad \text{and} \\ Y &= X^{d_2} \end{aligned} \tag{2}$$

Substituting one equation in the other, we find $X^{d_1d_2} - \kappa\, X = 0$. Thus dividing by X we obtain the desired Kummer extension. If x denotes as above the image of X in $\mathbb{F}_{q^n}$, the image of Y is $y = x^{d_2}$. Once again, since we are considering $D = 1$, our smoothness basis contains all the linear polynomials $x + a$ and $y + a$ with a in $\mathbb{F}_q$.

The Frobenius acts on the smoothness basis as follows:

$$\begin{aligned} (x+a)^q &= x^q + a = \mu\, x + a = \mu(x + a/\mu) \quad \text{and} \\ (y+a)^q &= y^q + a = \mu^{d_1}\, y + a = \mu^{d_1}(y + a/\mu^{d_1}). \end{aligned}$$

As a consequence, in the quotient group $\mathbb{F}_{q^n}^*/\mathbb{F}_q^*$, we have:

$$\begin{aligned} \log(x + a/\mu) &= q\log(x+a) \quad \text{and} \\ \log(y + a/\mu^{d_1}) &= q\log(y+a). \end{aligned}$$

These relations allow us to divide the number of unknowns in the linear system that we need to solve by a factor essentially equal to n. Indeed, all elements in the factor base except x and y have precisely n conjuguates (including themselves). Moreover, since $x^{n(q-1)} = 1$ and $y^{n(q-1)} = 1$, the logarithms of x and y are equal to 0 modulo any large prime dividing the order of the quotient group.

Advanced Pinpointing: Generating Equations in Kummer Extensions. As in the one-sided case, we consider the space of candidates generated by XY, X, Y and 1. Due to our specific choices, the renormalized candidates can be rewritten in a slightly simpler form:

$$\begin{aligned}&XY + aY + bX + c = \\ &X^{d_2+1} + aX^{d_2} + bX + c = Y^{d_1+1}/\kappa + b\,Y^{d_1}/\kappa + aY + c.\end{aligned}$$

We now remark that the polynomial on the X side splits, if and only if, $U^{d_2+1} + U^{d_2} + b\,a^{-d_2}\,U + c\,a^{-d_2-1}$ splits. Moreover, the polynomial on the Y side splits, if and only if, $V^{d_1+1}/\kappa + V^{d_1}/\kappa + a\,b^{-d_1}\,V + c\,b^{-d_1-1}$ splits.

Let $\lambda = c/(ab)$, then the polynomials in U and V can respectively be rewritten as:

$$U^{d_2+1} + U^{d_2} + b\,a^{-d_2}\,(U + \lambda) \quad \text{and} \quad (V^{d_1+1} + V^{d_1})/\kappa + a\,b^{-d_1}(V + \lambda).$$

Conversely, choose a triple (A, B, λ), with $A \neq 0$ and $B \neq 0$ and AB^{d_2} an n-th power in $\mathbb{F}_q$ such that:

$$U^{d_2+1} + U^{d_2} + A\,(U + \lambda) \quad \text{and} \quad (V^{d_1+1} + V^{d_1})/\kappa + B(V + \lambda)$$

both split. Then, we can recover a unique (up to Frobenius action) triple (a, b, c) corresponding to a candidate that yields an equation in the finite field. We first recover a and b. Putting together the two equations $A = ba^{-d_2}$ and $B = ab^{-d_1}$, we find $b^n = b^{d_1 d_2 - 1} = 1/(AB^{d_2})$. Since, by hypothesis, AB^{d_2} is an n-th power this equation has n distinct solutions. Choose one arbitrary solution for b, then we necessarily have $a = Bb^{d_1}$ and $c = \lambda ab$. We thus obtain a valid candidate (a, b, c). To show the unicity up to Frobenius action, we start from another solution $\mu^i b$ and obtain the triple $(\mu^{d_1 i}, \mu^i b, \mu^{(d_1+1)i} c)$. Now, let the Frobenius act j times on:

$$X^{d_2+1} + aX^{d_2} + bX + c$$

and renormalize to obtain:

$$X^{d_2+1} + a\mu^{-jd_2} X^{d_2} + b\mu^{-jd_1 d_2} X + c\mu^{-jd_1(d_2+1)}.$$

Since $d_1 d_2 \equiv 1 \pmod{n}$, we see that for $j \equiv -i \pmod{n}$, the action of Frobenius yields that same equation as the new choice for b.

Note. Once λ is fixed, finding the triples (A, B, λ) which satisfy the property that AB^{d_2} is an n-th power is a simple matter. Indeed, it suffices to partition the list of possible values for A and B in n sublists depending on the discrete logarithms of A (resp. B) modulo n. Since n is small, these values are easily computed by comparing $A^{(q^n-1)/n}$ (resp. $B^{(q^n-1)/n}$) with the possible n-th root of unity in $\mathbb{F}_Q$.

The d_1d_2+1 Variant. For a Kummer extension of degree n with $n = d_1d_2+1$, we can proceed in a very similar way defining the finite field by the relations:

$$\begin{aligned} X &= \kappa/Y^{d_1} \quad \text{and} \\ Y &= X^{d_2}, \end{aligned}$$

where κ again denotes a non n-th power. It is easy to adapt the action of Frobenius and the generation of equations to deal with this variant. See Section 5.2 for an example.

Further Generalization. We can also remark that the advanced form of pinpointing can also be used for some extension fields which are not Kummer extension. Indeed, when $d_1d_2 \pm 1$ does not divide the order of $\mathbb{F}_q$, choosing $X = Y^{d_1}/\kappa$ (resp. $X = \kappa/Y^{d_1}$) and $Y = X^{d_2}$ cannot define an extension of degree $d_1d_2 \pm 1$ because the polynomial $X^{d_1d_2} - \kappa X$ has two roots in $\mathbb{F}_q$. However, it can yield a extension of lower degree, depending on the factorization of the polynomial $X^{d_1d_2\pm1} - \kappa$ in $\mathbb{F}_q$. The main drawback compared to the case of Kummer extensions is that we can no longer use the action of Frobenius to reduce the size of the smoothness basis.

Cost Considerations. For each value of λ, creating the list of A-values costs $O(q)$ operations and the list contains about $(q-1)/(d_2+1)!$ elements. Similarly, the list of B-values costs $O(q)$ operations and contains approximately $(q-1)/(d_1+1)!$ elements. For a fixed λ, the total number of (A, B) pairs that yields a good triple (A, B, λ) is approximately:

$$\frac{(q-1)^2}{n(d_1+1)!(d_2+1)!}.$$

As a consequence, the average cost of constructing one relation is:

$$1 + O\left(\frac{n(d_1+1)!(d_2+1)!}{(q-1)}\right). \tag{3}$$

If we remember that the factor n in the second term is compensated by the fact that we only need q/n relations instead of q, we see that the other term is reduced from $(d_1+1)!$ to 1. As a consequence, the gain compared to sieving is at least $(q-1)/2$.

An interesting side-effect of this advanced pinpointing is that once the list of A and B values have been stored, the equations can be regenerated for a constant cost. This is interesting, because these lists are smaller than the list of equations. As a consequence, rather than storing the equations, it becomes preferable to recompute them on the fly whenever they are needed, thus saving disk space (and disk access time).

3.4 Complexity of Relation Construction Using Pinpointing

We first recall that the cost of sieving from [18]:

$$L_Q\left(\frac{1}{3}, \alpha + \frac{2}{3\sqrt{\alpha}}\right).$$

Moreover it is only applicable for $\alpha \geq 3^{-\frac{2}{3}}$.

Using One-Sided Pinpointing. As in [18], we now consider the complexity of computing discrete logarithms in a field $\mathbb{F}_Q$, with $Q = q^n$, assuming that the parameter α defined as:

$$\alpha = \frac{1}{n}\left(\frac{\log Q}{\log\log Q}\right)^{\frac{2}{3}}$$

is fixed. In this setting, we have $q = L_Q(\frac{1}{3}, \alpha)$. Since the smoothness basis has size $2q$, the cost of the linear algebra is the same as in [18], i.e., it is $L_Q(\frac{1}{3}, c_2)$ with $c_2 = 2\alpha$.

However, the complexity of collecting the relations is reduced compared to sieving. Indeed, the cost of collecting approximately $2q$ relations becomes:

$$2(d_1+1)!(q + (d_2+1)!).$$

Using the usual choice for d_1 and d_2, this can be written as:

$$L_Q\left(\frac{1}{3}, \frac{1}{3\sqrt{\alpha}} + \max\left(\alpha, \frac{1}{3\sqrt{\alpha}}\right)\right)$$

Note that this can be further improved by choosing the degrees d_1 and d_2 as follows:

$$d_1 \approx \frac{1}{3\alpha^2}\left(\frac{\log(Q)}{\log\log(Q)}\right)^{\frac{1}{3}} \quad \text{and} \quad d_2 \approx 3\alpha\left(\frac{\log(Q)}{\log\log(Q)}\right)^{\frac{1}{3}}.$$

For $\alpha \geq 3^{-\frac{2}{3}}$, this reduces the complexity to

$$L_Q\left(\frac{1}{3}, \alpha + \frac{1}{9\alpha^2}\right)$$

Using Advanced Pinpointing. To determine the asymptotic complexity of the advanced pinpointing method, we can ignore the action of Frobenius. Indeed, despite offering a very useful practical improvement, it does not provide an asympotic gain. The cost of collecting enough relations in this case is:

$$2(q + (d_1+1)!(d_2+1)!).$$

We choose:

$$d_1 \approx d_2 \approx \alpha^{-\frac{1}{2}}\left(\frac{\log(Q)}{\log\log(Q)}\right)^{\frac{1}{3}}$$

As a consequence, the cost of building the relations becomes:

$$L_Q\left(\frac{1}{3}, \max\left(\alpha, \frac{2}{3\sqrt{\alpha}}\right)\right).$$

Direct Access to Relations. When $\alpha \geq \frac{2}{3\sqrt{\alpha}}$, i.e. $\alpha \geq (2/3)^{\frac{2}{3}}$, the cost of building relations becomes equal to the number of relations. In other words, the right summand in equation (3) becomes negligible and each relation can be built in constant time. In this context, the pinpointing technique gives direct access to multiplicative relations. It is weird to note that, in this best case for pinpointing, there is no improvement on the full complexity, as shown in the next paragraph.

Impact on the Full Discrete Logarithm Complexity. In order to define the asymptotic complexity of the discrete logarithm computation for the algorithm with parameter $D = 1$, we also need to take into account the complexity of the linear algebra $L_Q(\frac{1}{3}, 2\alpha)$. For $\alpha \geq 3^{-\frac{2}{3}}$, this cost is higher than the cost of pinpointing in either version. As a consequence, in this range, the full complexity of discrete logarithm computation becomes $L_Q(\frac{1}{3}, 2\alpha)$. When α is in the interval $[3^{-\frac{2}{3}}; (2/3)^{\frac{2}{3}}[$, this is better than the algorithm of [18] whose cost is dominated by sieving. In particular, for $\alpha = 3^{-\frac{2}{3}}$, the cost is reduced from $L_Q(\frac{1}{3}, 3^{\frac{1}{3}}) \approx L_Q(\frac{1}{3}, 1.44)$ to $L_Q(\frac{1}{3}, (2/3)^{\frac{2}{3}}) \approx L_Q(\frac{1}{3}, 0.96)$.

4 Generalization to $D > 1$

The one-sided pinpointing technique presented above can be generalized to the case where $D > 1$ in a straightforward way. More precisely, it suffices to remark that a polynomial:

$$X^d + \sum_{i=0}^{d-1} a_i X^i,$$

can be decomposed into a product of polynomials of degree at most D, if and only if, the polynomial:

$$U^d + U^{d_1} + \sum_{i=0}^{d-2} a_i\, a_{d-1}^{d-i} U^i$$

can be decomposed into a product of polynomials of degree at most D.

As a consequence, we can essentially save a factor $q-1$ compared to a sieving approach if we use a pinpointing approach in this general case.

Resulting Complexity. As in [18], we consider the case where Equation (1) is satisfied. The amortized cost of constructing one relation is:

$$\frac{S_D(d_1+D)+(q-1)}{(q-1)/S_D(Dd_2+1)} = \frac{S_D(d_1+D)S_D(Dd_2+1)}{q-1} + S_D(Dd_2+1),$$

where $S_D(T)$ denotes the inverse of the probability for a degree T polynomial to decompose as a product of polynomials of degree at most D. We recall that $S_D(T) \approx \exp((T/D)\log T/D)$ (see [18,21]). As a consequence, the runtime of the relation collection is approximated by:

$$S_D(d_1+D)S_D(Dd_2+1)q^{D-1}+S_D(Dd_2+1)q^D.$$

For the usual choice, $d_1 \approx \sqrt{Dn}$ and $d_2 \approx \sqrt{n/D}$ and writing $q = L_Q(\frac{1}{3}, \alpha\, D)$ this becomes:

$$L_Q\left(\frac{1}{3}, D(D-1)\alpha + \frac{1}{3D\sqrt{\alpha}} + \max\left(\frac{1}{3D\sqrt{\alpha}}, D\alpha\right)\right).$$

Depending on the exact value of α, it can be re-optimized by changing the value of d_1 and d_2. When possible, the complexity becomes:

$$L_Q\left(\frac{1}{3}, D^2\alpha + \frac{1}{9D^2\alpha^2}\right).$$

To test whether re-optimization is possible, it suffices to compare the two complexities and keep the smaller.

4.1 Kummer Extensions with $D > 1$

In the case where $D > 1$, it is clear that using Kummer extensions allows us to account for the action of Frobenius, as in the $D = 1$ case. However, it is less clear that a dual-sided approach is also possible in this case. It turns out that the method used for $D = 1$ remains applicable. More precisely, define the relation between X and Y as in equation 2 and consider the space of candidates $\mathcal{A}(Y)\,X + \mathcal{B}(Y)$, where $\mathcal{A}$ and $\mathcal{B}$ are polynomials of degree D and $\mathcal{A}$ is unitary. We write $\mathcal{A}(Y) = Y^D + aY^{D-1} + \cdots$ and $\mathcal{B}(Y) = bY^D + cY^{D-1} + \cdots$.

The X-side is:

$$X^{Dd_2+1} + bX^{Dd_2} + aX^{(D-1)d_2+1} + cX^{(D-1)d_2} + \cdots$$

It splits, if and only if:

$$U^{Dd_2+1} + U^{Dd_2} + \frac{a}{b^{d_2}}\left(U^{(D-1)d_2+1} + \frac{c}{ab}U^{(D-1)d_2}\right) + \cdots$$

also splits. Similarly, the Y-side is:

$$Y^{d_1+D}/\kappa + aY^{d_1+D-1}/\kappa + \cdots + bY^D + cY^{(D-1)} + \cdots$$

It splits, if and only if:

$$V^{d_1+D}/\kappa + V^{d_1+D-1}/\kappa + \cdots + \frac{b}{a^{d_1}}\left(V^D + \frac{c}{ab}V^{(D-1)}\right) + \cdots$$

also splits. As a consequence, given $\lambda = c/(ab)$, $A = b/a^{d_1}$ and $B = a/b^{d_2}$ such that AB^{d_1} is an n-th power, we can transform all smooth polynomials in U and V into smooth polynomials in X and Y form with matching values for a, b and c. If the other coefficients also match, we obtain a relation.

However, due to the cost of matching extra coefficients, this is not as favorable as in the case $D = 1$.

5 Application: Two Discrete Logarithm Records

In order to demonstrate the practicality of our algorithm, we give a x-couple of new records for discrete logarithms in finite fields, in the particularly favorable case of Kummer extensions. More precisely, we decided to improve on the discrete logarithm record in $\mathbb{F}_{370\,801^{30}}$ presented in [17], using larger base fields and larger extension degrees.

To the best of our knowledge, the previous discrete logarithm record in a finite field concerned $\mathbb{F}_{3^{582}}$, a 923-bit field (see [13]). Our two results thus increases the size of the previous record by more than 500 bits. In order to illustrate the running time improvements gained from our new technique, we compare in the sequel our running times and the running times from [13]. However, we wish to warn the reader that this comparison should be analyzed with care. Indeed, the finite fields we have chosen are especially well-suited to our new techniques.

5.1 A Finite Field of Size 1175 Bits

For this example, we decided to consider an extension field $\mathbb{F}_{p^{47}}$ given by a Kummer extension of degree $47 = 8 \times 6 - 1$. We then chose $p = 33\,553\,771$, with $p - 1$ divisible by 47.

As a consequence, we can define the extension field using the relations $Y = X^6$ and $Y^8 = 2X$. This allows us to use advanced pinpointing and take advantage of the action of Frobenius. We obtain a smoothness basis of 1.43M elements. The cardinality of the finite field is:

$$p^{47} - 1 = 47 \cdot 2069 \cdot 12409 \cdot (p-1) \cdot 132103049403319 \cdot C,$$

where C is a 1073-bit composite cofactor of unknow factorization[2]

By construction, X has order $47(p-1)$ and thus cannot serve as a base for discrete logarithm. However, $X - 3$ is very likely to have order $p^{47} - 1$. Indeed, none of the values $(X-3)^{(p^{47}-1)/f}$ is equal to 1, when f is chosen as one of the known factors of $p^{47} - 1$. This choice is validated by our computation since we can find logarithms of random elements in basis $X - 3$.

As expected, the construction of the multiplicative relations is extremely efficient. For this reason, it was performed on a single laptop, using one CPU. We used advanced pinpointing. The preparatory construction of smooth-polynomials, for 1000 different values of λ, took a little more than 3 hours on the laptop. Once this was done, we performed the computation of the relations together with the structured Gaussian elimination, in 2 minutes. The resulting linear system contains 829 405 unknowns.

As expected, the computation is dominated by the linear algebra step. We performed this step using a block Wiedemann approach (as in [22]), based on 32 independent series of matrix-vector evaluation. Each run in the series was

[2] At the time of the computation, the factorization of C was unknown. Since then, William Hart [12] has found a 178-bit factor of C; the remaining cofactor is still composite.

performed on a 16-core[3] node of Genci's Curie computer, using OMP threads, thus using a total of 512 processors. The initial matrix-vector products required almost 37 hours. Due to memory requirement, the computation of a relation using block Wiedemann was done on 64 cores of a larger node[4] of Curie: it took 9h30min. Finally the recovery of the solution took 32 additional matrix-vectors products of half length compared to the initial runs. Due to the extra cost of combining the intermediate values using the coefficients in the relation, this required almost 25 hours. The grand total amounts to about 32 000 CPU-hours. We give a comparison of the timings with the previous record in Table 1.

Table 1. Comparison between our computations and the previous record

	Bitsize	Total time (CPU.h)	Relation construction (CPU.h)	Linear algebra (CPU.h)	Indiv. Log. (CPU.h)
[13]	923 bits	813 000	270 000	483 000	60 000
This paper	1175 bits	32 000	3	32 000	4
This paper	1425 bits	32 000	6	32 000	< 12

The reader can find some typical discrete logarithms of base elements in base $x-3$ modulo C in the eprint version of this paper [15] and in the announcement on the number theory mailing list [14]. They have been removed from this version to improve its compactness and lisibility.

Individual Discrete Logarithm. The computation of individual discrete logarithms is unchanged from [18] and requires a moderate amount of computing power. We illustrate this by computing the logarithm of:

$$Z = \sum_{i=0}^{46} \left(\lfloor \pi\, p^{i+1} \rfloor \bmod p\right) X^i.$$

The first step is to find a value related to Z which can be expressed using polynomials in X of relatively low degree. Here, we find that:

$$Z \cdot (X+1)^{359} = \frac{N}{D},$$

where N and D can be factored into irreducible polynomials of degree at most 8.

Once, this is done, we use the descent procedure to express each factor using polynomials of lower degree in X and Y. The slowest step in the descent is the final step that expresses polynomials of degree 2 using linear polynomials. After the earlier steps of the descent, we have a total of 278 degree polynomials whose logarithms are required (156 in X and 122 in Y). In the final step, we consider

[3] More precisely, it was on Curie's thin nodes: each node contains two octocore Intel Sandy Bridge EP (E5-2680) processors at 2.7 GHz.

[4] Here, we used half of a Curie's xlarge node, i.e. eight octocore Intel Nehalem-EX X7560 processors at 2.26 GHz .

all polynomials of the form $XY + aY + bX + c$ that are multiples of the target polynomial and use sieving to find a relation between this target and linear polynomials. When not possible, we use a relation that also includes another degree 2 polynomial and restart from that polynomial. The total time to obtain all these logarithms on the laptop used for computing the relations is under 4 hours.

Finally, back-substituting all the logarithms of the linear polynomials, we derive the logarithm of Z modulo C. To ease verification, we have also computed this logarithm modulo the small factors and thus give its complete value. We have:

$\log(Z) =$

3566331271464940662632811347409494405717808078782395308309921125231404942775893475045554815091157495604731476318649637458779492102525688657986426490390470334620506275228133179370846621472279947563764521646088983036872873337915243309378992279523113002528828381737389659610454461801405732402316469144478992620991524885344807375680493337120881974709130541

82

5.2 A Finite Field of Size 1425 Bits

For this example, we decided to consider an extension field $\mathbb{F}_{p^{57}}$ given by a Kummer extension of degree $57 = 8 \times 7 + 1$. We then chose p to make sure that $p - 1$ is divisible by 57 and that $p^{57} - 1$ is easy to factor. Thus, we considered $\mathbb{F}_{p^{57}}$, with $p = 33\,341\,353$.

This allows us to define the extension field using the relations: $Y = X^7$ and $X = 2/Y^8$. This illustrates the $d_1 d_2 + 1$ variant of our technique on Kummer extension. The initial smoothness basis contains 1.17M elements. The cardinality of the finite field is:

$$\begin{aligned} p^{57} - 1 &= (p-1) \cdot (p^2 + p + 1) \cdot 19 \cdot p_1 \cdot p_2 \quad \text{where} \\ p_1 &= \Big(\sum_{i=0}^{18} p^i\Big)/19 \quad \text{and} \\ p_2 &= \sum_{i=0}^{12} p^{3i} - (p + p^{20}) \sum_{i=0}^{5} p^{3i}. \end{aligned}$$

The two primes p_1 and p_2 respectively have 446 and 900 bits.

Since, X has order $57(p-1)$, we use $X - 11$ as our basis for discrete logarithms. The construction of the multiplicative relations was performed on the same laptop as previously indicated. For the preparatory construction of smooth-polynomials, we used 2000 different values of λ, which took 6 hours on the laptop. Once this is done, we performed the computation of the relations together with the structured Gaussian elimination, in 2 minutes. The resulting linear system contains 714 931 unknowns.

Once again, the computation is dominated by the linear algebra step. This time, we split the computation into two independent parts, adressing p_1 and p_2 separately. For each of the two primes, the initial matrix-vector products required 18 hours and 30 minutes, using a 16-core node for each prime. The block Wiedemann step required 2h30m for p_1 and 6h10m for p_2, using 64 cores for each computation. The final run of matrix-vectors products took 12 hours for each prime. Once again, the grand total amounts to about 32 000 CPU-hours.

5.3 Individual Discrete Logarithm

The computation of individual discrete logarithms works as previously. However, for performance reasons, we reimplemented the descent procedure in C, instead of using a mix of PARI/GP scripts and C code as before. The computing power required for an individual logarithm remains moderate and could be parallelized if required. We illustrate this by computing the logarithm of:

$$Z = \sum_{i=0}^{56} \left(\lfloor \pi \, p^{i+1} \rfloor \bmod p \right) X^i.$$

We have:

$$Z \cdot (X-11)^{2859} = \frac{N}{D},$$

where N and D can be factored into irreducible polynomials of degree at most 10.

Once, this is done, we use the descent procedure to express each factor using polynomials of lower degree in X and Y. The slowest step in the descent is again the final step that expresses polynomials of degree 2 using linear polynomials. The total time to obtain all the logarithms on the laptop used for computing the relations is 11h20m hours.

Finally, back-substituting all the logarithms of the linear polynomials (see [16] for some example values), we derive the logarithm of Z modulo $p_1 p_2$. To ease verification, we have also computed this logarithm modulo the small factors and thus give its complete value. We have:

$\log(Z) =$

386967279548486723402519963435606166899215654120310832592175430644490314
474088839541268684766235143037749947353744120837921318939397547163151 74
248440299271293657607241850991250364535044122994973576012005246534842 97
578176879047978194029063396672957652694830528789608330411939696620270 00
582282674552286146825678667645600249361054829752906320008220524565954 22
7246144528633360702659845991018671162540834330782804384739924956552212 0202

6 Conclusion and Open Problems

In this paper, we have shown a new technique to replace sieving in some index calculus algorithms. This technique can be applied whenever the target discrete logarithm group is a finite field that contains a subfield of the right size. We have illustrated it with some new discrete logarithms records. Since we only know how to use this technique in the medium prime case of the function field sieve, it leaves open the problem of generalizing the approach to other index calculus algorithms. The natural targets are the function field sieve without a medium-size subfield and the number field sieve, either for factoring or computing discrete logarithms. It should be noted that the result presented in this paper was in fact inspired by the cubic sieve introduced in [3] (see [5] for more details) which can be seen as its remote precursor and also offers a partial answer to the question of pinpointing in the case of number field sieve. However, generalizing to the general formulation of the number field sieve seems to be a difficult problem.

Another open problem is to adapt our construction to take advantage of the action of Frobenius regardeless of the extension degree. In particular, it would be convenient to make it compatible with the Galois invariant smoothness approach proposed in [4].

Acknowledgements. The author would like to thank the organizers of WMC and SCC 2012 for setting up such a nice and inspiring workshop.

We acknowledge that the results in this paper have been achieved using the PRACE Research Infrastructure resource Curie based in France at TGCC, Bruyères-le-Chatel (project number 2011050868). Some preliminary computations related to this work were performed on the PRACE Research Infrastructure resource Jugene/Juqueen based in Germany at the Jülich Supercomputing Centre (project number 2011050868).

References

1. Adleman, L.M., Huang, M.-D.A.: Function field sieve method for discrete logarithms over finite fields. In: Information and Computation, vol. 151, pp. 5–16. Academic Press (1999)
2. Coppersmith, D.: Fast evaluation of logarithms in fields of characteristic two. IEEE Transactions on Information Theory IT-30(4), 587–594 (1984)
3. Coppersmith, D., Odlyzko, A.M., Schroeppel, R.: Discrete logarithms in GF(p). Algorithmica 1(1), 1–15 (1986)
4. Couveignes, J.-M., Lercier, R.: Galois invariant smoothness basis. In: Hirschfeld, J., Chaumine, J., Rolland, R. (eds.) Algebraic Geometry and its Applications, Proceedings of the First SAGA Conference, May 7-11. Number Theory and Its Applications, vol. 5, pp. 142–167. World Scientific, Papeete (2007)
5. Das, A., Veni Madhavan, C.E.: On the cubic sieve method for computing discrete logarithms over prime fields. Int. J. Comput. Math. 82(12), 1481–1495 (2005)
6. Diem, C.: The GHS attack in odd characteristic. J. Ramanujan Math. Soc. 18(1), 1–32 (2003)
7. Gaudry, P.: An algorithm for solving the discrete log problem on hyperelliptic curves. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 19–34. Springer, Heidelberg (2000)
8. Gaudry, P.: Index calculus for abelian varieties of small dimension and the elliptic curve discrete logarithm problem. J. Symbolic Computation (2008)

9. Gaudry, P., Hess, F., Smart, N.P.: Constructive and destructive facets of Weil descent on elliptic curves. J. Cryptology 15(1), 19–46 (2002)
10. Gaudry, P., Thomé, E., Thériault, N., Diem, C.: A double large prime variation for small genus hyperelliptic index calculus. Mathematics of Computation 76, 475–492 (2007)
11. Gordon, D.M.: Discrete logarithms in GF(p) using the number field sieve. SIAM J. Discrete Math. 6(1), 124–138 (1993)
12. Hart, W.: Re: Discrete logarithms in a 1175-bit finite field. NMBRTHRY list (January 2013)
13. Hayashi, T., Shimoyama, T., Shinohara, N., Takagi, T.: Breaking pairing-based cryptosystems using η_T pairing over $gF(3^{97})$. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 43–60. Springer, Heidelberg (2012)
14. Joux, A.: Discrete logarithms in a 1175-bit finite field. NMBRTHRY list (December 2012)
15. Joux, A.: Faster index calculus for the medium prime case. Application to 1175-bit and 1425-bit finite fields. Cryptology ePrint Archive, Report 2012/720 (2012)
16. Joux, A.: Discrete logarithms in a 1425-bit finite field. NMBRTHRY list (January 2013)
17. Joux, A., Lercier, R.: Discrete logarithms in GF(370 801^{30}). NMBRTHRY list (November 2005)
18. Joux, A., Lercier, R.: The function field sieve in the medium prime case. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 254–270. Springer, Heidelberg (2006)
19. Arjen, K., Lenstra Jr., H.W. (eds.): The development of the number field sieve. Lecture Notes in Mathematics, vol. 1554. Springer, Heidelberg (1993)
20. Murphy, B.A.: Polynomial selection for the number field sieve integer factorisation algorithm. PhD thesis, Australian national university (1999)
21. Panario, D., Gourdon, X., Flajolet, P.: An Analytic Approach to Smooth Polynomials over Finite Fields. In: Buhler, J.P. (ed.) ANTS 1998. LNCS, vol. 1423, pp. 226–236. Springer, Heidelberg (1998)
22. Thomé, E.: Subquadratic computation of vector generating polynomials and improvement of the block wiedemann algorithm. J. Symb. Comput. 33(5), 757–775 (2002)

Fast Cryptography in Genus 2

Joppe W. Bos[1], Craig Costello[1,⋆], Huseyin Hisil[2], and Kristin Lauter[1]

[1] Microsoft Research, Redmond, USA
[2] Yasar University, Izmir, Turkey

Abstract. In this paper we highlight the benefits of using genus 2 curves in public-key cryptography. Compared to the standardized genus 1 curves, or elliptic curves, arithmetic on genus 2 curves is typically more involved but allows us to work with moduli of half the size. We give a taxonomy of the best known techniques to realize genus 2 based cryptography, which includes fast formulas on the Kummer surface and efficient 4-dimensional GLV decompositions. By studying different modular arithmetic approaches on these curves, we present a range of genus 2 implementations. On a single core of an Intel Core i7-3520M (Ivy Bridge), our implementation on the Kummer surface breaks the 120 thousand cycle barrier which sets a new software speed record at the 128-bit security level for constant-time scalar multiplications compared to all previous genus 1 and genus 2 implementations.

1 Introduction

Since its invention in the 1980's, elliptic curve cryptography [36,42] has become a popular and standardized approach to instantiate public-key cryptography. The use of elliptic curves, or *genus* 1 *curves*, has been well studied and consequently all of the speed records for fast curve-based cryptography are for elliptic curves (cf. the ECRYPT online benchmarking tool eBACS [8]). Jacobians of hyperelliptic curves of high genus have also been considered for cryptographic purposes, but for large genus there are "faster-than-generic" attacks on the discrete logarithm problem [2,24,20,18]. Such attacks are not known, however, for *genus* 2 *curves*. In [26], Gaudry showed that scalar multiplication on the Kummer surface associated with the Jacobian of a genus 2 curve can be more efficient than scalar multiplication on the Jacobian itself. Thus, it was proposed (cf. [5]) that hyperelliptic curve cryptography in genus 2 has the potential to be competitive with its genus 1 elliptic curve cryptography counterpart. One significant hurdle for genus 2 cryptography to overcome is the difficulty of generating secure genus 2 curves: that is, such that the Jacobian has a large prime or almost prime group order. In particular, for fast cryptographic implementations it is advantageous to work over special prime fields, where the underlying field arithmetic is fast, and to generate curves over those fields with suitable group orders. A major

⋆ Part of this work was done while the second author was working in the Department of Mathematics and Computer Science at the Technische Universiteit Eindhoven, Netherlands.

catalyst for this work is that genus 2 point counting methods and complex multiplication (CM) methods for constructing genus 2 curves with a known group order have become more practical. Hence, the time is ripe to give a taxonomy and a cross-comparison of all of the best known techniques for genus 2 curves over prime fields. The focus on prime fields is motivated by the recommendations made by the United States' National Security Agency Suite B of Cryptographic Protocols [46].

In this paper we set new performance speed records at the 128-bit security level using genus 2 hyperelliptic curves. For instance, using the Kummer surface given by Gaudry and Schost [29], we present the fastest curve based scalar multiplication over prime fields to date — this improves on the recent prime field record for elliptic curves from Longa and Sica which was presented at Asiacrypt 2012 [40]. As an additional bonus, our implementations on the Kummer surface inherently run in constant-time, which is one of the major steps towards achieving a side-channel resistant implementation [37]. Thus, we present the fastest constant-time software for curve based cryptography compared to *all* prior implementations.

Another advantage for genus 2 curves is that the endomorphism ring is larger than for genus 1 curves, so higher dimensional scalar decomposition is possible without passing to an extension field [23,22]. For prime fields we implement 4-dimensional GLV decompositions on Buhler-Koblitz (BK) curves [14] and on Furukawa-Kawazoe-Takahashi (FKT) curves [21], both of which are faster than all prior eBACS-documented implementations. To optimize overall performance, we present implementations based on two different methods that allow fast modular arithmetic: one based on the special form of the prime using "NIST-like" reduction [52] and another based on the special form of the prime when using Montgomery multiplication [43].

In addition, we put forward a multi-faceted case for (a special class of) Buhler-Koblitz curves of the form $y^2 = x^5 + b$. The curves we propose are particularly flexible in applications because they facilitate both a Kummer surface implementation and a GLV decomposition. Thus, a simple Diffie-Hellman style key exchange can be instantiated using the fast formulas on the Kummer surface, but if a more complicated protocol requires further group operations, one has the option to instead exploit a 4-dimensional GLV implementation using the same curve.

We refer to the full-version of this paper [10] for the specifications of all curves used and more detailed information.

2 Preliminaries

In this section we recall some basic facts and notation concerning genus 2 curves and briefly review the main techniques used to compute scalar multiplications.

Genus-2 Curves. A hyperelliptic genus 2 curve over a field of odd characteristic K can be defined by an affine model $C : y^2 = f(x)$, where $f(x)$ has degree 5 or 6 and has no double roots. We call C a *real* hyperelliptic curve if the degree of

f is 6, and if such an $f(x)$ has a rational root in K, then we can birationally transform the curve so that f has degree 5 instead, in which case we say C is an *imaginary* hyperelliptic curve. Arithmetic is currently slightly faster in the imaginary case.

Unlike genus 1 elliptic curves, in genus 2, the points on the curve do not form a group. Roughly speaking, unordered pairs of points on the curve form a group, where the group operation adds two pairs of points by passing a cubic through the four points, finding the other two points of intersection with the curve, and then reflecting them over the x-axis. More formally, we denote this group by $\mathrm{Jac}(C)$, the Jacobian of C, which consists of degree zero divisors on the curve modulo principal divisors. Throughout this paper we use the *Mumford representation* of general divisors $D = (x^2 + u_1x + u_0, y - (v_1x + v_0)) \in \mathrm{Jac}(C)$, and instead write $D = (u_1, u_0, v_1, v_0)$. This avoids confusion when x and y are used as two of the Kummer coordinates in Section 5. When working in homogeneous projective space, we write such divisors as $D = (U_1 : U_0 : V_1 : V_0 : Z)$, where $u_i = U_i/Z$ and $v_i = V_i/Z$ for $i \in \{0, 1\}$ and $Z \neq 0$.

Scalar Multiplication. There are many different ways to compute the scalar multiplication. Most approaches, like the double-and-add algorithm, are based on *addition chains* [50] and a typical optimization to lower the number of point additions is using *windows* [12] of a certain width $w > 1$. Given the input point P, we compute a lookup table consisting of the multiples $[c]P$ such that $0 \leq c < 2^w$, and perform a point addition once every w bits (instead of at most once per bit). After adding a precomputed multiple, we can "slide" to the next set-bit in the binary representation of the scalar; such *sliding windows* [55] lower the number of point additions required and halve the size of the lookup table since only the odd multiples of P are required. When computing the negation of a group element is inexpensive, which is the case for both elliptic and genus 2 curves, we can either add or subtract the precomputed point[1], reducing the total number of group operations even further; this is called the *signed windows* approach [45]. See [7] for a summary of these techniques.

Adding an affine point to a projective point to obtain another projective point, often referred to as mixed addition, is usually faster than adding two projective points. In order to use these faster formulas, a common approach is to convert the precomputed projective points into their affine form. This requires an inversion for each point in the table. Using Montgomery's *simultaneous inversion* method [44], I independent inversions can be replaced by $3(I-1)$ multiplications and a single inversion, which is typically much faster.

3 Fast Modular Arithmetic Using Special Primes

When performing arithmetic modulo a prime p in practice, it is common to use primes of a special form since this may allow fast reduction. For instance, in

[1] The term 'point' becomes 'divisor' in the case of hyperelliptic curves, but remains as 'point' for Kummer surface arithmetic in Section 5.

the FIPS 186-3 standard [56], NIST recommends the use of five prime fields when using the elliptic curve digital signature algorithm (but see also [4]). Such special primes have been studied from both a theoretical and practical point of view. A study of a software implementation of the NIST-recommended elliptic curves over prime fields on the x86 architecture is given by Brown et al. [13], and in [9] a comparison is made between the performance when using Montgomery multiplication [43] and specialized multiplication using the NIST primes. In this section we describe two different approaches to obtain fast modular arithmetic. We use the prime $p_{1271} = 2^{127} - 1$ to illustrate both methods, since this prime is used in some of our implementations (cf. Section 4 and Section 5).

Generalized Mersenne Primes. Primes that enable fast reduction techniques are usually of the form $2^s \pm \delta$, where $s, \delta \in \mathbb{Z}^+$, and $\delta \ll 2^s$. The constant δ is also small compared to the word-size of the target architecture, which is typically 32 or 64 bits. Another popular choice is using a generalized Mersenne prime of the form $2^s + \sum_{i \in S} i$, where S is a set of integers $\pm 2^j$ such that $|2^j| < 2^s$ and the cardinality of S is small. For example, fast reduction modulo $p = 2^s - \delta$ can be done as follows. For integers $0 \le a, b, c_h, c_\ell, \delta < 2^s$, write $c = a \cdot b = c_h \cdot 2^s + c_\ell \equiv c_\ell + \delta c_h \pmod{2^s - \delta}$ where $0 \le c_\ell + \delta c_h < (\delta+1)2^s$. At the cost of a multiplication by δ (which might be a shift depending on the form of δ) and an addition, compute $c' \equiv c \pmod{p}$ where c' is (much) smaller than c, depending on the size of δ. This is the basic idea behind Solinas' reduction scheme [52], which is used to implement fast arithmetic modulo the NIST primes [56]. We refer to this type of reduction as *NIST-like reduction*. When computing $a \cdot b \bmod p_{1271}$ with $0 \le a, b < p_{1271}$, one can first compute the multiplication $c = a \cdot b = c_1 \cdot 2^{128} + c_0$, where $0 \le c_1, c_0 < 2^{128}$. A first reduction step can be computed as $c' = (c_0 \bmod 2^{127}) + 2 \cdot c_1 + \lfloor c_0 / 2^{127} \rfloor \equiv c \pmod{p_{1271}}$ such that $0 \le c' < 2^{128}$. One can then reduce c' further using conditional subtractions. Modular reduction in the case of p_{1271} can therefore be computed without using any multiplications.

Montgomery-Friendly Primes. Montgomery multiplication [43] involves transforming each of the operands into their Montgomery representations and replacing the conventional modular multiplications by Montgomery multiplications. One of the advantages of this method is that the computational complexity is usually better than the classical method by a constant factor.

Let $r = 2^b$ be the radix of the system and $b > 2$ be the bit-length of a word. Let p be an n-word odd prime such that $r^{n-1} \le p < r^n$, and suppose we have an integer $0 \le X < p$. The Montgomery radix $R = r^n$ is a fixed integer such that $\gcd(R, p) = 1$. The Montgomery residue of X is defined as $\widetilde{X} = X \cdot R \bmod p$. The Montgomery product of two integers is defined as $M(\widetilde{X}, \widetilde{Y}) = \widetilde{X} \cdot \widetilde{Y} \cdot R^{-1} \bmod p$. Practical instances of Montgomery multiplication use the precomputed value $\mu = -p^{-1} \bmod r$. The interleaved Montgomery multiplication algorithm, in which multiplication and reduction are combined, computes $C = M(A, B)$ for $0 \le A, B < p$. Let $A = \sum_{i=0}^{n-1} a_i \cdot r^i$, where $0 \le a_i < r$, and start with $C = 0$. For all $i \in \mathbb{Z}$ such that $0 \le i < n$, the result C is updated as

$$C \leftarrow C + a_i \cdot B, \quad C \leftarrow \Big(C + ((\mu \cdot C) \bmod r) \cdot p\Big)\Big/ r.$$

The division by r can be implemented by a shift since the precomputed value μ ensures that the least significant digit (b bits) of $(C+((\mu\cdot C) \bmod r)\cdot p)$ is zero. It can be shown that the final Montgomery product C is bounded as $0 \leq C < 2\cdot p$, and therefore a final conditional subtraction is needed when complete reduction is required. In order to avoid handling additional carries in the Montgomery multiplication, which requires more instructions, our implementations prefer 127-bit moduli over 128-bit moduli. In [39] it is noticed that fixing part of the modulus can have advantages for Montgomery multiplication. For instance, the precomputation of μ can be avoided when $-p^{-1} \equiv \pm 1 \pmod r$, which also avoids computing a multiplication by μ for every iteration inside the Montgomery multiplication routine. This technique has been suggested in [35,1,31] as well. When μ is small, e.g. $\mu = \pm 1$, one could lower the cost of the multiplication of p with $(\mu \cdot c_0) \bmod r$ by choosing the $n-1$ most significant words of p in a similar fashion as for the generalized Mersenne primes: $\lfloor p/2^b \rfloor = 2^s + \sum_{i\in S} i$.

Consider the prime p_{1271} on 64-bit architectures: $r = 2^{64}$ and we have $\mu = -p_{1271}^{-1} \bmod 2^{64} = 1$, so that the multiplication by μ can be avoided. Write $C = c_2 \cdot 2^{128} + c_1 \cdot 2^{64} + c_0$ with $0 \leq c_2, c_1, c_0 < 2^{64}$. Due to the shape of the most-significant word of $p_{1271} = (2^{63}-1)\cdot 2^{64} + (2^{64}-1)$, the result of $\frac{C+((\mu\cdot C) \bmod r)\cdot p}{r}$ can be obtained using only two shift and two 64-bit addition instructions by computing $c_2\cdot 2^{64}+c_0\cdot 2^{63}+c_1$. Similar to the NIST-like reduction, Montgomery reduction in the setting of p_{1271} can be computed without using any multiplications.

Modular Inversion. When using the regular representation of integers, one can either use the (binary) extended GCD algorithm to compute the modular inversion or use the special form of the modulus to compute the inverse by using modular exponentiations. For instance, in the case of p_{1271}, one can exploit the congruence $a^{2^{127}-2} \equiv a^{-1} \pmod{p_{1271}}$. The situation when working in Montgomery form is slightly different. Given the Montgomery form $\tilde{a} = a2^{bn} \bmod p$ of an integer a, we want to compute the Montgomery inverse $\tilde{a}^{-1}2^{2bn} \equiv a^{-1}2^{bn}$ (mod p). This would require a classical inversion and modular multiplication, however we found that the approach presented in [11] (which uses the binary version of the Euclidean algorithm from [33]) is faster in practice. The first step of this approach computes a value $\tilde{a}^{-1}2^k \equiv a^{-1}2^{k-bn} \pmod p$, for some $0 \leq k < 2bn$. This value is then corrected via a Montgomery multiplication with 2^{3bn-k}. This last multiplication typically requires a lookup table with the different precomputed values $2^{3rn-k} \bmod p$. In the case of $p = 2^{127}-1$, one can avoid this lookup table since $2^t \bmod 2^{127}-1 = 2^{t \bmod 127}$.

Modular Addition/Subtraction. Let $0 \leq a, b < 2^k - c$. We compute $(a + b) \bmod (2^k - c)$ as $((((a+c)+b) \bmod 2^k)-c\cdot(1-\texttt{carry}((a+c)+b, 2^k))) \bmod 2^k$. The carry function $\texttt{carry}(x, y)$ returns either zero or one if $x < y$ or $x \geq y$ respectively. The output is correct and bounded by $2^k - c$, since if $a+b+c < 2^k$, then $a+b < 2^k-c$, while if $a+b+c \geq 2^k$, then $(a+b+c) \bmod 2^k = a+b-(2^k-c) < 2^k - c$. Note that since $a + c < 2^k$, the addition requires no carry propagation. Furthermore, c is multiplied with either one or zero such that this multiplication amounts to data movement.

The modular subtraction $(a - b) \bmod (2^k - c)$ is performed by computing $(((a - b) \bmod 2^k) - c \cdot \texttt{borrow}(a - b)) \bmod 2^k$. Analogous to the carry function, the borrow function $\texttt{borrow}(x)$ returns zero or one if $x \geq 0$ or $x < 0$ respectively. If $a < b$, then $0 \leq (a - b) \bmod 2^k - c = a - b + (2^k - c) < 2^k - c$, and if $a \geq b$, then $0 \leq a - b < 2^k - c$. In some scenarios one can compute additions as $(((a + b) \bmod 2^k) + c \cdot \texttt{carry}((a + b), 2^k)) \bmod 2^k$, but we note that here the output may not be completely reduced and can be $\geq 2^k - c$.

4 "Generic" Genus-2 Curves and Their Arithmetic

To give a concrete idea of the advantage gained when working on the Kummer surface or when exploiting GLV endomorphisms, we also consider the generic scenario that employs neither technique. We make use of the fast formulas for arithmetic on imaginary quadratic curves from [17], which employ homogeneous projective coordinates, and focus on reducing the total number of multiplications in projective point doublings, point additions and mixed additions.[2]

We assume that our curves are of the form $C : y^2 = x^5 + f_3x^3 + f_2x^2 + f_1x + f_0$, and count multiplications by the f_i as full multiplications, unless they are zero.[3] Letting $\mathbf{m}$, $\mathbf{s}$ and $\mathbf{a}$ be the cost of $\mathbb{F}_p$-multiplications, $\mathbb{F}_p$-squarings and $\mathbb{F}_p$-additions or subtractions respectively, we summarize the modified counts as follows. For $D = (U_1 : U_0 : V_1 : V_0 : Z)$, one can compute $[2]D$ in $34\mathbf{m} + 6\mathbf{s} + 34\mathbf{a}$. For the special GLV curves in Section 6, which have $f_2 = f_3 = 0$, the projective doubling can be computed using $32\mathbf{m} + 6\mathbf{s} + 32\mathbf{a}$. For $D = (U_1 : U_0 : V_1 : V_0 : Z)$ and $D' = (U_1' : U_0' : V_1' : V_0' : Z')$, one can compute the projective addition $D + D'$ in $44\mathbf{m} + 4\mathbf{s} + 29\mathbf{a}$. For the mixed addition between the projective point $D = (U_1 : U_0 : V_1 : V_0 : Z)$ and the affine point $D' = (u_1' : u_0' : v_1' : v_0')$, one can compute the projective result $D + D'$ in $37\mathbf{m} + 5\mathbf{s} + 29\mathbf{a}$. Full and mixed additions cost the same on the special GLV curves. Given these operation counts, our "generic" implementations performed fastest when using 4-bit signed sliding windows (see Section 2).

5 The Kummer Surface

Gaudry [26] built on earlier observations by Chudnovsky and Chudnovsky [15] to show that scalar multiplication in genus 2 can be greatly accelerated by working on the Kummer surface associated to a Jacobian, rather than on the Jacobian itself. Although the Kummer surface is not technically a group, it is close enough to a group to be able to define scalar multiplications on it, and is therefore an attractive setting for Diffie-Hellman like protocols that do not require any further group operations [51].

[2] Note that the formulas to compute the projective doubling from [17] can be sped up since the first multiplication to compute UU is redundant.

[3] Over prime fields it is standard to zero the coefficient of the x^4 term via an appropriate substitution.

The Squares-only Kummer Routine. The Kummer surface that was originally proposed for cryptography in [26] is a surface whose constants are parameterized by the four *fundamental Theta constants* $(\vartheta_1(0), \vartheta_2(0), \vartheta_3(0), \vartheta_4(0))$, and whose coordinates come from the four *fundamental Theta functions* $(\vartheta_1(\mathbf{z}), \vartheta_2(\mathbf{z}), \vartheta_3(\mathbf{z}), \vartheta_4(\mathbf{z}))$, all of which are values of the classical genus 2 *Riemann Theta function.* Bernstein [5] pointed out that one can work entirely with the squares of the fundamental Theta constants without any loss of efficiency. This provides more flexibility when transforming a given genus 2 curve into an associated Kummer surface, and makes it easier to control the size of squared fundamental Theta constants, for which small values can give worthwhile speedups.

Cosset [16] formally presented the "squares-only" setting, in which the Kummer surface $\mathcal{K}$ is completely defined by the *squared fundamentals* $(a^2, b^2, c^2, d^2) = (\vartheta_1(0)^2, \vartheta_2(0)^2, \vartheta_3(0)^2, \vartheta_4(0)^2)$ as

$$\mathcal{K}\colon \quad E'xyzt = ((x^2+y^2+z^2+t^2) - F(xt+yz) - G(xz+yt) - H(xy+zt))^2,$$

$$\text{where} \quad E' = 4E^2a^2b^2c^2d^2, \quad E = \frac{ABCD}{(a^2d^2-b^2c^2)(a^2c^2-b^2d^2)(a^2b^2-c^2d^2)},$$

$$F = \frac{a^4-b^4-c^4+d^4}{a^2d^2-b^2c^2}, \quad G = \frac{a^4-b^4+c^4-d^4}{a^2c^2-b^2d^2}, \quad H = \frac{a^4+b^4-c^4-d^4}{a^2b^2-c^2d^2},$$

$$\begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} a^2 \\ b^2 \\ c^2 \\ d^2 \end{bmatrix}. \tag{1}$$

We write $(x\colon y\colon z\colon t) = (\vartheta_1(\mathbf{z})^2\colon \vartheta_2(\mathbf{z})^2\colon \vartheta_3(\mathbf{z})^2\colon \vartheta_4(\mathbf{z})^2)$ for the coordinates of a projective point on $\mathcal{K}$.

Extracting the Squared Kummer Surface Parameters. In [26] Gaudry showed the relationship between the Kummer surface and the isomorphic Rosenhain model of the genus 2 curve C, given as

$$C_{\mathrm{Ros}}\colon y^2 = x(x-1)(x-\lambda)(x-\mu)(x-\nu), \tag{2}$$

where the Rosenhain invariants λ, μ and ν are linked to the squared fundamentals by

$$\lambda = \frac{a^2c^2}{b^2d^2}, \ \mu = \frac{c^2(AB+CD)}{d^2(AB-CD)}, \ \nu = \frac{a^2(AB+CD)}{b^2(AB-CD)},$$

with A, B, C, D as in (1). Since the three Rosenhain invariants are functions of the four squared fundamentals, there is a degree of freedom when inverting the equations to compute (a^2, b^2, c^2, d^2) from (λ, μ, ν). Thus, we can set $d^2 = 1$ [27] and compute the other squared fundamentals as

$$c^2 = \sqrt{\frac{\lambda\mu}{\nu}}, \qquad b^2 = \sqrt{\frac{\mu(\mu-1)(\lambda-\nu)}{\nu(\nu-1)(\lambda-\mu)}}, \qquad a^2 = b^2c^2\frac{\nu}{\mu}.$$

Given a hyperelliptic curve C of genus 2, there are up to 120 unique Rosenhain triples λ, μ, ν that give an isomorphic representation $C_{\mathrm{Ros}} \cong C$ over the algebraic closure [25, §2.2]. So for a given curve with rational 2-torsion, we can expect that there may be at least one Rosenhain triple for which the square roots above lie in the same field as λ, μ and ν, such that the Kummer surface is also defined over the same field (but see Section 8). If the 2-torsion is rational, then 16 must divide the cardinality of $\text{Jac}(C)$ [26].

Twist Security. There is an additional security consideration when working on the Kummer surface because a random point on $\mathcal{K}$ can map to either the curve $C_{\text{Ros}} \cong C$ or its twist $C'_{\text{Ros}} \cong C'$ [26, §5.2]. As long as the public generator $P \in \mathcal{K}$ is chosen so that it maps back to $\text{Jac}(C_{\text{Ros}})$, then any honest party participating in a Diffie-Hellman style protocol computes with multiples of P that also map back to $\text{Jac}(C_{\text{Ros}})$. However, an attacker could feed a party another point $P' \in \mathcal{K}$ that (unbeknownst to the party) maps back to C'_{Ros}, and on return of $[s]P'$, attack the discrete logarithm problem on the twist instead. It is undesirable to include a check of which curve the Kummer points map to, because the maps are overly involved (see [10, §5]). The best solution is to compute curves where both $\text{Jac}(C)$ and $\text{Jac}(C')$ have large prime order subgroups. Such curves and their associated Kummer surfaces are called *twist-secure* [29,28].

Implementation Details and Side-channel Resistance. When computing the scalar multiplication on a Kummer surface, the combined double and pseudo-addition routine is called for every bit in the scalar, except the first one. The main branch, i.e. checking if the bit is set (or not), can be converted into straight-line code by masking (pointers to) the in- and output. Since no lookup tables are used, and all modern cache sizes are large enough to hold the intermediate values when using 128-bit arithmetic, the algorithm (and runtime) becomes independent of input almost for free. The only input-dependent value is the scalar n whose bit-size can differ, meaning that the total runtime could potentially leak the value of the most significant bits. In order to make the implementation run in constant time, we can either increase the scalar via addition of the subgroup order, or we can artificially increase the running time by computing on dummy values such that we compute the combined doubling and pseudo-addition a fixed number of times.

6 GLV in Genus-2

The Gallant-Lambert-Vanstone (GLV) method [23] significantly speeds up scalar multiplication on algebraic curves that admit an efficiently computable endomorphism ϕ of degree $d > 1$, by decomposing the scalar k into d "mini-scalars", all of which have bit-lengths that are approximately $1/d$ that of k. The d scalar multiplications corresponding to each of these mini-scalars can then be computed as one multi-scalar multiplication of length $\approx \log_2(k)/d$, which effectively reduces the number of required doublings by a factor of d.

Endomorphisms. In general, algebraic curves over prime fields do not come equipped with a useful endomorphism ϕ, which means we have to use special curves to take advantage of the GLV method. For genus 1 elliptic curves, Gallant *et al.* suggested the curves $y^2 = x^3 + b$ and $y^2 = x^3 + ax$, which both allow a 2-dimensional decompositions over prime fields. On the other hand, the genus 2 analogues of these curves, Buhler-Koblitz (BK) curves of the form $y^2 = x^5 + b$ [14] and Furukawa-Kawazoe-Takahashi (FKT) curves of the form $y^2 = x^5 + ax$ [21], have ϕ's whose minimal polynomials are of degree 4, which means that we can achieve 4-dimensional scalar decompositions on genus 2 curves over prime fields. Besides the two families above that offer 4-dimensional GLV decompositions, families of genus 2 curves with *real multiplication* (RM) facilitate 2-dimensional scalar decompositions [38,28]. To give an idea of the expected performance in such scenarios, we also present timings for a 2-dimensional GLV decomposition on FKT curves.

Scalar Decomposition Via Division. At Eurocrypt 2002, Park, Jeong and Lim [48] gave an algorithm for performing GLV decomposition via division in the ring $\mathbb{Z}[\phi]$ generated by ϕ. This algorithm is very simple and effective in decomposing the scalar k quickly: in 4-dimensional cases (BK and FKT) it takes 20 multiplications to fully decompose k, and in the 2-dimensional case the decomposition totals just 6 multiplications. For the curves we used, this algorithm performed slightly better on average than the (conservative) numbers quoted in [48, Table 4].

Computing the Scalar Multiplication. We describe two approaches to implement the scalar multiplication. The d-dimensional decomposition of the scalar k results in d smaller scalars k_ℓ, for $0 \leq \ell < d$. The first approach precomputes the 2^d different points $L_i = \sum_{\ell=0}^{d-1} \left(\left\lfloor \frac{i}{2^\ell} \right\rfloor \bmod 2 \right) \cdot P_\ell$ for $0 \leq i < 2^d$ and stores them in a lookup table. When processing the j^{th} bit of the scalar, the precomputed multiple L_i is added, for $i = \sum_{\ell=0}^{d-1} 2^\ell \left(\left\lfloor \frac{k_\ell}{2^j} \right\rfloor \bmod 2 \right)$. Hence, besides the minor bit-fiddling overhead to construct the lookup table index, this requires computing at most a single curve addition and a single curve doubling per bit of the maximum of the k_ℓ's. The second approach [22] is very similar to using signed windows for a single scalar (see Section 2). We start by precomputing the multiples $L_\ell(c) = [c]P_\ell$ for d different tables: one corresponding to each scalar k_ℓ. When computing the scalar multiplication, the j^{th} part (of width w bits) in the scalar k_ℓ determines which point needs to be added (or subtracted), namely $\sum_{\ell=0}^{d-1} \pm L_\ell \left(\left\lfloor \frac{k_\ell}{2^{wj}} \right\rfloor \bmod 2^w \right)$, where the addition or subtraction depends on the addition-subtraction chain used. Thus, an addition to the running value has to be made only once every w bits and combining the lookup table values takes at most $d - 1$ additions, so one needs at most d additions per w bits. The optimal value for w depends on the dimension d, the bit-size of k_ℓ and the cost of (mixed) additions and doublings. There are multiple ways to save computations in this latter approach. After computing the multiples in the first lookup table L_0, the values for the $d - 1$ other tables can be computed by applying the map ϕ to the individual point in the lookup table [22]. Since the computation of the map ϕ only takes three or four multiplications (depending on the curve used), this is a significant saving compared to computing the group operation which is an order of magnitude slower. Furthermore, since the endomorphism costs the same in affine or projective space, one can convert the points in L_0 to affine coordinates

Table 1. Performance timings in 10^3 cycles of various programs calculating a $\lceil \log_2(r) \rceil$-bit scalar multiplication, using genus g arithmetic. The curve characteristics, such as the prime p, the cardinality r, the size of the automorphism group #Aut, and the security level $s = \log_2(\sqrt{\frac{\pi r}{2\#\mathrm{Aut}}})$, are stated as well. Here $p_1 = 2^{256} - 2^{224} + 2^{192} + 2^{96} + 1$ and $p_2 = 2^{64} \cdot (2^{63} - 27443) + 1$. If an implementation runs in constant-time (CT), we indicate this with '✓', if not with '✗', and if unknown with '?'.

Primitive	g	CT	field char p	$\lceil \log_2(r) \rceil$	#Aut	s	10^3 cycles
curve25519 [4,6]	1	✓	$2^{255} - 19$	253	2	125.8	182
ecfp256e [32]	1	✗	$2^{256} - 587$	255	2	126.8	227
Longa-Sica 2-GLV [40]	1	✗	$2^{256} - 11733$	256	6	127.0	145
surf127eps [30]	2	✓	$2^{127} - 735$	251	2	124.8	236
NISTp-224 [56,34]	1	✓	$2^{224} - 2^{96} + 1$	224	2	111.8	302
NISTp-256 [56]	1	?	p_1	256	2	127.8	658
(a) generic127	2	✗	$2^{127} - 1$	254	2	126.8	295
(b) generic127	2	✗	$2^{127} - 1$	254	2	126.8	248
(b) generic128	2	✗	$2^{128} - 173$	257	2	127.8	364
(a) Kummer	2	✓	$2^{127} - 1$	251	2	124.8	139
(b) Kummer	2	✓	$2^{127} - 1$	251	2	124.8	117
(b) Kummer	2	✓	$2^{128} - 237$	253	2	125.8	166
(a) GLV-4-BK	2	✗	p_2	254	10	125.7	156
(a) GLV-4-FKT	2	✗	p_2	253	8	125.3	156
(a) GLV-2-FKT	2	✗	p_2	253	8	125.3	220
(b) GLV-4-BK	2	✗	$2^{128} - 24935$	256	10	126.7	164
(b) GLV-4-FKT	2	✗	$2^{128} - 24935$	255	8	126.3	167
(b) GLV-2-FKT	2	✗	$2^{128} - 24935$	255	8	126.3	261

using Montgomery's simultaneous inversion method (see Section 2), and obtain all of the affine points in the other lookup tables very efficiently through the application of ϕ. This means the faster mixed addition formulas can be applied when adding any element in a lookup table. In our implementations, the first approach is faster in the 4-dimensional case and the second approach is faster in the 2-dimensional case.

7 Results and Discussion

In this section we present our performance results and compare them with the current state-of-the-art.

Benchmark Setting and Code. All of the implementations in Table 1 were run on an Intel Core i7-3520M (Ivy Bridge) processor at 2893.484 MHz with hyperthreading turned off and over-clocking ("turbo boost") disabled. The implementations labeled (a) use the Montgomery-friendly primes. They have been compiled using Microsoft Visual Studio 2012 and run on 64-bit Windows, where the timings are obtained using the time stamp counter instruction `rdtsc` over several thousand scalar multiplications. The implementations labeled (b) use the NIST-like approach and have been compiled with gcc 4.6.3 to run on 64-bit

Linux, where the timings are obtained using the SUPERCOP toolkit for measuring the performance of cryptographic software (see [8]). The implementations labeled (b) are made publicly available through [8]. Both (a) and (b) perform a final modular inversion to ensure that the output point is in affine form: this is the standard setting when computing a Diffie-Hellman key-exchange.

Results. Table 1 summarizes the performance and characteristics of various genus g curve implementations. For the security estimate we assume that the fastest attacks possible are the "generic algorithms", where we specifically use the complexity of the Pollard rho [49] algorithm that exploits additional automorphisms [19,58]. If r is the largest prime factor of a group with #Aut automorphisms, we compute the security level s as $s = \log_2(\sqrt{\frac{\pi r}{2\#\text{Aut}}})$. We also indicate if the implementation runs in constant time, an important step towards achieving side channel resistance [37].

The implementations in the top part of the table are obtained from eBACS, except for [56] and [40]. The standardized NIST curves [56], one of which is at a lower security level, are both obtained from the benchmark program included in OpenSSL 1.0.1.[4] The implementation from [40] is not publicly available, but the authors gave us a precompiled binary which reported its own cycle count so that we could report numbers obtained in our test-environment. All of these implementations were run on our hardware.

Discussion. The first thing to observe from Table 1 is that the standard NISTp-256 curve and the genus 2 curve "generic128" (see Section 4) offer the highest level of security. This "generic" genus 2 implementation is our slowest performing implementation, yet is it still 1.80 times faster than the NIST curve at the same security level. Interestingly, all our Kummer and 4-dimensional GLV implementations manage to outperform the previous fastest genus 2 implementation [30]. Prior to this work, the fastest curve arithmetic reported on eBACS was due to Bernstein [4], whilst Longa and Sica [40] held the overall software speed record over prime fields. We note that the former implementation runs in constant time, while the latter does not. Even though our GLV implementations do not currently run in constant time, we note that they can be transformed into constant time implementations following, for instance, the techniques from [40]. Our approach (b) on the Kummer surface sets a new software speed record by breaking the 120k cycle barrier for constant time implementations at the 128-bit security level.

We note that Table 1 reports implementations over prime fields only. For elliptic curves defined over quadratic extensions of large prime fields, Longa and Sica [40] report a non-constant time scalar multiplication in 91,000 cycles on the Sandy Bridge architecture, while their constant time version runs in 137,000 cycles. Over binary fields, Aranha *et al.* [3] perform a scalar multiplication on the Koblitz curve K-283 in 99,000 cycles on Sandy Bridge, while Oliveira *et al.* [47] recently announced a new speed record of 75,000 cycles on the same architecture. We note that both of these binary field implementations do not run in constant time.

[4] Note, to enable this implementation, using the techniques described in [34], OpenSSL needs to be configured using "./Configure enable-ec_nistp_64_gcc_128".

With respect to the different arithmetic approaches from Section 3, we conclude that when using the prime $2^{127}-1$, the NIST-like approach is the way to go. In the more general comparison of $2^{128}-c_1$ versus $2^{64}\cdot(2^{63}-c_2)\pm 1$ for NIST-like and Montgomery-friendly primes respectively, we found that the Montgomery-friendly primes outperform the former in practice. This was a surprising outcome and we hope that implementers of cryptographic schemes will consider this family of primes as well. The implementations (b) of "generic" and Kummer highlight the practical advantage of the prime $2^{127}-1$ over the prime $2^{128}-c_1$: in both instances the former is around 1.4 times faster than the latter.

8 Kummer Chameleons

In this section we explore curves that facilitate *both* efficient scalar multiplications on the Kummer surface and efficient scalar multiplications on the Jacobian using a GLV decomposition. Such curves give cryptographers the option of taking either route depending on the protocol at hand: for Diffie-Hellman protocols, working on the associated Kummer surface is the most efficient option, but if the pseudo-addition law on the Kummer surface is insufficient, the GLV method can be used on an associated curve. Since these curves can morph depending on the scenario, we call them *Kummer chameleons*.

We primarily focus on the two families that facilitate 4-dimensional GLV decompositions. We start with the FKT family of curves to show an unfortunate drawback which prohibits us from using this Kummer/GLV duality over prime fields. We then move to the BK family of curves which does allow this duality in practice. For these special families, we also show the benefits of computing the Kummer surface parameters analytically (i.e. over $\mathbb{C}$). This approach tells us when we can (or cannot) expect to find practical Kummer parameters using the technique of extracting $\mathcal{K}$ from C_{Ros} in Section 5. It can additionally reveal when we are likely to find small surface constants, which guarantees solid speedups in practice. For an overview of computations involving the analytic Jacobian of a hyperelliptic curve, we refer to [57].

Recognising Kummer Parameters Over $\mathbb{C}$. We use an analytic approach to assist us in generating Kummer surfaces which are associated to a particular CM field. For each CM field, there is a collection of period matrices which correspond to the isomorphism classes of Jacobians of genus 2 curves with CM by that field, and thus with known possible group orders (see [57]). The theta functions can be evaluated at these period matrices, and approximations of the complex values of quotients of the associated theta constants can be used to recognize the minimal polynomials that they satisfy.

Although it can be difficult to analytically recognize the theta constants themselves, for special families it is often possible to recognize *quotients* of certain theta constants. In Tables 2 and 3, we give the minimal polynomials satisfied by all of the parameters required for the Kummer surface implementation for the FKT and BK families: the values E', F, G, H which define the surface (see Section 5), and the constants y_0, z_0, t_0, y_0', z_0' and t_0' which are needed in the

Table 2. Kummer parameters (and their minimal polynomials) over $\mathbb{C}$ for the FKT family

$\mathcal{K}$ param.	E	F, G, H	y_0, t_0	z_0	y_0', t_0'	z_0'
Value $\in \mathbb{C}$	$17+31i$	$(3+i)/2$	1	$1-i$	$3+4i$	$-3-4i$
Min. poly.	$x^2-34x+1250$	$2x^2-6x+5$	$x-1$	x^2-2x+2	$x^2-6x+25$	$x^2+6x+25$

doubling and pseudo-addition operations (see [10, §5]). The coefficients of these minimal polynomials can be reduced modulo any prime p, so for any p for which the polynomials have a consistent choice of roots modulo p, they can be used to define a Kummer surface over $\mathbb{F}_p$ such that the associated group order of $\text{Jac}(C)$ is known (from the CM field).

The Kummer Surface of FKT Curves. For curves of the form $y^2 = x^5 + ax$, the complex values (and corresponding minimal polynomials) of the required Kummer parameters are given in Table 2. We note that once we choose $i = \sqrt{-1}$ by sufficiently extending $\mathbb{F}_p$ (if necessary), all of the required constants are determined. Observe that two of the six surface constants are 1, which immediately results in two fewer multiplications (see [10, §5]).

Mapping points from the Kummer surface to the associated Jacobian(s) actually takes points on $\mathcal{K}$ to divisors on $\text{Jac}(C_{\text{Ros}})$ or $\text{Jac}(C'_{\text{Ros}})$, where C_{Ros} : $y^2 = x(x-1)(x-\lambda)(x-\mu)(x-\nu)$, and for which we can also recognize the Rosenhain invariants in $\mathbb{C}$ as $\lambda = (i+1)/2$, $\mu = i$ and $\nu = i+1$. Now, if $p \equiv 1 \pmod 4$, then $i = \sqrt{-1} \in \mathbb{F}_p$ and the Rosenhain model defined by those values is defined over $\mathbb{F}_p$. The curve $C : y^2 = x^5 + ax$ can be rewritten as $y^2 = x(x-\alpha)(x+\alpha)(x-\alpha i)(x+\alpha i)$, where α is a non-trivial fourth root of $-a$. Clearly C and C_{Ros} can only be isomorphic over $\mathbb{F}_p$ if $\alpha \in \mathbb{F}_p$, which implies that $\text{Jac}(C)$ is isogenous over $\mathbb{F}_p$ to the product of two elliptic curves [21, Lemma 4]. Thus C is not suitable for cryptographic applications in this case, since the group order of $\text{Jac}(C)$ is a product of factors of at most half the size of the total. If instead $p \equiv 3 \pmod 4$, then $i \in \mathbb{F}_{p^2} \backslash \mathbb{F}_p$, and from Table 2 it follows that the Kummer surface is defined over $\mathbb{F}_{p^2}$, which destroys the arithmetic efficiency of the group law algorithms. Therefore, we conclude that the FKT family does not yield a secure and efficient Kummer surface over prime fields.

The Kummer Surface of BK Curves. For curves of the form $y^2 = x^5 + b$, the minimal polynomials for the required Kummer parameters are given in Table 3. Since these polynomials have degree larger than two, writing down the correct root corresponding to each Kummer parameter becomes more involved. Furthermore, these polynomials tell us that we can not expect any Kummer constants to automatically be small. Nevertheless, they do help us deduce when it is possible to find practical Kummer parameters. For example, t_0 is a root of $\Phi_5(-x^2)$, which does not have any roots in $\mathbb{F}_p$ when $p \equiv 11 \pmod{20}$, yet splits into linear factors when $p \equiv 1 \pmod{20}$. In fact, all of the polynomials in Table 3 split into linear factors in $\mathbb{F}_p$ for $p \equiv 1 \pmod{20}$; this agrees with our experiments which always extracted working Kummer parameters for BK curves when $p \equiv 1 \pmod{20}$, and always failed to do so when $p \equiv 11 \pmod{20}$.

Table 3. Kummer parameters (and their minimal polynomials) over $\mathbb{C}$ for the Buhler-Koblitz family

Kummer parameter	Minimal polynomial
E, F	$x^2 - 20x - 400$, $x^8 - 11x^6 + 46x^4 - 96x^2 + 121$
G, H	$x^8 - 11x^6 + 46x^4 - 96x^2 + 121$, $x^2 + x - 1$
y_0, z_0	$x^4 - x^3 + x^2 - x + 1$, $x^8 - 4x^6 + 6x^4 + x^2 + 1$
t_0, y'_0	$x^8 - x^6 + x^4 - x^2 + 1$, $x^4 - 16x^3 + 46x^2 - 16x + 1$
z'_0, t'_0	$25x^8 - 100x^7 + 460x^6 + 580x^5 + 286x^4 + 36x^3 - 4x^2 - 4x + 1$

The only minor drawback for the Kummer surface associated to the BK family is that, for primes congruent to 1 modulo 5, if the 2-torsion of $\mathrm{Jac}(C)$ or $\mathrm{Jac}(C')$ is defined over $\mathbb{F}_p$, then 5 divides at least one of the two group orders. Hence, even in the best case the two group orders have cofactors of 16 and 80, which means either the curve or its twist will be around 1 bit less secure than the other. In this case, generators on the Kummer surface should be chosen which map back to the curve with cofactor 16.

Kummer Chameleons with 2-dimensional GLV. Although we have focused on two families of genus 2 curves that offer 4-dimensional GLV over prime fields, there are many more families that offer 2-dimensional GLV [38,53,28]. We especially mention the family due to Mestre [41], which was studied further in [28, §4.4]. This family might be particularly attractive since the techniques in [28] make it practical to find twist-secure instances over $\mathbb{F}_p$ with $p = 2^{127} - 1$. Working analytically, we observed that small Kummer constants are often obtained if we take special instances of the families with efficiently computable RM. An example from the family due to Tautz, Tops and Verberkmoes [54] (also see [38, §5.1]) is the Kummer surface associated to the curve $y^2 = x(x^4 - x^2 + 1)$, which yields $t_0 = 1$ over $\mathbb{C}$, so the techniques in [28, §4.4] could be used (over many primes) to find twist-secure curves that can take advantage of this.

9 Conclusions

We have given a taxonomy of the state-of-the-art in genus-2 arithmetic over prime fields, with respect to its application in public-key cryptography. We studied two different approaches to achieve fast modular arithmetic and implemented these techniques in three settings: on "generic" genus-2 curves, on special genus-2 curves facilitating 2-and 4-dimensional GLV decompositions, and on the Kummer surface proposed by Gaudry [26]. Furthermore, we presented *Kummer chameleons*; curves which allow fast arithmetic on the Kummer surface as well as efficient arithmetic on the Jacobian that results from a GLV decomposition. Ultimately, we highlighted the practical benefits of genus-2 curves with our Kummer surface implementation - this sets a new software speed record at the 128-bit security level for computing constant time scalar multiplications compared to all previous elliptic curve and genus-2 implementations.

Acknowledgements. We wish to thank Pierrick Gaudry for his Kummer help when this project began, Dan Bernstein and Tanja Lange for several fruitful discussions during the preparation of this work, Patrick Longa for his advice on optimizing the GLV routines and extensive comments on this work, Michael Naehrig for proofreading early versions of this paper, and the anonymous Eurocrypt reviewers for their useful comments.

References

1. Acar, T., Shumow, D.: Modular reduction without pre-computation for special moduli. Technical report, Microsoft Research (2010)
2. Adleman, L., DeMarrais, J., Huang, M.: A subexponential algorithm for discrete logarithms over hyperelliptic curves of large genus over GF(q). Theoretical Computer Science 226(1-2), 7–18 (1999)
3. Aranha, D.F., Faz-Hernández, A., López, J., Rodríguez-Henríquez, F.: Faster implementation of scalar multiplication on Koblitz curves. In: Hevia, A., Neven, G. (eds.) LATINCRYPT 2012. LNCS, vol. 7533, pp. 177–193. Springer, Heidelberg (2012)
4. Bernstein, D.J.: Curve25519: New Diffie-Hellman speed records. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.) PKC 2006. LNCS, vol. 3958, pp. 207–228. Springer, Heidelberg (2006)
5. Bernstein, D.J.: Elliptic vs. Hyperelliptic, part I. Talk at ECC, slides at (September 2006), `http://cr.yp.to/talks/2006.09.20/slides.pdf`
6. Bernstein, D.J., Duif, N., Lange, T., Schwabe, P., Yang, B.-Y.: High-speed high-security signatures. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 124–142. Springer, Heidelberg (2011)
7. Bernstein, D.J., Lange, T.: Analysis and optimization of elliptic-curve single-scalar multiplication. In: *Finite Fields and Applications*. Contemporary Mathematics Series, vol. 461, pp. 1–19. American Mathematical Society (2008)
8. Bernstein, D.J., Lange, T. (eds). eBACS: ECRYPT Benchmarking of Cryptographic Systems, `http://bench.cr.yp.to` (accessed October 4, 2012)
9. Bos, J.W.: High-performance modular multiplication on the Cell processor. In: Hasan, M.A., Helleseth, T. (eds.) WAIFI 2010. LNCS, vol. 6087, pp. 7–24. Springer, Heidelberg (2010)
10. Bos, J.W., Costello, C., Hisil, H., Lauter, K.: Two is greater than one. Cryptology ePrint Archive, Report 2012/670 (2012), `http://eprint.iacr.org/`
11. Bos, J.W., Kaihara, M.E., Kleinjung, T., Lenstra, A.K., Montgomery, P.L.: Solving a 112-bit prime elliptic curve discrete logarithm problem on game consoles using sloppy reduction. Int. J. of Applied Cryptography 2(3), 212–228 (2012)
12. Brauer, A.: On addition chains. Bulletin of the American Mathematical Society 45, 736–739 (1939)
13. Brown, M., Hankerson, D., López, J., Menezes, A.: Software implementation of the NIST elliptic curves over prime fields. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 250–265. Springer, Heidelberg (2001)
14. Buhler, J., Koblitz, N.: Lattice basis reduction, Jacobi sums and hyperelliptic cryptosystems. Bull. of the Australian Math. Soc. 58(1), 147–154 (1998)
15. Chudnovsky, D.V., Chudnovsky, G.V.: Sequences of numbers generated by addition in formal groups and new primality and factorization tests. Advances in Applied Mathematics 7, 385–434 (1986)
16. Cosset, R.: Factorization with genus 2 curves. Math. Comp. 79(270), 1191–1208 (2010)
17. Costello, C., Lauter, K.: Group law computations on Jacobians of hyperelliptic curves. In: Miri, A., Vaudenay, S. (eds.) SAC 2011. LNCS, vol. 7118, pp. 92–117. Springer, Heidelberg (2012)

18. Diem, C.: On the discrete logarithm problem in class groups of curves. Math. Comp. 80, 443–475 (2011)
19. Duursma, I.M., Gaudry, P., Morain, F.: Speeding up the discrete log computation on curves with automorphisms. In: Lam, K.-Y., Okamoto, E., Xing, C. (eds.) ASIACRYPT 1999. LNCS, vol. 1716, pp. 103–121. Springer, Heidelberg (1999)
20. Enge, A.: Computing discrete logarithms in high-genus hyperelliptic Jacobians in provably subexponential time. Math. Comp. 71, 729–742 (2002)
21. Furukawa, E., Kawazoe, M., Takahashi, T.: Counting points for hyperelliptic curves of type $y^2 = x^5 + ax$ over finite prime fields. In: SAC 2003. LNCS, vol. 3006, pp. 26–41. Springer, Heidelberg (2003)
22. Galbraith, S.D., Lin, X., Scott, M.: Endomorphisms for faster elliptic curve cryptography on a large class of curves. J. Cryptology 24(3), 446–469 (2011)
23. Gallant, R.P., Lambert, R.J., Vanstone, S.A.: Faster point multiplication on elliptic curves with efficient endomorphisms. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 190–200. Springer, Heidelberg (2001)
24. Gaudry, P.: An algorithm for solving the discrete log problem on hyperelliptic curves. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 19–34. Springer, Heidelberg (2000)
25. Gaudry, P.: Algorithmique des courbes hyperelliptiques et applications à la cryptologie. PhD thesis, École polytechnique (2000), `http://www.lix.polytechnique.fr/Labo/Pierrick.Gaudry/publis/`
26. Gaudry, P.: Fast genus 2 arithmetic based on theta functions. Journal of Mathematical Cryptology 1(3), 243–265 (2007)
27. Gaudry, P.: Personal communication (2011)
28. Gaudry, P., Kohel, D.R., Smith, B.A.: Counting points on genus 2 curves with real multiplication. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 504–519. Springer, Heidelberg (2011)
29. Gaudry, P., Schost, É.: Genus 2 point counting over prime fields. J. Symb. Comp. 47(4), 368–400 (2012)
30. Gaudry, P., Thomé, E.: The mp$\mathbb{F}_q$ library and implementing curve-based key exchanges. In: SPEED 2007, pp. 49–64 (2007), `http://www.loria.fr/~gaudry/publis/mpfq.pdf`
31. Hamburg, M.: Fast and compact elliptic-curve cryptography. Cryptology ePrint Archive, Report 2012/309 (2012), `http://eprint.iacr.org/`
32. Hisil, H., Wong, K.K.-H., Carter, G., Dawson, E.: Twisted Edwards curves revisited. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 326–343. Springer, Heidelberg (2008)
33. Kaliski Jr., B.S.: The Montgomery inverse and its applications. IEEE Transactions on Computers 44(8), 1064–1065 (1995)
34. Käsper, E.: Fast elliptic curve cryptography in openSSL. In: Danezis, G., Dietrich, S., Sako, K. (eds.) FC 2011 Workshops 2011. LNCS, vol. 7126, pp. 27–39. Springer, Heidelberg (2012)
35. Knežević, M., Vercauteren, F., Verbauwhede, I.: Speeding up bipartite modular multiplication. In: Hasan, M.A., Helleseth, T. (eds.) WAIFI 2010. LNCS, vol. 6087, pp. 166–179. Springer, Heidelberg (2010)
36. Koblitz, N.: Elliptic curve cryptosystems. Math. Comp. 48(177), 203–209 (1987)
37. Kocher, P.C.: Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996)

38. Kohel, D.R., Smith, B.A.: Efficiently computable endomorphisms for hyperelliptic curves. In: Hess, F., Pauli, S., Pohst, M. (eds.) ANTS 2006. LNCS, vol. 4076, pp. 495–509. Springer, Heidelberg (2006)
39. Lenstra, A.K.: Generating RSA moduli with a predetermined portion. In: Ohta, K., Pei, D. (eds.) ASIACRYPT 1998. LNCS, vol. 1514, pp. 1–10. Springer, Heidelberg (1998)
40. Longa, P., Sica, F.: Four-dimensional Gallant-Lambert-Vanstone scalar multiplication. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 718–739. Springer, Heidelberg (2012)
41. Mestre, J.-F.: Couples de Jacobiennes isogenes de courbes hyperelliptiques. Preprint, arXiv (2009), `http://arxiv.org/abs/0902.3470`
42. Miller, V.S.: Use of elliptic curves in cryptography. In: Williams, H.C. (ed.) CRYPTO 1985. LNCS, vol. 218, pp. 417–426. Springer, Heidelberg (1986)
43. Montgomery, P.L.: Modular multiplication without trial division. Math. Comp. 44(170), 519–521 (1985)
44. Montgomery, P.L.: Speeding the Pollard and elliptic curve methods of factorization. Math. Comp. 48(177), 243–264 (1987)
45. Morain, F., Olivos, J.: Speeding up the computations on an elliptic curve using addition-subtraction chains. Informatique Théorique et Applications/Theoretical Informatics and Applications 24, 531–544 (1990)
46. National Security Agency. Fact sheet NSA Suite B Cryptography (2009), `http://www.nsa.gov/ia/programs/suiteb_cryptography/index.shtml`
47. Oliveira, T., Rodríguez-Henríquez, F., López, J.: New timings for scalar multiplication using a new set of coordinates. In: Rump Session Talk at ECC 2012 (2012)
48. Park, Y.-H., Jeong, S., Lim, J.: Speeding up point multiplication on hyperelliptic curves with efficiently-computable endomorphisms. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 197–208. Springer, Heidelberg (2002)
49. Pollard, J.M.: Monte Carlo methods for index computation (mod p). Math. Comp. 32(143), 918–924 (1978)
50. Scholz, A.: Aufgabe 253. Jahresbericht der deutschen Mathematiker-Vereingung 47, 41–42 (1937)
51. Smart, N.P., Siksek, S.: A fast Diffie-Hellman protocol in genus 2. J. Cryptology 12(1), 67–73 (1999)
52. Solinas, J.A.: Generalized Mersenne numbers. Technical Report CORR 99–39, Centre for Applied Cryptographic Research, University of Waterloo (1999)
53. Takashima, K.: A new type of fast endomorphisms on Jacobians of hyperelliptic curves and their cryptographic application. IEICE Trans. 89-A(1), 124–133 (2006)
54. Tautz, W., Top, J., Verberkmoes, A.: Explicit hyperelliptic curves with real multiplication and permutation polynomials. Canad. J. Math 43(5), 1055–1064 (1991)
55. Thurber, E.G.: On addition chains $l(mn) \leq l(n) - b$ and lower bounds for $c(r)$. Duke Mathematical Journal 40, 907–913 (1973)
56. U.S. Department of Commerce/National Institute of Standards and Technology. Digital Signature Standard (DSS). FIPS-186-3 (2009), `http://csrc.nist.gov/publications/fips/fips186-3/fips_186-3.pdf`
57. van Wamelen, P.: Computing with the analytic Jacobian of a genus 2 curve. In: *Discovering Mathematics with Magma*. Algorithms and Computation in Mathematics, vol. 19, pp. 117–135. Springer, Heidelberg (2006)
58. Wiener, M.J., Zuccherato, R.J.: Faster attacks on elliptic curve cryptosystems. In: Tavares, S., Meijer, H. (eds.) SAC 1998. LNCS, vol. 1556, pp. 190–200. Springer, Heidelberg (1999)

Graph-Theoretic Algorithms for the "Isomorphism of Polynomials" Problem

Charles Bouillaguet[1], Pierre-Alain Fouque[2], and Amandine Véber[3]

[1] University of Lille-1
charles.bouillaguet@univ-lille1.fr
[2] University of Rennes-1
pierre-alain.fouque@univ-rennes1.fr
[3] CMAP Lab, CNRS and Ecole Polytechnique
amandine.veber@cmap.polytechnique.fr

Abstract. We give three new algorithms to solve the "isomorphism of polynomial" problem, which was underlying the hardness of recovering the secret-key in some multivariate trapdoor one-way functions. In this problem, the adversary is given two quadratic functions, with the promise that they are equal up to linear changes of coordinates. Her objective is to compute these changes of coordinates, a task which is known to be harder than Graph-Isomorphism. Our new algorithm build on previous work in a novel way. Exploiting the birthday paradox, we break instances of the problem in time $q^{2n/3}$ (rigorously) and $q^{n/2}$ (heuristically), where q^n is the time needed to invert the quadratic trapdoor function by exhaustive search. These results are obtained by turning the algebraic problem into a combinatorial one, namely that of recovering partial information on an isomorphism between two exponentially large graphs. These graphs, derived from the quadratic functions, are new tools in multivariate cryptanalysis.

1 Introduction

The notion of *equivalent linear maps* is a basic concept in linear algebra; two linear functions f and g over vector spaces are equivalent if and only if there exist two other linear bijective functions S and T such that $f = T \circ g \circ S$. Geometrically speaking, this means that f and g are essentially the same function, but with coordinates expressed in different bases. The computational problem consisting in testing the equivalence of two linear functions (given by matrices) is easy, because it is well-known that two linear maps are equivalent if and only if they have the same rank.

This notion of equivalent linear maps lends itself to an obvious generalization, by dropping the requirement that the functions shall be linear. Then, given two vector spaces U and V, of respective dimension n and m, two functions $f, g : U \to V$ are said to be equivalent if there exist an invertible $n \times n$ matrix S and an invertible $m \times m$ matrix T such that $g = T \circ f \circ S$. Again, the geometric interpretation of this notion is that g and f are "the same function", up to linear changes of coordinates. However, deciding the equivalence of two such functions is no longer easy in general.

The case where f and g are polynomial maps is particularly relevant, not only because it is a natural generalization of the linear case, but also because f and g admit a compact representation. It is understood that a polynomial map f is such that each coordinate of the vector $f(x)$ is a polynomial expression of the coordinates of the vector x. Testing the equivalence of two polynomial maps has been called the "Isomorphism of Polynomials" (IP) problem by Patarin in 1996 [28], and later the "Polynomial Linear Equivalence" (PLE) problem by Faugère et al. in 2006 [19].

One aspect of PLE that makes it a bit difficult to study is that depending on the parameters (dimensions and base field of the vector spaces, degree of the polynomials, special restrictions, etc.), the problem can take very different forms. We will thus focus on the case where the base field of the vector space is finite (of size q), where polynomials are quadratic, and where their domain and codomain are the same, *i.e.*, where $f, g : (\mathbb{F}_q)^n \rightarrow (\mathbb{F}_q)^n$ are quadratic maps. This is the setting that appears in most cryptographic constructions. In the sequel we will call this particular restriction the Quadratic Maps Linear Equivalence (QMLE) problem. In order to make our exposition simpler, we will furthermore assume that q, the size of the finite field, is a power of two. The theory of quadratic forms presents itself very differently for odd characteristic and for characteristic two, and in order not to expose two variants of each of our results, we chose the most computer-oriented setting.

The first "multivariate" cryptographic schemes relied on a somewhat heuristic construction to build Trapdoor One-Way Functions, whose security was based on the hardness of QMLE. Starting with an easy-to-invert quadratic map f, one builds an apparently random-looking one by setting $g = T \circ f \circ S$. The idea is that the changes of coordinate would hide the structure of f that makes it easy to invert, so that g would look random. Inverting random quadratic maps is extremely hard, and the best options in general are exhaustive search (if q is small), or the computation of a Groebner basis (when q is large), both techniques being exponential in n. This construction backed one of the advertized goals of multivariate cryptography, namely the ability to encrypt or sign n-bit blocks while offering n bits of security, as opposed to, *e.g.* RSA.

In this setting, g (and eventually f) is the public key, while S and T are the secret key. When f is public, then recovering the secret-key precisely means solving an instance of QMLE. Several cryptosystems have been built on this idea [6, 29, 41, 13], but they have *all* been broken [22, 18, 5, 26, 21]. The main reason behind this fiasco is that the specific instances of QMLE exposed by these schemes were weak because f was too special, so that polynomial-time and/or efficient algorithms to crack them have eventually been designed.

In a different direction, Patarin also proposed to use the hardness of *arbitrarily chosen* instances of the PLE problem to design a public-key identification scheme, thus potentially avoiding the aforementioned disaster. A *prover*, who has generated a pair of private/public keys (PK, SK), wants to prove her identity to a *verifier* who knows PK. In fact the prover aims to convince that she knows SK, but without revealing any information about SK to the verifier, or to anybody else. In 1986, Goldreich, Micali and Wigderson [24] built an elegant zero-knowledge proof system for Graph Isomorphism (GI) and used it to build an identification scheme. There, PK is a pair of isomorphic graphs, and SK is the isomorphism (a permutation of the vertices). In order for this system to be secure, it must be hard to solve the instance of GI formed by the public-key. Despite a large research effort, until now no algorithm has been able to solve instances of GI in worst-case polynomial, which is certainly encouraging. However, most instances of GI, and in particular random instances, are *extremely easy* to solve. Thus, the identification scheme of [24] relied on a presumably hard problem for which we do not know how to generate non-trivial instances...

Patarin's suggestion was that Graph Isomorphism could be replaced by QMLE, with the hope that *random instances* of the problem would then be hard, and that key-generation would then be straightforward. There was apparently nothing to lose with the new problem, because it was shown to be harder than GI [30]. Using random instances would in principle avoid the weak instances that had been broken. The resulting QMLE-based identification scheme is not particularly efficient, and does not enjoy very attractive key-sizes, but it is quite simple. It also has a few interesting features compared to other identification schemes based on NP-hard combinatorial problems such as [33–38]: most notably, it does not require hash functions nor commitment schemes, and it does not require the parties to share a (usually large) public common string describing an instance of the NP-complete problem.

1.1 Related Work

The QMLE problem is reminiscent of the Even-Mansour cipher [17], which turns a fixed n-bit permutation P into an n-bit block-cipher with $2n$-bit key by setting $E_{k_1,k_2}(x) = P(x + k_1) + k_2$. Attacks against this construction aim to recover the keys while only having black-box access to E and P. One of its distinctive features is that the performance of a successful adversary running in time t and sending q queries is limited by $t \cdot q \geq 2^n$, under the assumption that P is a random permutation. The known attacks match this bound [12, 16]. As mentioned above, the hardness of QMLE would allow a similar construction where a fixed and public quadratic permutation P is turned into a public-key encryption primitive $E_{S,T} = T \circ P \circ S$. In this context, adversaries not only have oracle to E and P, but know their full description.

Essentially two non-trivial algorithms have been proposed so far for QMLE: the "To-and-Fro" approach [30] on the one hand, and the "Groebner Basis" approach [19] on the other hand. There are also several, more efficient algorithms for the special case where the secret T matrix is known to be the identity matrix [23, 32, 9, 27]. This sub-problem is also GI-hard, even in very restricted settings [1]. The article [2] considers the particular case of testing whether two boolean functions are equal modulo a permutation of their inputs. It shows that $2^{n/2}$ queries are necessary if one only has black-box access to the boolean functions.

Back to the full QMLE problem, the "To-and-Fro" algorithm, while being simple, was exposed on a toy example, without pseudo-code nor detailed analyzis. We are convinced that the algorithm works when the polynomial maps f and g are *bijective*, but it cannot work as-is when they are not (the authors of [19] made the same observation). Note that a random polynomial map is not bijective with overwhelming probability. As is it given in [30], the "to-and-fro" algorithm is thus not applicable to random instances of QMLE. We found out that it *is* nevertheless possible to adapt the algorithm to work in the non-bijective case, but there are several ways to do so, and some are more efficient than others. Figuring that out required some work, and exposing it requires some space, so we will not go deeper into this issue in this paper. In any case, the authors of [30] claim that the complexity of their algorithm is of order $\mathcal{O}\left(q^{2n}\right)$ when $q > 2$ and $\mathcal{O}\left(2^{3n}\right)$ when $q = 2$, and we agree with them. The algorithm was later independently rediscovered under the form of a procedure to test the linear equivalence of S-boxes [7].

The "Groebner basis" algorithm, on the other hand is not heuristic, and is well-specified. It consists in identifying coefficient-wise the equation $T^{-1} \circ g = f \circ S$, which relates two vectors of n quadratic forms. It is therefore equivalent to about n^3 quadratic equations in the $2n^2$ coefficients of the unknown changes of coordinates. These equations are then solved through the computation of a Groebner basis. The complexity of Groebner basis algorithms is notoriously tricky to study, and the authors of [19] did not give any definitive results. However, they empirically observed an important fact, namely that when f and g are *inhomogeneous* quadratic maps, *i.e.*, when f and g contains non-zero linear and constant terms, then their algorithm terminated in polynomial time $\mathcal{O}\left(n^9\right)$. In the homogeneous case, the authors of [19] conjectured that their algorithm is subexponential, without providing any argument nor any evidence that it is the case. This assertion is impossible to verify in practice because the complexities are too high, but our own reasoning makes us more inclined to believe that the algorithm is plainly exponential. Assuming that the equations form a semi-regular sequence would allow to estimate the complexity of the Groebner basis computation [4]; doing so results in a total complexity of $\mathcal{O}\left(2^{18n}\right)$, yet assuming that the equations are semi-regular is probably a bit of a stretch. Establishing the complexity of this algorithm is thus essentially an open problem.

In the sequel, we will nevertheless take for granted that inhomogeneous instances of QMLE are tractable and can be solved in polynomial time, using the "Groebner-based" algorithm for instance.

It must be noted that in [30], the existence of an algorithm based on the birthday paradox and running in time $\mathcal{O}\left(q^{n/2}\right)$ is asserted, and that this algorithm is itself partially described in [31], where it is called the "combined powers attack". This algorithm is sometimes acknowledged for in the literature (*e.g.* in [19]). However, it is underspecified to the point that it is impossible to implement it, and some of the bits that are specified have major problems. Some of them deterministically fail to meet their goal, and the whole construction relies on heuristic assumptions that are empirically false (sometimes provably). This “algorithm” should thus be disregarded.

1.2 Our Results

We give three algorithms to solve QMLE in the homogeneous case. All these algorithms work by reducing the solution of a homogeneous (hard) instance into that of one or several inhomogeneous (easy) instances after some preprocessing. We will thus assume that we are given a (black-box) *Inhomogeneous solver* that presumably works in polynomial time, and we will count the number of *inhomogeneous queries* sent to this oracle. We are well-aware that this assumption is quite strong. The empirical success of the algorithm of [19] convinced us that it works in polynomial-time on average, yet moving from there to “worst-case polynomial time” seems like a leap of faith. However, this assumption eases our exposition considerably, and in practice there does not seem to be any problem (probably because the queries sent to the inhomogeneous oracle are random enough).

Our three algorithms differ by the number of queries they send to the oracle, by the amount of computation they perform themselves, and by their success probability.

Algo.	Preprocessing	Inhom. queries	success prob.	
1		q^n	1	
2	$\mathcal{O}\left(n^3 \cdot q^{2n/3}\right)$	$q^{2n/3}$	62%	
3	$\mathcal{O}\left(n^5 \cdot q^{n/2}\right)$	1	62 %	only when $q = 2$

Algorithm 1 is deterministic, and essentially performs an exhaustive search in $(\mathbb{F}_q)^n$, sending one inhomogeneous query per vector. Using the algorithm of [19] to deal with the inhomogeneous instances, the resulting complexity is $\mathcal{O}\left(n^9 \cdot q^n\right)$, which already improves on the “to-and-fro” algorithm of [30].

Algorithms 2 and 3 rely on the birthday paradox to improve on exhaustive search and break the q^n barrier. To this end, two exponentially large isomorphic graphs are derived from the two quadratic maps. Recovering a bit of information on an isomorphism allows to make the problem inhomogeneous, and thus easy to solve. The trick is that this partial information must be extracted without knowing the full graphs, because they are too large. The construction of these graphs borrows from the differential techniques that have broken SFLASH, amongst others.

Algorithm 2 is relatively easy to analyze and we rigorously establish its complexity and success probability when dealing with random instances of the problem. Algorithm 3 is more efficient but more sophisticated and harder to analyze (as well as somewhat heuristic). We provide an as-rigorous-as-possible complexity analysis under a conjecture on random quadratic maps, and we verify experimentally that we are not off by too much.

Because our algorithms are exponential in n, we do not fully break Patarin’s identification scheme (it is of no practical value anyway), even though its key-sizes should in principle be doubled. The construction of a Trapdoor One-Way

Function from QMLE outlined above has already been bludgeoned to death by cryptanalysts, and it now lies on the autopsy table. We take the role of the medical examiner that appears in every good police drama, only to discover that the corpse had a fatal disease even before being brutally assaulted. We indeed believe that our algorithms condemn this generic construction of a Trapdoor One-Way Function *post-mortem*, and give a theoretical reason not to try again, besides the obvious "they have all been broken" argument. Our algorithms indeed break the QMLE instance and retrieve the secret-key (asymptotically) much faster than inverting the quadratic map by exhaustive search. This shows in passing that this construction can only offer $n/2$ bits of security, instead of the n that was its original objective.

2 A First Algorithm Based on Dehomogenization

Confronted with a *homogeneous* instance of QMLE, our strategy throughout this paper is to build an *inhomogeneous instance* admitting the exact same solutions. This inhomogeneous instance can in turn be solved in polynomial time, and reveals the solution(s) of the original problem. The downside of this approach is that the image of S must be known at one arbitrary point of the vector space. Indeed, if $\beta = S \cdot \alpha$, then:

$$\forall x.\ g(x) = T \cdot f(S \cdot x) \qquad \Longleftrightarrow \qquad \forall x.\ g(x+\alpha) = T \cdot f(S \cdot x + \beta).$$

Thus defining $g'(x) = g(x+\alpha)$ and $f' = f(x+\beta)$ yields an equivalent problem, *i.e.*, an instance that has the same solutions as the original one. In addition, the new instance is inhomogeneous. This follows from the simple observation that although x^2 is a homogeneous polynomial, $(x+\alpha)^2 = x^2 + 2\alpha x + \alpha^2$ is not since it has a non-trivial linear term αx and a non-trivial constant term α^2.

It follows that solving (homogeneous) instances of QMLE essentially boils down to finding $S\alpha$, for some known and non-zero vector α. Exhaustive search is the first option that comes to mind, leading to Algorithm 1. This algorithm sends q^n queries to the inhomogeneous solver in the worst case, and finds the solutions when they exist. This algorithm terminates with probability one in time $\mathcal{O}\left(n^9 \cdot q^n\right)$ if the Groebner-based algorithm of [19] is used to solve the inhomogeneous instances. Despite being extremely simple, Algorithm 1 is asymptotically q^n times faster than to the "to-and-fro" algorithm of [30].

This *dehomogenization* technique exposes a crucial asymmetry in the problem: it is apparently much more critical to obtain knowledge on S than on T. This is not new: the "To-and-Fro" algorithm relies on the ability to transfer knowledge of a relation $\beta = S \cdot \alpha$ to a relation $g(\alpha) = T \cdot f(\beta)$.

3 Moving the Problem into a Graphic World

Using the birthday paradox is a natural idea to improve on exhaustive search algorithms in many scenarii, with the hope to halve the exponent in the complexity. Here, we wish to use the birthday paradox to obtain the image of S at

Algorithm 1. Simple algorithm based on dehomogenization

```
function EXHAUSTIVE-DEHOMOGENIZATION($f, g$)
    $x \leftarrow$ random non-zero vector in $(\mathbb{F}_q)^n$
    for all $0 \neq y \in (\mathbb{F}_q)^n$ do
        $f'(z) \leftarrow f(z+y)$
        $g'(z) \leftarrow g(z+x)$
        query IQMLE-SOLVER with $(f', g'')$
        if solution $(S, T)$ found then return $(S, T)$
    return "Not Equivalent"
```

one point, and build a dehomogenized instance, just as we did in the previous section. One difficulty is that we want to focus only on S, and leave T alone. To this end, we introduce a tool which is, to the best of our knowledge, new. We associate a *graph* G_h to any quadratic map $h : (\mathbb{F}_q)^n \mapsto (\mathbb{F}_q)^n$. Its vertices are the elements of $(\mathbb{F}_q)^n$, and there is an edge between $x, y \in (\mathbb{F}_q)^n$ if and only if $h(x+y) = h(x) + h(y)$. To some extent, G_h expresses the "linear behavior" of h (even though h is not linear) and thus we call these graphs the "linearity graphs" of the associated quadratic maps.

These graphs are natural objects associated to quadratic maps. For instance, the distinguisher of [15] to determine whether a given quadratic map f is an HFE public key can be rephrased as follows: pick a random node in G_f, and count its neighbors. If their number exceeds a given bound (which depends on the degree of the internal HFE polynomial), then return "random", else return "HFE". With the right bound on the number of neighbors, this algorithm achieves subexponential advantage.

The essential interest of linearity graphs for our purposes is that the two graphs G_f and G_g are connected by the secret matrix S.

Lemma 1. *If $T \circ g = f \circ S$ then S is a graph isomorphism that sends G_f to G_g.*

Proof. Indeed, if $x \leftrightarrow y$ in G_g, then by definition $g(x+y) = g(x) + g(y)$, and it follows that $T \circ g(x+y) = T \circ g(x) + T \circ g(y)$, and thus that $f(S \cdot x + S \cdot y) = f(S \cdot x) + f(S \cdot y)$. This in turn means that $S \cdot x \leftrightarrow S \cdot y$ in G_f. It follows that S is a graph isomorphism between G_f and G_g.

Linearity graphs thus allows a formulation of the problem where the other secret matrix T is no longer present. We have two (exponentially large) isomorphic graphs G_f and G_g, and we ultimately need to recover the whole isomorphism S. However, thanks to the dehomogenization technique of the previous section, and thanks to the ease with which inhomogeneous instances can be solved, it turns out that recovering just a little bit of information on the isomorphism is enough to find it completely. More precisely, we just need to know how the isomorphism S transforms one arbitrary vertex.

Of course, completely building these graphs is prohibitively expensive (they have q^n vertices). It turns out that this is never necessary, because it is possible to walk in these graphs without fully knowing them.

Walking in Linearity Graphs. The function $\psi(x,y) = f(x+y)+f(x)+f(y)$ is a generalization of the *polar form* of a quadratic form to vectors thereof, in characteristic two. It is easy to check that ψ is bilinear. Given a (non-zero) vertex $x \in (\mathbb{F}_q)^n$ in the graph, the function:

$$\mathrm{D}_x f : y \mapsto \psi(x,y) = f(x+y) + f(x) + f(y)$$

is a familiar object in multivariate cryptology, called the *Differential of f at x* [20, 15, 14, 22]. It is a linear function from $(\mathbb{F}_q)^n$ to $(\mathbb{F}_q)^n$, which is then conveniently represented by a matrix. The set of nodes adjacent to x in G_f is in fact the kernel of $\mathrm{D}_x f$. Note that x always belong to $\ker \mathrm{D}_x f$, because $x+x=0$. The main reason we chose to focus on the case where $q=2^e$ is that this fact is not true when q is not a power of two.

The matrix $\mathrm{D}_x f$ is easy to compute given f and x. If f is a (homogeneous) quadratic map, then it is in fact a vector of n quadratic forms, which can conveniently be described by a collection of n matrices $F_1, \dots, F_n$, that are interpreted as follows: $F_k[i,j]$ is the coefficient of $x_i x_j$ in the k-th component of f. If ${}^tM$ denotes the transpose of M, then the matrix representation of the differential of f at x is given by:

$$\mathrm{D}_x f = \left(x \cdot (F_1 + {}^tF_1) \Big| \dots \Big| x \cdot (F_n + {}^tF_n) \right).$$

Thus, given a vector x, finding the neighbors of x in G_f can be done in time $\mathcal{O}(n^3)$: computing the matrix $\mathrm{D}_x f$ requires n matrix-vector products, and determining its kernel classically takes $\mathcal{O}(n^3)$ operations. It is thus possible to crawl the linearity graphs by spending a polynomial number of elementary operations on each traversed vertex.

Structure in Linearity Graphs. Linearity graphs possess a rich structure, thanks to their algebraic origin. Recall that in G_f, two nodes x and y are adjacent if $\psi(x,y)=0$, where ψ is the symmetric bilinear map defined above. The bilinearity of ψ induces a lot of structure in G_f. For instance, we always have $\psi(x,x)=0$, and by bilinearity $\psi(\lambda x, \mu x) = \lambda\mu\psi(x,x) = 0$, so that the q multiples of a vector x form a clique in G_f. The set of all multiples of x are thus topologically indifferentiable (they all have the exact same neighborhood).

Furthermore, the same reasoning shows that if two vectors x and y are adjacent in G_f, then the set of q^2 linear combinations $\lambda x + \mu y$ form a clique in G_f of size q^2.

Degree Distribution. If a quadratic map f is randomly chosen (amongst the finite number of possibilities), then the resulting linearity graph G_f follows a certain —mostly unknown— probability distribution, and any property of G_f can be seen as a random variable. One of the most interesting properties of G_f is the distribution of the *degree* (*i.e.*, of the number of neighbors) of vertices in

G_f. This result is stated in terms of the probability that a random $n \times n$ matrix over $\mathbb{F}_q$ is invertible. We denote it by $\lambda(n)$:

$$\lambda(n) = \prod_{i=1}^{n} \left(1 - \frac{1}{q^i}\right)$$

Lemma 2 (theorem 2 in [15]). *Let $x \in (\mathbb{F}_q)^n$ be a non-zero vector, and $f : (\mathbb{F}_q)^n \to (\mathbb{F}_q)^n$ be a uniformly random quadratic map. Then $\mathrm{D}_x f$ is a uniformly random matrix vanishing over x. As a consequence, the probability that $\mathrm{D}_x f$ has a kernel of dimension $k \geq 1$ is:*

$$\frac{\lambda(n)\lambda(n-1)}{\lambda(k)\lambda(k-1)\lambda(n-k)} q^{-k(k-1)}$$

Because $\lambda(n)$ is a decreasing function of n that converges to a finite limit bounded away from zero, then the ratio of the λ-expressions lives in a small interval, independently of q, n and k, so that the probability is in fact of order $q^{-k(k-1)}$. Of course, over $\mathbb{F}_q$, a k-dimensional vector space contains q^k elements, so that if $\dim \ker \mathrm{D}_x f = k$, then the vertex x has q^k neighbors.

Sparsity. Computing the expectation and the variance of the degree is technical, but feasible:

$$\mathbb{E}\left[\text{degree}\right] = q - \frac{1}{q^{n-2}} \qquad\qquad \sigma^2 = q^2(q-1)\left(1 - \frac{q^2+1}{q^n} + \frac{q^2}{q^{2n}}\right)$$

Establishing these two expressions is somewhat technical, yet because both are sums of q-hypergeometric terms, they can be computed by “creative telescoping” thanks to the q-analog of Zeilberger’s algorithm [42]. It follows that the expected number of edges of G_f is essentially $q^{n+1}/2$. In other terms, G_f is a very sparse graph that has barely more edges than it has vertices.

Disconnecting Linearity Graphs. A linearity graph G_f is fully connected, because all vertices are adjacent to the “zero” vertex. This “zero” vertex is not very interesting (since it is adjacent to *every* other vertex), and, as a matter of fact, it even turns out to be a bit annoying. Thus, it seems that there is nothing to lose by removing it. In addition, we could also get rif of the self-loops ; they are useless since *every* vertex has one.

We thus denote by G_f^* the simple graph G_f in which the zero vertex has been removed, and where self-edges are removed. It is interesting to note that the resulting graph is no longer connected, and that there are in fact very many connected components. Indeed, if $\dim \ker \mathrm{D}_x f = 1$, then the only neighbors of x are its multiples, and x belong to a connected component of size $q-1$. Lemma 2 tells us that this happens with probability $\lambda(n)/\lambda(1)$, and this converges to a finite limit bounded away from zero when n goes to infinity. Thus, a constant fraction of the vertices belong to “small” connected components of size $q-1$. Working a bit on the λ functions reveals that this proportion grows like $1 - 1/q^2$.

4 Just Count Your Neighbors

It is well-know that if two graphs (V_1, E_1) and (V_2, E_2) are isomorphic, and if ρ is an isomorphism between them, then $u \in V_1$ and $\rho(u) \in V_2$ have the same *degree*, *i.e.*, the same number of neighbors. It follows that if $u \in V_1$ and $v \in V_2$ do not have the same degree, then they cannot be related by ρ.

We adapt this simple idea in the context of QMLE, under the form of Algorithm 2. The main idea in this algorithm is to target vertices in the linearity graphs of f and g that have a specific degree: we only look for a "right pair" $y = S \cdot x$ amongst vertices x, y that have a prescribed degree (chosen to optimise the complexity of the algorithm). The remaining of this section is devoted to establishing the properties of this algorithm, which are summarized in the following theorem.

Algorithm 2. First Birthday Based Algorithm

```
1: function SampleSet($h$)
2:     $L \leftarrow \emptyset$
3:     repeat
4:         repeat
5:             $x \leftarrow$ random vertex of $G_h$
6:         until $x$ has $q^{\sqrt{n/3}}$ neighbors
7:         $L \leftarrow L \cup \{x\}$
8:     until $|L| = \sqrt{2}q^{n/3}$
9:     return $L$
10: function Neighbor-Counting-QMLE($f, g$)
11:     $U \leftarrow$ SampleSet$(f)$
12:     $V \leftarrow$ SampleSet$(g)$
13:     for all $(x, y) \in U \times V$ do
14:         $f'(z) \leftarrow f(z + y)$
15:         $g'(z) \leftarrow g(z + x)$
16:         query IQMLE-Solver with $(f', g')$
17:         if solution $(S, T)$ found then return $(S, T)$
18:     return "Probably not equivalent"
```

Theorem 1. *Algorithm 2 performs $\mathcal{O}\left(q^{2n/3}\right)$ units of computations on average, sends at most $q^{2n/3}$ queries to the inhomogeneous solver, and succeeds with probability $1 - 1/e$.*

The helper function SampleSet returns a set of $\mathcal{O}\left(q^{n/3}\right)$ vertices of G_f (resp. G_g), each having $q^{\sqrt{n/3}}$ neighbors in the graph. It follows that there are $q^{2n/3}$ queries to the inhomogeneous solver, because this is the size of the cartesian product $U \times V$.

It remains to establish the complexity of SampleSet, and the success probability of the algorithm. As explained above, since we are looking for a "right pair” $y = S \cdot x$, it is safe to restrict our attention to vertices x, y that have a specific degree (as long as vertices with such a degree exist in the graphs).

Lemma 2 gives us the expected number iterations of the innermost loop of SampleSet that are required to find a random vertex with the required degree. Up to a constant factor, finding a vertex with degree q^k requires $q^{k(k-1)}$ trials, so that finding each new random vertex requires $\mathcal{O}\left(q^{n/3}\right)$ rank computations on $n \times n$ matrices, hence $\mathcal{O}\left(n^3 \cdot q^{n/3}\right)$ operations.

Lemma 2 also tells us that there are on average $q^{n-k(k-1)}$ vertices in G_f each having degree q^k. In Algorithm 2 we look specifically at vertices of degree $q^{\sqrt{n/3}}$, and we thus expect G_f to contain $q^{2n/3}$ of them. Since the number of iterations of the outermost **repeat**...**until** loop is roughly the square root of this number, we do not expect more than a constant number of “extra” iterations finding an already-known vector x. Putting everything together, we conclude that SampleSet terminates after $\mathcal{O}\left(n^3 q^{2n/3}\right)$ operations.

Now, the birthday bound tells us that $U \times V$ contains a “right pair” $y = Sx$ with probability greater than 63%, because both U and V contain about the square root of the total number of vertices with degree $q^{\sqrt{n/3}}$ (see [39] for a precise statement of this specific version of the birthday paradox).

Practical Results. We have implemented Algorithm 2 inside the MAGMA computer algebra system[8], running on one core of a 2.8 Ghz Xeon machine. As shown in Table 1, we found out that in practice it is difficult to balance the cost of building U and V on the one hand, and going through the candidate pair on the other hand, because the target degree can only take $\sqrt{n}$ integer values. We could nevertheless verify in practice that the complexity of building the lists and the expected number of right pairs in them is consistent with our expectations. The source code is in the public domain, and is available on the webpage of the first author. It uses an unpublished algorithm to solve the inhomogeneous instances.

Table 1. Experimental results on Algorithm 2

n	q	generating U and V	total time	$\log_q$ (target degree)	$\|U\|$	# pairs
16	2	0s	68s	3	1	4
22	2	28s	9h45m	4	13	400
28	2	4913s	2h15m	5	8	64

5 Map Your Neighborhood

We have seen in section 3 that the linearity graphs, once deprived from the “zero” vertex, contain many small connected components. Of course, if $y = Sx$, then the connected component of x is isomorphic to the connected component of y. In this section, we describe an algorithm that builds upon this idea—instead of just looking at immediate neighbors, as we did in algorithm 2, we now try to look at the whole connected component, in order to distinguish between vertices of the same degree.

Canonical Graph Labeling. Given a graph G, a *Canonical Labeling algorithm* relabels the vertices of G, thus producing a graph $Canon(G)$, which is by definition isomorphic to G. The result is canonical in the sense that if G and H are isomorphic graphs, then $Canon(G) = Canon(H)$. The canonical labels are therefore complete invariants of the isomorphism class, and as such, computing a canonical labeling is necessarily harder than checking if two graphs are isomorphic. However, computing a canonical labeling can be done in average linear time [3], because except for an exponentially small fraction of all graphs, it can be done with a very simple linear algorithm.

Back to our more specific problem, let us denote by C_x (resp C_y) the connected component of x in G_f^* (resp. of y in G_g^*). The key idea of the algorithm presented in this section is that $y = Sx$ implies $Canon(C_x) = Canon(C_y)$. Thus, it seems that the function $H : u \mapsto Canon(C_u)$ could be used as a "hash function". In fact, in algorithm 2, we used the degree as such a "hash function", but it was not very discriminating, because the degree does not contain enough entropy. We hope that H behaves as a good hash function, and that *false positives*, *i.e.*, pairs (x, y) such that $H(x) = H(y)$ but $y \neq Sx$, should be very rare.

One problem is that H does not distinguish between vertices of the same connected component. To improve it, we would need a way to single out a specific vertex in the connected component. Fortunately, most canonical labeling algorithm return the isomorphism (say ρ) between their argument G and $Canon(G)$. To single a vertex x out in G, it is sufficient to send $\rho(x)$ along with the canonical labeling of G.

A Canonical-Labeling-Based Algorithm. As discussed in section 3, G_f^* contains many small connected components that are all isomorphic to each others, since they are all cliques of size $q - 1$. Therefore, if we want our "hash function" to be discriminating, we must avoid small connected components. Our "hash function" will thus reject the vector x if there is no simple path starting from x and of length at least r. In the other direction, we cannot exclude the existence of a giant connected component of exponential size. Therefore, we only consider the radius-r neighborhood of the vertex x we are interested in, *i.e.*, the set of all vertices that can be reached from x by crossing at most r edges. This is the basis of algorithm 3.

Remarks on Algorithm 3. Establishing the complexity and success probability of algorithm 3 is surprisingly difficult, probably because is relies on topological properties of G_f^*, which is a somewhat random but very structured graph.

Algorithm 3 has been written in a generic way, independently of the actual value of q. However, we have only been able to discuss its properties when $q = 2$. We have verified that the algorithm works as we expected in this case, but the situation when $q \neq 2$ is not so clear. We tend to believe that the complexity and/or success probability degrade exponentially fast when q grows, but we fall short of definitive conclusion.

Algorithm 3. Canonical Labeling/Birthday Based Algorithm

```
1: function $\textsc{Hashable}^{[r]}(G, x)$
2:     Perform a Breadth-First Search in $G$ starting from $x$
3:     return $\textsc{True}$ if the BFS hits a vertex $r$ edges away from $x$

4: function $\mathrm{H}^{[r]}(G, x)$
5:     $C_x \leftarrow$ subgraph of $G$ formed by all vertices at most $r$ edges away from $x$.
6:     $\rho, \mathcal{G} \leftarrow \textsc{CanonicalLabeling}(C_x)$
7:     return $\left(\mathcal{G}, \rho(x)\right)$

8: function $\textsc{SampleHashTable}(h)$
9:     $L \leftarrow \emptyset$
10:    repeat
11:        repeat
12:            $x \leftarrow$ random vertex of $G_h^*$
13:        until $\textsc{Hashable}^{[r]}(G_h^*, x)$
14:        $L\left[\mathrm{H}^{[r]}\left(G_h^*, x\right)\right] \leftarrow x$
15:    until $|L| = \sqrt{2} q^{n/2}$
16:    return $L$

17: function $\textsc{Canonical-Labeling-QMLE}(f, g)$
18:    $U \leftarrow \textsc{SampleHashTable}\,(f)$
19:    $V \leftarrow \textsc{SampleHashTable}\,(g)$
20:    for all $(h_1 \mapsto x) \in U, (h_2 \mapsto y) \in V$ such that $h_1 = h_2$ do
21:        $f'(z) \leftarrow f(z + y)$
22:        $g'(z) \leftarrow g(z + x)$
23:        query IQMLE-$\textsc{Solver}$ with $(f', g')$
24:        if solution $(S, T)$ found then return $(S, T)$
25:    return "Probably not equivalent"
```

When $q = 2$, the structure of G_f^* seems to be richer. For instance, we already alluded to the fact that the fraction of nodes whose connected component is of size only $q - 1$, grows like $1 - 1/q^2$. In addition, as we will see in the next section, setting $q = 2$ allows us to turn most more-or-less-random graphs into trees, which are much easier to deal with.

Preliminary Analysis of Algorithm 3. When $q = 2$, the correctness of the algorithm is implied by the following three heuristic statements.

Claim. *i)* $\textsc{Hashable}^{[r]}\left(G_f^*, x\right)$ is true with probability $\approx 1/r$ over the random choice of f (assuming $x \neq 0$).

ii) Both $\textsc{Hashable}^{[r]}$ and $\mathrm{H}^{[r]}$ can be evaluated in expected time $\mathcal{O}\left(rn^3\right)$.

iii) When restricted to elements that are $\textsc{Hashable}^{[r]}$, then $\mathrm{H}^{[r]}\left(G_f^*, \cdot\right)$ is an ε^r-*almost universal hash function family* (indexed by f) for some $\varepsilon < 1$.

The notion of almost universal hash function is usually useful when the hash function is "less injective" than a random function. In this paper though, $\mathrm{H}^{[r]}$ can become *more injective* than a random function, as soon as r becomes sufficiently large.

It follows from claim i that the expected number of iterations of the loop of lines 11–13 is $\mathcal{O}(r)$, and it follows from claim ii that finding one admissible vector x requires $\mathcal{O}\left(r^2 n^3\right)$ operations on average. Claim iii then guarantees that if we choose r to be a bit larger than n, then the probability to find hash collisions can be made smaller than 2^{-n}, and standard birthday-type results guarantee that the number of expected hash collisions in the execution of SampleHashTable is constant. From this, we conclude that SampleHashTable runs in expected time $\mathcal{O}\left(r^2 n^3 q^{n/2}\right)$.

It follows from the birthday paradox [39] that there is a "right pair" in $U \times V$, *i.e.*, a pair (x, y) with $y = Sx$, with probability greater than $1 - 1/e$. This is because $(\mathbb{F}_q)^n$ has q^n elements and that the sizes of both U and V are essentially $q^{n/2}$. This guarantees the success probability of the algorithm.

Let us denote by $\mathcal{N}$ the number of bogus inhomogeneous queries, *i.e.*, the number of pairs $x \neq y \in U \times V$ with the same hash. It follows from Markov's inequality and claim iii that $\mathbb{P}\left[\mathcal{N} \geq 1\right] \leq 2q^n \cdot \varepsilon^r$. Thus, as soon as r is asymptotically larger than n, *e.g.* $r = n \log \log n$, then the probability that $\mathcal{N} \geq 1$ gets exponentially small. This concludes our preliminary analysis: algorithm 3 runs in time $\mathcal{O}\left(n^5 q^{n/2}\right)$, and sends a constant number of inhomogeneous queries. It now remains to show that our claims are valid, but we first find it reassuring to show that the practical behavior of the algorithm is very consistent with our expectations.

Discussion of the Claims. Because of space limitations, we discuss these claims in the extended version of the paper [10].

Experimental Results. We have implemented Algorithm 3 using the MAGMA computer algebra system [8], and we found out that it works well in practice, as Table 2 shows. The experiment clearly shows that $\mathcal{N}$ is constant, as expected. This justify our heuristic analysis *a posteriori*. The implementation is in the public domain and is available on the webpage of the first author.

Table 2. Experimental results on Algorithm 3

n	q	generating U and V	finding collisions	$\lvert U \rvert$	$\mathcal{N}$
16	2	3.6 s	1s	64	6
24	2	123 s	13s	836	5
32	2	61 min	200s	11585	2
40	2	31 h	2h	165794	7

References

1. Agrawal, M., Saxena, N.: Equivalence of f-algebras and cubic forms. In: Durand, B., Thomas, W. (eds.) STACS 2006. LNCS, vol. 3884, pp. 115–126. Springer, Heidelberg (2006)
2. Alon, N., Blais, E.: Testing boolean function isomorphism. In: Serna, M.J., Shaltiel, R., Jansen, K., Rolim, J.D.P. (eds.) APPROX 2010. LNCS, vol. 6302, pp. 394–405. Springer, Heidelberg (2010)
3. Babai, L., Kucera, L.: Canonical labelling of graphs in linear average time. In: FOCS, pp. 39–46. IEEE Computer Society (1979)
4. Bardet, M., Faugère, J.C., Salvy, B.: On the complexity of Gröbner basis computation of semi-regular overdetermined algebraic equations. In: Proc. International Conference on Polynomial System Solving (ICPSS), pp. 71–75 (2004)
5. Bettale, L., Faugère, J.-C., Perret, L.: Cryptanalysis of the trms signature scheme of pkc'05. In: Vaudenay, S. (ed.) AFRICACRYPT 2008. LNCS, vol. 5023, pp. 143–155. Springer, Heidelberg (2008)
6. Billet, O., Gilbert, H.: A traceable block cipher. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 331–346. Springer, Heidelberg (2003)
7. Biryukov, A., Cannière, C.D., Braeken, A., Preneel, B.: A toolbox for cryptanalysis: Linear and affine equivalence algorithms. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 33–50. Springer, Heidelberg (2003)
8. Bosma, W., Cannon, J.J., Playoust, C.: The Magma Algebra System I: The User Language. J. Symb. Comput. 24(3/4), 235–265 (1997)
9. Bouillaguet, C., Faugère, J.-C., Fouque, P.-A., Perret, L.: Practical cryptanalysis of the identification scheme based on the isomorphism of polynomial with one secret problem. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 473–493. Springer, Heidelberg (2011)
10. Bouillaguet, C., Fouque, P.A., Véber, A.: Graph-theoretic algorithms for the "isomorphism of polynomials" problem. Cryptology ePrint Archive, Report 2012/607 (2012), `http://eprint.iacr.org/`
11. Cramer, R. (ed.): EUROCRYPT 2005. LNCS, vol. 3494. Springer, Heidelberg (2005)
12. Daemen, J.: Limitations of the even-mansour construction. In: [25], pp. 495–498
13. Ding, J., Wolf, C., Yang, B.-Y.: ℓ-Invertible cycles for multivariate quadratic public key cryptography. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 266–281. Springer, Heidelberg (2007)
14. Dubois, V., Fouque, P.-A., Shamir, A., Stern, J.: Practical Cryptanalysis of SFLASH. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 1–12. Springer, Heidelberg (2007)
15. Dubois, V., Granboulan, L., Stern, J.: An efficient provable distinguisher for hfe. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4052, pp. 156–167. Springer, Heidelberg (2006)
16. Dunkelman, O., Keller, N., Shamir, A.: Minimalism in cryptography: The even-mansour scheme revisited. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 336–354. Springer, Heidelberg (2012)
17. Even, S., Mansour, Y.: A construction of a cioher from a single pseudorandom permutation. In: [25], pp. 210–224
18. Faugère, J.-C., Joux, A., Perret, L., Treger, J.: Cryptanalysis of the hidden matrix cryptosystem. In: Abdalla, M., Barreto, P.S.L.M. (eds.) LATINCRYPT 2010. LNCS, vol. 6212, pp. 241–254. Springer, Heidelberg (2010)

19. Faugère, J.-C., Perret, L.: Polynomial Equivalence Problems: Algorithmic and Theoretical Aspects. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 30–47. Springer, Heidelberg (2006)
20. Fouque, P.A., Granboulan, L., Stern, J.: Differential cryptanalysis for multivariate schemes. In: [11], pp. 341–353
21. Fouque, P.-A., Macario-Rat, G., Perret, L., Stern, J.: Total break of the ℓ-ic signature scheme. In: Cramer, R. (ed.) PKC 2008. LNCS, vol. 4939, pp. 1–17. Springer, Heidelberg (2008)
22. Fouque, P.-A., Macario-Rat, G., Stern, J.: Key Recovery on Hidden Monomial Multivariate Schemes. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 19–30. Springer, Heidelberg (2008)
23. Geiselmann, W., Meier, W., Steinwandt, R.: An Attack on the Isomorphisms of Polynomials Problem with One Secret. Int. J. Inf. Sec. 2(1), 59–64 (2003)
24. Goldreich, O., Micali, S., Wigderson, A.: Proofs that yield nothing but their validity and a methodology of cryptographic protocol design (extended abstract). In: FOCS, pp. 174–187. IEEE (1986)
25. Matsumoto, T., Imai, H., Rivest, R.L. (eds.): ASIACRYPT 1991. LNCS, vol. 739. Springer, Heidelberg (1993)
26. Joux, A., Kunz-Jacques, S., Muller, F., Ricordel, P.M.: Cryptanalysis of the tractable rational map cryptosystem. In: [40], pp. 258–274
27. Kayal, N.: Efficient algorithms for some special cases of the polynomial equivalence problem. In: Randall, D. (ed.) SODA, pp. 1409–1421. SIAM (2011)
28. Patarin, J.: Hidden fields equations (HFE) and isomorphisms of polynomials (IP): Two new families of asymmetric algorithms. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 33–48. Springer, Heidelberg (1996)
29. Patarin, J., Goubin, L., Courtois, N.T.: C_{-+}^{*} and HM: Variations around two schemes of T. Matsumoto and H. Imai. In: Ohta, K., Pei, D. (eds.) ASIACRYPT 1998. LNCS, vol. 1514, pp. 35–50. Springer, Heidelberg (1998)
30. Patarin, J., Goubin, L., Courtois, N.T.: Improved Algorithms for Isomorphisms of Polynomials. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 184–200. Springer, Heidelberg (1998)
31. Patarin, J., Goubin, L., Courtois, N.: Improved Algorithms for Isomorphisms of Polynomials – Extended Version (1998), `http://minrank.org/ip6long.pdf`
32. Perret, L.: A Fast Cryptanalysis of the Isomorphism of Polynomials with One Secret Problem. In: [11], pp. 354–370
33. Pointcheval, D.: A new identification scheme based on the perceptrons problem. In: Guillou, L.C., Quisquater, J.-J. (eds.) EUROCRYPT 1995. LNCS, vol. 921, pp. 319–328. Springer, Heidelberg (1995)
34. Sakumoto, K.: Public-key identification schemes based on multivariate cubic polynomials. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 172–189. Springer, Heidelberg (2012)
35. Sakumoto, K., Shirai, T., Hiwatari, H.: Public-key identification schemes based on multivariate quadratic polynomials. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 706–723. Springer, Heidelberg (2011)
36. Shamir, A.: An efficient identification scheme based on permuted kernels (extended abstract). In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 606–609. Springer, Heidelberg (1990)
37. Stern, J.: A new identification scheme based on syndrome decoding. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 13–21. Springer, Heidelberg (1994)

38. Stern, J.: Designing identification schemes with keys of short size. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 164–173. Springer, Heidelberg (1994)
39. Vaudenay, S.: A Classical Introduction to Cryptography: Applications for Communications Security. Springer-Verlag New York, Inc., Secaucus (2005)
40. Vaudenay, S. (ed.): PKC 2005. LNCS, vol. 3386. Springer, Heidelberg (2005)
41. Wang, L.C., Hu, Y.H., Lai, F., Yen Chou, C., Yang, B.Y.: Tractable rational map signature. In: [40], pp. 244–257
42. Wilf, H., Zeilberger, D.: An algorithmic proof theory for hypergeometric (ordinary and "q") multisum/integral identities. Inventiones Mathematicae 108, 575–633 (1992), 10.1007/BF02100618

Cryptanalysis of Full RIPEMD-128

Franck Landelle[1] and Thomas Peyrin[2,*]

[1] DGA MI, France
[2] Division of Mathematical Sciences, School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore
`landelle.franck@laposte.net, thomas.peyrin@gmail.com`

Abstract. In this article we propose a new cryptanalysis method for double-branch hash functions that we apply on the standard RIPEMD-128, greatly improving over know results. Namely, we were able to build a very good differential path by placing one non-linear differential part in each computation branch of the RIPEMD-128 compression function, but not necessarily in the early steps. In order to handle the low differential probability induced by the non-linear part located in later steps, we propose a new method for using the freedom degrees, by attacking each branch separately and then merging them with free message blocks. Overall, we present the first collision attack on the full RIPEMD-128 compression function as well as the first distinguisher on the full RIPEMD-128 hash function. Experiments on reduced number of rounds were conducted, confirming our reasoning and complexity analysis. Our results show that 16 years old RIPEMD-128, one of the last unbroken primitives belonging to the MD-SHA family, might not be as secure as originally thought.

Keywords: RIPEMD-128, collision, distinguisher, hash function.

1 Introduction

Recent impressive progresses in hash function cryptanalysis [25,28,29,27] led to the fall of most standardized primitives, such as MD4, MD5, SHA-0 and SHA-1. All these algorithms share the same design rationale for their compression functions (i.e. they incorporate additions, rotations, xors and boolean functions in an unbalanced Feistel network), and we usually refer to them as the MD-SHA family. As of today, among this family only SHA-2, RIPEMD-128 and RIPEMD-160 remain unbroken.

The notation RIPEMD represents several distinct hash functions related to the MD-SHA-family, the first representative being RIPEMD-0 [2] that was recommended in 1992 by the European *RACE Integrity Primitives Evaluation* (RIPE) consortium. Its compression function basically consists in two MD4-like [20] functions computed in parallel (but with different constant additions for the two branches), with 48 steps in total. Early cryptanalysis by Dobbertin on a reduced version

[*] Supported by the Singapore National Research Foundation Fellowship 2012 (NRF-NRFF2012-06).

of the compression function [9] seemed to indicate that RIPEMD-0 was a weak function and this was fully confirmed much later by Wang *et al.* [27] who showed that one can find a collision for the full RIPEMD-0 hash function with as few as 2^{16} computations.

However, in 1996, due to the cryptanalysis advances on MD4 and on the compression function of RIPEMD-0, the original RIPEMD-0 was reinforced by Dobbertin, Bosselaers and Preneel [10] to create two stronger primitives RIPEMD-128 and RIPEMD-160, with 128/160-bit output and 64/80 steps respectively (two other less known 256 and 320-bit output variants RIPEMD-256 and RIPEMD-320 were also proposed, but with a claimed security level equivalent to an ideal hash function with a twice smaller output size). The main novelty compared to RIPEMD-0 is that the two computation branches were made much more distinct by using not only different constants, but also different rotation values and boolean functions, which greatly hardens the attacker's task in finding good differential paths for both branches at a time. The security seems to have indeed increased since as of today no attack is known on the full RIPEMD-128 or RIPEMD-160 compression/hash functions and the two primitives are worldwide ISO/IEC standards [12].

Even though no result is known on the full RIPEMD-128 and RIPEMD-160 compression/hash functions yet, many analysis were conducted in the recent years. In [17], a preliminary study checked up to what extent can the known attacks [27] on RIPEMD-0 apply to RIPEMD-128 and RIPEMD-160. Then, following the extensive work on preimage attacks for MD-SHA family, [21,19,26] describe high complexity preimage attacks on up to 36 steps of RIPEMD-128 and 31 steps of RIPEMD-160. Collision attacks were considered in [16] for RIPEMD-128 and in [15] for RIPEMD-160, with 48 and 36 steps broken respectively. Finally, distinguishers based on non-random properties such as second-order collisions are given in [16,22,15], reaching about 50 steps with a very high complexity.

Our Contributions. In this article, we introduce a new type of differential path for RIPEMD-128 using one non-linear differential trail for both left and right branches and, in contrary to previous work, not necessarily located in the early steps (Section 3). The important differential complexity cost of these two parts is mostly avoided by using the freedom degrees in a novel way: some message words are used to handle the non-linear parts in both branches and the remaining ones are used to merge the internal states of the two branches (Section 4). Overall, we obtain the first cryptanalysis of the full 64-round RIPEMD-128 hash and compression functions. Namely, we provide a distinguisher based on a differential property for both the full 64-round RIPEMD-128 compression function and hash function (Section 5). Previously best-known results for non-randomness properties only applied to 52 steps of the compression function, 48 steps of the hash function. More importantly, we also derive a semi-free-start (SFS) collision attack on the full RIPEMD-128 compression function (Section 5), significantly improving the previous free-start (FS) collision attack on 48 steps. Any further improvement of our techniques is likely to provide a practical SFS collision attack on the RIPEMD-128 compression function. In order to increase the confidence in

Table 1. Summary of known and new results on RIPEMD-128 hash function

Function	Size	Key Setting	Target	#Steps	Complexity	Ref.
RIPEMD-128	128	comp. function	preimage	35	2^{112}	[19]
RIPEMD-128	128	hash function	preimage	35	2^{121}	[19]
RIPEMD-128	128	hash function	preimage	36	$2^{126.5}$	[26]
RIPEMD-128	128	comp. function	collision	48	2^{40}	[16]
RIPEMD-128	**128**	**comp. function**	**collision**	**60**	$2^{57.57}$	**new**
RIPEMD-128	**128**	**comp. function**	**collision**	**63**	$2^{59.91}$	**new**
RIPEMD-128	**128**	**comp. function**	**collision**	**Full**	$2^{61.57}$	**new**
RIPEMD-128	128	hash function	collision	38	2^{14}	[16]
RIPEMD-128	128	comp. function	non-rand.	52	2^{107}	[22]
RIPEMD-128	**128**	**comp. function**	**non-rand.**	**Full**	$2^{59.57}$	**new**
RIPEMD-128	128	hash function	non-rand.	48	2^{70}	[16]
RIPEMD-128	**128**	**hash. function**	**non-rand.**	**Full**	$2^{105.40}$	**new**

our reasoning, we implemented independently the two main parts of the attack (the merge and the probabilistic part) and the observed complexity matched our predictions. Our results and previous works complexities are given in Table 1 for comparison.

2 Description of RIPEMD-128

RIPEMD-128 [10] is a 128-bit hash function that uses the Merkle-Damgård construction as domain extension algorithm: the hash function H is built by iterating a 128-bit compression function h that takes as input a 512-bit message block m_i and a 128-bit chaining variable cv_i: $cv_{i+1} = h(cv_i, m_i)$, where the message m to hash is padded beforehand to a multiple of 512 bits[1] and the first chaining variable is set to a predetermined initial value $cv_0 = IV$.

We refer to [10] for a complete description of RIPEMD-128. In the rest of this article, we denote by $[Z]_i$ the i-th bit of a word Z, starting the counting from 0. $\boxplus$ and $\boxminus$ represent the modular addition and subtraction on 32 bits, and $\oplus$, $\vee$, $\wedge$, the bitwise "exclusive or", the bitwise "or", and the bitwise "and" function respectively.

2.1 RIPEMD-128 Compression Function

The RIPEMD-128 compression function is based on MD4, with the particularity that it uses two parallel instances of it. We differentiate these two computation branches by left and right branch and we denote by X_i (resp. Y_i) the 32-bit word of left branch (resp. right branch) that will be updated during step i of the compression function. The process is composed of 64 steps divided into 4 rounds of 16 steps each in both branches.

Initialization. The 128-bit input chaining variable cv_i is divided into 4 words h_i of 32 bits each, that will be used to initialize the left and right branch 128-bit

[1] The padding is the same as for MD4: a "1" is first appended to the message, then x "0" bits (with $x = 512 - (|m| + 1 + 64 \pmod{512})$) are added, and finally the message length $|m|$ coded on 64 bits is appended as well.

internal state: $X_{-3} = h_0$, $X_{-2} = h_1$, $X_{-1} = h_2$, $X_0 = h_3$, $Y_{-3} = h_0$, $Y_{-2} = h_1$, $Y_{-1} = h_2$, $Y_0 = h_3$

The Message Expansion. The 512-bit input message block is divided into 16 words M_i of 32 bits each. Each word M_i will be used once in every round in a permuted order (similarly to MD4) and for both branches. We denote by W^l_i (resp. W^r_i) the 32-bit expanded message word that will be used to update the left branch (resp. right branch) during step i. We have for $0 \leq j \leq 3$ and $0 \leq k \leq 15$: $W^l_{j\cdot 16+k} = M_{\pi^l_j(k)}$ and $W^r_{j\cdot 16+k} = M_{\pi^r_j(k)}$, where π^l_j and π^r_j are permutations.

The Step Function. At every step i, the registers X_{i+1} and Y_{i+1} are updated with functions f^l_j and f^r_j that depends on the round j in which i belongs:

$$X_{i+1} = (X_{i-3} \boxplus \Phi^l_j(X_i, X_{i-1}, X_{i-2}) \boxplus W^l_i \boxplus K^l_j)^{\lll s^l_i},$$
$$Y_{i+1} = (Y_{i-3} \boxplus \Phi^r_j(Y_i, Y_{i-1}, Y_{i-2}) \boxplus W^r_i \boxplus K^r_j)^{\lll s^r_i},$$

where K^l_j, K^r_j are 32-bit constants defined for every round j and every branch, s^l_i, s^r_i are rotation constants defined for every step i and every branch, Φ^l_j, Φ^r_j are 32-bit boolean functions defined for every round j and every branch. All these constants and functions, as well as the IV and the permutations π^l_j and π^r_j can be found in the original RIPEMD-128 documentation [10].

The Finalization. A finalization and a feed-forward is applied when all 64 steps have been computed in both branches. The four 32-bit words h'_i composing the output chaining variable are finally obtained by: $h'_0 = X_{63} \boxplus Y_{62} \boxplus h_1$, $h'_1 = X_{62} \boxplus Y_{61} \boxplus h_2$, $h'_2 = X_{61} \boxplus Y_{64} \boxplus h_3$, $h'_3 = X_{64} \boxplus Y_{63} \boxplus h_0$

3 A New Family of Differential Paths for RIPEMD-128

3.1 The General Strategy

The first task for an attacker looking for collisions in some compression function is to set a good differential path. In the case of RIPEMD and more generally double or multi-branches compression functions, this can be quite a difficult task because the attacker has to find a good path for all branches at the same time. This is exactly what multi-branches functions designers are hoping: it is unlikely that good differential paths exist in both branches at the same time when the branches are made distinct enough (note that the weakness of RIPEMD-0 is that both branches are almost identical and the same differential path can be used for the two branches at the same time).

Differential paths in recent collision attacks on MD-SHA family are composed of two parts: a low probability non-linear part in the first steps and a high probability linear part in the remaining ones. Only the latter will be handled probabilistically and impact the overall complexity of the collision finding algorithm, since during the first steps the attacker can choose message words independently.

This strategy proved to be very effective because it allows to find much better linear parts than before by relaxing many constraints on them. The previous approaches for attacking `RIPEMD-128` [17,16] are based on the same strategy, building good linear paths for both branches, but without including the first round (i.e. the first 16 steps). The first round in each branch will be covered by a non-linear differential path and this is depicted left in Figure 1. The collision search is then composed of two subparts, the first handling the low-probability non-linear paths with the message blocks (step ① and then the remaining steps in both branches are verified probabilistically (step ②).

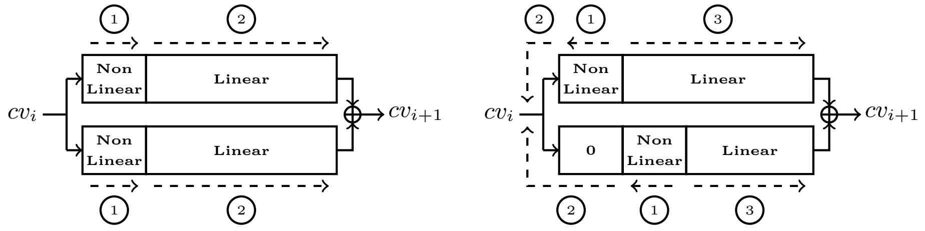


Fig. 1. The previous (left-hand side) and new (right-hand side) approach for collision search on double-branch compression functions

This differential path search strategy is natural when one will handle the non-linear parts in a classic way (i.e. computing only forward) during the collision search, but in Section 4 we will describe a new approach for using the available freedom degrees provided by the message words in double-branch compression functions (see right in Figure 1): instead of handling the first rounds of both branches at the same time during the collision search, we will satisfy them independently (step ①), then use some remaining free message words to merge the two branches (step ②) and finally handle the remaining steps in both branches probabilistically (step ③). This new approach broadens the search area of good linear differential parts, and provides us better candidates in the case of `RIPEMD-128`.

3.2 Finding a Good Linear Part

Since any active bit in a linear differential path (i.e. a bit containing a difference) is likely to cause many conditions in order to control its spread, most successful collision searches start with a low-weight linear differential path, therefore reducing the complexity as much as possible. `RIPEMD-128` is no exception, and because every message word is used once in every round of every branch in `RIPEMD-128`, the best would be to insert only a single-bit difference in one of them. This was considered in [16], but the authors concluded that none of all single-word differences leads to a good choice and they eventually had to utilize one active bit in two message words instead, therefore doubling the amount of differences inserted during the compression function computation and reducing the overall number of steps they could attack. By relaxing the constraint that both non-linear parts must necessarily be located in the first round, we show that a single-word difference in M_{14} is actually a very good choice.

Boolean Functions. Analyzing the various boolean functions in `RIPEMD-128` rounds is very important. Indeed, there are three distinct functions: `XOR`, `ONX` and `IF`, with all very distinct behavior. The function `IF` is non-linear and can absorb differences (one difference on one of its input can be blocked from spreading to the output by setting some appropriate bit value conditions). In other words, one bit difference in the internal state during an `IF` round can be forced to create only a single bit difference 4 steps later, thus providing no diffusion at all. In the contrary, `XOR` is arguably to most problematic function in our situation because it can not absorb any difference. Thus, one bit difference in the internal state during an `XOR` round will double the number of bit differences every step and quickly lead to an unmanageable amount of conditions. Moreover, the linearity of the `XOR` function makes it problematic when using the non-linear part search tool that strongly leverages non-linear behavior to obtain a solution. In between, the `ONX` function is non-linear for two inputs and can absorb difference up to some extent. We can easily conclude that the goal for the attacker will be to locate the biggest proportion of differences in the `IF` or if needed in the `ONX` functions, and try to avoid the `XOR` parts as much as possible.

Choosing a Message Word. We would like to find the best choice for the single-message word difference insertion. The `XOR` function located in the 4th round of right branch must be avoided, so we are looking for a message word that is incorporated either very early (so we can propagate the difference backward) or very late (so we can propagate the difference forward) in this round. Similarly, the `XOR` function located in the 1st round of left branch must be avoided, so we are looking for a message word that is incorporated either very early (for a FS collision attack) or very late (for a SFS collision attack) in this round as well. It is easy to check that M_{14} is a perfect candidate, being the last inserted in 4th round of right branch and the second-to-last in 1st round of left branch.

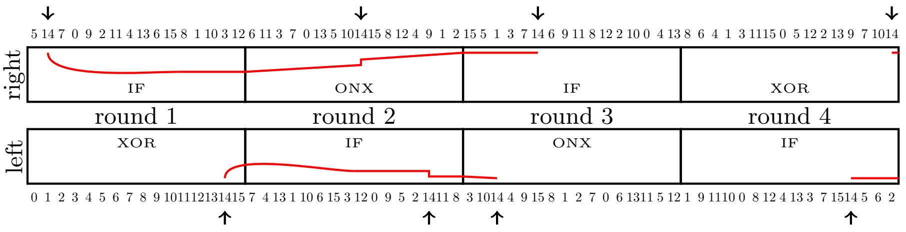


Fig. 2. The shape of our differential path for `RIPEMD-128`. The numbers are the message words inserted at each step and the red curves represent the rough amount differences in the internal state during each steps. The arrows show where the bit differences are injected with M_{14}.

Building the Linear Part. Once we chose that the only message difference will be a single bit in M_{14}, we need to build the whole linear part of the differential path in the internal state. By linear we mean that all modular additions will be modeled as a bitwise XOR function. Moreover, when a difference is input of a boolean function, it is absorbed when possible in order to remain as low weight as possible (though, for a few special bit positions it might be more interesting to not absorb the difference if it can erase another difference in later steps). We give the rough skeleton of our differential path in Figure 2. Both differences inserted in the 4^{th} round of left and right branches are simply propagated forward for a few steps and we are very lucky that this linear propagation leads to two final internal states whose difference can be mutually erased after application of the compression function finalization and feed-forward. All differences inserted in the 3^{rd} and 2^{nd} rounds of left and right branches are propagated linearly backward and will be later connected to the bit difference inserted in the 1^{st} round by the non-linear part. Note that since a non-linear part usually has a low differential probability, we will try to make it as thin as possible. No difference will be present in the input chaining variable, so the trail is well suited for a SFS collision attack. We had to choose the bit position for the message M_{14} difference insertion and among the 32 possible choices, the most significant bit was selected because it is the one maximizing the differential probability of the linear part we just built (this finds an explanation by the fact that at the most significant bit position many conditions due to carry control in modular additions are avoided).

3.3 The Non-linear Differential Part Search Tool

Finding non-linear differential path is a very complex task, but we implemented a tool similar to [4] for SHA-1 in order to perform this task in an automated way. Since RIPEMD-128 also belongs to the MD-SHA family, the original technique works well, in particular when used in a round with a non-linear boolean function such as IF.

We have to find one non-linear part in each branch and note that they can be handled independently. We included the special constraint that the non-linear parts should be as thin as possible (i.e. spreading on the fewer possible amount of steps), so as to later reduce the overall complexity (linear parts have higher differential probability than non-linear ones).

3.4 The Final Differential Path Skeleton

Applying our non-linear part search tool and reusing notations from [4], we obtain the differential path in Figure 3, for which we provide at each step i the differential probability $\mathrm{P}^l[i]$ and $\mathrm{P}^r[i]$ of left and right branch respectively. Also, we give for each step i the accumulated probability $\mathrm{P}[i]$ starting from last step, i.e. $\mathrm{P}[i] = \prod_{j=63}^{j=i}(\mathrm{P}^r[j] \cdot \mathrm{P}^l[j])$.

One can check that the trail has differential probability $2^{-85.09}$ (that is $\prod_{i=0}^{63} \mathrm{P}^l[i] = 2^{-85.09}$) in the left branch and 2^{-145} (i.e. $\prod_{i=0}^{63} \mathrm{P}^r[i] = 2^{-145}$) in the right branch. Its overall differential probability is $2^{-230.09}$ and since we have 511 bits of message with unspecified value (one bit of M_4 is already set to "1"), plus 127 unrestricted bits of chaining variable (one bit of $X_0 = Y_0 = h_3$ is already set to "0"), we expect many solutions to exist (about $2^{407.91}$).

```
Step              X_i               Π^l_i  P^l[i]               Y_i                Π^r_i  P^r[i]   P[i]

 -3: --------------------------------
 -2: --------------------------------
 -1: --------------------------------
 00: -----------------------0-------- |  0   0.00 | -----------------------0-------- |  5  -1.00 | -230.09
 01: -------------------------------- |  1   0.00 | -----------------------1-------- | 14  -1.00 | -229.09
 02: -------------------------------- |  2   0.00 | -----------------------n-------- |  7   0.00 | -228.09
 03: -------------------------------- |  3   0.00 | -------------------------------- |  0  -7.00 | -228.09
 04: -------------------------------- |  4   0.00 | --0000000----------------------- |  9  -8.00 | -221.09
 05: -------------------------------- |  5   0.00 | --1111111----------------------- |  2  -7.00 | -213.09
 06: -------------------------------- |  6   0.00 | --nuuuuuu----------------------- | 11  -6.00 | -206.09
 07: -------------------------------- |  7   0.00 | --01-----------------------0-000 |  4  -5.00 | -200.09
 08: -------------------------------- |  8   0.00 | -01------------------------0-011 | 13 -14.00 | -195.09
 09: -------------------------------- |  9   0.00 | -1----------------10-0-----n-nnn |  6 -11.00 | -181.09
 10: -------------------------------- | 10   0.00 | 1n010000----------11-1---------- | 15 -14.00 | -170.09
 11: -------------------------------- | 11   0.00 | 00111111-----00--0nu-n---------- |  8 -17.00 | -156.09
 12: -------------------------------- | 12   0.00 | nuuuuuuu-----11--11--0---------- |  1  -6.00 | -139.09
 13: -------------------------------- | 13   0.00 | -------1-----nn--un--u---------- | 10  -5.00 | -133.09
 14: -------------------------------- | 14  -1.00 | -------1----01----u------------- |  3 -11.00 | -128.09
 15: -----------------------n-------- | 15  -7.00 | -------u----10----0------------- | 12  -6.00 | -116.09
 16: -----------unnnn-------0-------- |  7 -12.09 | -----0-u----u------------------- |  6  -3.00 | -103.09
 17: -------n---00000-------1-------- |  4  -7.00 | -----u-0----u------------------- | 11  -2.00 | -88.00
 18: -------0---01111---------------- | 13  -4.00 | -----u------0------------------- |  3  -2.00 | -79.00
 19: ---u---1-------n------------1--- |  1  -4.00 | 0----0-------------------------- |  7  -1.00 | -73.00
 20: ---0-----------0------------0--- | 10  -3.00 | u------------------------------- |  0  -2.00 | -68.00
 21: ---1-----------1------------n--- |  6  -6.00 | u----------------------------0-- | 13  -2.00 | -63.00
 22: ---------------unnnn--------0--- | 15 -10.00 | 0----------------------------u-- |  5  -1.00 | -55.00
 23: ---------------00000--------u--- |  3  -7.00 | -----------------------------1-- | 10  -2.00 | -44.00
 24: -------------n-11101--------1--- | 12  -4.00 | ----------------------0------0-- | 14  -1.00 | -35.00
 25: -----------n-0--------------1--- |  0  -4.00 | ----------------------u--------- | 15  -1.00 | -30.00
 26: -------u---0-1------------------ |  9  -5.00 | ----------------------u--------- |  8  -1.00 | -25.00
 27: 1------0---1-u------------------ |  5  -3.00 | ----------------------0--------- | 12   0.00 | -19.00
 28: 0------1-----0------------------ |  2  -2.00 | -------------------------------- |  4  -1.00 | -16.00
 29: n------------1------------------ | 14  -1.00 | -------0------------------------ |  9  -1.00 | -13.00
 30: u------------------------------- | 11  -1.00 | -------u------------------------ |  1  -1.00 | -11.00
 31: u------------------------------- |  8  -1.00 | -------1------------------------ |  2  -1.00 |  -9.00
```

Fig. 3. The differential path for `RIPEMD-128`, after the non-linear parts search. The notations are the same as in [4] and $\mathrm{P}^l[i]$, $\mathrm{P}^r[i]$ and $P[i]$ are given in $\log_2()$.

In order for the path to provide a collision, the bit difference in X_{61} must erase the one in Y_{64} during the finalization phase of the compression function: $h'_2 = X_{61} \boxplus Y_{64} \boxplus h_3$. Since the signs of these two bit differences are not specified, this happens with probability 2^{-1} and the overall probability to follow our differential path and to obtain a collision for a randomly chosen input is $2^{-231.09}$.

4 Utilization of the Freedom Degrees

In the differential path from Figure 3, the difference mask is already entirely set, but almost all message bits and chaining variable bits have no constraint with regards to their value. All these freedom degrees can be used to reduce the complexity of the straightforward collision search (i.e. choosing random 512-bit message values) that requires about $2^{231.09}$ `RIPEMD-128` step computations. We will utilize these freedom degrees in three phases:

- **Phase 1**: we first fix some internal state and message bits in order to prepare the attack. This will allow us to handle in advance some conditions in the differential path as well as facilitating the merging phase. This preparation phase is done once for all.
- **Phase 2**: we will fix iteratively the internal state words X_{21}, X_{22}, X_{23}, X_{24} from left branch, and Y_{11}, Y_{12}, Y_{13},Y_{14} from right branch, as well as message words M_{12}, M_3, M_{10}, M_1, M_8, M_{15}, M_6, M_{13}, M_4, M_{11} and M_7 (the ordering

is important). This will provide us a starting point for the merging phase and due to a lack of freedom degrees, we will need to perform this phase several times in order to get enough starting points to eventually find a solution for the entire differential path.

- **Phase 3**: we use the remaining unrestricted message words M_0, M_2, M_5, M_9 and M_{14} to efficiently merge the internal states of the left and right branches.

4.1 Phase 1: Preparation

Before starting to fix a lot of message and internal state bit values, we need to prepare the differential path from Figure 3 so that the merge can later be done efficiently and so that the probabilistic part will not be too costly. Understanding these constraints requires a deep insight of the differences propagation and conditions fulfillment inside the `RIPEMD-128` step function. Therefore, the reader not interested in the details of the differential path construction is advised to skip this subsection.

The first constraint that we set is $Y_3 = Y_4$. The effect is that the `IF` function at step 4 of the right branch, $\mathtt{IF}(Y_2, Y_4, Y_3) = (Y_2 \wedge Y_3) \oplus (\overline{Y_2} \wedge Y_4) = Y_3 = Y_4$, will not depend on Y_2 anymore. We will see in Section 4.3 that this constraint is crucial in order for the merge to be performed efficiently.

The second constraint is $X_{24} = X_{25}$ (except the two bit positions of X_{24} and X_{25} that contain differences), and the effect is that the `IF` function at step 26 of the left branch (when computing X_{27}), $\mathtt{IF}(X_{26}, X_{25}, X_{24}) = (X_{26} \wedge X_{25}) \oplus (\overline{X_{26}} \wedge X_{24}) = X_{24} = X_{25}$, will not depend on X_{26} anymore. Before the final merging phase starts, we will not know M_0, and having this $X_{24} = X_{25}$ constraint will allow us to directly fix the conditions located on X_{27} without knowing M_0 (since X_{26} directly depends on M_0). Moreover, we fix the 12 first bits of X_{23} and X_{24} to "`01000100u001`" and "`01000011110`" respectively because this choice is among the few that minimizes the number of bits of M_9 that needs to be set in order to verify many of the conditions located on X_{27}.

The third constraint consists in setting the bits 18 to 30 of Y_{20} to zero. The effect is that for these 13 bit positions, the `ONX` function at step 21 of the right branch (when computing Y_{22}), $\mathtt{ONX}(Y_{21}, Y_{20}, Y_{19}) = (Y_{21} \vee \overline{Y_{20}}) \oplus Y_{19}$, will not depend on the 13 corresponding bits of Y_{21} anymore. Again, because we will not know M_0 before the merging phase starts, this constraint will allow us to directly fix the conditions on Y_{22} without knowing M_0 (since Y_{21} directly depends on M_0).

Finally, the last constraint that we enforce is that the first two bits of Y_{22} are set to "`10`" and the first three bits of M_{14} are set to "`011`". This particular choice of bit values is among the ones that reduces the most the spectrum of possible carries during the addition of step 24 (when computing Y_{25}) and we obtain a probability improvement to reach "`u`" in Y_{25} from 2^{-1} to $2^{-0.25}$.

We observe that all the constraints set in this subsection consume in total $32 + 51 + 13 + 5 = 101$ bits of freedom degrees, and a huge amount of solutions (about $2^{306.91}$) are still expected to exist.

4.2 Phase 2: Generating a Starting Point

Once the differential path properly prepared in phase 1, we would like to utilize the huge amount of freedom degrees available to fulfill directly as many conditions as possible. Our approach is to fix the value of the internal states in both the left and right branches (they can be handled independently), exactly in the middle of the non-linear parts where the number of conditions is important. Then, we will fix the message words one by one following a particular scheduling, and propagating the bit values forward and backward from the middle of the non-linear parts in both branches.

Fixing the Internal State. We chose to start by setting the values of X_{21}, X_{22}, X_{23}, X_{24} in the left branch, and Y_{11}, Y_{12}, Y_{13}, Y_{14} in the right branch, because they are located right in the middle of the non-linear parts. We take the first word X_{21} and randomly set all of its unrestricted "`-`" bits to "`0`" or "`1`" and check if any direct inconsistency is created with this choice. If that is the case, we simply pick another candidate until no direct inconsistency is deduced. Otherwise, we can go to the next word X_{22}, etc. If too many tries are failing for a particular internal state word, we can backtrack and pick another choice for the previous word. Finally, if no solution is found after a certain amount of time, we just restart the whole process, so as to avoid being blocked in a particularly bad subspace with no solution.

Fixing the Message Words. Similarly to the internal state words, we randomly fix the value of message words M_{12}, M_3, M_{10}, M_1, M_8, M_{15}, M_6, M_{13}, M_4, M_{11} and M_7 (following this particular ordering that facilitates the convergence towards a solution). The difference here is that the left and right branch computations are no more independent since the message words are used in both of them. However, this does not change anything to our algorithm and the very same process is applied: for each new message word randomly fixed, we compute forward and backward from the known internal state values and check for any inconsistency, using backtracking and reset if needed.

Overall, finding one new solution for this entire phase 2 takes about 5 minutes of computation on a recent PC with a naive implementation[2]. However, when one starting point is found, we can generate many for a very cheap cost by randomizing message words M_4, M_{11} and M_7 since the most difficult part is to fix the 8 first message words of the schedule. For example, once a solution is found, one can directly generate 2^{18} new starting points by randomizing a certain portion of M_7 (because M_7 has no impact on the validity of the non-linear part in the left branch, while in the right branch one has only to ensure that the last 14 bits of Y_{20} are set to "`u0000000000000`") and this was verified experimentally.

[2] Our message word fixing approach is certainly not optimal, but this phase is not the bottleneck of our attack and we preferred to aim for simplicity when possible. In case a very fast implementation is needed, a more efficient but more complex strategy would be to find a bit per bit scheduling instead of a word-wise one.

We give an example of such a starting point in Figure 4 and we emphasize that by "solution" or "starting point" we mean a differential path instance with **exactly** the same probability profile as this one. The 3 constrained bit values in M_{14} are coming from the preparation in phase 1, and the 3 constrained bit values in M_9 are necessary conditions in order to fulfill step 26 when computing X_{27}. It is also important to remark that whatever instance found during this second phase, the position of these 3 constrained bit values will always be the same thanks to our preparation in phase 1.

The probabilities displayed in Figure 4 for early steps (steps 0 to 14) are not meaningful here since they assume an attacker only computing forward, while in our case we will compute backwards from the non-linear parts to the early steps. However, we can see that the uncontrolled accumulated probability (i.e. step $\textcircled{3}$ in right side of Figure 1) is now improved to $2^{-29.32}$, or $2^{-30.32}$ if we add the extra condition for the collision to happen at the end of the RIPEMD-128 compression function.

```
Step              X_i                                W^l_i               Π^l_i             Y_i                                W^r_i              Π^r_i]  P[i]

 -3: --------------------------------
 -2: --------------------------------
 -1: --------------------------------
 00: -----------------------0-------- | --------------------------------  0 | -----------------------0-------- | --------------------------------  5 | -287.32
 01: -------------------------------- | 00000101111011100000110011000111  1 | ----------------------01-------- | x----------------------------011 14 | -285.32
 02: -------------------------------- | --------------------------------  2 | -----------------------n-------- | 01000010101100100011001110010110  7 | -284.32
 03: -------------------------------- | 00101100100000110100001001011110  3 | 00000000001100101010101011000000 | --------------------------------  0 | -252.32
 04: -------------------------------- | 11110000101100100000101111111100  4 | 00000000001100101010101011000000 | ----------------------0---1-1---  9 | -220.32
 05: -------------------------------- | --------------------------------  5 | 10111111101001001001010100111100 | --------------------------------  2 | -189.32
 06: -------------------------------- | 00100101011001000111000001010101  6 | 00nuuuuuu11000110111011001100100 | 10111001010001001100100111001100 11 | -157.32
 07: -------------------------------- | 01000010101100100011001110010110  7 | 00011011111110111011001001110000 | 11110000101100100000101111111100  4 | -157.32
 08: -------------------------------- | 00111100101111111010001110110000  8 | 10101101110101010010000001001011 | 01100011101010100010110001110011 13 | -157.32
 09: -------------------------------- | ----------------------0---1-1---  9 | 1110001100110111001010101l0n0nnn | 00100101011001000111000001010101  6 | -157.32
 10: -------------------------------- | 10001010101010011100001100111101 10 | 1n010000110010011011010100011110 | 00000110110000101001110101001010 15 | -157.32
 11: -------------------------------- | 10111001010001001100100111001100 11 | 0011111110111000l0nu1n1000110110 | 00111100101111111010001110110000  8 | -157.32
 12: 00111010101011111111101110101000 | 01101001001010010010111011101100 12 | nuuuuuuu011101111111110110111100l | 00000101111011100000110011000111  1 | -125.32
 13: 01110011001001011011001011011110 | 01100011101010100010110001110011 13 | 010111111010lnn10un11u1001001110 | 10001010101010011100001100111101 10 | -93.32
 14: 11110100011110100101101111011100 | x----------------------------011 14 | 01011111111001000 0u1001100000001 | 00101100100000110100001001011110  3 | -61.32
 15: 0110101010111100010111 1n00110110 | 00000110110000101001110101001010 15 | 1010100u111110000000100011100110 0 | 01101001001010010010111011101100 12 | -29.32
 16: 01010110010unnnn0010011000101111 | 01000010101100100011001110010110  7 | 1100101u1111u0011110011000010000 | 00100101011001000111000001010101  6 | -29.32
 17: 0100101n011000000000000111111001 | 11110000101100100000101111111100  4 | 11101u101111u0011111001011000010 | 10111001010001001100100111001100 11 | -29.32
 18: 10100010110011111110100000101000 | 01100011101010100010110001110011 13 | 11010u110000000101001100110001111 | 00101100100000110100001001011110  3 | -29.32
 19: 001u10010000101n0111000101111111 | 00000101111011100000110011000111  1 | 01010000011111101010011111100100 | 01000010101100100011001110010110  7 | -29.32
 20: 01100001101001101110001100100101 | 10001010101010011100001100111101 10 | u0000000000000000000000000000000 | --------------------------------  0 | -29.32
 21: 1011110100111111111001101001n110 | 00100101011001000111000001010101  6 | u----------------------------0-- | 01100011101010100010110001110011 13 | -27.32
 22: 1011101110001 01unnnn011010000111 | 00000110110000101001110101001010 15 | 0111101111101111010000010100 0u10 | --------------------------------  5 | -27.32
 23: 10111111101110000000001000100u001 | 00101100100000110100001001011110  3 | -----------------------------1-- | 10001010101010011100001100111101 10 | -26.32
 24: 010000101101 1n0111010010000111 10 | 01101001001010010010111011101100 12 | ---------------------10------0-- | x----------------------------011 14 | -24.32
 25: 01000010110n10011101001000011110 | --------------------------------  0 | ----------------------u--------- | 00000110110000101001110101001010 15 | -24.08
 26: -------u---0-1------------------ | ----------------------0---1-1---  9 | ----------------------u--------- | 00111100101111111010001110110000  8 | -21.08
 27: 1------0---1-u------------------ | --------------------------------  5 | ----------------------0--------- | 01101001001010010010111011101100 12 | -19.08
 28: 0------1-----0------------------ | --------------------------------  2 | -------------------------------- | 11110000101100100000101111111100  4 | -16.08
 29: n------------1------------------ | x----------------------------011 14 | -------0------------------------ | ----------------------0---1-1---  9 | -13.08
 30: u------------------------------- | 10111001010001001100100111001100 11 | -------u------------------------ | 00000101111011100000110011000111  1 | -11.00
 31: u------------------------------- | 00111100101111111010001110110000  8 | -------1------------------------ | --------------------------------  2 |  -9.00
 32: 1------------------------------- | 00101100100000110100001001011110  3 | -------1------------------------ | 00000110110000101001110101001010 15 |  -7.00
 33: -------------------------------- | 10001010101010011100001100111101 10 | -------------------------------- | --------------------------------  5 |  -7.00
 34: -------------------------------- | x----------------------------011 14 | u------------------------------- | 00000101111011100000110011000111  1 |  -6.00
 35: -------------------------------- | 11110000101100100000101111111100  4 | 0------------------------------- | 00101100100000110100001001011110  3 |  -4.00
 36: -------------------------------- | ----------------------0---1-1---  9 | 1------------------------------- | 01000010101100100011001110010110  7 |  -3.00
 37: -------------------------------- | 00000110110000101001110101001010 15 | -------------------------------- | x----------------------------011 14 |  -3.00
 38: -------------------------------- | 00111100101111111010001110110000  8 | -------------------------------- | 00100101011001000111000001010101  6 |  -3.00
 39: -------------------------------- | 00000101111011100000110011000111  1 | -------------------------------- | ----------------------0---1-1---  9 |  -3.00
 40: -------------------------------- | --------------------------------  2 | -------------------------------- | 10111001010001001100100111001100 11 |  -3.00
 41: -------------------------------- | 01000010101100100011001110010110  7 | -------------------------------- | 00111100101111111010001110110000  8 |  -3.00
 42: -------------------------------- | --------------------------------  0 | -------------------------------- | 01101001001010010010111011101100 12 |  -3.00
 43: -------------------------------- | 00100101011001000111000001010101  6 | -------------------------------- | --------------------------------  2 |  -3.00
 44: -------------------------------- | 01100011101010100010110001110011 13 | -------------------------------- | 10001010101010011100001100111101 10 |  -3.00
 45: -------------------------------- | 10111001010001001100100111001100 11 | -------------------------------- | --------------------------------  0 |  -3.00
 46: -------------------------------- | --------------------------------  5 | -------------------------------- | 11110000101100100000101111111100  4 |  -3.00
 47: -------------------------------- | 01101001001010010010111011101100 12 | -------------------------------- | 01100011101010100010110001110011 13 |  -3.00
 48: -------------------------------- | 00000101111011100000110011000111  1 | -------------------------------- | 00111100101111111010001110110000  8 |  -3.00
 49: -------------------------------- | ----------------------0---1-1---  9 | -------------------------------- | 00100101011001000111000001010101  6 |  -3.00
 50: -------------------------------- | 10111001010001001100100111001100 11 | -------------------------------- | 11110000101100100000101111111100  4 |  -3.00
 51: -------------------------------- | 10001010101010011100001100111101 10 | -------------------------------- | 00000101111011100000110011000111  1 |  -3.00
 52: -------------------------------- | --------------------------------  0 | -------------------------------- | 00101100100000110100001001011110  3 |  -3.00
 53: -------------------------------- | 00111100101111111010001110110000  8 | -------------------------------- | 10111001010001001100100111001100 11 |  -3.00
 54: -------------------------------- | 01101001001010010010111011101100 12 | -------------------------------- | 00000110110000101001110101001010 15 |  -3.00
 55: -------------------------------- | 11110000101100100000101111111100  4 | -------------------------------- | --------------------------------  0 |  -3.00
 56: -------------------------------- | 01100011101010100010110001110011 13 | -------------------------------- | --------------------------------  5 |  -3.00
 57: -------------------------------- | 00101100100000110100001001011110  3 | -------------------------------- | 01101001001010010010111011101100 12 |  -3.00
 58: -------------------------------- | 01000010101100100011001110010110  7 | -------------------------------- | --------------------------------  2 |  -3.00
 59: ------------------------0------- | 00000110110000101001110101001010 15 | -------------------------------- | 01100011101010100010110001110011 13 |  -2.00
 60: ------------------------1------- | x----------------------------011 14 | -------------------------------- | ----------------------0---1-1---  9 |  -1.00
 61: ------------------------x------- | --------------------------------  5 | -------------------------------- | 01000010101100100011001110010110  7 |  -1.00
 62: -------------------------------- | 00100101011001000111000001010101  6 | -------------------------------- | 10001010101010011100001100111101 10 |  -1.00
 63: -------------------------------- | --------------------------------  2 | -------------------------------- | x----------------------------011 14 |  -1.00
 64: -------------------------------- |                                       | ------------------------x-------
```

Fig. 4. The differential path for RIPEMD-128, after the second phase of the freedom degree utilization. The notations are the same as in [4] and $P[i]$ is given in $\log_2()$.

4.3 Phase 3: Merging Left and Right Branches

At the end of the second phase, we have several starting points equivalent to the one from Figure 4, with many conditions already verified and an uncontrolled accumulated probability of $2^{-30.32}$. Our goal for this third phase is now to use remaining free message words M_0, M_2, M_5, M_9, M_{14} and make sure that both left and right branches start with the same chaining variable.

We recall that during the first phase we enforced that $Y_3 = Y_4$, and for the merge we will require an extra constraint $X_5^{\ggg 5} \boxminus M_4 = \texttt{0xffffffff}$. The message words M_{14} and M_9 will be utilized to fulfill this constraint, and message words M_0, M_2 and M_5 will be used to perform the merge of the two branches only with a few operations, and with a success probability of 2^{-34}.

Handling the Extra Constraint with M_{14} and M_9. First, let us deal with the constraint $X_5^{\ggg 5} \boxminus M_4 = \texttt{0xffffffff}$, which can be rewritten as $X_5 = (\texttt{0xffffffff} \boxplus M_4)^{\lll 5}$ and then $(\texttt{0xffffffff} \boxplus M_4)^{\lll 5} = X_9^{\ggg 11} \boxminus (X_8 \oplus X_7 \oplus X_6) \boxminus M_8 \boxminus K_0^l$ by replacing M_5 using update formula of step 8 in left branch. Finally, isolating and replacing X_6 using update formula of step 9 in left branch:

$$M_9 = X_{10}^{\ggg 13} \boxminus ((X_9^{\ggg 11} \boxminus M_8 \boxminus K_0^l \boxminus (\texttt{0xffffffff} \boxplus M_4)^{\lll 5}) \oplus X_8 \oplus X_7) \boxminus K_0^l \boxminus (X_9 \oplus X_8 \oplus X_7). \quad (1)$$

All values on the right side of this equation are known if M_{14} is fixed. Therefore, so as to fulfill our extra constraint, what we could do is to simply pick a random value for M_{14}, and then directly deduce the value of M_9 thanks to equation (1). However, one can see in Figure 4 that 3 bits are already fixed in M_9 (the last one being the 10^{th} bit of M_9) and thus a valid solution would be found only with probability 2^{-3}. In order to avoid this extra complexity factor, we will first randomly fix the first 24 bits of M_{14} and this will allow us to directly deduce the first 10 bits of M_9 that fulfill our extra constraint up to the 10^{th} bit (because knowing the first 24 bits of M_{14} will lead to the first 24 bits of X_{11}, X_{10}, X_9, X_8 and the first 10 bits of X_7, which is exactly what we need according to equation (1)). Once a solution is found after 2^3 tries on average, we can randomize the remaining M_{14} unrestricted bits (the 8 most significant bits) and eventually deduce the 22 most significant bits of M_9 with equation (1). With this method, we completely remove the extra 2^3 factor, because the cost is amortized by the final randomization of the 8 most significant bits of M_{14}.

Merging the branches with M_0, M_2 and M_5. Once M_9 and M_{14} fixed, we still have message words M_0, M_2 and M_5 to determine for the merging. One can see that with only these three message words undetermined, all internal state values except X_2, X_1, X_0, X_{-1}, X_{-2}, X_{-3} and Y_2, Y_1, Y_0, Y_{-1}, Y_{-2}, Y_{-3} are fully known when computing backwards from the non-linear parts in each branch.

This is where our first constraint $Y_3 = Y_4$ comes into play. Indeed, when writing Y_1 from the equation from step 4 in right branch, we have:

$$Y_1 = Y_5^{\ggg 13} \boxminus (Y_4 \wedge Y_2 \oplus Y_3 \wedge \overline{Y_2}) \boxminus M_9 \boxminus K_0^r = Y_5^{\ggg 13} \boxminus Y_3 \boxminus M_9 \boxminus K_0^r$$

which means that Y_1 is already completely determined at this point (the bit condition present in Y_1 in Figure 4 is actually handled for free when fixing M_{14}

and M_9, since it requires to know the 9 first bits of M_9). In other words, the constraint $Y_3 = Y_4$ allowed Y_1 to not depend on Y_2 which is currently undetermined. Another effect of this constraint can be seen when writing Y_2 from the equation from step 5 in right branch:

$$Y_2 = Y_6^{\ggg 15} \boxminus (Y_5 \wedge Y_3 \oplus Y_4 \wedge \overline{Y_3}) \boxminus M_2 \boxminus K_0^r = Y_6^{\ggg 15} \boxminus (Y_5 \wedge Y_3) \boxminus M_2 \boxminus K_0^r = C_0 \boxminus M_2$$

where $C_0 = Y_6^{\ggg 15} \boxminus (Y_5 \wedge Y_3) \boxminus K_0^r$ is a constant.

Our second constraint $X_5^{\ggg 5} \boxminus M_4 = \texttt{0xffffffff}$ is useful when writing X_1 and X_2 from the equations from step 4 and 5 in left branch

$$\begin{aligned} X_2 &= X_6^{\ggg 8} \boxminus (X_5 \oplus X_4 \oplus X_3) \boxminus M_5 = C_1 \boxminus M_5 \\ X_1 &= X_5^{\ggg 5} \boxminus (X_4 \oplus X_3 \oplus X_2) \boxminus M_4 = \overline{X_4} \oplus X_3 \oplus X_2 = \overline{X_4} \oplus X_3 \oplus (C_1 \boxminus M_5) \end{aligned}$$

where $C_1 = X_6^{\ggg 8} \boxminus (X_5 \oplus X_4 \oplus X_3)$ is a constant.

Finally, our ultimate goal for the merge is to ensure that $X_{-3} = Y_{-3}$, $X_{-2} = Y_{-2}$, $X_{-1} = Y_{-1}$ and $X_0 = Y_0$, knowing that all other internal states are determined when computing backwards from the non-linear parts in each branch, except $Y_2 = C_0 \boxminus M_2$, $X_2 = C_1 \boxminus M_5$ and $X_1 = \overline{X_4} \oplus X_3 \oplus (C_1 \boxminus M_5)$. We therefore write the equations relating these eight internal state words:

$$\begin{aligned} X_0 &= X_4^{\ggg 12} \boxminus (X_3 \oplus X_2 \oplus X_1) \boxminus M_3 = X_4^{\ggg 12} \boxminus \overline{X_4} \boxminus M_3 \\ = Y_0 &= Y_4^{\ggg 11} \boxminus (Y_3 \wedge Y_1 \oplus Y_2 \wedge \overline{Y_1}) \boxminus M_0 \boxminus K_0^r = Y_4^{\ggg 11} \boxminus (Y_3 \wedge Y_1 \oplus (C_0 \boxminus M_2) \wedge \overline{Y_1}) \boxminus M_0 \boxminus K_0^r \\[1ex] X_{-1} &= X_3^{\ggg 15} \boxminus (X_2 \oplus X_1 \oplus X_0) \boxminus M_2 = X_3^{\ggg 15} \boxminus (\overline{X_4} \oplus X_3 \oplus X_0) \boxminus M_2 \\ = Y_{-1} &= Y_3^{\ggg 9} \boxminus (Y_2 \wedge Y_0 \oplus Y_1 \wedge \overline{Y_0}) \boxminus M_7 \boxminus K_0^r = Y_3^{\ggg 9} \boxminus ((C_0 \boxminus M_2) \wedge X_0 \oplus Y_1 \wedge \overline{X_0}) \boxminus M_7 \boxminus K_0^r \\[1ex] X_{-2} &= X_2^{\ggg 14} \boxminus (X_1 \oplus X_0 \oplus X_{-1}) \boxminus M_1 = (C_1 \boxminus M_5)^{\ggg 14} \boxminus (\overline{X_4} \oplus X_3 \oplus (C_1 \boxminus M_5) \oplus X_0 \oplus X_{-1}) \boxminus M_1 \\ = Y_{-2} &= Y_2^{\ggg 9} \boxminus (Y_1 \wedge Y_{-1} \oplus Y_0 \wedge \overline{Y_{-1}}) \boxminus M_{14} \boxminus K_0^r = (C_0 \boxminus M_2)^{\ggg 9} \boxminus (Y_1 \wedge X_{-1} \oplus X_0 \wedge \overline{X_{-1}}) \boxminus M_{14} \boxminus K_0^r \\[1ex] X_{-3} &= X_1^{\ggg 11} \boxminus (X_0 \oplus X_{-1} \oplus X_{-2}) \boxminus M_0 = (\overline{X_4} \oplus X_3 \oplus (C_1 \boxminus M_5))^{\ggg 11} \boxminus (X_0 \oplus X_{-1} \oplus X_{-2}) \boxminus M_0 \\ = Y_{-3} &= Y_1^{\ggg 8} \boxminus (Y_0 \wedge Y_{-2} \oplus Y_{-1} \wedge \overline{Y_{-2}}) \boxminus M_5 \boxminus K_0^r = Y_1^{\ggg 8} \boxminus (X_0 \wedge X_{-2} \oplus X_{-1} \wedge \overline{X_{-2}}) \boxminus M_5 \boxminus K_0^r \end{aligned}$$

If these four equations are verified, then we have merged left and right branch to the same input chaining variable. We first remark that X_0 is already fully determined and thus the second equation $X_{-1} = Y_{-1}$ only depends on M_2. Moreover, it is a T-function in M_2 (any bit i of the equation depends only on the i first bits of M_2) and can therefore be solved very efficiently bit per bit. We explain in the full version how to solve this T-function and our average cost in order to find one M_2 solution is one `RIPEMD-128` step computations.

Since X_0 is already fully determined, from the M_2 solution previously obtained we directly deduce the value of M_0 to satisfy the first equation $X_0 = Y_0$. From M_2 we can compute the value of Y_{-2} and we know that $X_{-2} = Y_{-2}$ and we calculate X_{-3} from M_0 and X_{-2}. At this point, the two first equations are fulfilled and we still have the value of M_5 to choose.

The third equation can be rewritten $V^{\ggg 14} = (V \oplus C_2) \boxplus C_3$, where $V = X[2] = (C_1 \boxminus M_5)$ and C_2, C_3 are two constants. Similarly, the fourth equation can be rewritten $V^{\ggg 11} = (V \boxplus C_4) \oplus C_5$, where C_4, C_5 are two constants. Solving either of these two equations with regards to V can be costly because of the rotations, so we combine them to create simpler one: $((V \oplus C_2) \boxplus C_3)^{\lll 3} = (V \boxplus C_4) \oplus C_5$. This equation is easier to handle because the rotation coefficient is small: we guess the 3 most significant bits of $((V \oplus C_2) \boxplus C_3)$ and we solve simply the equation 3-bit layer per 3-bit layer, starting from the least significant bit. From the value of V deduced, we straightforwardly obtain $M_5 = C_1 \boxminus V$ and the cost of recovering M_5 is equivalent to 8 `RIPEMD-128` step computations (the 3-bit guess implies a factor of 8, but the resolution can be implemented very efficiently with tables).

When all three message words M_0, M_2 and M_5 have been fixed, the first, second and a combination of the third and fourth equalities are necessarily verified. However, we have a probability 2^{-32} that both the third and fourth equations will be fulfilled. Moreover, one can check in Figure 4 that there is one bit condition on $X_0 = Y_0$ and one bit condition on Y_2 and this further adds up a factor 2^{-2}. We evaluate the whole process to cost about 19 `RIPEMD-128` step computations on average: there are 17 steps to compute backwards after having identified a proper couple M_{14}, M_9, and the 8 `RIPEMD-128` step computations to obtain M_5 are only done 1/4 of the time because the two bit conditions on Y_2 and $X_0 = Y_0$ are filtered before.

To summarize the merging: we first compute a couple M_{14}, M_9 that satisfies a special constraint, we find a value of M_2 that verifies $X_{-1} = Y_{-1}$, then we directly deduce M_0 to fulfill $X_0 = Y_0$, and we finally obtain M_5 to satisfy a combination of $X_{-2} = Y_{-2}$ and $X_{-3} = Y_{-3}$. Overall, with only 19 `RIPEMD-128` step computations on average, one can merge the branches with probability 2^{-34}.

5 Results and Implementation

5.1 Complexity Analysis and Implementation

After the quite technical description of the attack in previous section, we would like to rewrap everything to get a clearer view of the attack complexity, the amount of freedom degrees etc. Given a starting point from phase 2, the attacker can perform 2^{26} merge processes (because 3 bits are already fixed in both M_9 and M_{14}, and the extra constraint consumes 32 bits) and since one merge process succeeds only with probability of 2^{-34}, he obtains a solution with probability 2^{-8}. Since he needs $2^{30.32}$ solutions from the merge to have a good chance to verify the probabilistic part of the differential path, a total of $2^{38.32}$ starting points will have to be generated and handled.

From the end of phase 1, he generates $2^{38.32}$ starting points in phase 2, that is $2^{38.32}$ differential paths like the one from Figure 4 (with the same step probabilities). For each of them, in phase 3 he tries 2^{26} times to find a solution for the merge with an average complexity of 19 `RIPEMD-128` step computations for each try. The SFS collision final complexity is $19 \cdot 2^{26+38.32}$ `RIPEMD-128` step computations, which corresponds to $(19/128) \cdot 2^{64.32} = 2^{61.57}$ `RIPEMD-128` compression function computations (there are 64 steps in each branch).

The merge process has been implemented and we give in the full version of the article an example of a message and chaining variable couple that verifies the merge (i.e. they follow the differential path from Figure 3 until step 25 of the left branch and step 20 of the right branch). We measured the efficiency of our implementation in order to confront it to our theoretic complexity estimation. As point of reference, we observed that on the same computer, an optimized implementation of `RIPEMD-160` (OpenSSL v.1.0.1c) performs $2^{21.44}$ compression function computations per second. With 4 rounds instead of 5 and about 3/4 less operations per step, we extrapolated that `RIPEMD-128` would perform at $2^{22.17}$ compression function computations per second. Our implementation performs $2^{24.61}$ merge process (both phase 2 and phase 3) per second on average, which therefore corresponds to a SFS collision final complexity of $2^{61.88}$ `RIPEMD-128` compression computations. While our practical results confirm our theoretical estimations, we emphasize that the latter are a bit pessimistic, since our attack implementation is not optimized. As a side note, we also verified experimentally that the probabilistic part in both left and right branch can be fulfilled.

A last point needs to be checked: the complexity estimation for the generation of the starting points. Indeed, as much as $2^{38.32}$ starting points are required at the end of phase 2 and the algorithm being quite heuristic, it is hard to analyze precisely. The amount of freedom degrees is not an issue since we already saw in Section 4.1 that about $2^{306.91}$ solutions are expected to exist for the differential path at the end of phase 1. A completely new starting point takes about 5 minutes to be outputted on average with our implementation, but from one such path we can directly generate 2^{18} equivalent ones by randomizing M_7. Using the OpenSSL implementation as reference, this amounts to $2^{50.72}$ `RIPEMD-128` computations to generate all the starting points that we need in order to find a SFS collision. This gross estimation is extremely pessimistic since its doesn't even take in account the fact that once a starting point is found, one can also randomize M_4 and M_{11} to find many other valid candidates with a few operations. Finally, one may argue that with this method the starting points generated are not independent enough (in backward direction when merging and/or in forward direction for verifying probabilistically the linear part of the differential path). However, no such correlation was detected during our experiments and previous attacks on similar hash functions [13,14] showed that only a few rounds were enough to observe independence between bit conditions. In addition, even if some correlations existed, since we are looking for many solutions, the effect would be averaged among good and bad candidates.

Collision for the `RIPEMD-128` Compression Function. We described in previous sections a SFS collision attack for the full `RIPEMD-128` compression function with $2^{61.57}$ computations. It is clear from Figure 4 that we can remove the 4 last steps of our differential path in order to attack a 60-step reduced variant of the `RIPEMD-128` compression function. No difference will be present in the internal state at the end of the computation and we directly get a collision, saving a factor 2^4 over the full `RIPEMD-128` attack complexity.

We also give in the full version of the article a slightly different freedom degrees utilization when attacking 63 steps of the `RIPEMD-128` compression function (the first step being taken out), that saves a factor $2^{1.66}$ over the collision attack.

Distinguishers. We provide in the full version of this article a limited-birthday distinguisher [11] on the full `RIPEMD-128` compression but also hash function.

Conclusion

In this article, we proposed a new cryptanalysis technique for `RIPEMD-128`, that led to a collision attack on the full compression function as well as a distinguisher for the full hash function. We believe that our method still presents room for improvements, and we expect a practical collision attack for the full `RIPEMD-128` compression function to be found during the incoming years. While our results don't endanger the collision resistance of the `RIPEMD-128` hash function as a whole, we emphasize that SFS collision attacks are a strong warning sign which indicates that `RIPEMD-128` might not be as secure as the community expected. Considering the history of the attacks on the `MD5` compression function [7,8], `MD5` hash function [29], and then `MD5`-protected certificates [24], we believe that another function than `RIPEMD-128` should be used for new security applications. Future works include reducing the attack complexity and applying our methods to `RIPEMD-160` and other parallel branches-based functions.

References

1. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Keccak specifications. Submission to NIST (2008)
2. Bosselaers, A., Preneel, B. (eds.): RIPE 1992. LNCS, vol. 1007. Springer, Heidelberg (1995)
3. Brassard, G. (ed.): CRYPTO 1989. LNCS, vol. 435. Springer, Heidelberg (1990)
4. De Cannière, C., Rechberger, C.: Finding SHA-1 Characteristics: General Results and Applications. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 1–20. Springer, Heidelberg (2006)
5. Cramer, R. (ed.): EUROCRYPT 2005. LNCS, vol. 3494. Springer, Heidelberg (2005)
6. Damgård, I.: A Design Principle for Hash Functions. In: Brassard, pp. 416–427
7. den Boer, B., Bosselaers, A.: Collisions for the Compression Function of MD5. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 293–304. Springer, Heidelberg (1994)
8. Dobbertin, H.: Cryptanalysis of MD5 compress. In: Rump Session of Advances in Cryptology EUROCRYPT 1996 (1996)
9. Dobbertin, H.: RIPEMD with Two-Round Compress Function is Not Collision-Free. J. Cryptology 10(1), 51–70 (1997)
10. Dobbertin, H., Bosselaers, A., Preneel, B.: RIPEMD-160: A Strengthened Version of RIPEMD. In: Gollmann, D. (ed.) FSE 1996. LNCS, vol. 1039, pp. 71–82. Springer, Heidelberg (1996)

11. Gilbert, H., Peyrin, T.: Super-Sbox Cryptanalysis: Improved Attacks for AES-Like Permutations. In: Hong, S., Iwata, T. (eds.) FSE 2010. LNCS, vol. 6147, pp. 365–383. Springer, Heidelberg (2010)
12. ISO. ISO/IEC 10118-3:2004: Information technology — Security techniques — Hash-functions — Part 3: Dedicated hash-functions. pub-ISO (February 2004)
13. Joux, A., Peyrin, T.: Hash Functions and the (Amplified) Boomerang Attack. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 244–263. Springer, Heidelberg (2007)
14. Manuel, S., Peyrin, T.: Collisions on SHA-0 in One Hour. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 16–35. Springer, Heidelberg (2008)
15. Mendel, F., Nad, T., Scherz, S., Schläffer, M.: Differential Attacks on Reduced RIPEMD-160. In: Gollmann, D., Freiling, F.C. (eds.) ISC 2012. LNCS, vol. 7483, pp. 23–38. Springer, Heidelberg (2012)
16. Mendel, F., Nad, T., Schläffer, M.: Collision Attacks on the Reduced Dual-Stream Hash Function RIPEMD-128. In: Canteaut, A. (ed.) FSE 2012. LNCS, vol. 7549, pp. 226–243. Springer, Heidelberg (2012)
17. Mendel, F., Pramstaller, N., Rechberger, C., Rijmen, V.: On the Collision Resistance of RIPEMD-160. In: Katsikas, S.K., López, J., Backes, M., Gritzalis, S., Preneel, B. (eds.) ISC 2006. LNCS, vol. 4176, pp. 101–116. Springer, Heidelberg (2006)
18. Merkle, R.C.: One Way Hash Functions and DES. In: Brassard [3], pp. 428–446
19. Ohtahara, C., Sasaki, Y., Shimoyama, T.: Preimage Attacks on Step-Reduced RIPEMD-128 and RIPEMD-160. In: Lai, X., Yung, M., Lin, D. (eds.) Inscrypt 2010. LNCS, vol. 6584, pp. 169–186. Springer, Heidelberg (2011)
20. Rivest, R.L.: The MD4 message-digest algorithm. Request for Comments (RFC) 1320, Internet Activities Board, Internet Privacy Task Force (April 1992)
21. Sasaki, Y., Aoki, K.: Meet-in-the-Middle Preimage Attacks on Double-Branch Hash Functions: Application to RIPEMD and Others. In: Boyd, C., González Nieto, J. (eds.) ACISP 2009. LNCS, vol. 5594, pp. 214–231. Springer, Heidelberg (2009)
22. Sasaki, Y., Wang, L.: Distinguishers beyond Three Rounds of the RIPEMD-128/-160 Compression Functions. In: Bao, F., Samarati, P., Zhou, J. (eds.) ACNS 2012. LNCS, vol. 7341, pp. 275–292. Springer, Heidelberg (2012)
23. Shoup, V. (ed.): CRYPTO 2005. LNCS, vol. 3621. Springer, Heidelberg (2005)
24. Stevens, M., Sotirov, A., Appelbaum, J., Lenstra, A., Molnar, D., Osvik, D.A., de Weger, B.: Short Chosen-Prefix Collisions for MD5 and the Creation of a Rogue CA Certificate. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 55–69. Springer, Heidelberg (2009)
25. Wang, X., Yu, H., Lisa Yin, Y.: Efficient Collision Search Attacks on SHA-0. In: Shoup [23], pp. 1–16
26. Wang, L., Sasaki, Y., Komatsubara, W., Ohta, K., Sakiyama, K. (Second) Preimage Attacks on Step-Reduced RIPEMD/RIPEMD-128 with a New Local-Collision Approach. In: Kiayias, A. (ed.) CT-RSA 2011. LNCS, vol. 6558, pp. 197–212. Springer, Heidelberg (2011)
27. Wang, X., Lai, X., Feng, D., Chen, H., Yu, X.: Cryptanalysis of the Hash Functions MD4 and RIPEMD. In: Cramer [5], pp. 1–18
28. Wang, X., Lisa Yin, Y., Yu, H.: Finding Collisions in the Full SHA-1. In: Shoup [23], pp. 17–36
29. Wang, X., Yu, H.: How to Break MD5 and Other Hash Functions. In: Cramer [5], pp. 19–35

New Collision Attacks on SHA-1 Based on Optimal Joint Local-Collision Analysis

Marc Stevens

CWI, Amsterdam, The Netherlands
marc@marc-stevens.nl

Abstract. The main contributions of this paper are two-fold.

Firstly, we present a novel direction in the cryptanalysis of the cryptographic hash function SHA-1. Our work builds on previous cryptanalytic efforts on SHA-1 based on combinations of local collisions. Due to dependencies, previous approaches used heuristic corrections when combining the success probabilities and message conditions of the individual local collisions. Although this leads to success probabilities that are seemingly sufficient for feasible collision attacks, this approach most often does not lead to the maximum success probability possible as desired. We introduce novel techniques that enable us to determine the theoretical maximum success probability for a given set of (dependent) local collisions, as well as the smallest set of message conditions that attains this probability. We apply our new techniques and present an implemented open-source near-collision attack on SHA-1 with a complexity equivalent to $2^{57.5}$ SHA-1 compressions.

Secondly, we present an identical-prefix collision attack and a chosen-prefix collision attack on SHA-1 with complexities equivalent to approximately 2^{61} and $2^{77.1}$ SHA-1 compressions, respectively.

1 Introduction

A series of breakthrough attacks on hash functions started in 2004 when the first collisions for MD4, MD5, HAVAL-128 and RIPEMD were presented by Wang et al.[WFLY04, WY05]. This was soon followed by the first SHA-0 collision by Biham et al. [BCJ$^+$05]. Soon thereafter, Wang et al. published a more efficient collision attack on SHA-0 [WYY05c]. In the same year, the first collision attack on full SHA-1 [WYY05b] was presented by Wang et al. with an estimated complexity of 2^{69} compressions. A later unpublished result by Wang et al. claimed an attack with a complexity of 2^{63} compressions [WYY05a] which was later partly verified by Cochran [Coc07]. This was further improved by Mendel et al. with an unpublished attack with a complexity of $2^{60.x}$ compressions [MRR07]. Although later withdrawn, McDonald et al. published an attack with claimed complexity of 2^{52} compressions [MHP09]. Rafi Chen claims to be able to find collisions in

2^{58} [Che11] [1]. For reduced step variants of SHA-1 more progress has been made [CR06, CMR07, Gre10] and collisions have been found for up to 75 steps [GA11].

So far, it seems some kind of barrier has been reached at around 2^{61} SHA-1 compressions. Unfortunately, as Polk et al. [PCTH11] point out, these cryptanalytic advancements are not fully reflected in the literature so far.

2 Our Contributions

This paper aims to renew the cryptanalytic efforts to construct a feasible collision attack on SHA-1 and find an actual collision pair. The main contributions of this paper are two-fold.

Firstly, we present a novel direction in the cryptanalysis of SHA-1 that we believe will allow collision attacks with complexity well below the 2^{61} barrier. Collision attacks on SHA-1 are constructed in roughly two parts: a non-linear part (over approximately the first 20 steps) and a linear part (over approximately the last 60 steps). The linear part is constructed using a linear combination of local collisions as described by a disturbance vector [CJ98]. So far, to obtain the success probability of these combinations, the local collisions are first studied independently (e.g., see [MPRR06]) and then combined. As the success probabilities of local collisions can be dependent (e.g., see [Man11]), current approaches make some heuristic corrections when joining probabilities and message conditions. Although this is seemly sufficient to construct feasible collision attack on SHA-1, it may not lead to the desired maximum success probability possible and thereby leads to sub-optimal collision attacks. We introduce novel techniques that enable the computation of the maximum success probability for a given set of (dependent) local collisions, as well as the smallest set of message conditions that attains this probability. That our new approach provides a distinct advantage over the previous approach is showcased in our second contribution.

Our second contribution is an implemented near-collision attack for SHA-1 with a complexity equivalent to $2^{57.5}$ compressions[2]. We show how this near-collision attack can be used to construct a SHA-1 identical-prefix collision attack with a complexity of 2^{61} compressions. Furthermore, we present the first SHA-1 chosen-prefix collision attack with a complexity of $2^{77.1}$ compressions.

Our attack distinguishes itself from previous claims on several aspects. Firstly, we aimed to optimize the complexity over the linear part and (so far) not over the non-linear part. Secondly, our novel direction has resulted in a competitive attack complexity without exploiting nearly all degrees of freedoms. In fact there are well over 50 from the 512 message bits left as degrees of freedom that can be further exploited in future work. Lastly, it is the first public implementation

[1] We like to note that using our methods we have proven that the highest probability attainable over the last 8 steps is $2^{-8.356}$. But Chen (see Ch. 9.5) actually uses a factor $\frac{100}{3000} \approx 2^{-4.9}$, suggesting his claim may be a factor $2^{3.5}$ too optimistic.

[2] This complexity is not based on a purely theoretical cost analysis, but directly determined from the measured performance over the non-linear part and the (implementation verified) theoretical success probabilities over the linear part, see Sect. 5.1.

of a SHA-1 collision attack: the source code is available online [Ste12b]. This allows the public verification of the correctness and the complexity of our implementation and we also hope it leads to better understanding and improvements by the scientific community. Due to space considerations, many technical details have been omitted here, but these can be found in [Ste12a]. Despite this, we briefly discuss how the correctness of our implementation as well as our claimed complexity can be verified using our publicly available source code.

3 Preliminaries

Notation. SHA-1 is defined using 32-bit words $X = (x_i)_{i=0}^{31} \in \{0,1\}^{32}$ that are identified with elements $X = \sum_{i=0}^{31} x_i 2^i$ of $\mathbb{Z}/2^{32}\mathbb{Z}$ (for addition and subtraction). A *binary signed digit representation* (BSDR) for $X \in \mathbb{Z}/2^{32}\mathbb{Z}$ is a sequence $Z = (z_i)_{i=0}^{31} \in \{-1,0,1\}^{32}$ for which $X = \sum_{i=0}^{31} z_i 2^i$. We use the following notation: $Z[i] = z_i$, $RL(Z,n)$ and $RR(Z,n)$ (cyclic left and right rotation), $w(Z)$ (Hamming weight), $\sigma(Z) = X = \sum_{i=0}^{31} k_i 2^i \in \mathbb{Z}/2^{32}\mathbb{Z}$.

In collision attacks we consider two related messages M and M'. For any variable X related to the SHA-1 calculation of M, we use X' to denote the corresponding variable for M'. Furthermore, for such a 'matched' variable $X \in \mathbb{Z}/2^{32}\mathbb{Z}$ we define $\delta X = X' - X$ and $\Delta X = (X'[i] - X[i])_{i=0}^{31}$.

SHA-1's Compression Function. The input for SHA-1's Compress consists of an intermediate hash value $IHV_{\text{in}} = (a,b,c,d,e)$ of five 32-bit words and a 512-bit message block B. The 512-bit message block B is partitioned into 16 consecutive 32-bit strings which are interpreted as 32-bit words $W_0, W_1, \ldots, W_{15}$ (using big-endian), and expanded to $W_0, \ldots, W_{79}$ as follows:

$$W_t = RL(W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}, 1), \quad \text{for } 16 \le t < 80. \tag{1}$$

We describe SHA-1's compression function Compress in an 'unrolled' version. For each step $t = 0, \ldots, 79$ it uses a working state consisting of five 32-bit words Q_t, Q_{t-1}, Q_{t-2}, Q_{t-3} and Q_{t-4} and calculates a new state word Q_{t+1}. The working state is initialized before the first step as

$$(Q_0, Q_{-1}, Q_{-2}, Q_{-3}, Q_{-4}) = (a, b, RR(c,30), RR(d,30), RR(e,30)).$$

For $t = 0, 1, \ldots, 79$ in succession, Q_{t+1} is calculated as follows:

$$\begin{aligned} F_t &= f_t(Q_{t-1}, RL(Q_{t-2},30), RL(Q_{t-3},30)), \\ Q_{t+1} &= F_t + AC_t + W_t + RL(Q_t,5) + RL(Q_{t-4},30). \end{aligned} \tag{2}$$

These 80 steps are grouped in 4 rounds of 20 steps each. Here, AC_t is the constant $\texttt{5a827999}_{16}$, $\texttt{6ed9eba1}_{16}$, $\texttt{8f1bbcdc}_{16}$ or $\texttt{ca62c1d6}_{16}$ for the 1st, 2nd, 3rd and 4th round, respectively. The non-linear function $f_t(X,Y,Z)$ is defined as $(X \wedge Y) \oplus (\overline{X} \wedge Z)$, $X \oplus Y \oplus Z$, $(X \wedge Y) \vee (Z \wedge (X \vee Y))$ or $X \oplus Y \oplus Z$ for the 1st, 2nd, 3rd and 4th round, respectively. Finally, the output intermediate hash value δIHV_{out} is determined as:

$$\delta IHV_{\text{out}} = (a+Q_{80},\ b+Q_{79},\ c+RL(Q_{78},30),\ d+RL(Q_{77},30),\ e+RL(Q_{76},30)).$$

4 Joint Local-Collision Analysis

4.1 Local Collisions and the Disturbance Vector

In 1998, Chabaud and Joux [CJ98] constructed a collision attack on SHA-0 based on local collisions. A local collision over 6 steps for SHA-0 and SHA-1 consists of a disturbance $\delta Q_{t+1} = 2^b$ created in some step t by a message word bit difference $\delta W_t = 2^b$. This disturbance is corrected over the next five steps, so that after those five steps no differences occur in the five working state words. They were able to interleave many of these local collisions such that the message word differences $(\Delta W_t)_{t=0}^{79}$ conform to the message expansion (cf. Eq. 1). For more convenient analysis, they consider the *disturbance vector* which is a non-zero vector $(DV_t)_{t=0}^{79}$ conform to the message expansion where every '1'-bit $DV_t[b]$ marks the start of a local collision based on the disturbance $\delta W_t[b] = \pm 1$. We denote by $(DW_t)_{t=0}^{79}$ the message word bit differences without sign (i.e., $DW_t = W_t' \oplus W_t$) for a disturbance vector $(DV_t)_{t=0}^{79}$:

$$DW_t := \bigoplus_{(i,r)\in\mathcal{R}} RL(DV_{t-i}, r), \quad \mathcal{R} = \{(0,0),(1,5),(2,0),(3,30),(4,30),(5,30)\}$$

Note that in differential paths we work with differences δW_t instead of DW_t. We say that a message word difference δW_t is *compatible* with DW_t if there are coefficients $c_0, \ldots, c_{31} \in \{-1, 1\}$ such that $\delta W_t = \sum_{j=0}^{31} c_j \cdot DW_t[j]$. The set $\mathcal{W}_t$ of all compatible message word differences given DW_t is defined as:

$$\mathcal{W}_t := \{\sigma(X) \mid \text{BSDR } X, \ \ X[i] \in \{-DW_t[i], +DW_t[i]\}, \ \ i \in \{0,\ldots,31\}\}$$

4.2 Disturbance Vector Classes

Manuel [Man11] has classified previously found interesting disturbance vectors into two classes. A disturbance vector from the first class denoted by $\mathrm{I}(K, b)$ is defined by $DV_K = \ldots = DV_{K+14} = 0$ and $DV_{K+15} = 2^b$. Similarly, a disturbance vector from the second class denoted by $\mathrm{II}(K, b)$ is defined by $DV_{K+1} = DV_{K+3} = RL(2^{31}, b)$ and $DV_{K+15} = 2^b$ and $DV_{K+i} = 0$ for $i \in \{0, 2, 4, 5, \ldots, 14\}$. For both classes, the remaining $DV_0, \ldots, DV_{K-1}$ and $DV_{K+16}, \ldots, DV_{79}$ are determined through the (reverse) message expansion relation (Eq. 1).

4.3 Dependencies of Local Collisions

Local collisions can interact in the following three ways.

- *Message differences.* Firstly, they may use message word differences in the same bit position of the same message word. E.g., consider the disturbance vector for which $DV_{50}[0]$ and $DV_{55}[30]$ are the only '1'-bits. Then as $DW_{55} = DV_{55} \oplus RL(DV_{50}, 30) = 0$, this means the message word differences in step 55 of the two local collisions must be chosen to cancel each other.

- *Working state differences.* Secondly, local collisions starting in the same step directly interact with each other due to carries. E.g., Wang et al. [WYY05b] introduced a bit compression technique. They use opposite signs for two local collisions that start in the same step at two subsequent bit positions (say $DV_{25}[0] = DV_{25}[1] = 1$) to turn it into a single local collision.
- *Boolean function differences.* Thirdly, two 'close' disturbances can interact in the boolean function. E.g., consider the disturbance vector for which $DV_{25}[31]$ and $DV_{26}[31]$ are the only '1'-bits. Then these local collisions interact as in the first case as the message word differences in steps 29 and 30 cancel each other out. Moreover, in step 29 it is also guaranteed that $\delta F_{29} = 0$ as the two disturbances input to the XOR boolean function cancel each other. In contrast, when analyzing these two local collisions independently, each has a probability of 0.5 that the difference δF_{29} has the opposite sign from δW_{29}. The product of the independent success probabilities is thereby *lower* than the maximum joint probability of these two local collisions by a factor $0.5 \cdot 0.5 = 0.25$ (see also [Man11, Table 9]). This particular example does not involve any carries, which in other cases may have a further impact on the maximum success probability.

Although these examples are quite easy to analyze, disturbance vectors have a higher density of disturbances at the beginning and the end. For these higher density areas, it is significantly more difficult to analyze the exact impact of these interactions on the maximum success probability. In this paper we take a new direction in the cryptanalysis of SHA-1 in which we do not analyze these interactions directly, but use a rather general approach to determine the maximum success probability that incorporates these interactions.

4.4 Optimal Joint Local-Collision Analysis

We start at the relatively easy and well understood analysis of a single local collision. Given the single bit disturbance $\Delta Q_{t+1}[b] = \pm 1$ created in the first step t, one analyzes the necessary message conditions to cancel this disturbance in the subsequent steps. Most importantly, one determines what the probability is of a successful cancellation under these message conditions. Higher success probabilities are obtained by also considering carries in ΔQ_{t+1} from bit position b to higher positions.

One approach that obtains exact success probabilities is to sum the exact success probabilities of *all* possible differential paths over these 6 steps $t, \ldots, t+5$ with $\delta Q_{t-4} = \ldots = \delta Q_t = 0$, $\delta Q_{t+1} \neq 0$ and $\delta Q_{t+2} = \ldots = \delta Q_{t+6} = 0$ using a given message difference vector $(\delta W_i)_{i=t}^{t+5}$. Although there are quite a few of such differential paths for a single local collision, these can easily be enumerated.

We propose to study combinations of local collisions in a very similar way. That is, we propose to analyze the set of *all* possible differential paths over a given range of steps $t_b, \ldots, t_e$ that contain disturbances as prescribed by the disturbance vector using message word differences δW_t compatible with DW_t. Next, this set is partitioned based on the values for the starting and ending working state differences and the message word differences. We distinguish thus only on the pre-conditions (the differences in the starting working state and the message words) and the post-condition (the differences in the ending working state) of differential paths that matches how they are used in an actual collision attack. For each partition, we compute the sum of the probabilities of its differential paths. One can thus interpret this total partition probability as the total probability that the ending working state differences are obtained after step t_e given that both the differences in the starting working state at step t_b and the differences in the message words for steps $t_b, \ldots, t_e$ hold. Hence, the desired maximum success probability is the maximum over all total partition probabilities.

4.5 Definitions

More formally, we define a differential path $\mathcal{P}$ over steps $t = t_b, \ldots, t_e$ to be given as $\mathcal{P} = ((\Delta Q_t)_{t=t_b-4}^{t_e+1}, (\Delta F_t)_{t=t_b}^{t_e}, (\delta W_t)_{t=t_b}^{t_e})$[3], under the following restrictions:

– correct differential steps for $t = t_b, \ldots, t_e$:

$$\sigma(\Delta Q_{t+1}) = \sigma(RL(\Delta Q_t, 5)) + \sigma(RL(\Delta Q_{t-4}, 30)) + \sigma(\Delta F_t) + \delta W_t. \quad (3)$$

– $\Delta F_t[31] \in \{0, 1\}$ and a non-zero value represents $\Delta F_t[31] \in \{-1, +1\}$. [4]

The success probability $\Pr[\mathcal{P}]$ of a differential path $\mathcal{P}$ over steps $t_b, \ldots, t_e$ is informally defined as the probability that the given path $\mathcal{P}$ holds exactly for $(\widehat{Q}_{t_b-4}, \widehat{Q}'_{t_b-4}), \ldots, (\widehat{Q}_{t_e+1}, \widehat{Q}'_{t_e+1})$ for uniformly-randomly chosen $\widehat{Q}_{t_b-4}, \ldots, \widehat{Q}_{t_b}$ and $\widehat{W}_{t_b}, \ldots, \widehat{W}_{t_e}$. The $\widehat{Q}'_{t_b-4}, \ldots, \widehat{Q}'_{t_b}$ and $\widehat{W}'_{t_b}, \ldots, \widehat{W}'_{t_e}$ are determined through the first five working state differences $\delta Q_{t_b}, \ldots, \delta Q_{t_e}$ and the message differences δW_i (for $i = t_b, \ldots, t_e$). The remaining $(\widehat{Q}_{t_b+1}, \widehat{Q}'_{t_b+1}), \ldots, (\widehat{Q}_{t_e+1}, \widehat{Q}'_{t_e+1})$ are computed using the step function (Eq. 2). We refer to [Ste12a, Ch. 7.5] for an equivalent definition and how to efficiently determine the probability $\Pr[\mathcal{P}]$.

As we are interested in differential paths with prescribed disturbances, we define the set $\mathcal{Q}_t$ as the set of all allowed differences ΔQ_t given $(DV_i)_{i=0}^{79}$:

$$\mathcal{Q}_t := \left\{ \text{BSDR } Y \;\middle|\; {\scriptstyle \sigma(Y)=\sigma(Z), \atop Z[i] \in \{-DV_{t-1}[i], DV_{t-1}[i]\},\ i=0,\ldots,31} \right\}.$$

We are now ready to define the set of *all* possible differential paths over steps $t_b, \ldots, t_e$ that we will base our analysis on:

$$\mathcal{D}_{[t_b,t_e]} := \left\{ \widehat{\mathcal{P}} \;\middle|\; \Delta\widehat{Q_i} \in \mathcal{Q}_i,\ \delta\widehat{W_j} \in \mathcal{W}_j,\ \Pr[\widehat{\mathcal{P}}] > 0 \right\}$$

We define three functions ψ, ϕ and ω that return beginning working state differences, ending working state differences and message word differences:

[3] In practice, we use a strictly smaller representation wherein ΔQ_{t_b-4} and δQ_{t_e+1} are replaced by $\delta(RL(Q_{t-4}, 30))$ and δQ_{t_e+1}, respectively. We use a simplification here to ease presentation.

[4] Here both -1 and $+1$ result in the same contribution in $\sigma(\Delta F_t)$.

$$\psi(\mathcal{P}) = (\Delta Q_i)_{i=t_b-4}^{t_b}, \quad \omega(\mathcal{P}) = (\delta W_i)_{i=t_b}^{t_e},$$

$$\phi(\mathcal{P}) = (d_i)_{i=t_e-3}^{t_e+1}, \text{ where } d_i = \begin{cases} \sigma(RL(\Delta Q_i, 30)), & i = t_e-3, t_e-2, t_e-1; \\ \delta Q_i, & i = t_e, t_e+1. \end{cases}$$

We have chosen this particular definition for the ending working state differences $\phi(\mathcal{P})$ as this matches δIHV_{out} exactly. We denote by $\psi(\mathcal{D})$, $\phi(\mathcal{D})$ and $\omega(\mathcal{D})$ the sets found by applying ψ, ϕ or ω to all differential paths in the set $\mathcal{D}$.

For a given disturbance vector $(DV_t)_{t=0}^{79}$, the desired maximum success probability over steps $t_b, \ldots, t_e$ denoted by $\text{FDC}_{[t_b,t_e]}\left((DV_t)_{t=0}^{79}\right)$ is:

$$\text{FDC}_{[t_b,t_e]}\left((DV_t)_{t=0}^{79}\right) = \max_{b,e,w} \sum_{\substack{\widehat{\mathcal{P}} \in \mathcal{D}_{[t_b,t_e]} \\ \psi(\widehat{\mathcal{P}})=b,\ \phi(\widehat{\mathcal{P}})=e,\ \omega(\widehat{\mathcal{P}})=w}} \Pr[\widehat{\mathcal{P}}] \cdot c(b),$$

where $c(b) = c((\Delta Q_i)_{i=t_b-4}^{t_e})$ is the correction factor $c(b) = \prod_{i=t_b-4}^{t_b-2} 2^{w(\Delta \widehat{Q}_i)}$. This correction factor $c(b)$ ensures that FDC is the maximum success probability assuming all working state bit conditions are fulfilled for Q_{t_b-4}, Q_{t_b-3} and Q_{t_b-2}.[5] This is due to the fact that a collision attack fulfills working state bit conditions step by step, using message freedoms to speed up the attack, until these freedoms cannot be exploited anymore. At that point, it is more beneficial to compute all remaining steps and verify whether the desired δIHV_{out} is obtained. FDC returns the maximum success probability obtainable for these remaining steps.

4.6 Differential Path Reduction

Unfortunately, analyzing a single local collision in the above manner is very feasible, whereas analyzing multiple local collisions quickly results in a prohibitively large set of possible differential paths. We exploit the large amount of redundancy among the possible differential paths to be able to efficiently compute the desired maximum success probability even when there are many local collisions.

Note that we are only interested in the total success probability for given pre- and post-conditions and not in the differential paths themselves per se. We therefore propose to break up a differential path $\mathcal{P}$ into two valid differential paths $\widehat{\mathcal{P}}$ and $\widetilde{\mathcal{P}}$ with the following properties:

- $\widehat{\mathcal{P}}$ and $\widetilde{\mathcal{P}}$ are 'disjoint' and 'add' to $\mathcal{P}$. More specifically, we want that either $\Delta\widehat{Q}_i[b]$ or $\Delta\widetilde{Q}_i[b]$ to be equal to $\Delta Q_i[b]$ and the other to be zero (or all three to be zero). The same holds for $\Delta F_i[b]$. Furthermore, $\delta W_i = \delta\widehat{W}_i + \delta\widetilde{W}_i$;
- the success probabilities of $\widehat{\mathcal{P}}$ and $\widetilde{\mathcal{P}}$ are independent: $\Pr[\mathcal{P}] = \Pr[\widehat{\mathcal{P}}] \cdot \Pr[\widetilde{\mathcal{P}}]$;
- $\psi(\mathcal{P}) = \psi(\widehat{\mathcal{P}})$ and $\phi(\mathcal{P}) = \phi(\widehat{\mathcal{P}})$;
- the success probability $\Pr[\widehat{\mathcal{P}}]$ is maximal under the above restraints.

[5] Note that if bit conditions up to Q_{t_b-2} are fulfilled then ΔF_{t_b-1} has been ensured, but not ΔF_{t_b}.

Algorithm 4-1. Iterative construction of reduced differential path sets

```
1. Let $\widehat{t}$ be some step in the range $[t_b, t_e]$.
2. Construct the entire set $\mathcal{D}_{[\widehat{t},\widehat{t}]}$ of all possible differential paths over step $\widehat{t}$.
3. Compute $\mathcal{R}_{[\widehat{t},\widehat{t}]} = \{\text{Reduce}(\mathcal{P}) \mid \mathcal{P} \in \mathcal{D}_{[\widehat{t},\widehat{t}]}\}$.
4. For $i = \widehat{t}, \widehat{t}+1, \ldots, t_e - 1$, using the set $\mathcal{R}_{[\widehat{t},i]}$ we compute: $\mathcal{R}_{[\widehat{t},i+1]}$:
   (a) Let $A := \emptyset$.
   (b) For all $\mathcal{P} \in \mathcal{R}_{[\widehat{t},i]}$ and for all choices $\Delta Q_{i+2} \in \mathcal{Q}_{i+2}$, $\delta W_{i+1} \in \mathcal{W}_{i+1}$, $\Delta F_{i+1} \in \{-1,0,1\}^{31} \times \{0,1\}$ let $\widehat{\mathcal{P}}$ be the differential path over steps $\widehat{t}, \ldots, i+1$ given as $\mathcal{P}$ appended with $\Delta Q_{i+2}$, $\Delta F_{i+1}$ and $\delta W_{i+1}$.
       If $\Pr[\widehat{\mathcal{P}}] > 0$ then let $A := A \cup \{\text{Reduce}(\widehat{\mathcal{P}})\}$.
   (c) $\mathcal{R}_{[\widehat{t},i+1]} := A$.
5. For $i = \widehat{t}, \widehat{t}-1, \ldots, t_b + 1$, using the set $\mathcal{R}_{[i,t_e]}$ we compute $\mathcal{R}_{[i-1,t_e]}$:
   (a) Let $A := \emptyset$.
   (b) For all $\mathcal{P} \in \mathcal{R}_{[i,t_e]}$ and for all choices $\Delta Q_{i-5} \in \mathcal{Q}_{i-5}$, $\delta W_{i-1} \in \mathcal{W}_{i-1}$, $\Delta F_{i-1} \in \{-1,0,1\}^{31} \times \{0,1\}$ let $\widehat{\mathcal{P}}$ be the differential path over steps $i-1, \ldots, t_e$ given as $\mathcal{P}$ prepended with $\Delta Q_{i-5}$, $\Delta F_{i-1}$ and $\delta W_{i-1}$.
       If $\Pr[\widehat{\mathcal{P}}] > 0$ then let $A := A \cup \{\text{Reduce}(\widehat{\mathcal{P}})\}$.
   (c) $\mathcal{R}_{[i-1,t_e]} := A$.
6. Output $\mathcal{R}_{[t_b,t_e]}$.
```

One can interpret $\widehat{\mathcal{P}}$ as the differential path $\mathcal{P}$ with all differences removed that do not interact with the differences that constitute the starting and ending working state differences $\psi(\mathcal{P})$ and $\phi(\mathcal{P})$. We denote $\widehat{\mathcal{P}}$ as Reduce($\mathcal{P}$) and $\widetilde{\mathcal{P}}$ as $\mathcal{P} - \widehat{\mathcal{P}}$. In our proposed methodology, instead of directly computing the differential paths in $\mathcal{D}_{[t_b,t_e]}$ and their probabilities, we propose to work with the set of reduced differential paths $\mathcal{R}_{[t_b,t_e]} := \{\text{Reduce}(\mathcal{P}) \mid \mathcal{P} \in \mathcal{D}_{[t_b,t_e]}\}$ and cumulative probabilities $p_{(\mathcal{P},w)}$ for each reduced differential path $\mathcal{P}$ and w defined as:

$$p_{(\mathcal{P},w)} = \sum_{\substack{\mathcal{P}' \in \mathcal{D}_{[t_b,t_e]} \\ \mathcal{P}=\text{Reduce}(\mathcal{P}'), w=\omega(\mathcal{P}')}} \Pr[\mathcal{P}' - \mathcal{P}].$$

These cumulative probabilities have an easy interpretation using the equation:

$$\Pr[\mathcal{P}] \cdot p_{(\mathcal{P},w)} = \sum_{\substack{\mathcal{P}' \in \mathcal{D}_{[t_b,t_e]} \\ \mathcal{P}=\text{Reduce}(\mathcal{P}'), w=\omega(\mathcal{P}')}} \Pr[\mathcal{P}] \cdot \Pr[\mathcal{P}' - \mathcal{P}] = \sum_{\substack{\mathcal{P}' \in \mathcal{D}_{[t_b,t_e]} \\ \mathcal{P}=\text{Reduce}(\mathcal{P}'), w=\omega(\mathcal{P}')}} \Pr[\mathcal{P}']$$

As the working state differences $\phi(\mathcal{P})$ and $\psi(\mathcal{P})$ are unaffected by Reduce($\mathcal{P}$), the set of reduced differential paths and the cumulative probabilities are sufficient to determine the total success probability of any partition (b, e, w) of $\mathcal{D}_{[20,79]}$.

Moreover, the set $\mathcal{R}_{[t_b,t_e]}$ of reduced differential paths can be computed efficiently in an iterative manner as shown in Alg. 4-1. The cumulative probabilities can also be computed iteratively, but unfortunately the number of possible message difference vectors $w \in (\mathcal{W}_i)_{i=t_b}^{t_e}$ still grows exponentially in the number of local collisions over these steps.

Message Difference Vector Classes. To solve the problem of the exponential growth of possible message difference vectors, we consider classes $\overline{w}$ of message difference vectors w over steps $i, \ldots, j$, where any two $w \neq w'$ are in the same class $\overline{w}$ if and only if $p_{(\mathcal{P},w)} = p_{(\mathcal{P},w')}$ for all $\mathcal{P} \in \mathcal{R}_{[i,j]}$. It then suffices to compute the cumulative probabilities for only one representative $w \in \overline{w}$ for each class $\overline{w}$ over steps $t_b, \ldots, t_e$.

Let $\overline{W}_{[i,j]}$ be the set of all message difference vector classes $\overline{w}$ over steps $i, \ldots, j$. An important insight is that for any class $\overline{w}_{[i,j]} \in \overline{W}_{[i,j]}$ and any two $w, w' \in \overline{w}_{[i,j]}$ it holds that the extensions $w||\delta W_{j+1}$ and $w'||\delta W_{j+1}$ of w and w' with a difference δW_{j+1} are both in the same class $\overline{w}_{[i,j+1]} \in \overline{W}_{[i,j+1]}$. An analogous statement holds for prepending a δW_{i-1} to w and w'. These insights imply that it is sufficient to consider only one representative of each class in $\overline{W}_{[i,j]}$ to determine the sets $\overline{W}_{[i-1,j]}$ and $\overline{W}_{[i,j+1]}$.

In conclusion, with our two key techniques of differential path reduction and message difference vector classes, we are able to efficiently compute $\mathrm{FDC}_{[t_b,t_e]}$.

4.7 Results

We have computed $\mathrm{FDC}_{[20,79]}$ for several interesting disturbance vectors. These results are shown in Sect. B and show the maximum success probability of these disturbance vectors over the last 60 steps. Although the total complexity of a collision attack also depends on the complexity over the non-linear part, these results provide important insights which of these disturbance vectors may possibly lead to the fastest collision attack.

4.8 Improvements for the Last Few Steps of SHA-1

A common approach in constructing SHA-1 collision attacks is to remove the conditions for the last few steps as this will decrease the attack's overall complexity. The heuristic behind this effect is that for the last few steps some other differential paths that do not follow the disturbance vector actually have a higher success probability. Our approach can be adjusted by extending the sets $\mathcal{Q}_{76}, \ldots, \mathcal{Q}_{80}$ with differences ΔQ_i from these more likely alternative differential paths. We denote by $\mathrm{FDC}'_{[t_b,t_e]}$, $\mathcal{D}'_{[t_b,t_e]}$ and $\mathcal{R}'_{[t_b,t_e]}$ the respective function and sets wherein the extended sets $\mathcal{Q}'_{76}, \ldots, \mathcal{Q}'_{80}$ are used instead of $\mathcal{Q}_{76}, \ldots, \mathcal{Q}_{80}$. Algorithms that efficiently determine such extended sets $\mathcal{Q}'_{76}, \ldots, \mathcal{Q}'_{80}$ using ideas similar to the analysis in Sect. 4 are omitted here, but can be found in [Ste12a, Ch. 7.5].

5 New Collision Attacks on SHA-1

5.1 Open-Source Near-Collision Attack

In this section we present our near-collision attack on SHA-1 with an average complexity of $2^{57.5}$ compressions. Our near-collision attack is based on disturbance vector II(52,0). Below we describe how we used our new approach from

Table 5-1. SHA-1 near-collision differential path - round 1

t	Bitconditions: $\mathfrak{q}_t[31] \ldots \mathfrak{q}_t[0]$	ΔW_t
$-4, -3, -2$	`........ ........ ........ ........`	
-1	`...1.... ........ ........ ....0...`	
0	`.^.0.1.. .....0.1 ...00.10 .1..1..1`	$\{1, 26, 27\}$
1	`.0.+^-^^ ^^^^^1^0 ^^^11^10 .0..1.+0`	$\{\overline{4}, \overline{30}, 31\}$
2	`1-...+-- -------- -------- --.-1.+0`	$\{\overline{2}, 3, \overline{4}, \overline{26}, 28, 29, 31\}$
3	`.-.-.0.1 11111111 11110++1 +-1-00-0`	$\{\overline{2}, 26, 27, 28, \overline{29}\}$
4	`.-...1.0 11111111 1111-+++ ++0.1.+1`	$\{1, \overline{3}, 4, 26, \overline{27}, \overline{28}, \overline{29}, 31\}$
5	`.-...0.. ........ ......0. .+.+10+0`	$\{\overline{4}, \overline{29}\}$
6	`.-.+.... ........ ......01 100-.0+.`	$\{2, 3, 4, 26, \overline{29}\}$
7	`-1...1.. ........ ........ ...0.0..`	$\{2, 4, \overline{26}, \overline{27}, 29, 30, 31\}$
8	`1.1-.1.. ........ ........ .....1..`	$\{\overline{1}, 26, \overline{27}\}$
9	`..-..0.. ........ ........ ........`	$\{\overline{4}, 30, 31\}$
10	`^...00.. ........ ........ .......1`	$\{\overline{2}, \overline{3}, 4, \overline{26}, \overline{28}, 29, 31\}$
11	`..-.1... ........ ........ .......0`	$\{2, \overline{26}, 27, \overline{29}\}$
12	`0-..1... ........ ........ ......!.`	$\{\overline{3}, 4, \overline{26}, \overline{27}, \overline{28}, 29, 31\}$
13	`+..01... ........ ........ ........`	$\{\overline{4}, 28, 29, 31\}$
14	`..-1.... ........ ........ ......!.`	$\{\overline{2}, 3\}$
15	`+.0.1... ........ ........ ......!^`	$\{\overline{4}, \overline{27}, \overline{28}, \overline{29}, 31\}$
16	`+-0.0... ........ ........ ......!.`	$\{3, \overline{4}, \overline{27}\}$
17	`+..1.... ........ ........ ......^.`	$\{\overline{4}, \overline{27}, \overline{28}, \overline{29}, 30\}$
18	`-.+0.... ........ ........ ........`	$\{\overline{2}, 4, \overline{27}\}$
19	`-....... ........ ........ ........`	$\{4, 28, 29, \overline{30}\}$
20	`..+..... ........ ........ ........`	

Note: ΔW_t uses a compact notation, e.g., $\Delta W_t = +2^5 - 2^{10}$ is notated as $\{5, \overline{10}\}$.

Sect. 4 to determine which message bitrelations and δIHV_{out} to use and how we constructed the first round differential path. Collision search algorithms and various improvements using message modification techniques have already been covered extensively in the literature. We refer to our open-source implementation [Ste12b, Ste12a] for these details due to space considerations.

To apply our analysis of Sect. 4, we have chosen to use $t_b = 20$ (and $t_e = 79$). We use the improvements mentioned in Sect. 4.8 as this leads to higher success probabilities by a factor $2^{1.2}$. Let $\mathcal{D}' := \mathcal{D}'_{[20,79]}$, for $b \in \psi(\mathcal{D}')$, $e \in \phi(\mathcal{D}')$ and $w \in \omega(\mathcal{D}')$ we define $p_{b,e,w}$ and $p_{\max}$ as:

$$p_{b,e,w} = \sum_{\substack{\widehat{\mathcal{P}} \in \mathcal{D}' \\ \psi(\widehat{\mathcal{P}})=b,\ \phi(\widehat{\mathcal{P}})=e,\ \omega(\widehat{\mathcal{P}})=w}} \Pr[\widehat{\mathcal{P}}] \cdot c(b), \quad p_{\max} = \max_{b,e,w} p_{b,e,w}.$$

We algorithmically find a differential path over the first 20 steps that starts from $\delta IHV_{\text{in}} = 0$ and ends with working state differences $b \in \psi(\mathcal{D}')$ for which there are e and w such that $p_{b,e,w} = p_{\max}(=\text{FDC}'_{[20,79]}(\text{II}(52, 0)))$. The differential path over the first round that we selected for our near-collision attack is shown in Tbl. 5-1 and fixes a specific value $\widehat{b}$ and specific message differences $\delta\widehat{W}_0, \ldots, \delta\widehat{W}_{19}$.

Table 5-2. SHA-1 near-collision attack target δIHV_{diff} values

$$\begin{aligned}
\mathcal{I}_1 = \big\{&(2^{11}{+}2^4{-}2^2, 2^6, 2^{31}, 2^1, 2^{31}), (2^{12}{+}2^{11}{+}2^9{+}2^4{-}2^2, 2^7{+}2^6{+}2^4, 2^{31}, 2^1, 2^{31}),\\
&(2^{12}{+}2^3{+}2^1, 2^7, 0, 2^1, 2^{31}), (2^{12}{+}2^{11}{+}2^4{-}2^2, 2^7{+}2^6, 2^{31}, 2^1, 2^{31}),\\
&(2^{12}{+}2^4{-}2^1, 2^7, 0, 2^1, 2^{31}), (2^{11}{+}2^9{+}2^4{-}2^2, 2^6{+}2^4, 2^{31}, 2^1, 2^{31}),\\
&(2^{12}{+}2^9{+}2^3{+}2^1, 2^7{+}2^4, 0, 2^1, 2^{31}), (2^{12}{+}2^9{+}2^4{-}2^1, 2^7{+}2^4, 0, 2^1, 2^{31})\big\};\\
\mathcal{I}_2 = \big\{&(v_1{+}c_1\cdot 2^3{-}c_2\cdot 2^5, v_2, v_3, v_4{-}c_3\cdot 2^2, v_5) \;\big|\; (v_i)_{i=1}^5 \in \mathcal{I}_1,\ c_1, c_2, c_3 \in \{0,1\}\big\};\\
\mathcal{I}_3 = \big\{&(v_1{-}c\cdot 2^{13}, v_2{-}c\cdot 2^8, v_3, v_4, v_5) \;\big|\; (v_i)_{i=1}^5 \in \mathcal{I}_2,\ c \in \{0,1\}\big\};\\
\widetilde{\mathcal{I}} = \big\{&(v_1{-}c\cdot 2^9, v_2{-}c\cdot 2^4, v_3, v_4, v_5) \;\big|\; (v_i)_{i=1}^5 \in \mathcal{I}_3,\ c \in \{0,1\}\big\};
\end{aligned}$$

Note: the resulting set $\widetilde{\mathcal{I}}$ has 192 unique target δIHV_{diff} values. Furthermore, for any $\delta IHV_{\text{diff}} \in \widetilde{\mathcal{I}}$ also $-\delta IHV_{\text{diff}} \in \widetilde{\mathcal{I}}$.

To maximize the success probability, we only accept δIHV_{out} in the set $\{e \in \phi(\mathcal{D}') \mid \exists w : p_{\widehat{\mathfrak{b}},e,w} > 0.9p_{\max}\}$. We can further decrease overall complexity by only allowing w that maximize the number of $e = \delta IHV_{\text{out}}$ with $p_{\widehat{\mathfrak{b}},e,w} > 0.9p_{\max}$. The near-collision attack gains a speed up due to the fact that it always has several chances of finding a target δIHV_{out}. Note that a possible second near-collision attack (for an identical-prefix collision attack) does not have the benefit of the speedup as it targets one specific $\delta IHV_{\text{out}} = 0$. More formally, for each $w \in \omega(\mathcal{D}')$, we count the number N_w of values e for which $p_{\widehat{\mathfrak{b}},e,w} > 0.9p_{\max}$. Let $N_{\max} := \max_w N_w$ (which is 6 in our case) then we limit the allowed message difference vectors to the set $\mathfrak{W}_{[20,79]} = \{w \mid N_w = N_{\max}\}$. Hence, we only accept values for δIHV_{out} in the set $\{e \in \phi(\mathcal{D}') \mid \exists w \in \mathfrak{W}_{[20,79]} : p_{\widehat{\mathfrak{b}},e,w} > 0.9p_{\max}\}$. In this manner we have found 192 target δIHV_{out}-values (see Tbl. 5-2).

With the differential path and the set of allowed δIHV_{out} known, we only need the message bit relations to construct a collision attack. We translate the set $\mathfrak{W}_{[20,79]}$ and the vector $(\widehat{\delta W_i})_{i=0}^{19}$ into a smallest sufficient set of linear bit relations on the message words bits using linear algebra (see Sect. A).

Using the differential path, the message bitrelations and the set of allowed δIHV_{out}, we have implemented a near-collision attack. For more details, we refer to the source code which is available online at [Ste12b]. For more convenient analysis, the attack is split in four subsequent stages:

1. The first stage finds a message block pair satisfying the message bitrelations and which results in $\delta Q_i = 0$ for $i = 29, 30, 31, 32, 33$. This stage is the most complex and contains all speed ups using message modification techniques.
2. The second stage is to find a message block pair that satisfies the message bitrelations and results in $\delta Q_i = 0$ for $i = 49, 50, 51, 52, 53$.
3. The third stage is to find a message block pair that satisfies the message bitrelations and results in $\delta Q_i = 0$ for $i = 57, 58, 59, 60, 61$.
4. The fourth and final stage is to find a message block pair that results in one of the 192 target δIHV_{out} in Tbl. 5-2.

The last three stages cannot use any freedoms anymore and thereby either are or are not successful with some probability. The average total complexity of our

Table 5-3. Example message pair each consisting of an identical-prefix block and a near-collision block satisfying our differential path up to step 66

First message	Second message
`bc7e393a0470f684 e0a484dea556875a`	`bc7e393a0470f684 e0a484dea556875a`
`cddff9c82d02016b 860ee7f911e18418`	`cddff9c82d02016b 860ee7f911e18418`
`71bfbff1067095c9 ed44afee78122409`	`71bfbff1067095c9 ed44afee78122409`
`a3b2eb2e16c0cfc2 06c5202810383c2b`	`a3b2eb2e16c0cfc2 06c5202810383c2b`
`73e6e2c8437fb13e 4e4d5db6e383e01d`	`7fe6e2ca837fb12e fa4d5daadf83e019`
`7bea242c2bb63054 6845b1430c2194ab`	`c7ea24360bb63044 4c45b15fe02194bf`
`fb5236be2bc91e19 1d11bf8f665ef9ab`	`f75236bcebc91e09 a911bf934a5ef9af`
`9f8fe36a402cbf39 d77c1fb43cb00872`	`238fe372f02cbf29 d77c1fb884b00862`

near-collision attack is thus the average complexity of the first stage divided by the product of the success probabilities for the last three stages. Our implementation outputs the throughput of the first stage in #/s as 'timeavg 40', and the success probabilities of the last three stages as 'avg 53 stats', 'avg 61 stats' and 'avg 80 stats', respectively. Using these numbers one can easily determine the average complexity in SHA-1 compressions to find a near-collision. With profiling and tuned optimization flags for the compiler and many hours-long runs, we determined an average complexity of the first stage to be $2^{20.91}$ SHA-1 compressions per message block pair. Using our novel analysis for step ranges [33,52], [53,60] and [61,79] and $N_{\max} = 6$, we determined the exact success probabilities for the last three stages, namely, $2^{-20.91}$, 2^8 and $2^{16.65}$, respectively. These probabilities were verified by our implemented attack. Hence, the total complexity of our near-collision is $2^{11.97} \cdot 2^{20.91} \cdot 2^{8.00} \cdot 2^{16.65} = 2^{57.53}$ SHA-1 compressions. Finally, we like to note that with more than 50 bits of the 512 message bits left as degrees of freedom, there is ample room to further optimize the first stage with message modification techniques.

We provide an example message pair in Tbl. 5-3 that successfully passed the first three stages of our near-collision attack (at a cost of about $2^{40.9}$ compressions).

5.2 Identical-Prefix Collision Attack on SHA-1

The near-collision attack of Sect. 5.1 can directly be used in a two-block identical-prefix collision attack on SHA-1.[6] The second near-collision block of the two blocks cancels the δIHV_{out} resulting from the first near-collision block.

For the second near-collision block, we follow the steps as described in Sect. 5.1 with two modifications. Firstly, in Sect. 5.1 we allow only $\delta IHV_{\text{out}} = 0$ (thus δIHV_{in} is canceled). This leads to $N_{\max} = 1$ and a different set of optimal message difference vectors $\mathfrak{W}_{[20,79]}$. Hence, the total complexity over the last

[6] An additional identical-prefix block is used to satisfy a few bitconditions on the *IHV* (see Tbl. 5-1) and furthermore to simplify implementation and to allow very easy parallelization. It should be possible to remove this prefix block with only a negligible impact on the attack complexity.

three stages increases by a factor 6. Secondly, instead of using a differential path starting with $\delta IHV_{\text{in}} = 0$ in Sect. 5.1, we use a differential path that starts with the (IHV, IHV') resulting from the first near-collision block.

A lower-bound for the complexity of a complete two-block identical-prefix collision attack based on our current near-collision implementation is about $(1 + 6) \cdot 2^{57.5} \approx 2^{60.3}$ compressions, as the first near-collision attack has the luxury of six allowed values for δIHV_{out} for each possible $(\delta W_t)_{t=0}^{79}$, whereas the second near-collision attack must target one specific δIHV_{out}. As the second-block differential path will differ roughly only up to $\mathfrak{q}_4$, almost all used message modification techniques will be unaffected. Also, there will still be a relative large amount of freedoms left to further apply message modification techniques. Hence, it is reasonable to expect a similar complexity in the first stage (first 32 steps). Nevertheless, leaving room for a small set back, we estimate the average complexity of our identical-prefix collision attack for SHA-1 to be equivalent to 2^{61} SHA-1 compressions.

5.3 Chosen-Prefix Collision Attack

We present a chosen-prefix collision attack on SHA-1 using the second near-collision attack of Sect. 5.2 that does the following. Given chosen prefixes P and P', we first append bit strings S_{r} and S'_{r} such that the bit lengths of $P||S_{\text{r}}$ and $P'||S'_{\text{r}}$ are both equal to $N \cdot 512 - 119$. By processing the first $N - 1$ blocks of $P||S_{\text{r}}$ and $P'||S'_{\text{r}}$, we obtain IHV_{N-1} and IHV'_{N-1}, resp. Furthermore, let B and B' be the last $512 - 119$ bits of $P||S_{\text{r}}$ and $P'||S'_{\text{r}}$, resp. The next step is to perform a birthday search as explained in [vOW99] using a search space V and a step function $f : V \to V$. We define $V = \{0,1\}^{119}$ and f (based on Tbl. 5-2) as:

$$f(v) = \begin{cases} \phi\big(\text{Compress}(IHV_{N-1}, B||v)\big) & \text{if } w(v) = 0 \bmod 2; \\ \phi\big(\text{Compress}(IHV'_{N-1}, B'||v) - (0,0,0,0,2^{31})\big) & \text{if } w(v) = 1 \bmod 2, \end{cases}$$

$$\phi(a,b,c,d,e) = (a[i])_{i=19}^{31}||(b[i])_{i=14}^{31}||(c[i])_{i=0}^{30}||(d[i])_{i=7}^{31}||e$$

The probability that a birthday collision results in one of the 192 target δIHV_{out} is found to be approximately $2^{-33.46}$ using Monte Carlo simulations. Therefore, a birthday search collision pair v, w with $f(v) = f(w)$ has a probability of $q = 2^{-33.46-1}$ that $\tau(v) \neq \tau(w)$ and δIHV_N is one of the 192 target δIHV_{out}-values. Using the analysis from [vOW99], this implies that the expected birthday search complexity in SHA-1 compressions is $\sqrt{\pi \cdot |V|/(2 \cdot q)} \approx 2^{77.06}$.

To complete the chosen-prefix collision attack it remains to find a near-collision block that cancels δIHV_N. But as δIHV_N is one of the 192 target δIHV_{out}, we can directly use the construction of the second near-collision block of Sect. 5.2, whose complexity is significantly lower than $2^{77.06}$. Hence, the overall cost of a chosen-prefix collision attack on SHA-1 is dominated by the expected $2^{77.1}$ SHA-1 compressions required for the birthday search.

6 Concluding Remarks

We have presented new collision attacks on SHA-1, most importantly an identical-prefix collision attack with an average complexity of 2^{61} compressions. With the construction of these attacks, we focused mostly on obtaining the highest success probability that is theoretically possible over the linear part. Our novel direction in the cryptanalysis of SHA-1 is essentially based on an exhaustive and exact analysis of all possible differential paths that follow the disturbance vector. This is in contrast to previous approaches that combine success probabilities and conditions of individual local collisions with heuristic corrections. In this paper we have introduced the foundations of our novel direction. For a complete and rigorous mathematical treatment we refer to the full version [Ste12a].

As our attacks have still over 50 out of the 512 message bits left as degrees of freedom for further improvements using message modification techniques, we hope that our novel methods provide the necessary advantage to construct attacks with complexity well below 2^{61} compressions and thereby contributes to the search for the long-anticipated first SHA-1 collision.

References

[BCJ+05] Biham, E., Chen, R., Joux, A., Carribault, P., Lemuet, C., Jalby, W.: Collisions of SHA-0 and reduced SHA-1. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 36–57. Springer, Heidelberg (2005)

[Che11] Chen, R.: New Techniques for Cryptanalysis of Cryptographic Hash Functions, Ph.D. thesis, Technion (August 2011)

[CJ98] Chabaud, F., Joux, A.: Differential Collisions in SHA-0. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 56–71. Springer, Heidelberg (1998)

[CMR07] De Cannière, C., Mendel, F., Rechberger, C.: Collisions for 70-Step SHA-1: On the Full Cost of Collision Search. In: Adams, C., Miri, A., Wiener, M. (eds.) SAC 2007. LNCS, vol. 4876, pp. 56–73. Springer, Heidelberg (2007)

[Coc07] Cochran, M.: Notes on the Wang et al. 2^{63} SHA-1 Differential Path, Cryptology ePrint Archive, Report 2007/474 (2007)

[CR06] De Cannière, C., Rechberger, C.: Finding SHA-1 Characteristics: General Results and Applications. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 1–20. Springer, Heidelberg (2006)

[GA11] Grechnikov, E.A., Adinetz, A.V.: Collision for 75-step SHA-1: Intensive Parallelization with GPU, Cryptology ePrint Archive, Report 2011/641 (2011)

[Gre10] Grechnikov, E.A.: Collisions for 72-step and 73-step SHA-1: Improvements in the Method of Characteristics, Cryptology ePrint Archive, Report 2010/413 (2010)

[Man11] Manuel, S.: Classification and generation of disturbance vectors for collision attacks against SHA-1. Des. Codes Cryptography 59(1-3), 247–263 (2011)

[MHP09] McDonald, C., Hawkes, P., Pieprzyk, J.: Differential Path for SHA-1 with complexity $O(2^{52})$, Cryptology ePrint Archive, Report 2009/259 (2009)

[MPRR06] Mendel, F., Pramstaller, N., Rechberger, C., Rijmen, V.: The Impact of Carries on the Complexity of Collision Attacks on SHA-1. In: Robshaw, M. (ed.) FSE 2006. LNCS, vol. 4047, pp. 278–292. Springer, Heidelberg (2006)

[MRR07] Mendel, F., Rechberger, C., Rijmen, V.: Update on SHA-1, Rump session of CRYPTO 2007 (2007)

[PCTH11] Polk, T., Chen, L., Turner, S., Hoffman, P.: Security Considerations for the SHA-0 and SHA-1 Message-Digest Algorithms, Internet Request for Comments, RFC 6194 (March 2011)

[Ste12a] Stevens, M.: Attacks on Hash Functions and Applications, Ph.D. thesis, Leiden University (June 2012)

[Ste12b] Stevens, M.: SHA-1 near collision attack source code (2012), `https://hashclash.googlecode.com/files/sha1_nearcoll_attack.zip`

[vOW99] van Oorschot, P.C., Wiener, M.J.: Parallel Collision Search with Cryptanalytic Applications. J. Cryptology 12(1), 1–28 (1999)

[WFLY04] Wang, X., Feng, D., Lai, X., Yu, H.: Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD, Cryptology ePrint Archive, Report 2004/199 (2004)

[WY05] Wang, X., Yu, H.: How to Break MD5 and Other Hash Functions. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 19–35. Springer, Heidelberg (2005)

[WYY05a] Wang, X., Yao, A.C., Yao, F.: Cryptanalysis on SHA-1, NIST Cryptographic Hash Workshop Presentation (2005)

[WYY05b] Wang, X., Yin, Y.L., Yu, H.: Finding collisions in the full SHA-1. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 17–36. Springer, Heidelberg (2005)

[WYY05c] Wang, X., Yu, H., Yin, Y.L.: Efficient Collision Search Attacks on SHA-0. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 1–16. Springer, Heidelberg (2005)

A Deriving Message Bitrelations

For each $\widehat{w} = (\widehat{\delta W_i})_{i=20}^{79} \in \mathfrak{W}_{[20,79]}$ we define the set $\mathcal{V}_{\widehat{w}}$ as the set of all $(W_i)_{i=0}^{79}$ that 'result' in $\widehat{w}$, i.e., $(W_i \oplus DW_i) - W_i = \widehat{\delta W_i}$ for all $i \in \{20, \ldots, 79\}$. Let the set $\mathcal{V} = \bigcup_{w \in \mathfrak{W}_{[20,79]}} \mathcal{V}_w$ consist of all $(W_t)_{t=0}^{79}$ that are compatible with some $w \in \mathfrak{W}_{[20,79]}$. Furthermore, let $\mathcal{V}'$ be the set consisting of all elements of $\mathcal{V}$ mapped to $\mathbb{F}_2^{32\cdot 80}$. We search for an affine subspace $y + \mathcal{U} \subseteq \mathcal{V}'$ which is as large as possible. Choose any basis of $\mathcal{U}^\perp$ of size k and let the k rows of the matrix $A_{[20,79]} \in \mathbb{F}_2^{k\times(32\cdot 80)}$ consist of the k basis vectors of $\mathcal{U}^\perp$. It follows that $x \in \mathcal{U} \Leftrightarrow A_{[20,79]} \cdot x = 0$ and thus $x \in y + \mathcal{U} \Leftrightarrow A_{[20,79]} \cdot x = A_{[20,79]} \cdot y$. The matrix equation $A_{[20,79]} \cdot x = c_{[20,79]}$ with $c_{[20,79]} = A_{[20,79]} \cdot y$ describes sufficient linear bit relations for steps 20 up to 79.[7]

The set $\mathfrak{W}_{[0,19]} = \{(\widehat{\delta W_i})_{i=0}^{19}\}$ similarly leads to a matrix equation $A_{[0,19]} \cdot x = c_{[0,19]}$. The two matrix equations can be combined into a single matrix

[7] Although this seems to be impractical, we can compute this efficiently by splitting it into independent parts and using well chosen representations.

equation $A_{[0,79]} \cdot x = c_{[0,79]}$ that defines our message search space. Finally, this matrix equation over the $32 \cdot 80$ message words bits is reduced using the message expansion relation to a matrix equation over the 512 message block bits.

B SHA-1 Disturbance Vector Analysis

Tbl. B-1 is based on the disturbance vector cost function $\mathrm{FDC}_{[t_b,t_e],u}$ that is defined as similar to $\mathrm{FDC}_{[t_b,t_e]}$, but under the additional constraint that only up to u carries are allowed in the working state differences ΔQ_i. More formally, we define:

$$\mathcal{Q}_{t,u} := \left\{ \text{BSDR } Y \;\middle|\; \substack{\sigma(Y)=\sigma(Z), \\ Z[i]\in\{-DV_{t-1}[i],DV_{t-1}[i]\},\ i=0,\ldots,31, \\ w(Y)\le u+\min_{X\in\mathcal{Q}_t} w(X).} \right\};$$

$$\mathcal{D}_{[t_b,t_e],u} := \left\{ \widehat{\mathcal{P}} \;\middle|\; \Delta\widehat{Q_i} \in \mathcal{Q}_{i,u},\ \delta\widehat{W_j} \in \mathcal{W}_j,\ \Pr[\widehat{\mathcal{P}}] > 0 \right\};$$

$$\mathrm{FDC}_{[t_b,t_e],u}\left((DV_t)_{t=0}^{79}\right) = \max_{b,e,w} \sum_{\substack{\widehat{\mathcal{P}}\in\mathcal{D}_{[t_b,t_e],u} \\ \psi(\widehat{\mathcal{P}})=b,\ \phi(\widehat{\mathcal{P}})=e,\ \omega(\widehat{\mathcal{P}})=w}} \Pr[\widehat{\mathcal{P}}] \cdot c(b),$$

where $c(b) = c((\Delta Q_i)_{i=t_b-4}^{t_e})$ is the correction factor $c(b) = \prod_{i=t_b-4}^{t_b-2} 2^{w(\Delta\widehat{Q}_i)}$.

The tables below contain notes $\epsilon = 0, 1/8, 1/4, 1/2$ for each entry. This note indicates whether in our algorithms to compute $\mathrm{FDC}_{[t_b,t_e],u}$ we removed certain message difference vectors w that had a 'total success probability of w' less than ϵ times the highest 'total success probability over all w''. Although, we won't go into the details of the notationally heavy definition of this 'total success probability', it is clear that choosing $\epsilon = 0$ will cause no message difference vector to be removed. Choosing $\epsilon > 0$ will result in that the maximum taken in $\mathrm{FDC}_{[t_b,t_e],u}$ will actually be taken over a subset of all values w. Hence, choosing $\epsilon > 0$ can only affect the outcome in a negative way, i.e., a smaller maximum success probability. Although for ϵ close to 1, this removal of message difference vectors does affect the outcome (in a negative way), we have not seen this happen for $\epsilon \le 0.5$ for all selected studied cases. Choosing $\epsilon > 0$ allows us to compute lower-bounds for $\mathrm{FDC}_{[t_b,t_e],u}$ for disturbance vectors and values for u that were otherwise prohibitive for our particular machine due to memory requirements. We argue that for up to $\epsilon \le 0.5$ these values are not just lower-bounds, but in fact the correct outcome for $\mathrm{FDC}_{[t_b,t_e],u}$, which is backed-up by the fact that for increasing u these outcomes increase as expected and no sudden decrease is seen (or, when taking the $-\log_2$, decrease as expected and no sudden increase is seen).

Table B-1. Disturbance vector results

DV	u				
	0	1	2	3	7
I(42, 0)	82.68 $\epsilon=0$	78.67 $\epsilon=0$	78.36 $\epsilon=1/4$		
I(43, 0)	82.00 $\epsilon=0$	77.65 $\epsilon=0$	77.31 $\epsilon=1/8$		
I(44, 0)	81.00 $\epsilon=0$	77.41 $\epsilon=0$	77.1 $\epsilon=0$	76.98 $\epsilon=0$	76.89 $\epsilon=1/8$
I(45, 0)	81.00 $\epsilon=0$	76.91 $\epsilon=0$	76.66 $\epsilon=0$	76.54 $\epsilon=0$	76.45 $\epsilon=1/8$
I(46, 0)	79.00 $\epsilon=0$	75.02 $\epsilon=0$	74.92 $\epsilon=0$	74.84 $\epsilon=0$	74.83 $\epsilon=1/8$
I(47, 0)	79.00 $\epsilon=0$	75.15 $\epsilon=0$	74.83 $\epsilon=0$	74.71 $\epsilon=0$	74.61 $\epsilon=0$
I(**48, 0**)	75.00 $\epsilon=0$	71.84 $\epsilon=0$	71.61 $\epsilon=0$	71.51 $\epsilon=0$	71.42 $\epsilon=0$
I(**49, 0**)	76.00 $\epsilon=0$	72.59 $\epsilon=0$	72.34 $\epsilon=0$	72.24 $\epsilon=0$	72.15 $\epsilon=0$
I(**50, 0**)	75.00 $\epsilon=0$	72.02 $\epsilon=0$	71.95 $\epsilon=0$	71.93 $\epsilon=0$	71.92 $\epsilon=0$
I(51, 0)	77.00 $\epsilon=0$	73.76 $\epsilon=0$	73.53 $\epsilon=0$	73.43 $\epsilon=0$	73.34 $\epsilon=0$
I(52, 0)	79.00 $\epsilon=0$	76.26 $\epsilon=0$	76.24 $\epsilon=0$	76.24 $\epsilon=0$	76.24 $\epsilon=0$
I(53, 0)	82.83 $\epsilon=0$	78.86 $\epsilon=0$	78.79 $\epsilon=0$	78.77 $\epsilon=0$	78.77 $\epsilon=0$
I(54, 0)	82.83 $\epsilon=0$	79.60 $\epsilon=0$	79.38 $\epsilon=0$	79.28 $\epsilon=0$	79.19 $\epsilon=0$
I(55, 0)	81.54 $\epsilon=0$	78.67 $\epsilon=0$	78.42 $\epsilon=0$	78.32 $\epsilon=0$	78.23 $\epsilon=0$
I(56, 0)	81.54 $\epsilon=0$	79.10 $\epsilon=0$	79.03 $\epsilon=0$	79.01 $\epsilon=0$	79.01 $\epsilon=0$
I(42, 2)	85.09 $\epsilon=0$	82.17 $\epsilon=1/4$	81.84 $\epsilon=1/2$	81.72 $\epsilon=1/2$	
I(43, 2)	84.42 $\epsilon=0$	81.15 $\epsilon=1/4$	80.78 $\epsilon=1/2$		
I(44, 2)	84.42 $\epsilon=0$	81.92 $\epsilon=0$	81.57 $\epsilon=1/4$	81.45 $\epsilon=1/2$	81.36 $\epsilon=1/2$
I(45, 2)	83.42 $\epsilon=0$	80.80 $\epsilon=0$	80.52 $\epsilon=0$	80.41 $\epsilon=1/4$	80.32 $\epsilon=1/2$
I(46, 2)	80.42 $\epsilon=0$	78.10 $\epsilon=0$	78.00 $\epsilon=0$	77.99 $\epsilon=1/8$	77.99 $\epsilon=1/4$
I(47, 2)	79.68 $\epsilon=0$	77.01 $\epsilon=0$	76.68 $\epsilon=0$	76.56 $\epsilon=0$	76.47 $\epsilon=1/8$
I(48, 2)	76.68 $\epsilon=0$	74.27 $\epsilon=0$	73.99 $\epsilon=0$	73.88 $\epsilon=0$	73.79 $\epsilon=0$
I(49, 2)	77.00 $\epsilon=0$	74.30 $\epsilon=0$	74.02 $\epsilon=0$	73.92 $\epsilon=0$	73.83 $\epsilon=0$
I(50, 2)	77.00 $\epsilon=0$	74.74 $\epsilon=0$	74.63 $\epsilon=0$	74.61 $\epsilon=0$	74.60 $\epsilon=0$
I(51, 2)	80.00 $\epsilon=0$	77.47 $\epsilon=0$	77.21 $\epsilon=0$	77.11 $\epsilon=0$	77.03 $\epsilon=0$
I(52, 2)	82.00 $\epsilon=0$	79.98 $\epsilon=0$	79.93 $\epsilon=0$	79.92 $\epsilon=0$	79.92 $\epsilon=0$
I(53, 2)	84.00 $\epsilon=0$	81.91 $\epsilon=0$	81.80 $\epsilon=0$	81.78 $\epsilon=0$	81.78 $\epsilon=0$
I(54, 2)	84.00 $\epsilon=0$	81.37 $\epsilon=0$	81.06 $\epsilon=0$	80.95 $\epsilon=0$	80.85 $\epsilon=0$
I(55, 2)	84.00 $\epsilon=0$	81.78 $\epsilon=0$	81.53 $\epsilon=0$	81.43 $\epsilon=0$	81.34 $\epsilon=0$
I(56, 2)	82.00 $\epsilon=0$	80.22 $\epsilon=0$	80.13 $\epsilon=0$	80.12 $\epsilon=0$	80.11 $\epsilon=0$
II(44, 0)	87.00 $\epsilon=0$	79.51 $\epsilon=1/2$			
II(45, 0)	83.00 $\epsilon=0$	75.45 $\epsilon=1/8$	74.82 $\epsilon=1/2$		
II(**46, 0**)	76.00 $\epsilon=0$	71.85 $\epsilon=0$	71.83 $\epsilon=1/2$		
II(47, 0)	81.42 $\epsilon=0$	76.23 $\epsilon=0$	75.87 $\epsilon=1/2$		
II(48, 0)	80.00 $\epsilon=0$	76.11 $\epsilon=0$	75.89 $\epsilon=0$	75.79 $\epsilon=1/8$	
II(49, 0)	80.00 $\epsilon=0$	75.04 $\epsilon=0$	74.72 $\epsilon=0$	74.60 $\epsilon=0$	74.51 $\epsilon=1/2$
II(**50, 0**)	78.00 $\epsilon=0$	73.52 $\epsilon=0$	73.23 $\epsilon=0$	73.12 $\epsilon=0$	73.02 $\epsilon=0$
II(**51, 0**)	77.00 $\epsilon=0$	72.55 $\epsilon=0$	72.18 $\epsilon=0$	72.02 $\epsilon=0$	71.88 $\epsilon=0$
II(**52, 0**)	75.00 $\epsilon=0$	71.88 $\epsilon=0$	71.87 $\epsilon=0$	71.76 $\epsilon=0$	71.75 $\epsilon=0$
II(53, 0)	76.96 $\epsilon=0$	73.65 $\epsilon=0$	73.34 $\epsilon=1/8$	73.23 $\epsilon=1/8$	73.14 $\epsilon=1/8$
II(54, 0)	77.96 $\epsilon=0$	73.97 $\epsilon=0$	73.74 $\epsilon=1/8$	73.64 $\epsilon=1/8$	73.55 $\epsilon=1/8$
II(55, 0)	77.96 $\epsilon=0$	75.22 $\epsilon=1/8$	74.99 $\epsilon=1/2$	74.89 $\epsilon=1/2$	74.80 $\epsilon=1/2$
II(56, 0)	76.96 $\epsilon=0$	74.48 $\epsilon=1/2$	74.18 $\epsilon=1/2$	74.07 $\epsilon=1/2$	73.97 $\epsilon=1/2$
II(45, 2)	85.00 $\epsilon=0$	78.64 $\epsilon=1/2$			
II(46, 2)	82.00 $\epsilon=0$	77.51 $\epsilon=1/2$			
II(47, 2)	85.42 $\epsilon=0$	79.83 $\epsilon=1/2$			
II(48, 2)	83.00 $\epsilon=0$	78.81 $\epsilon=1/2$	78.46 $\epsilon=1/2$		
II(49, 2)	83.00 $\epsilon=0$	78.09 $\epsilon=0$	77.74 $\epsilon=1/2$		
II(50, 2)	81.00 $\epsilon=0$	76.51 $\epsilon=0$	76.16 $\epsilon=1/8$	76.03 $\epsilon=1/8$	
II(51, 2)	82.00 $\epsilon=0$	77.74 $\epsilon=0$	77.36 $\epsilon=1/8$	77.20 $\epsilon=1/8$	
II(52, 2)	82.00 $\epsilon=0$	79.07 $\epsilon=0$	78.96 $\epsilon=0$	78.94 $\epsilon=0$	78.93 $\epsilon=1/2$
II(53, 2)	83.00 $\epsilon=0$	79.60 $\epsilon=0$	79.30 $\epsilon=0$	79.18 $\epsilon=0$	79.09 $\epsilon=1/8$
II(54, 2)	84.00 $\epsilon=0$	80.49 $\epsilon=0$	80.21 $\epsilon=0$	80.10 $\epsilon=0$	80.00 $\epsilon=1/8$
II(55, 2)	84.00 $\epsilon=0$	81.20 $\epsilon=0$	80.88 $\epsilon=0$	80.76 $\epsilon=0$	80.67 $\epsilon=1/8$
II(56, 2)	85.00 $\epsilon=0$	82.69 $\epsilon=1/4$	82.39 $\epsilon=1/4$	82.27 $\epsilon=1/4$	82.18 $\epsilon=1/4$

Note: the columns are the negative $\log_2$ results of the cost function $\mathrm{FDC}_{[20,79],u}$.

Improving Local Collisions: New Attacks on Reduced SHA-256

Florian Mendel, Tomislav Nad, and Martin Schläffer

IAIK, Graz University of Technology, Austria
florian.mendel@iaik.tugraz.at

Abstract. In this paper, we focus on the construction of semi-free-start collisions for SHA-256, and show how to turn them into collisions. We present a collision attack on 28 steps of the hash function with practical complexity. Using a two-block approach we are able to turn a semi-free-start collision into a collision for 31 steps with a complexity of at most $2^{65.5}$. The main improvement of our work is to extend the size of the local collisions used in these attacks. To construct differential characteristics and confirming message pairs for longer local collisions, we had to improve the search strategy of our automated search tool. To test the limits of our techniques we present a semi-free-start collision for 38 steps.

Keywords: hash functions, SHA-2, cryptanalysis, collisions, semi-free-start collisions, differential characteristics, automatic search tool.

1 Introduction

Since 2005, many collision attacks have been shown for commonly used and standardized hash functions. In particular, the collision attacks of Wang et al. [17,18] on MD5 and SHA-1 have convinced many cryptographers that these widely deployed hash functions can no longer be considered secure. As a consequence, NIST has proposed the transition from SHA-1 to the SHA-2 family. Many companies and organization follow this advice and migrate to SHA-2. Additionally, SHA-2 is faster on many platforms than the recently chosen winner of the SHA-3 competition [14]. Hence, NIST explicitly recommends both, SHA-2 and SHA-3. Therefore, the cryptanalysis of SHA-2 is still of high interest.

In the last few years several cryptanalytic results have been published for SHA-256. The security of SHA-256 against preimage attacks was first studied by Isobe and Shibutani in [5]. They have presented a preimage attack on 24 out of 64 steps. This was improved by Aoki et al. to 43 steps in [1] and later extended to 45 steps by Khovratovich et al. in [6]. In [8] Li et al. have shown how a particular preimage attack can be used to construct a free-start collision attack for 52 steps of SHA-256. All attacks are only slightly faster than the generic attack which has a complexity of 2^{256} for preimages and 2^{128} for free-start collisions. Furthermore, second-order differential collisions for SHA-256 up to 47 steps with practical complexity have been shown in [2,7].

In [15] Nikolić and Biryukov, studied the security of SHA-256 with respect to collision attacks. They found a differential characteristic resulting in a collision attack for 23 steps of SHA-256. Later this approach was extended to a collision attack on 24 steps [4, 16]. All these results use rather simple local collisions spanning over 9 steps, which are constructed mostly manually or using basic cryptanalytic tools. However, as pointed out in [4] it is unlikely that this approach can be extended beyond 24 steps.

Recently, Mendel et al. proposed a technique to use local collisions spanning over a higher number of steps. The main improvement is to use an automated search tool to construct the differential characteristics and to find confirming message pairs. Using local collisions spanning over more than 9 steps, a collision attack on 27 steps and a semi-free-start collision attack on 32 steps of SHA-256 has been shown. Both attacks have practical complexity. Currently, these are the best collision attacks on SHA-256 with practical complexity.

In this paper, we improve upon these collision attacks on SHA-256. We present collisions for the hash function on up to 31 out of 64 steps with complexity of at most $2^{65.5}$, and semi-free-start collisions on 38 steps with complexity of 2^{37}. We get these attacks by extending the size of the local collision to up to 18 steps. Furthermore, we try to ensure that the first message words do not contain any differences. This way, we can convert most of our semi-free-start collision attacks into collision attacks on the hash function.

The remainder of this paper is structured as follows. A description of the hash function is given in Sect. 2. A high-level overview of our attacks is given in Sect. 3. In Sect. 4 we show how we construct local collisions spanning over a higher number of steps. We show how to construct the differential characteristics in Sect. 5. In Sect. 6 we present our results and show how to turn (some of) the semi-free-start collision into collisions. Finally, we conclude in Sect. 7.

2 Description of SHA-256

SHA-256 is an iterated hash function that pads and processes the input message using t 512-bit message blocks m_j. The 256-bit hash value is computed using the compression function f:

$$\begin{aligned} h_0 &= IV \\ h_{j+1} &= f(h_j, m_j) \qquad \text{for } 0 \le j < t \\ hash &= h_t \end{aligned}$$

In the following, we briefly describe the compression function f of SHA-256. It basically consists of two parts: the message expansion and the state update transformation. A more detailed description of SHA-256 is given in [13].

2.1 Message Expansion

The message expansion of SHA-256 splits the 512-bit message block into 16 words M_i, $i = 0, \ldots, 15$, and expands them into 64 expanded message words W_i as follows:

$$W_i = \begin{cases} M_i & 0 \le i < 16 \\ \sigma_1(W_{i-2}) + W_{i-7} + \sigma_0(W_{i-15}) + W_{i-16} & 16 \le i < 64 \end{cases}$$

The functions $\sigma_0(x)$ and $\sigma_1(x)$ are given by

$$\sigma_0(x) = (x \ggg 7) \oplus (x \ggg 18) \oplus (x \gg 3)$$
$$\sigma_1(x) = (x \ggg 17) \oplus (x \ggg 19) \oplus (x \gg 10)$$

2.2 State Update Transformation

We use the alternative description of the SHA-256 state update given in [11], which is illustrated in Fig. 1.

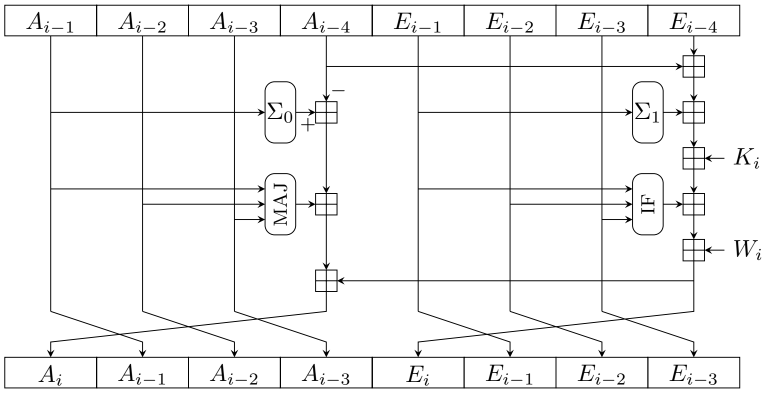


Fig. 1. The state update transformation of SHA-256

The state update transformation starts from the previous 256-bit chaining value $h_j = (A_{-4}, \dots, A_{-1}, E_{-4}, \dots, E_{-1})$ which is updated by applying the step functions 64 times. In each step $i = 0, \dots, 63$ the expanded 32-bit word W_i is used to compute the two state variables E_i and A_i as follows:

$$E_i = A_{i-4} + E_{i-4} + \Sigma_1(E_{i-1}) + \mathrm{IF}(E_{i-1}, E_{i-2}, E_{i-3}) + K_i + W_i$$
$$A_i = E_i - A_{i-4} + \Sigma_0(A_{i-1}) + \mathrm{MAJ}(A_{i-1}, A_{i-2}, A_{i-3}).$$

For the definition of the step constants K_i we refer to [13]. The bitwise Boolean functions IF and MAJ used in each step are defined by

$$\mathrm{IF}(x, y, z) = xy \oplus xz \oplus z$$
$$\mathrm{MAJ}(x, y, z) = xy \oplus yz \oplus xz,$$

and the linear functions Σ_0 and Σ_1 are defined as follows:

$$\Sigma_0(x) = (x \ggg 2) \oplus (x \ggg 13) \oplus (x \ggg 22)$$
$$\Sigma_1(x) = (x \ggg 6) \oplus (x \ggg 11) \oplus (x \ggg 25).$$

After the last step of the state update transformation, the previous chaining value is added to the output of the state update (Davies-Meyer construction). The result is the chaining value h_{j+1} for the next message block m_{j+1}.

3 Basic Attack Strategy

In this section, we give a high-level overview of our collision attacks on SHA-256. We first construct semi-free-start collisions for SHA-256 based on a local collision and then turn these collisions on the compression function into collisions on the hash function. This is possible if enough message words can be chosen freely at the input of the compression function. The main difficulty lies in the construction of semi-free-start collisions which offer enough free message words to turn them into collisions.

3.1 Constructing Local Collisions

If the local collision starts at step n, we get n free message words at the beginning of a differential characteristic. In this case message words $W_0, \dots, W_{n-1}$ do not contain any differences. Hence, they can be chosen (almost) freely to match the initial value or the chaining value of the first block [4]. Note that in some cases, a few bits of these message words may be needed to fulfill conditions in the message expansion.

For our collision attack on 28 steps, we need a local collision with $t = 11$ steps which starts in step $n = 8$ and ends in step 18 (see Fig. 2). For our attack on 31 steps, we use a local collision with $t = 14$ steps which starts in step $n = 5$ and ends in step 18 (see Fig. 2). Although, these local collision spans over fewer words in E_i ($t - 4$ words) and A_i ($t - 8$ words), a lot of freedom is still needed to fulfill all the conditions imposed by the differential characteristic.

Therefore, we use local collisions which are sparse, especially in steps greater than 16. Since the message difference has the largest influence, we aim for a small number of message words which contain differences (see Sect. 4). Additionally, we try to keep the number of differences in the later words of A_i and E_i low. This significantly reduces the search space and improves the running time of our automatic search algorithm (see Sect. 5).

3.2 Turning Semi-Free-Start Collisions into Collisions

In SHA-256 a semi-free-start collision can easily be turned into a collision, if the first 8 message words can be chosen freely. In this case, we can choose the 8 message words to match the 8 words of the initial value [4]. We use this approach

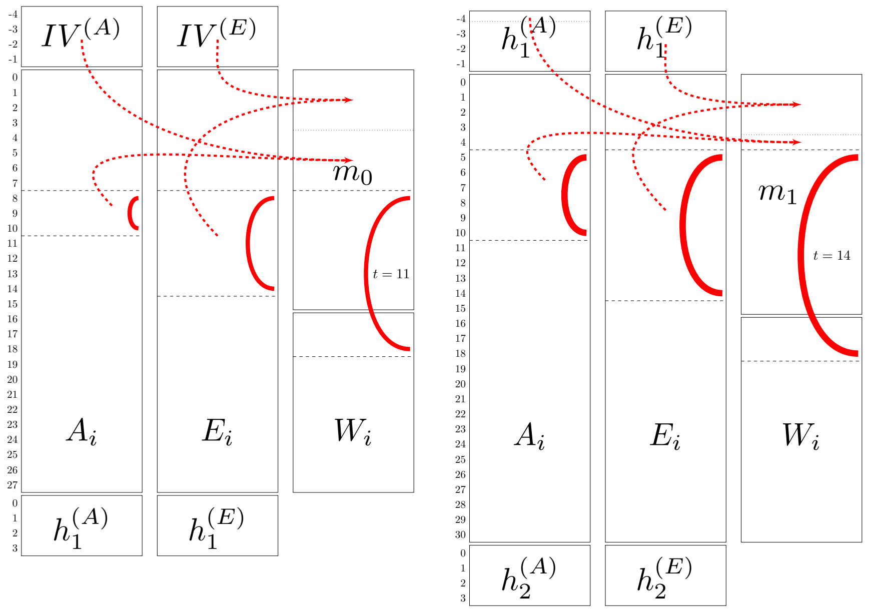


Fig. 2. Two approaches to construct collisions for SHA-256. The left approach uses 8 free message words to turn a semi-free-start collision on 28 steps into a collision on the hash function. The right approach uses random first blocks to increase the freedom and thus, the size of the local collision.

in our collision attack on 28 steps of SHA-256 (see Fig. 2). However, by attacking a higher number of steps a lot of freedom is needed for message modification, in particular after step 15. Therefore, we aim for local collisions resulting in only a few conditions after step 15, which limits the size of the local collision.

However, to get an attack on a higher number of steps, we need to increase the size of the local collision. This is possible by using a 2-block approach. In this case, we need less free message words at the beginning of the compression function. More specifically, by using an unbalanced meet-in-the-middle approach and computing 2^k random first blocks, we only need $256 - k$ free message bits to match the previous chaining value. If the complexity of finding a confirming message pair for the semi-free-start collision is 2^x, the total complexity of this approach will be $\max(2^k, 2^x \cdot 2^{128-k})$. We use this approach in our attack on 31 steps of SHA-256 (see Fig. 2). We construct the local collision such that the first 5 message words can be chosen freely.

3.3 Searching for Differential Characteristics and Message Pairs

A differential attack usually consists of two main steps: constructing a differential characteristic and finding a confirming message pair. Unfortunately, both steps are difficult and depend highly on the hash function under attack. For our collision attacks on SHA-2 we proceed according to the following high-level overview:

- **Find a differential characteristic**
  1. Choose (sparse) message difference
  2. Determine sparse part of the differential characteristic
  3. Determine dense part of the differential characteristic
- **Find a confirming message pair**
  4. Use message modification in dense part of the characteristic
  5. Perform random trials in sparse part of the characteristic
  6. Use free message words to match the initial value (optional)

To search for differential characteristics and confirming message pairs, we use the same approach and automatic search tool as in [11]. However, the selection of starting points for the search (message words which contain differences) and the search strategy have been improved. We try to minimize both, the number of message words which contain differences and the conditions on the expanded message words (see Sect. 4). This allows us to construct sparser differential characteristics and increases the freedom in the message words.

We start the search by first constructing a differential characteristic for the message expansion. Next, we guess on the last few state words of the local collision to keep the differential characteristic sparse towards the end. This improved guessing strategy is essential for our attack to work. Finally, we continue the search to find a confirming message pair.

4 Determining Message Words with Differences

In this section, we show how to construct local collisions, which result in a (semi-free-start) collision attack on a high number of rounds. The previously best results on SHA-256 have been published by Mendel et al. in [11]. Using a local collision on $t = 11$ steps with differences in 5 message words they have shown a collision for 27 steps of the SHA-256 hash function with practical complexity. Additionally, a local collision on $t = 16$ steps with differences in 8 expanded message words has been used to construct semi-free-start collisions on 32 steps of SHA-256. To improve on these attacks, we need to increase the length t of the local collision, while still keeping the part without differences large.

4.1 Constructing Local Collisions in SHA-256

Compared to previous attacks [4,11,15,16], we are able to find differential characteristics over a much larger number of steps in the state update of SHA-256 (see Sect. 5). This allows us to increase the size t of the local collision. However, for large values of t the complexity of a local collision increases significantly. The larger the value of t, the more freedom is needed to find a confirming message pair. Since we additionally need free message words to turn a semi-free-start collision into a collision, we need to reduce the complexity of the local collision.

Table 1. Message word differences and message word conditions for the attacks on 28 steps (left), 31 steps (middle) and 38 steps (right) steps of SHA-256. Rows show the individual steps of the message expansion to compute W_i. Columns (and highlighted rows) show those expanded message words which contain a difference. An occurrence of a message word in the message expansion equation is denoted by 'x'. For all rows which are not highlighted but contain an 'x', the message differences must cancel.

W_i	8	9	13	16	18
0					
1					
2					
3					
4					
5					
6					
7					
8	x				
9		x			
10					
11					
12					
13			x		
14					
15					
16		x		x	
17					
18				x	x
19					
20			x		x
21					
22					
23	x			x	
24	x	x			
25		x			x
26					
27					

W_i	5	6	7	8	9	16	18
0							
1							
2							
3							
4							
5	x						
6		x					
7			x				
8				x			
9					x		
10							
11							
12							
13							
14							
15							
16					x	x	
17							
18						x	x
19							
20	x						x
21	x	x					
22		x	x				
23			x	x		x	
24				x	x		
25					x		x
26							
27							
28							
29							
30							

W_i	7	8	10	15	23	24
0						
1						
2						
3						
4						
5						
6						
7	x					
8		x				
9						
10			x			
11						
12						
13						
14						
15				x		
16						
17			x	x		
18						
19						
20						
21						
22	x			x		
23	x	x			x	
24		x				x
25			x		x	
26			x			x
27						
28						
29						
30				x	x	
31				x		x
32						
33						
34						
35						
36						
37						

This is possible be reducing the number of message words which contain differences. The freedom of every message word which does not contain a difference can be used more easily for message modification. Additionally, we require that the expanded message words cancel each other, such that we get a high number of steps which do not contain any differences at all. This cancellation of message word differences results in conditions on the message expansion, and fewer message words with differences result in fewer conditions.

To find good candidates for local collisions and message word differences, we first need to determine the initial constraints. To turn a semi-free-start collision into a collision attack, we need to have no differences in the first (ideally 8) message words. Furthermore, we consider only local collisions with $t > 9$ which result in (semi-free-start) collision attacks on more than 27 steps. We prefer local collisions with small values of t and which result in fewer conditions on the expanded message words. In particular, we try to avoid many conditions (and thus, message words with differences) after step 16 since they are more difficult to fulfill.

4.2 Candidates of Good Local Collisions

For each candidate of t, we identify those message words which must have differences such that the differential characteristic holds for a large number of steps. Then, we minimize the number of conditions on the expanded message and the number of message words with differences. Using this strategy, we get many possible candidates for good local collisions. For the three candidates shown in Table 1 we are able to construct differential characteristics (see Sect. 5) and present collision and semi-free-start collision attacks (see Sect. 6).

The first local collision in Table 1 spans over $t = 11$ steps (step 8-18) and results in a collision on 28 steps of SHA-256. The local collision has differences in only 5 message words ($W_8, W_9, W_{13}, W_{16}, W_{18}$) and we get 4 conditions in step 20, 23, 24, and 25 of the message expansion. The first 8 message words do not contain any differences and therefore, we can turn a semi-free-start collision into a real collision for SHA-256.

To extend the collision attacks on SHA-256 to more steps, we drop the condition that the first 8 message words should have no differences. If only the 5 first message words contain no differences, we can still get an attack below the birthday bound using a 2-block approach (see Sect. 3). We get a local collision spanning over $t = 14$ steps (step 5-18) with differences in only 7 expanded message words ($W_5, \ldots, W_9, W_{16}, W_{18}$) which results in a collision attack on 31 steps of SHA-256. We get 6 conditions in step 20–25 of the message expansion (see Table 1).

Finally, if we only search for local collisions which result in a semi-free-start collision, we can extend the number of steps to attack even further. Using a local collision spanning over $t = 18$ steps with differences in only 6 expanded message words ($W_7, W_8, W_{10}, W_{15}, W_{23}, W_{24}$) we can get an attack on up to 38 steps of SHA-256. We get conditions in 6 steps of the message expansion (see Table 1).

5 Finding Differential Characteristics

After the message words which contain differences are fixed, we need to find a differential characteristic. Due to the increased complexity of SHA-2 compared to other members of the MD4 family, finding good (high probability) differential characteristics by hand is almost impossible. We use the techniques developed by Mendel et al. which have been applied in several attacks on ARX-based hash functions [9–12]. Using an automated search tool, complex nonlinear differential characteristics can be found. Additionally, the tool can be used for solving nonlinear equations involving conditions on state and message words (i.e. for message modification).

5.1 Automated Search for Differential Characteristics

The basic idea of the search algorithm is to pick and guess previously unrestricted bits. After each guess, the information due to these restrictions is propagated to other bits. If an inconsistency occurs, the algorithm backtracks to an earlier state of the search and tries to correct it. Similar to [11], we denote these three parts of the search by decision (guessing), deduction (propagation), and backtracking (correction). Then, the search algorithm proceeds as follows:

```
Let $U$ be a set of bits. Repeat the following until $U$ is empty:
Decision (Guessing)
    1. Pick randomly (or according to some heuristic) a bit in $U$.
    2. Impose new constraints on this bit.
Deduction (Propagation)
    3. Propagate the new information to other variables and equations as described in [11].
    4. If an inconsistency is detected start backtracking, else continue with step 1.
Backtracking (Correction)
    5. Try a different choice for the decision bit.
    6. If all choices result in an inconsistency, mark the bit as critical.
    7. Jump back until the critical bit can be resolved.
    8. Continue with step 1.
```

In the deduction, we use generalized conditions on bits [3] to propagate information. A generalized condition takes all 16 possible conditions on a pair of bits into account. Table 2 lists these conditions and introduces a notation for them. We also use generalized conditions to represent the differential characteristics in the remainder of this paper. During the search, we propagate information and backtrack as proposed in [11]. Similar to [11], we additionally consider linear conditions on two related bits ($X_j \oplus X_k = \{0, 1\}$) during the search for a confirming message pair.

The main difficulty in finding a long differential characteristic lies in the fine-tuning of the search algorithm. There are a lot of variations possible which can

Table 2. Notation for all generalized conditions on a pair of bits [3]

$(X_i, X_i{}^*)$	(0,0)	(1,0)	(0,1)	(1,1)	(X_i, X_i^*)	(0,0)	(1,0)	(0,1)	(1,1)
?	✓	✓	✓	✓	3	✓	✓	-	-
-	✓	-	-	✓	5	✓	-	✓	-
x	-	✓	✓	-	7	✓	✓	✓	-
0	✓	-	-	-	A	-	✓	-	✓
u	-	✓	-	-	B	✓	✓	-	✓
n	-	-	✓	-	C	-	-	✓	✓
1	-	-	-	✓	D	✓	-	✓	✓
#	-	-	-	-	E	-	✓	✓	✓

decide whether the search succeeds or fails. The main improvement compared to [11] results from an improved decision (guessing) part of the search algorithm. Instead of randomly choosing bits from the whole set of unrestricted bits, we split the set in specific sub-sets. These sub-sets are chosen such that the resulting differential characteristic gets sparser and the search terminates faster. We cover these improvements which led to the new results on SHA-256 in the following section.

5.2 Improved Decision Strategy for SHA-256

Note that already the starting point has a large influence on the efficiency of finding a differential characteristic. In the case of our attacks on SHA-256, we have chosen a starting point where the local collision is not so long and only a few message words contain differences. This significantly reduces the search space for the automatic search tool. To further improve the efficiency of the tool, we reduce the search space further by separating the search into 3 stages, compared to two stages in [11].

In each stage, we define a different set of unrestricted bits in U_i. This way, we can control the order in which we guess bits. For the local collisions given in the previous section, the best strategy is to first search for a differential characteristic in the message expansion. Then, we continue by searching for a sparse characteristic in the state update and finally, we search for a confirming message pair. The stages are executed sequentially but we dynamically switch between them if a contradiction occurs. If necessary, we try to correct contradictions by backtracking into the previous stages and continue the search from there. Additionally, we restart the search from scratch after a certain amount of inconsistencies occurred. This terminates search branches for which it is unlikely that a solution can be found. The three stages can be summarized as follows:

Stage 1: We first search for a consistent differential characteristic in the message expansion. Hence, we only add unconstrained bits '?' or 'x' of W_i to the set U_1. Furthermore, we try to reduce the number of conditions after step 15 in the message expansion. In this case, it is more likely to find confirming message pairs in the last stage of the search. To get a sparser characteristic in this area, we pick decision bits more often from the last few steps of the message expansion.

Stage 2: Once we have found a differential characteristic for the message expansion, we continue with searching for a differential characteristic in the state update. We add all unconstrained bits '`?`' or '`x`' of A and E to the set U_2. Note that we pick decision bits more often from A, since this results in sparser characteristics for A. Similar to Stage 1, experiments have shown that in this case, confirming message pairs are easier to find in the last stage.

Stage 3: In the last stage, we search for confirming inputs. We only pick decision bits '`-`' which are constraint by linear two-bit conditions, similar as in [11]. This ensures that those bits which influence a lot of other bits are guessed first. Additionally, at least all bits influenced by two-bit conditions propagate as well. This way, inconsistent characteristics can be detected earlier and valid solutions are found faster.

Note that after Stage 3 finishes, we already get a confirming message pair which results in a semi-free-start collision. The corresponding differential characteristics for 28, 31 and 38 steps of SHA-256, including bits marked with linear conditions on two bits are given in Table 3, 4 and 5. Note that for SHA-2, the characteristics are in general too complex to list all conditions (including non-linear conditions on two or more bits). Therefore, all our characteristics are verified by providing conforming message pairs in the appendix.

6 Finding Confirming Message Pairs

In this section, we present our results and show how to turn (some of) the semi-free-start collision into collisions on the hash function. To confirm our claims, we present confirming message pairs for those steps of our attacks which have practical complexity.

6.1 Collision for 28 Steps of SHA-256

Using the starting point given in Table 1 and the search strategy described in Sect. 5, we can find a semi-free-start collision for 28 steps of SHA-256. Finding the differential characteristic took about 5 hours on a single cpu, which is equivalent to about $2^{36.3}$ SHA-256 evaluations using the current version of OpenSSL as a reference point.

Since we can (almost) freely choose the first 8 message words, this semi-free-start collision can be turned into a collision on the hash function with almost no cost. The confirming message pair is given in Table 6 and the according differential characteristic is given in Table 3.

6.2 Collision for 31 Steps of SHA-256

Using the starting point given in Table 1, we can find a semi-free-start collision on 31 steps of SHA-256, where the first 5 message words can be chosen freely.

Table 3. Differential characteristic for the collision attack on SHA-256 reduced to 28 steps. Bits with gray background have at least one additional condition.

i	∇A_i	∇E_i	∇W_i
-4	--------------------------------	--------------------------------	
-3	--------------------------------	--------------------------------	
-2	--------------------------------	--------------------------------	
-1	--------------------------------	--------------------------------	
0	--------------------------------	--------------------------------	--------------------------------
1	--------------------------------	--------------------------------	--------------------------------
2	--------------------------------	--------------------------------	--------------------------------
3	--------------------------------	--------------------------------	--------------------------------
4	--------------------------------	--------------------------------	--------------------------------
5	--------------------------------	--------------------------------	--------------------------------
6	------------------------1-------	-------------0----------00------	--------------------------------
7	------------------------1-------	----------1--1-10-------11--1---	--------------------------------
8	-0uu-nuu-u-uu-nnn-uu---nu-n-----	-0uu-1nu-uun-u-0uu-n0---nnu11---	nnnu----n-nn-u-u-n-n----n-n-----
9	-0-------n--u----------n1-------	-1-1011-u1u0--0nn-0-u-n0010uu-0-	11--un1-un0nn--110-11u01uunnu---
10	nnnnnn----u-u--------u--u--uu---	0n010n1011101011n011u1110n0nu01u	--------------------------------
11	-1----------------------1-------	-n-1un1-n111000n10n0110n10001-u0	--------------------------------
12	-1----------------------1-------	0011n111n00u0n11u0uu10110uu10-00	--------------------------------
13	--------------------------------	000100010n011nuuuuuuuu1n11011101	------n--n----------n--1n--n----
14	--------------------------------	11-00u--0un0u000-00-u0nn-nnnu-0-	---1----------------------------
15	--------------------------------	-----1---10-11001011-00--0001---	--------------------------------
16	--------------------------------	-----1---01-1-------0-00-1111---	1----u---0uun-----10un01uun-n---
17	--------------------------------	--------------------------------	--------------------------------
18	--------------------------------	--------------------------------	-----n----n-n------1-n01n-n-u---
19	--------------------------------	--------------------------------	--------------------------------
20	--------------------------------	--------------------------------	--------------------------------
21	--------------------------------	--------------------------------	--------------------------------
22	--------------------------------	--------------------------------	--------------------------------
23	--------------------------------	--------------------------------	--------------------------------
24	--------------------------------	--------------------------------	--------------------------------
25	--------------------------------	--------------------------------	--------------------------------
26	--------------------------------	--------------------------------	--------------------------------
27	--------------------------------	--------------------------------	--------------------------------

Finding the differential characteristic and confirming message pair for the semi-free-start collision took about 21 hours on a single CPU, which is equivalent to about $2^{38.4}$ SHA-256 evaluations. The resulting differential characteristic is given in Table 4. To demonstrate that we can indeed choose the 5 first message words, we have set the last 5 chaining words to 0 (see Table 7).

In the characteristic on 31 steps, we have no differences in the first 5 message words. Using a single semi-free-start collision and a two-block approach, we can construct a collision for the SHA-256 hash function reduced to 31 steps with a complexity of about $2^{99.5}$ compression function evaluations. We start this part of the attack with the differential characteristic given in Table 4 and continue as follows:

1. Use the automatic tool to determine all expanded message words and state variables in steps 5–12. This also determines the state words E_1–E_4 and A_{-3}–A_4. Note that this step of the attack takes only seconds and does not contribute to the final attack complexity.
2. Compute 2^{96} arbitrary first message blocks to fulfill the 96 conditions on the chaining input A_{-3}–A_{-1}. This step of the attack has a complexity of about 2^{96} SHA-256 evaluations and also determines the expanded message words W_0–W_4.

Table 4. Differential characteristic for the collision attack on SHA-256 reduced to 31 steps. Bits with gray background have at least one additional condition.

i	∇A_i	∇E_i	∇W_i
-4	--------------------------------	--------------------------------	
-3	--------------------------------	--------------------------------	
-2	--------------------------------	--------------------------------	
-1	--------------------------------	--------------------------------	
0	--------------------------------	--------------------------------	--------------------------------
1	--------------------------------	--------------------------------	--------------------------------
2	--------------------------------	--------------------------------	--------------------------------
3	------------------------------0-	-0----------------0-------------	--------------------------------
4	------------------------------00	-1---------1-----01---1-0--0--10	--------------------------------
5	-nnn-n-n-11----n--nu-1-------0n-	0nnnn1uu-0-1101n-1nu--0-11-1-0n1	u---uunu-------n---n----------n-
6	unnnn-------------------------0-	n-n10111n--u11u00n10u1n-nn1n-1uu	nn1-n---nu-nn--1u--0-un0--n0-nn-
7	-------------------n--------n-0u	101u0nn10-11011u-n111n110un1-nnn	00nn0n101-n1nnn1u0nn-n011u-1n0--
8	--------------------------------	1-uu11110--0u10110n-10101010-0n0	0001u0001-000nuuun1n01nn-01nuuuu
9	--------------------------------	101100uu111111nu111001--011110nn	-----1---------un---0-----11un--
10	----------------u------------u--	1-00u1101001101un00--0001--u1n00	---0--------------------------1-
11	--------------------------------	010100u0nu1uuuuuu1001000000n1u10	--------------------------------
12	--------------------------------	111nuuuuuuuuuuuuu001111101100n00	--------------------------------
13	--------------------------------	---101-11-1-----1----------0-0--	--------------------------------
14	--------------------------------	---100---0011111u-------1----u--	--------------------------------
15	--------------------------------	----------------0------------0--	--------------------------------
16	--------------------------------	----------------1------------1--	-------------unnnunnnnnnnnnnnn--
17	--------------------------------	--------------------------------	--------------------------------
18	--------------------------------	--------------------------------	----------------n------------n--
19	--------------------------------	--------------------------------	--------------------------------
20	--------------------------------	--------------------------------	--------------------------------
21	--------------------------------	--------------------------------	--------------------------------
22	--------------------------------	--------------------------------	--------------------------------
23	--------------------------------	--------------------------------	--------------------------------
24	--------------------------------	--------------------------------	--------------------------------
25	--------------------------------	--------------------------------	--------------------------------
26	--------------------------------	--------------------------------	--------------------------------
27	--------------------------------	--------------------------------	--------------------------------
28	--------------------------------	--------------------------------	--------------------------------
29	--------------------------------	--------------------------------	--------------------------------
30	--------------------------------	--------------------------------	--------------------------------

3. At this point, the chaining value and the message words W_0–W_{12} are chosen. Next, we use the freedom in the message words W_{13}–W_{15} to fulfill the conditions on E_{13}–E_{15} and W_{16}, W_{18}. However, this step of the attack succeeds only with a probability of about 1/12 (verified experimentally) since we have just not enough freedom in W_{13}–W_{15}. If this step fails then we go back to step 2.

To summarize, we can find a 2-block collision for SHA-256 reduced to 31 steps with a complexity of about $12 \cdot 2^{96} \approx 2^{99.5}$ SHA-256 evaluations. Note that the complexity of the attack can be improved significantly by using a meet-in-the-middle approach.

Instead of computing only one solution in the first step of the attack, we compute ℓ solutions and save them in a list L. In step 2 of the attack where we compute an arbitrary first message blocks, we check for a match in the list L. By increasing the size ℓ of the list, we can reduce the number of first message blocks that we need to compute by a factor of $1/\ell$.

The main question is how many entries in L can be computed in our attack. Using our unoptimized code we have already found $\ell > 2^{19.5}$ different solutions for step 1 of the attack with an average cost of $2^{25.5}$. This reduces the attack

Table 5. Differential characteristic for the semi-free-start collision attack on SHA-256 reduced to 38 steps. Bits with gray background have at least one additional condition.

i	∇A_i	∇E_i	∇W_i
-4	--------------------------------	--------------------------------	
-3	--------------------------------	--------------------------------	
-2	--------------------------------	--------------------------------	
-1	--------------------------------	--------------------------------	
0	--------------------------------	--------------------------------	--------------------------------
1	--------------------------------	--------------------------------	--------------------------------
2	--------------------------------	--------------------------------	--------------------------------
3	--------------------------------	--------------------------------	--------------------------------
4	--------------------------------	--------------------------------	--------------------------------
5	--------------------------------	--------------------1----1------	--------------------------------
6	--------------------------------	1--00--1-------1-10111---0-1--0-	--------------------------------
7	-n----u-n-u---u-------n-nn------	nu-11-0uuun101-uuuuuuu1u-u-1--01	--nnnnn--nn--un----nuuu-nn------
8	----nn-n--n--un-uu-u-u-u-n------	n01nn-1n10000-1u1uuunu00n0-nn0n1	0000011011101--1100nuuuuuuuuuu00
9	u---u-n--nuuu---n-uuu--u-------n	000n00u10001n101nu0u000111-u11un	--------------------------------
10	--------------------------------	00unn00110n1001u11u-101u111u0uun	--------unnnnnnnu---------------
11	--------------------------------	1u1-11u11-n01n100n10u1-11u100011	------1-------------------------
12	--------------------------------	0uu1u1u0u1uu1n01nn111u011n-01010	----------1-----01--------------
13	--------------------------------	n0010uu01-00n1-01n0nu10u10-1-nuu	----------------0--------------1
14	--------------------------------	101-110-1-0010-10--111-010---100	--------------------------------
15	----u----------n----------------	0-1-u01----00--n-00----10----111	-----------------------------u--
16	-----------------------------u--	----n-n1---01-un11------nuu-nu01	--------------------------------
17	--------------------------------	-0--n-1---0nnnnn-nuu1---011-1-un	--------------------------------
18	--------------------------------	----0-1---00000--000--1011101100	--------------------------------
19	--------------------------------	0---u-00nuuuuuuu0001--0011011011	--------------------------------
20	--------------------------------	1--1--11100111---1--0unnnnnnnn0-	--------------------------------
21	--------------------------------	----1---11111111-----000000000--	--------------------------------
22	--------------------------------	---------------------111111111--	--------------------------------
23	--------------------------------	--------------------------------	----n---------un----------------
24	--------------------------------	--------------------------------	-----------------------------n--
25	--------------------------------	--------------------------------	--------------------------------
26	--------------------------------	--------------------------------	--------------------------------
27	--------------------------------	--------------------------------	--------------------------------
28	--------------------------------	--------------------------------	--------------------------------
29	--------------------------------	--------------------------------	--------------------------------
30	--------------------------------	--------------------------------	--------------------------------
31	--------------------------------	--------------------------------	--------------------------------
32	--------------------------------	--------------------------------	--------------------------------
33	--------------------------------	--------------------------------	--------------------------------
34	--------------------------------	--------------------------------	--------------------------------
35	--------------------------------	--------------------------------	--------------------------------
36	--------------------------------	--------------------------------	--------------------------------
37	--------------------------------	--------------------------------	--------------------------------

complexity to about 2^{80} SHA-256 evaluations. However, our experiments indicate that we can expect to find about 2^{34} different solutions, which results in a total attack complexity of about $2^{65.5}$ SHA-256 evaluations.

6.3 Semi-Free-Start Collision for 38 Steps of SHA-256

Finally, we have also improved the best semi-free-start collision attack on SHA-256. Using the starting point given in Table 1, we can find a semi-free-start collision for 38 steps of SHA-256. Finding the differential characteristic and the confirming message pair took about 8 hours on a single CPU. This is equivalent to about 2^{37} SHA-256 evaluations. The differential characteristic is shown in Table 5 and the resulting semi-free-start collision in Table 8.

7 Conclusions

In this paper, we have improved the best known collision attacks on the SHA-256 hash function from 27 to 31 steps. We focus on the construction of semi-free-start collisions which can be turned into collisions on the hash function. Our results are hash function collisions on 31 steps of SHA-256 with complexity $2^{65.5}$, and semi-free-start collisions on 38 steps with complexity 2^{37}. We have verified all our attacks by providing practical examples whenever this was possible.

Our results were obtained by extending the size of the local collision up to 18 steps. Furthermore, we ensure that the first message words do not contain any differences and try to reduce the number of conditions on the expanded message words. To find differential characteristics and confirming message pairs for local collisions spanning over more steps we have improved the efficiency of the automatic search tool used by Mendel et al. in their attacks on reduced SHA-256 in several ways. Most importantly, we have improved the search strategy to find sparser differential characteristics by guessing primarily bits towards the end of the local collision.

Acknowledgments. Part of this work was done while Florian Mendel was with KU Leuven. The work has been supported in part by the Secure Information Tech- nology Center-Austria (A-SIT) and by the Austrian Science Fund (FWF), project P21936-N23.

References

1. Aoki, K., Guo, J., Matusiewicz, K., Sasaki, Y., Wang, L.: Preimages for Step-Reduced SHA-2. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 578–597. Springer, Heidelberg (2009)
2. Biryukov, A., Lamberger, M., Mendel, F., Nikolić, I.: Second-Order Differential Collisions for Reduced SHA-256. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 270–287. Springer, Heidelberg (2011)
3. De Cannière, C., Rechberger, C.: Finding SHA-1 Characteristics: General Results and Applications. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 1–20. Springer, Heidelberg (2006)
4. Indesteege, S., Mendel, F., Preneel, B., Rechberger, C.: Collisions and Other Non-random Properties for Step-Reduced SHA-256. In: Avanzi, R.M., Keliher, L., Sica, F. (eds.) SAC 2008. LNCS, vol. 5381, pp. 276–293. Springer, Heidelberg (2009)
5. Isobe, T., Shibutani, K.: Preimage Attacks on Reduced Tiger and SHA-2. In: Dunkelman, O. (ed.) FSE 2009. LNCS, vol. 5665, pp. 139–155. Springer, Heidelberg (2009)
6. Khovratovich, D., Rechberger, C., Savelieva, A.: Bicliques for Preimages: Attacks on Skein-512 and the SHA-2 Family. In: Canteaut, A. (ed.) FSE 2012. LNCS, vol. 7549, pp. 244–263. Springer, Heidelberg (2012)
7. Lamberger, M., Mendel, F.: Higher-Order Differential Attack on Reduced SHA-256. Cryptology ePrint Archive, Report 2011/037 (2011), `http://eprint.iacr.org/`
8. Li, J., Isobe, T., Shibutani, K.: Converting Meet-In-The-Middle Preimage Attack into Pseudo Collision Attack: Application to SHA-2. In: Canteaut, A. (ed.) FSE 2012. LNCS, vol. 7549, pp. 264–286. Springer, Heidelberg (2012)

9. Mendel, F., Nad, T., Scherz, S., Schläffer, M.: Differential Attacks on Reduced RIPEMD-160. In: Gollmann, D., Freiling, F.C. (eds.) ISC 2012. LNCS, vol. 7483, pp. 23–38. Springer, Heidelberg (2012)
10. Mendel, F., Nad, T., Schläffer, M.: Cryptanalysis of Round-Reduced HAS-160. In: Kim, H. (ed.) ICISC 2011. LNCS, vol. 7259, pp. 33–47. Springer, Heidelberg (2012)
11. Mendel, F., Nad, T., Schläffer, M.: Finding SHA-2 Characteristics: Searching through a Minefield of Contradictions. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 288–307. Springer, Heidelberg (2011)
12. Mendel, F., Nad, T., Schläffer, M.: Collision Attacks on the Reduced Dual-Stream Hash Function RIPEMD-128. In: Canteaut, A. (ed.) FSE 2012. LNCS, vol. 7549, pp. 226–243. Springer, Heidelberg (2012)
13. National Institute of Standards and Technology: FIPS PUB 180-3: Secure Hash Standard. Federal Information Processing Standards Publication 180-3, U.S. Department of Commerce (October 2008), http://www.itl.nist.gov/fipspubs
14. National Institute of Standards and Technology: SHA-3 Selection Announcement (October 2012), http://csrc.nist.gov/groups/ST/hash/sha-3/sha-3_selection_announcement.pdf
15. Nikolić, I., Biryukov, A.: Collisions for Step-Reduced SHA-256. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 1–15. Springer, Heidelberg (2008)
16. Sanadhya, S.K., Sarkar, P.: New Collision Attacks against Up to 24-Step SHA-2. In: Chowdhury, D.R., Rijmen, V., Das, A. (eds.) INDOCRYPT 2008. LNCS, vol. 5365, pp. 91–103. Springer, Heidelberg (2008)
17. Wang, X., Yin, Y.L., Yu, H.: Finding Collisions in the Full SHA-1. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 17–36. Springer, Heidelberg (2005)
18. Wang, X., Yu, H.: How to Break MD5 and Other Hash Functions. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 19–35. Springer, Heidelberg (2005)

A Resulting Message Pair Examples

Table 6. Example of a collision for 28 steps of SHA-256

h_0	6a09e667	bb67ae85	3c6ef372	a54ff53a	510e527f	9b05688c	1f83d9ab	5be0cd19
m	14c48440	b3c3277f	ad69812d	c3d4dffa	7eae690b	7f9fe027	832aece8	9a489458
	1607a45c	db81bdc8	8786e031	d8f22801	72b6be5e	45a2652f	f3fbb17a	2ce70f52
m^*	14c48440	b3c3277f	ad69812d	c3d4dffa	7eae690b	7f9fe027	832aece8	9a489458
	e6b2f4fc	d759b930	8786e031	d8f22801	72b6be5e	47e26dbf	f3fbb17a	2ce70f52
Δm	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
	f0b550a0	0cd804f8	00000000	00000000	00000000	02400890	00000000	00000000
h_1	01470131	cd0062bc	7e8f8c21	98938652	3d49075a	327f38e8	11f0d36d	58601725

Table 7. Example of a semi-free-start collision for 31 steps of SHA-256 with the last 5 chaining variables set to 0

h_0	532f13f5	6a28c3c0	e301fab5	00000000	00000000	00000000	00000000	00000000
m	d55c884f	faf18f34	b772b323	af46235b	3d8bd87b	dd3e8271	26618488	02d189d0
	1883a4af	4f99167b	271b11c7	81b8363d	b27e389d	2155a533	8b811348	4a8da291
m^*	d55c884f	faf18f34	b772b323	af46235b	3d8bd87b	523f9273	eeb902ae	36ff3d98
	108477b0	4f989677	271b11c7	81b8363d	b27e389d	2155a533	8b811348	4a8da291
Δm	00000000	00000000	00000000	00000000	00000000	8f011002	c8d88626	342eb448
	0807d31f	0001800c	00000000	00000000	00000000	00000000	00000000	00000000
h_1	6ff5a9a7	9d014158	12938ebb	dbf8dc76	29fb2c4c	b48b053e	1c4377a9	e21554c1

Table 8. Example of a semi-free-start collision for 38 steps of SHA-256

h_0	ba75b4ac	c3c9fd45	fce04f3a	6d620fdb	42559d01	b0a0cd10	729ca9bc	b284a572
m	4f5267f8	8f8ec13b	22371c61	56836f2b	459501d1	8078899e	98947e61	4015ef31
	06e98ffc	4babda4a	27809447	3bf9f3be	7b3b74e1	065f711d	6c6ead5e	a1781d54
m^*	4f5267f8	8f8ec13b	22371c61	56836f2b	459501d1	8078899e	98947e61	7e73f1f1
	06e99000	4babda4a	277f1447	3bf9f3be	7b3b74e1	065f711d	6c6ead5e	a1781d50
Δm	00000000	00000000	00000000	00000000	00000000	00000000	00000000	3e661ec0
	00001ffc	00000000	00ff8000	00000000	00000000	00000000	00000000	00000004
h_1	baa8df17	9f9f64dd	d57d5c2c	7b232c81	1f3916e6	7a03a2be	7afb1d86	6b0eced6

Dynamic Proofs of Retrievability via Oblivious RAM[⋆]

David Cash[1,⋆⋆], Alptekin Küpçü[2,⋆⋆⋆], and Daniel Wichs[3,†]

[1] Rutgers University
[2] Koç University
[3] Northeastern University

Abstract. Proofs of retrievability allow a client to store her data on a remote server (e.g., "in the cloud") and periodically execute an efficient *audit* protocol to check that all of the data is being maintained correctly and can be recovered from the server. For efficiency, the *computation* and *communication* of the server and client during an audit protocol should be significantly smaller than reading/transmitting the data in its entirety. Although the server is only asked to access a few locations of its storage during an audit, it must *maintain full knowledge of all client data* to be able to pass.

Starting with the work of Juels and Kaliski (CCS '07), all prior solutions require that the client data is *static* and do not allow it to be efficiently updated. Indeed, they store a redundant encoding of the data on the server, so that the server must delete a large fraction of its storage to 'lose' any actual content. Unfortunately, this means that even a single bit modification to the original data will need to modify a large fraction of the server storage, which makes updates highly inefficient.

In this work, we give the first solution providing proofs of retrievability for *dynamic* storage, where the client can perform arbitrary reads/writes on any location within her data by running an efficient protocol with the server. At any point in time, the client can also execute an audit protocol to ensure that the server maintains the *latest version* of its data. The computation and communication complexity of the server and client in our protocols is only *polylogarithmic* in the size of the data. Our main idea is to split up the data into small blocks and redundantly encode each block of data individually, so that an update inside any data block only affects a few codeword symbols. The main difficulty is to prevent the server from identifying and deleting too many codeword symbols belonging to any single data block. We do so by hiding where the various codeword symbols are stored on the server and when they are being accessed by the client, using the techniques of *oblivious RAM*.

⋆ A full version of this work is available as ePrint report 2012/550.

⋆⋆ Research conducted while at IBM Research, T.J. Watson.

⋆⋆⋆ Supported by TÜBİTAK, the Scientific and Technological Research Council of Turkey, under project number 112E115.

† Research conducted while at IBM Research, T.J. Watson and supported by DARPA under agreement number FA8750-11-C-0096.

1 Introduction

Cloud storage systems (Amazon S3, Dropbox, Google Drive etc.) are becoming increasingly popular as a means of storing data reliably and making it easily accessible from any location. Unfortunately, even though the remote storage provider may not be trusted, current systems provide few security or integrity guarantees. Guaranteeing the *privacy* and *authenticity* of remotely stored data while allowing efficient access and updates is non-trivial, and relates to the study of *oblivious RAMs* and *memory checking*, which we will return to later. The main focus of this work, however, is an orthogonal question: How can we efficiently verify that the entire client data is being stored on the remote server in the first place? In other words, what prevents the server from deleting some portion of the data (say, an infrequently accessed sector) to save on storage?

PROVABLE STORAGE. Motivated by the questions above, there has been much cryptography and security research in creating a provable storage mechanism, where an untrusted server can *prove* to a client that her data is kept intact. More precisely, the client can run an efficient *audit* protocol with the untrusted server, guaranteeing that the server can only pass the audit if it maintains full *knowledge* of the entire client data. This is formalized by requiring that the data can be efficiently *extracted* from the server given its state at the beginning of any successful audit. One may think of this as analogous to the notion of extractors in the definition of *zero-knowledge proofs of knowledge* [15,4].

One trivial audit mechanism, which accomplishes the above, is for the client to simply download all of her data from the server and check its authenticity (e.g., using a MAC). However, for the sake of efficiency, we insist that the *computation* and *communication* of the server and client during an audit protocol is much smaller than the potentially huge size of the client's data. In particular, the server shouldn't even have to *read* all of the client's data to run the audit protocol, let alone *transmit* it. A scheme that accomplishes the above is called a *Proof of Retrievability* (PoR).

PRIOR TECHNIQUES. The first PoR schemes were defined and constructed by Juels and Kaliski [18], and have since received much attention. We review the prior work and and closely related primitives (e.g., *sublinear authenticators* [21] and *provable data possession* [1]) in Section 1.2.

On a very high level, all PoR constructions share essentially the same common structure. The client stores some *redundant encoding* of her data under an erasure code on the server, ensuring that the server must delete a significant fraction of the encoding before losing any actual data. During an audit, the client then checks a few random locations of the encoding, so that a server who deleted a significant fraction will get caught with overwhelming probability.

More precisely, let us model the client's input data as a string $\mathbf{M} \in \Sigma^\ell$ consisting of ℓ symbols from some small alphabet Σ, and let $\mathsf{Enc} : \Sigma^\ell \to \Sigma^{\ell'}$ denote an erasure code that can correct the erasure of up to $\frac{1}{2}$ of its output symbols. The client stores $\mathsf{Enc}(\mathbf{M})$ on the server. During an audit, the client selects a small random subset of t out of the ℓ' locations in the encoding, and challenges the server to respond with the corresponding values, which it then checks for authenticity (e.g., using MAC tags). Intuitively, if the server deletes more than half of the values in the encoding, it will get caught with overwhelming probability $> 1 - 2^{-t}$ during the audit, and otherwise it retains knowledge of the original data because of the redundancy of the encoding. The complexity of the audit protocol is only proportional to t which can be set to the *security parameter* and is independent of the size of the client data.[1]

Difficulty of Updates. One of the main limitations of all prior PoR schemes is that they do not support efficient updates to the client data. Under the above template for PoR, if the client wants to modify even a single location of $\mathbf{M}$, it will end up needing to change the values of at least half of the locations in $\mathsf{Enc}(\mathbf{M})$ on the server, requiring a large amount of work (linear in the size of the client data). Constructing a PoR scheme that allows for efficient updates was stated as the main open problem by Juels and Kaliski [18]. We emphasize that, in the setting of updates, the audit protocol must ensure that the server correctly maintains knowledge of the *latest version* of the client data, which includes all of the changes incurred over time. Before we describe our solution to this problem, let us build some intuition about the challenges involved by examining two natural but *flawed* proposals.

First Proposal. A natural attempt to overcome the inefficiency of updating a huge redundant encoding is to encode the data "locally" so that a change to one position of the data only affects a small number of codeword symbols. More precisely, instead of using an erasure code that takes all ℓ data symbols as input, we can use a code $\mathsf{Enc} : \Sigma^k \to \Sigma^n$ that works on small blocks of only $k \ll \ell$ symbols encoded into n symbols. The client divides the data $\mathbf{M}$ into $L = \ell/k$ *message blocks* $(\mathbf{m}_1, \ldots, \mathbf{m}_L)$, where each block $\mathbf{m}_i \in \Sigma^k$ consists of k symbols. The client redundantly encodes each message block $\mathbf{m}_i$ individually into a corresponding *codeword block* $\mathbf{c}_i = \mathsf{Enc}(\mathbf{m}_i) \in \Sigma^n$ using the above code with small inputs. Finally the client concatenates these codeword blocks to form the value $\mathbf{C} = (\mathbf{c}_1, \ldots, \mathbf{c}_L) \in \Sigma^{Ln}$, which it stores on the server. Auditing works as before: The client randomly chooses t of the $L \cdot n$ locations in $\mathbf{C}$ and challenges the server to respond with the corresponding codeword symbols in these locations, which it then tests for authenticity.[2] The client can now read/write to any location within her data by simply reading/writing to the n relevant codeword symbols on the server.

The above proposal can be made secure when the block-size k (which determines the complexity of reads/updates) and the number of challenged locations t (which determines the complexity of the audit) are both set to $\Omega(\sqrt{\ell})$ where ℓ is the size of the data (see the full version [8] for details). This way, the audit

[1] Some of the more advanced PoR schemes (e.g., [24,11]) optimize the communication complexity of the audit even further by cleverly compressing the t codeword symbols and their authentication tags in the server's response.

[2] This requires that we can efficiently check the *authenticity* of the remotely stored data $\mathbf{C}$, while supporting efficient updates on it. This problem is solved by *memory checking* (see our survey of related work in Section 1.2).

is likely to check sufficiently many values in *each* codeword block $\mathbf{c}_i$. Unfortunately, if we want a truly efficient scheme and set $n, t = o(\sqrt{\ell})$ to be small, then this solution becomes completely insecure. The server can delete a single codeword block $\mathbf{c}_i$ from $\mathbf{C}$ entirely, losing the corresponding message block $\mathbf{m}_i$, but still maintain a good chance of passing the above audit as long as none of the t random challenge locations coincides with the n deleted symbols, which happens with good probability.

Second Proposal. The first proposal (with small n, t) was insecure because a cheating server could easily identify the locations within $\mathbf{C}$ that correspond to a single message block and delete exactly the codeword symbols in these locations. We can prevent such attacks by pseudo-randomly permuting the locations of all of the different codeword-symbols of different codeword blocks together. That is, the client starts with the value $\mathbf{C} = (\mathbf{C}[1], \ldots, \mathbf{C}[Ln]) = (\mathbf{c}_1, \ldots, \mathbf{c}_L) \in \Sigma^{Ln}$ computed as in the first proposal. It chooses a pseudo-random permutation $\pi : [Ln] \to [Ln]$ and computes the permuted value $\mathbf{C}' := (\mathbf{C}[\pi(1)], \ldots, \mathbf{C}[\pi(Ln)])$ which it then stores on the server in an encrypted form (each codeword symbol is encrypted separately). The audit still checks t out of Ln random locations of the server storage and verifies authenticity.

It may seem that the server now cannot immediately identify and *selectively* delete codeword-symbols belonging to a single codeword block, thwarting the attack on the first proposal. Unfortunately, this modification only re-gains security in the static setting, when the client never performs any operations on the data.[3] Once the client wants to update some location of $\mathbf{M}$ that falls inside some message block $\mathbf{m}_i$, she has to reveal to the server where all of the n codeword symbols corresponding to $\mathbf{c}_i = \mathsf{Enc}(\mathbf{m}_i)$ reside in its storage since she needs to update exactly these values. Therefore, the server can later selectively delete exactly these n codeword symbols, leading to the same attack as in the first proposal.

Impossibility? Given the above failed attempts, it may even seem that truly efficient updates could be inherently incompatible with efficient audits in PoR. If an update is efficient and only changes a small subset of the server's storage, then the server can always just *ignore* the update, thereby failing to maintain knowledge of the latest version of the client data. All of the prior techniques appear ineffective against such attack. More generally, any audit protocol which just checks *a small subset of random* locations of the server's storage is unlikely to hit any of the locations involved in the update, and hence will not detect such cheating, meaning that it cannot be secure. However, this does not rule out the possibility of a very efficient solution that relies on a more clever audit protocol, which is likelier to check recently updated areas of the server's storage and therefore detect such an attack. Indeed, this property will be an important component in our actual solution.

[3] A variant of this idea was actually used by Juels and Kaliski [18] for extra efficiency in the static setting.

1.1 Our Results and Techniques

Overview of Result. In this work, we give the first solution to *dynamic PoR* that allows for efficient updates to client data. The client only keeps some short local state, and can execute arbitrary read/write operations on any location within the data by running a corresponding protocol with the server. At any point in time, the client can also initiate an audit protocol, which ensures that a passing server must have complete knowledge of the *latest version* of the client data. The cost of any read/write/audit execution in terms of server/client work and communication is only *polylogarithmic* in the size of the client data. The server's storage remains linear in the size of the client data. Therefore, our scheme is optimal in an asymptotic sense, up to polylogarithmic factors. See Section 6 for a detailed efficiency analysis.

PoR via Oblivious RAM. Our dynamic PoR solution starts with the same idea as the first proposal above, where the client redundantly encodes small blocks of her data individually to form the value $\mathbf{C} = (\mathbf{c}_1, \ldots, \mathbf{c}_L) \in \Sigma^{Ln}$, consisting of L codeword blocks and $\ell' = Ln$ codeword symbols, as defined previously. The goal is to then store $\mathbf{C}$ on the server in some "clever way" so that that the server cannot selectively delete too many symbols within any single codeword block $\mathbf{c}_i$, even after observing the client's read and write executions (which access exactly these symbols). As highlighted by the second proposal, simply permuting the locations of the codeword symbols of $\mathbf{C}$ is insufficient. Instead, our main idea it to store all of the individual codeword symbols of $\mathbf{C}$ on the server using an *oblivious RAM* scheme.

Overview of ORAM. Oblivious RAM (ORAM), initially defined by Goldreich and Ostrovsky [14], allows a client to outsource her *memory* to a remote server while allowing the client to perform random-access reads and writes in a *private* way. More precisely, the client has some data $\mathbf{D} \in \Sigma^d$, which she stores on the server in some carefully designed privacy-preserving form, while only keeping a short local state. She can later run efficient protocols with the server to read or write to the individual entries of $\mathbf{D}$. The read/write protocols of the ORAM scheme should be efficient, and the client/server work and communication during each such protocol should be small compared to the size of $\mathbf{D}$ (e.g., *polylogarithmic*). A secure ORAM scheme not only hides the *content* of $\mathbf{D}$ from the server, but also the *access pattern* of which *locations* in $\mathbf{D}$ the client is reading or writing in each protocol execution. Thus, the server cannot discern any correlation between the physical locations of its storage that it is asked to access during each read/write protocol execution and the logical location inside $\mathbf{D}$ that the client wants to access via this protocol.

In our work, we will also always use ORAM schemes that are *authenticated*, which means that the client can detect if the server ever sends an incorrect value. In particular, authenticated ORAM schemes ensure that the most recent version of the data is being retrieved in any accepting read execution, preventing the server from "rolling back" updates.

Construction of Dynamic PoR. A detailed technical description of our construction appears in Section 5, and below we give a simplified overview. In our PoR construction, the client starts with data $\mathbf{M} \in \Sigma^{\ell}$ which she splits into small message blocks $\mathbf{M} = (\mathbf{m}_1, \ldots, \mathbf{m}_L)$ with $\mathbf{m}_i \in \Sigma^k$ where the block size $k \ll \ell = Lk$ is only dependant on the security parameter. She then applies an error correcting code $\mathsf{Enc} : \Sigma^k \rightarrow \Sigma^n$ that can efficiently recover $\frac{n}{2}$ erasures to each message block individually, resulting in the value $\mathbf{C} = (\mathbf{c}_1, \ldots, \mathbf{c}_L) \in \Sigma^{Ln}$ where $\mathbf{c}_i = \mathsf{Enc}(\mathbf{m}_i)$. Finally, she initializes an ORAM scheme with the initial data $\mathbf{D} = \mathbf{C}$, which the ORAM stores on the server in some clever privacy-preserving form, while keeping only a short local state at the client.

Whenever the client wants to read or write to some location within her data, she uses the ORAM scheme to perform the necessary reads/writes on each of the n relevant codeword symbols of $\mathbf{C}$ (see details in Section 5). To run an audit, the client chooses t ($\approx$ security parameter) random locations in $\{1, \ldots, Ln\}$ and runs the ORAM read protocol t times to read the corresponding symbols of $\mathbf{C}$ that reside in these locations, checking them for authenticity.

Catching Disregarded Updates. First, let us start with a sanity check, to explain how the above construction can thwart a specific attack in which the server simply disregards the latest update. In particular, such attack should be caught by a subsequent audit. During the audit, the client runs the ORAM protocol to read t *random* codeword symbols and these are *unlikely* to coincide with any of the n codeword symbols modified by the latest update (recall that t and n are both small and independent of the data size ℓ). However, the ORAM scheme stores data on the server in a highly organized data-structure, and ensures that the most recently updated data is accessed during *any* subsequent "read" execution, even for an unrelated logical location. This is implied by ORAM security since we need to hide whether or not the location of a read was recently updated or not. Therefore, although the audit executes the "ORAM read" protocols on random logical locations inside $\mathbf{C}$, the ORAM scheme will end up scanning recently updated ares of the server's actual storage and check them for authenticity, ensuring that recent updates have not been disregarded.

Security and "Next-Read Pattern Hiding". The high-level security intuition for our PoR scheme is quite simple. The ORAM hides from the server where the various locations of $\mathbf{C}$ reside in its storage, even after observing the access pattern of read/write executions. Therefore it is difficult for the server to reach a state where it will fail on read executions for most locations within some single codeword block (lose data) without also failing on too many read executions altogether (lose the ability to pass an audit).

Making the above intuition formal is quite subtle, and it turns out that standard notion of ORAM security does *not* suffice. The main issue is that that the server may be able to somehow delete *all* (or most) of the n codeword symbols that fall within *some* codeword block $\mathbf{c}_i = (\mathbf{C}[j+1], \ldots, \mathbf{C}[j+n])$ without knowing *which* block it deleted. Therefore, although the server will fail on any subsequent read if and only if its location falls within the range $\{j+1, \ldots, j+n\}$, it will not learn anything about the location of the read itself since it does not know the index j. Indeed, we will give an example of a contrived ORAM scheme where such an attack is possible and our resulting construction of PoR using this ORAM is *insecure.*

We show, however, that the intuitive reasoning above can be salvaged if the ORAM scheme achieves a new notion of security that we call *next-read pattern hiding (NRPH)*, which may be of independent interest. NRPH security considers an adversarial server that first gets to observe many read/write protocol executions performed sequentially with the client, resulting in some final client configuration $\mathcal{C}_{\mathsf{fin}}$. The adversarial server then gets to see various possibilities for how the "*next read*" operation would be executed by the client for various distinct locations, where each such execution starts from the same *fixed* client configuration $\mathcal{C}_{\mathsf{fin}}$.[4] The server should not be able to discern any *relationship* between these executions and the locations they are reading. For example, two such "next-read" executions where the client reads two consecutive locations should be indistinguishable from two executions that read two random and unrelated locations. This notion of NRPH security will be used to show that server cannot reach a state where it can *selectively* fail to respond on read queries whose location falls within some small range of a single codeword block (lose data), but still respond correctly to most completely random reads (pass an audit).

Proving Security via an Extractor. We now give a high-level overview of how our PoR extractor works. In particular, we claim that we can take any adversarial server that has a "good" chance of passing an audit and use the extractor to efficiently recover the latest version of the client data from it. The extractor initializes an "empty array" $\mathbf{C}$. It then executes random audit protocols with the server, by acting as the honest client. In particular, it chooses t random locations within the array and runs the corresponding ORAM read protocols. If the execution of the audit is successful, the extractor fills in the corresponding values of $\mathbf{C}$ that it learned during the audit execution. In either case, it then rewinds the server and runs a fresh execution of the audit, repeating this step for several iterations.

Since the server has a good chance of passing a random audit, it is easy to show that the extractor can eventually recover a large fraction, say $> \frac{3}{4}$, of the entries inside $\mathbf{C}$ by repeating this process sufficiently many times. Because of the *authenticity* of the ORAM, the recovered values are the correct ones, corresponding to the latest version of the client data. Now we need to argue that there is no codeword block $\mathbf{c}_i$ within $\mathbf{C}$ for which the extractor recovered fewer than $\frac{1}{2}$ of its codeword symbols, as this would prevent us from applying erasure decoding and recovering the underlying message block. Let FAILURE denote the above bad event. If all the recovered locations (comprising $> \frac{3}{4}$ fraction of the total) were distributed uniformly within $\mathbf{C}$ then FAILURE would occur with negligible probability, as long as the codeword size n is sufficiently large in the security parameter. We can now rely on the NRPH security of the ORAM to ensure that FAILURE also happens with negligible probability in our case.

[4] This is in contrast to the standard sequential operations where the client state is updated after each execution.

We can think of the FAILURE event as a function of the locations queried by the extractor in each audit execution, and the set of executions on which the server fails. If the malicious server can cause FAILURE to occur, it means that it can distinguish the pattern of locations actually queried by the extractor during the audit executions (for which the FAILURE event occurs) from a randomly permuted pattern of locations (for which the FAILURE event does not occur with overwhelming probability). Note that the use of rewinding between the audit executions of the extractor forces us to rely on NRPH security rather than just standard ORAM security.

The above presents the high-level intuition and is somewhat oversimplified. See Section 4 for the formal definition of NRPH security and Section 5 for the formal description of our dynamic PoR scheme and a rigorous proof of security.

Achieving Next-Read Pattern Hiding. We show that standard ORAM security does *not* generically imply NRPH security, by giving a contrived scheme that satisfies the former but not the latter. Nevertheless, all natural ORAM constructions in the literature *do* essentially satisfy NRPH security. In the full version [8], we look at one particularly efficient ORAM construction of Goodrich and Mitzenmacher [16] in depth, and prove that (with minor modifications) it is NRPH secure.

Contributions. We call our final scheme PORAM since it combines the techniques and security of PoR and ORAM. In particular, other than providing provable dynamic cloud storage as was our main goal, our scheme also satisfies the strong *privacy* guarantees of ORAM, meaning that it hides all contents of the remotely stored data as well as the access pattern of which locations are accessed when. It also provides strong *authenticity* guarantees (same as *memory checking*; see Section 1.2), ensuring that any "read" execution with a malicious remote server is guaranteed to return the latest version of the data (or detect cheating). In brief, our contributions can be summarized as follows:

- We give the first asymptotically efficient solution to PoR for outsourced dynamic data, where a successful audit ensures that the server knows the latest version of the client data. In particular:
    - Client storage is small and independent of the data size.
    - Server storage is linear in the data size, expanding it by only a small constant factor.
    - Communication and computation of client and server during *read*, *write*, and *audit* executions are polylogarithmic in the size of the client data.
- Our scheme also achieves strong *privacy* and *authenticity* guarantees, matching those of *oblivious RAM* and *memory checking*.

We mention that the PORAM scheme is simple to implement and has low concrete efficiency overhead *on top of* an underlying ORAM scheme with NRPH security. There is much recent and ongoing research activity in instantiating/ implementing truly practical ORAM schemes, which are likely to yield correspondingly practical instantiations of our PORAM protocol.

1.2 Related Work

Proofs of retrievability for *static* data were initially defined and constructed by Juels and Kaliski [18], building on a closely related notion called sublinear-authenticators of Naor and Rothblum [21]. Concurrently, Ateniese et al. [1] defined another related primitive called *provable data possession* (PDP). Since then, there has been much ongoing research activity on PoR and PDP schemes.

PoR vs. PDP. The main difference between PoR and PDP is the notion of security that they achieve. A PoR audit guarantees that the server maintains knowledge of *all* of the client data, while a PDP audit only ensures that the server is storing *most* of the client data. For example, in a PDP scheme, the server may lose a small portion of client data (say 1 MB out of a 10 GB file) and may maintain an high chance of passing a future audit. On a technical level, the main difference in most prior PDP/PoR constructions is that PoR schemes store a *redundant encoding* of the client data on the server. For a detailed comparison, see Küpçü [19,20].

Static Data. PoR and PDP schemes for static data (without updates) have received much research attention [24,11,7,2], with works improving on communication efficiency and exact security, yielding essentially optimal solutions. Another interesting direction has been to extend these works to the multi-server setting [6,9,10] where the client can use the audit mechanism to identify faulty machines and recover the data from the others.

Dynamic Data. The works of Ateniese et al. [3], Erway et al. [13] and Wang et al. [27] show how to achieve PDP security for *dynamic data*, supporting efficient updates. This is closely related to work on memory checking [5,21,12], which studies how to authenticate remotely stored dynamic data so as to allow efficient reads/writes, while being able to verify the authenticity of the latest version of the data (preventing the server from "rolling back" updates and using an old version). Unfortunately, these techniques alone cannot be used to achieve the stronger notion of PoR security. Indeed, the main difficulty that we resolve in this work, how to efficiently update *redundantly encoded data*, does not come up in the context of PDP.

A recent work of Stefanov et al. [26] considers PoR for dynamic data, but in a more complex setting where an additional trusted "portal" performs some operations on behalf of the client, and can cache updates for an extended period of time. It is not clear if these techniques can be translated to the basic client/server setting, which we consider here. However, even in this modified setting, the complexity of the updates and the audit in that work is proportional to *square-root* of the data size, whereas ours is *polylogarithmic*.

2 Preliminaries

Notation. Throughout, we use λ to denote the *security parameter*. We identify *efficient* algorithms as those running in (probabilistic) polynomial time in λ and their input lengths, and identify *negligible* quantities (e.g., acceptable error probabilities) as $\mathsf{negl}(\lambda) = 1/\lambda^{\omega(1)}$, meaning that they are asymptotically smaller than $1/\lambda^c$ for every constant $c > 0$. For $n \in \mathbb{N}$, we define the set $[n] \overset{\text{def}}{=} \{1, \ldots, n\}$. We use the notation $(k \bmod n)$ to denote the unique integer $i \in \{0, \ldots, n-1\}$ such that $i = k \pmod{n}$.

Erasure Codes. We say that $(\mathsf{Enc}, \mathsf{Dec})$ is an $(n, k, d)_\Sigma$*-code with efficient erasure decoding* over an alphabet Σ if the original message can always be recovered from a corrupted codeword with at most $d-1$ erasures. That is, for every *message* $\mathbf{m} = (m_1, \ldots, m_k) \in \Sigma^k$ giving a *codeword* $\mathbf{c} = (c_1, \ldots, c_n) = \mathsf{Enc}(\mathbf{m})$, and every corrupted codeword $\tilde{\mathbf{c}} = (\tilde{c}_1, \ldots, \tilde{c}_n)$ such that $\tilde{c}_i \in \{c_i, \perp\}$ and the number of erasures is $|\{i \in [n] : \tilde{c}_i = \perp\}| \leq d-1$, we have $\mathsf{Dec}(\tilde{\mathbf{c}}) = \mathbf{m}$. We say that a code is *systematic* if, for every message $\mathbf{m}$, the codeword $\mathbf{c} = \mathsf{Enc}(\mathbf{m})$ contains $\mathbf{m}$ in the first k positions $c_1 = m_1, \ldots, c_k = m_k$. A systematic variant of the Reed-Solomon code achieves the above for any integers $n > k$ and any *field* Σ of size $|\Sigma| \geq n$ with $d = n - k + 1$.

Virtual Memory. We think of *virtual memory* $\mathbf{M}$, with *word-size* w and *length* ℓ, as an array $\mathbf{M} \in \Sigma^\ell$ where $\Sigma \overset{\text{def}}{=} \{0,1\}^w$. We assume that, initially, each location $\mathbf{M}[i]$ contains the special *uninitialized symbol* $\mathbf{0} = 0^w$. Throughout, we will think of ℓ as some large polynomial in the security parameter, which upper bounds the amount of memory that can be used.

Outsourcing Virtual Memory. In the next two sections, we look at two primitives: *dynamic PoR* and *ORAM.* These primitives allow a client to *outsource* some virtual memory $\mathbf{M}$ to a remote server, while providing useful security guarantees. Reading and writing to some location of $\mathbf{M}$ now takes on the form of a protocol execution with the server. The goal is to provide security while preserving efficiency in terms of client/server computation, communication, and the number of server-memory accesses per operation, which should all be *poly-logarithmic* in the length ℓ. We also want to optimize the size of the client storage (independent of ℓ) and server storage (not much larger than ℓ).

3 Dynamic PoR

A *Dynamic PoR* scheme consists of protocols $\mathbf{P}\mathsf{Init}, \mathbf{P}\mathsf{Read}, \mathbf{P}\mathsf{Write}, \mathsf{Audit}$ between two *stateful* parties: a client $\mathcal{C}$ and a server $\mathcal{S}$. The server acts as the curator for some virtual memory $\mathbf{M}$, which the client can *read*, *write* and *audit* by initiating the corresponding interactive protocols:

- $\mathbf{P}\mathsf{Init}(1^\lambda, 1^w, \ell)$: This protocol corresponds to the client initializing an (empty) virtual memory $\mathbf{M}$ with word-size w and length ℓ, which it supplies as inputs.
- $\mathbf{P}\mathsf{Read}(i)$: This protocol corresponds to the client reading $v = \mathbf{M}[i]$, where it supplies the input i and outputs some value v at the end.
- $\mathbf{P}\mathsf{Write}(i, v)$: This protocol corresponds to setting $\mathbf{M}[i] := v$, where the client supplies the inputs i, v.
- Audit: This protocol is used by the client to verify that the server is maintaining the memory contents correctly so that they remain retrievable. The client outputs a decision $b \in \{\texttt{accept}, \texttt{reject}\}$.

The client $\mathcal{C}$ in the protocols may be *randomized*, but we assume (w.l.o.g.) that the honest server $\mathcal{S}$ is deterministic. At the conclusion of the $\mathbf{P}\mathsf{Init}$ protocol, both the client and the server create some long-term local state, which each party will update during the execution of each of the subsequent protocols. The client may also output `reject` during the execution of the $\mathbf{P}\mathsf{Init}$, $\mathbf{P}\mathsf{Read}$, $\mathbf{P}\mathsf{Write}$ protocols, to denote that it detected some misbehavior of the server. Note that we assume that the virtual memory is initially *empty*, but if the client has some initial data, she can write it onto the server block-by-block immediately after initialization. For ease of presentation, we may assume that the state of the client and the server always contains the security parameter, and the memory parameters $(1^\lambda, 1^w, \ell)$. We now define the three properties of a dynamic PoR scheme: *correctness*, *authenticity* and *retrievability*. For these definitions, we say that $P = (op_0, op_1, \dots, op_q)$ is a dynamic PoR *protocol sequence* if $op_0 = \mathbf{P}\mathsf{Init}(1^\lambda, 1^w, \ell)$ and, for $j > 0$, $op_j \in \{\mathbf{P}\mathsf{Read}(i), \mathbf{P}\mathsf{Write}(i, v), \mathsf{Audit}\}$ for some index $i \in [\ell]$ and value $v \in \{0,1\}^w$.

Correctness. If the client and the server are both *honest* and $P = (op_0, \dots, op_q)$ is some protocol sequence, then we require the following to occur with probability 1 over the randomness of the client:

- Each execution of a protocol $op_j = \mathbf{P}\mathsf{Read}(i)$ results in the client outputting the correct value $v = \mathbf{M}[i]$, matching what would happen if the corresponding operations were performed directly on a memory $\mathbf{M}$. In particular, v is the value contained in the most recent prior write operation with location i, or, if no such prior operation exists, $v = \mathbf{0}$.
- Each execution of the Audit protocol results in the decision $b =$ `accept`.

Authenticity. We require that the client can always *detect* if any protocol message sent by the server deviates from honest behavior. More precisely, consider the following game $\mathtt{AuthGame}_{\tilde{\mathcal{S}}}(\lambda)$ between a malicious server $\tilde{\mathcal{S}}$ and a challenger:

- The malicious server $\tilde{\mathcal{S}}(1^\lambda)$ specifies a valid protocol sequence $P = (op_0, \dots, op_q)$.
- The challenger initializes a copy of the honest client $\mathcal{C}$ and the (deterministic) honest server $\mathcal{S}$. It sequentially executes $op_0, \dots, op_q$ between $\mathcal{C}$ and the malicious server $\tilde{\mathcal{S}}$ while, in parallel, also passing a copy of every message from $\mathcal{C}$ to the honest server $\mathcal{S}$.
- If, at any point during the execution of some op_j, any protocol message given by $\tilde{\mathcal{S}}$ differs from that of $\mathcal{S}$, and the client $\mathcal{C}$ does not output `reject`, the adversary wins and the game outputs 1. Else 0.

For any efficient adversarial server $\tilde{\mathcal{S}}$, we require $\Pr[\mathtt{AuthGame}_{\tilde{\mathcal{S}}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$. Note that authenticity and correctness together imply that the client will always either read the correct value corresponding to the latest contents of the virtual memory or reject whenever interacting with a malicious server.

Retrievability. Finally we define the main purpose of a dynamic PoR scheme, which is to ensure that the client data remains retrievable. We wish to guarantee that, whenever the malicious server is in a state with a reasonable probability δ

of successfully passing an audit, he must *know* the entire content of the client's virtual memory $\mathbf{M}$. As in "proofs of knowledge", we formalize *knowledge* via the existence of an efficient *extractor* $\mathcal{E}$ which can recover the value $\mathbf{M}$ given (black-box) access to the malicious server.

More precisely, we define the game $\texttt{ExtGame}_{\tilde{\mathcal{S}},\mathcal{E}}(\lambda, p)$ between a malicious server $\tilde{\mathcal{S}}$, extractor $\mathcal{E}$, and challenger:

- The malicious server $\tilde{\mathcal{S}}(1^\lambda)$ specifies a protocol sequence $P = (op_0, \ldots, op_q)$. Let $\mathbf{M} \in \Sigma^\ell$ be the correct value of the memory contents at the end of executing P.
- The challenger initializes a copy of the honest client $\mathcal{C}$ and sequentially executes $op_0, \ldots, op_q$ between $\mathcal{C}$ and $\tilde{\mathcal{S}}$. Let $\mathcal{C}_{\mathsf{fin}}$ and $\tilde{\mathcal{S}}_{\mathsf{fin}}$ be the final configurations (states) of the client and malicious server at the end of this interaction, including all of the random coins of the malicious server. Define the success-probability $\mathbf{Succ}(\tilde{\mathcal{S}}_{\mathsf{fin}}) \overset{\text{def}}{=} \Pr\left[\tilde{\mathcal{S}}_{\mathsf{fin}} \overset{\mathsf{Audit}}{\longleftrightarrow} \mathcal{C}_{\mathsf{fin}} = \texttt{accept}\right]$ as the probability that an execution of a subsequent Audit protocol between $\tilde{\mathcal{S}}_{\mathsf{fin}}$ and $\mathcal{C}_{\mathsf{fin}}$ results in the latter outputting $\texttt{accept}$. The probability is only over the random coins of $\mathcal{C}_{\mathsf{fin}}$ during this execution.
- Run $\mathbf{M}' \leftarrow \mathcal{E}^{\tilde{\mathcal{S}}_{\mathsf{fin}}}(\mathcal{C}_{\mathsf{fin}}, 1^\ell, 1^p)$, where the extractor $\mathcal{E}$ gets *black-box rewinding access* to the malicious server in its final configuration $\tilde{\mathcal{S}}_{\mathsf{fin}}$, and attempts to extract out the memory contents as $\mathbf{M}'$.[5]
- If $\mathbf{Succ}(\tilde{\mathcal{S}}_{\mathsf{fin}}) \geq 1/p$ and $\mathbf{M}' \neq \mathbf{M}$ then output 1, else 0.

We require that there exists a probabilistic-poly-time extractor $\mathcal{E}$ such that, for every efficient malicious server $\tilde{\mathcal{S}}$ and every polynomial $p = p(\lambda)$ we have $\Pr[\texttt{ExtGame}_{\tilde{\mathcal{S}},\mathcal{E}}(\lambda, p) = 1] \leq \mathsf{negl}(\lambda)$.

The above says that whenever the malicious server reaches some state $\tilde{\mathcal{S}}_{\mathsf{fin}}$ in which it maintains a $\delta \geq 1/p$ probability of passing the *next audit*, the extractor $\mathcal{E}$ will be able to extract out the correct memory contents $\mathbf{M}$ from $\tilde{\mathcal{S}}_{\mathsf{fin}}$, meaning that the server must retain full *knowledge* of $\mathbf{M}$ in this state. The extractor is efficient, but can run in time polynomial in p and the size of the memory ℓ.

A Note on Adaptivity. We defined the above *authenticity* and *retrievability* properties assuming that the sequence of read/write operations is adversarial, but is chosen *non-adaptively*, before the adversarial server sees any protocol executions. This seems to be sufficient in most realistic scenarios, where the server is unlikely to have any influence on which operations the client wants to perform. It also matches the security notions in prior works on ORAM. Nevertheless, we note that our final results also achieve adaptive security, where the attacker can choose the sequence of operations op_i adaptively after seeing the execution of previous operations, if the underlying ORAM satisfies this notion. Indeed, most prior ORAM solutions seem to do so, but it was never included in their analyses.

[5] This is similar to the extractor in zero-knowledge proofs of knowledge. In particular $\mathcal{E}$ can execute protocols with the malicious server in its state $\tilde{\mathcal{S}}_{\mathsf{fin}}$ and rewind it back to this state at the end of the execution.

4 Oblivious RAM with Next-Read Pattern Hiding

An ORAM consists of protocols ($\mathbf{O}\mathsf{Init}, \mathbf{O}\mathsf{Read}, \mathbf{O}\mathsf{Write}$) between a client $\mathcal{C}$ and a server $\mathcal{S}$, with the same syntax as the corresponding protocols in PoR. We will also extend the syntax of $\mathbf{O}\mathsf{Read}$ and $\mathbf{O}\mathsf{Write}$ to allow for reading/writing from/to multiple distinct locations simultaneously. That is, for arbitrary $t \in \mathbb{N}$, we define the protocol $\mathbf{O}\mathsf{Read}(i_1, \ldots, i_t)$ for *distinct* indices $i_1, \ldots, i_t \in [\ell]$, in which the client outputs $(v_1, \ldots, v_t)$ corresponding to reading $v_1 = \mathbf{M}[i_1], \ldots, v_t = \mathbf{M}[i_t]$. Similarly, we define the protocol $\mathbf{O}\mathsf{Write}(i_t, \ldots, i_t; v_1, \ldots, v_t)$ for *distinct* indices $i_1, \ldots, i_t \in [\ell]$, which corresponds to setting $\mathbf{M}[i_1] := v_1, \ldots, \mathbf{M}[i_t] := v_t$.

We say that $P = (op_0, \ldots, op_q)$ is an *ORAM protocol sequence* if $op_0 = \mathbf{O}\mathsf{Init}(1^\lambda, 1^w, \ell)$ and, for $j > 0$, op_j is a valid (multi-location) read/write operation. We require that an ORAM construction needs to satisfy *correctness* and *authenticity*, which are defined the same way as in PoR. (Traditionally, authenticity is not always defined/required for ORAM. However, it is crucial for our use. As noted in several prior works, it can often be added at almost no cost to efficiency. It can also be added generically by running a *memory checking* scheme on top of ORAM.) We now define a new property called *next-read pattern hiding.*

Next-Read Pattern Hiding. Consider an *honest-but-curious* server $\mathcal{A}$ who observes the execution of some protocol sequence P with a client $\mathcal{C}$ resulting in the final client configuration $\mathcal{C}_{\mathsf{fin}}$. At the end of this execution, $\mathcal{A}$ gets to observe how $\mathcal{C}_{\mathsf{fin}}$ would execute the *next* read operation $\mathbf{O}\mathsf{Read}(i_1, \ldots, i_t)$ for various different t-tuples $(i_1, \ldots, i_t)$ of locations, but always starting in the same client state $\mathcal{C}_{\mathsf{fin}}$. We require that $\mathcal{A}$ cannot observe any correlation between these next-read executions and their locations, up to *equality.* That is, $\mathcal{A}$ should not be able to distinguish if $\mathcal{C}_{\mathsf{fin}}$ instead executes the next-read operations on *permuted locations* $\mathbf{O}\mathsf{Read}(\pi(i_1), \ldots, \pi(i_t))$ for a permutation $\pi : [\ell] \to [\ell]$.

More formally, we define $\mathtt{NextReadGame}^b_{\mathcal{A}}(\lambda)$, for $b \in \{0, 1\}$, between an adversary $\mathcal{A}$ and a challenger:

- The attacker $\mathcal{A}(1^\lambda)$ chooses an ORAM protocol sequence $P_1 = (op_0, \ldots, op_{q_1})$. It also chooses a sequence $P_2 = (rop_1, \ldots, rop_{q_2})$ of valid multi-location read operations, where each operation is of the form $rop_j = \mathbf{O}\mathsf{Read}(i_{j,1}, \ldots, i_{j,t_j})$ with t_j distinct locations. Lastly, it chooses a permutation $\pi : [\ell] \to [\ell]$. For each rop_j in P_2, define a permuted version $rop'_j := \mathbf{O}\mathsf{Read}(\pi(i_{j,1}), \ldots, \pi(i_{j,t_j}))$. The game now proceeds in two stages.
- *Stage I.* The challenger initializes the honest client $\mathcal{C}$ and the (deterministic) honest server $\mathcal{S}$. It sequentially executes the protocols $P = (op_0, \ldots, op_{q_1})$ between $\mathcal{C}$ and $\mathcal{S}$. Let $\mathcal{C}_{\mathsf{fin}}, \mathcal{S}_{\mathsf{fin}}$ be the final configuration of the client and server at the end.
- *Stage II.* For each $j \in [q_2]$: challenger either executes the original operation rop_j if $b = 0$, or the permuted operation rop'_j if $b = 1$, between $\mathcal{C}$ and $\mathcal{S}$. At the end of each operation execution it resets the configuration of the client and server back to $\mathcal{C}_{\mathsf{fin}}, \mathcal{S}_{\mathsf{fin}}$ respectively, before the next execution.
- The adversary $\mathcal{A}$ is given the transcript of all the protocol executions in stages I and II, and outputs a bit $\tilde{b}$ which we define as the output of the game.

Note that, since the honest server $\mathcal{S}$ is deterministic, seeing the protocol transcripts between $\mathcal{S}$ and $\mathcal{C}$ is the same as seeing the entire internal state of $\mathcal{S}$ at any point time.

We require that, for every efficient $\mathcal{A}$, we have

$$\left|\Pr[\texttt{NextReadGame}^0_{\mathcal{A}}(\lambda) = 1] - \Pr[\texttt{NextReadGame}^1_{\mathcal{A}}(\lambda) = 1]\right| \leq \mathsf{negl}(\lambda).$$

5 PORAM: Dynamic PoR via ORAM

We now give our construction of dynamic PoR, using ORAM. Since the ORAM security properties are preserved by the construction as well, we happen to achieve ORAM and dynamic PoR simultaneously. Therefore, we call our construction PORAM.

Overview of Construction. Let (Enc, Dec) be an $(n, k, d = n - k + 1)_\Sigma$ systematic code with efficient erasure decoding over the alphabet $\Sigma = \{0,1\}^w$ (e.g., the systematic Reed-Solomon code over $\mathbb{F}_{2^w}$). Our construction of dynamic PoR will interpret the memory $\mathbf{M} \in \Sigma^\ell$ as consisting of $L = \ell/k$ consecutive *message blocks*, each having k alphabet symbols (assume k is small and divides ℓ). The construction implicitly maps operation on $\mathbf{M}$ to operations on *encoded memory* $\mathbf{C} \in (\Sigma)^{\ell_{\mathsf{code}}=Ln}$, which consists of L *codeword blocks* with n alphabet symbols each. The L codeword blocks $\mathbf{C} = (\mathbf{c}_1, \dots, \mathbf{c}_L)$ are simply the encoded versions of the corresponding message blocks in $\mathbf{M} = (\mathbf{m}_1, \dots, \mathbf{m}_L)$ with $\mathbf{c}_q = \mathsf{Enc}(\mathbf{m}_q)$ for $q \in [L]$. This means that, for each $i \in [\ell]$, the value of the memory location $\mathbf{M}[i]$ can only affect the values of the encoded-memory locations $\mathbf{C}[j+1], \dots, \mathbf{C}[j+n]$ where $j = n \cdot \lfloor i/k \rfloor$. Furthermore, since the encoding is *systematic*, we have $\mathbf{M}[i] = \mathbf{C}[j+u]$ where $u = (i \mod k) + 1$. To read the memory location $\mathbf{M}[i]$, the client will use ORAM to read the codeword location $\mathbf{C}[j+u]$. To write to the memory location $\mathbf{M}[i] := v$, the client needs to update the entire corresponding codeword block. She does so by first using ORAM to read the corresponding codeword block $\mathbf{c} = (\mathbf{C}[j+1], \dots, \mathbf{C}[j+n])$, and decodes to obtain the original memory block $\mathbf{m} = \mathsf{Dec}(\mathbf{c})$. She then locally updates the memory block by setting $\mathbf{m}[u] := v$, re-encodes the updated memory block to get $\mathbf{c}' = (c_1, \dots, c_n) := \mathsf{Enc}(\mathbf{m})$ and uses the ORAM to write $\mathbf{c}'$ back into the encoded memory, setting $\mathbf{C}[j+1] := c'_1, \dots, \mathbf{C}[j+n] := c'_n$.

The Construction. Our PORAM construction is defined for some parameters $n > k, t \in \mathbb{N}$. Let $\mathbf{O} = (\mathbf{O}\mathsf{Init}, \mathbf{O}\mathsf{Read}, \mathbf{O}\mathsf{Write})$ be an ORAM. Let (Enc, Dec) be an $(n, k, d = n - k + 1)_\Sigma$ systematic code with efficient erasure decoding over the alphabet $\Sigma = \{0,1\}^w$ (e.g., the systematic Reed-Solomon code over $\mathbb{F}_{2^w}$).

- $\mathbf{P}\mathsf{Init}(1^\lambda, 1^w, \ell)$: Assume k divides ℓ and let $\ell_{\mathsf{code}} := n \cdot (\ell/k)$. Run the $\mathbf{O}\mathsf{Init}(1^\lambda, 1^w, \ell_{\mathsf{code}})$ protocol.
- $\mathbf{P}\mathsf{Read}(i)$: Let $i' := n \cdot \lfloor i/k \rfloor + (i \mod k) + 1$ and run the $\mathbf{O}\mathsf{Read}(i')$ protocol.
- $\mathbf{P}\mathsf{Write}(i, v)$: Set $j := n \cdot \lfloor i/k \rfloor$ and $u := (i \mod k) + 1$.
  - Run $\mathbf{O}\mathsf{Read}(j+1, \dots, j+n)$ and get output $\mathbf{c} = (c_1, \dots, c_n)$.
  - Decode $\mathbf{m} = (m_1, \dots, m_k) = \mathsf{Dec}(\mathbf{c})$.

  - Modify position u of $\mathbf{m}$ by locally setting $m_u := v$. Re-encode the modified message-block $\mathbf{m}$ by setting $\mathbf{c}' = (c'_1, \ldots, c'_n) := \mathsf{Enc}(\mathbf{m})$.
  - Run $\mathbf{O}\mathsf{Write}(j+1, \ldots, j+n; c'_1, \ldots, c'_n)$.
- Audit: Pick t distinct indices $j_1, \ldots, j_t \in [\ell_{\mathsf{code}}]$ at random. Run $\mathbf{O}\mathsf{Read}$ $(j_1, \ldots, j_t)$ and return `accept` iff the protocol finished without outputting `reject`.

If, any ORAM protocol execution in the above scheme outputs `reject`, the client enters a special rejection state in which it stops responding and automatically outputs `reject` for any subsequent protocol execution.

As our main result, we now prove that if the ORAM scheme satisfies next-read pattern hiding (NRPH) security then the PORAM construction above is also a secure dynamic PoR scheme. See the full version [8] for a proof of the following theorem.

Theorem 1. *Assume that* $\mathbf{O} = (\mathbf{O}\mathsf{Init}, \mathbf{O}\mathsf{Read}, \mathbf{O}\mathsf{Write})$ *is an ORAM with next-read pattern hiding (NRPH) security, and we choose parameters* $k = \Omega(\lambda)$, $k/n = (1 - \Omega(1))$, $t = \Omega(\lambda)$. *Then the above scheme* $\mathsf{PORAM} = (\mathbf{P}\mathsf{Init}, \mathbf{P}\mathsf{Read}, \mathbf{P}\mathsf{Write}, \mathsf{Audit})$ *is a dynamic PoR scheme.*

ORAM with NPRH Security. The notion of ORAM was introduced by Goldreich and Ostrovsky [14], who also introduced the so-called *hierarchical scheme*. Since then several improvements to the hierarchical scheme have been given, including improved rebuild phases and the use of advanced hashing techniques (e.g., [23,16] etc.).

In the full version of our work [8], we examine a particular ORAM scheme of Goodrich and Mitzenmacher [16] and show that (with minor modifications) it satisfies *next-read pattern hiding* security. Therefore, this scheme can be used to instantiate our PORAM construction. We note that most other ORAM schemes from the literature that follow the hierarchical structure also seemingly satisfy next-read pattern hiding, and we only focus on the above example for concreteness. However, in the full version of our work, we show that it is *not* the case that *every* ORAM scheme satisfies next-read pattern hiding, and in fact give an example of a contrived scheme which does not satisfy this notion and makes our construction of PORAM completely insecure. We also believe that there are natural schemes, such as the ORAM of Shi et al. [25], which do not satisfy this notion. Therefore, next-read pattern hiding is a meaningful property beyond standard ORAM security and must be examined carefully.

6 Efficiency

We now look at the efficiency of our PORAM construction (when instantiated with the ORAM scheme of Goodrich-Mitzemacher [16] with the worst-case complexity optimization [17,22]). We analyze efficiency in three ways: firstly, we look at the overhead of PORAM scheme on top of just storing the data inside of the ORAM , secondly, we look at the overall efficiency of PORAM, and thirdly, we compare it with dynamic PDP [13,27] which does not employ erasure codes and does not provide full retrievability guarantee. In the table below, ℓ denotes the size of the client data and λ is the security parameter. We assume that the ORAM scheme uses a PRF whose computation takes $O(\lambda)$ work.

PORAM Efficiency	**vs. ORAM**	**Overall**	**vs. Dynamic PDP** [13]
Client Storage	Same	$O(\lambda)$	Same
Server Storage	$\times\ O(1)$	$O(\ell)$	$\times\ O(1)$
Read Complexity	$\times\ O(1)$	$O(\lambda \log^2 \ell)$	$\times\ O(\log \ell)$
Write Complexity	$\times\ O(\lambda)$	$O(\lambda^2 \times \log^2 \ell)$	$\times\ O(\lambda \times \log \ell)$
Audit Complexity	Read $\times\ O(\lambda)$	$O(\lambda^2 \times \log^2 \ell)$	$\times\ O(\log \ell)$

By modifying the underlying ORAM to dynamically resize tables during rebuilds, the resulting PORAM instantiation can achieve the same efficiency measures as above with ℓ taken to be amount of memory currently used, rather than the maximum memory use.

Disclaimer

The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. Distribution Statement "A" (Approved for Public Release, Distribution Unlimited).

References

1. Ateniese, G., Burns, R.C., Curtmola, R., Herring, J., Kissner, L., Peterson, Z.N.J., Song, D.: Provable data possession at untrusted stores. In: Ning, P., di Vimercati, S.D.C., Syverson, P.F. (eds.) ACM CCS 2007, pp. 598–609. ACM Press (October 2007)
2. Ateniese, G., Kamara, S., Katz, J.: Proofs of storage from homomorphic identification protocols. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 319–333. Springer, Heidelberg (2009)
3. Ateniese, G., Pietro, R.D., Mancini, L.V., Tsudik, G.: Scalable and efficient provable data possession. Cryptology ePrint Archive, Report 2008/114 (2008)
4. Bellare, M., Goldreich, O.: On defining proofs of knowledge. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 390–420. Springer, Heidelberg (1993)
5. Blum, M., Evans, W.S., Gemmell, P., Kannan, S., Naor, M.: Checking the correctness of memories. Algorithmica 12(2/3), 225–244 (1994)
6. Bowers, K.D., Juels, A., Oprea, A.: HAIL: a high-availability and integrity layer for cloud storage. In: Al-Shaer, E., Jha, S., Keromytis, A.D. (eds.) ACM CCS 2009, pp. 187–198. ACM Press (November 2009)
7. Bowers, K.D., Juels, A., Oprea, A.: Proofs of retrievability: theory and implementation. In: Sion, R., Song, D. (eds.) CCSW, pp. 43–54. ACM (2009)
8. Cash, D., Kupcu, A., Wichs, D.: Dynamic proofs of retrievability via oblivious ram. Cryptology ePrint Archive, Report 2012/550 (2012)

9. Chen, B., Curtmola, R., Ateniese, G., Burns, R.C.: Remote data checking for network coding-based distributed storage systems. In: Perrig, A., Sion, R. (eds.) CCSW, pp. 31–42. ACM (2010)
10. Curtmola, R., Khan, O., Burns, R., Ateniese, G.: Mr-pdp: Multiple-replica provable data possession. In: ICDCS (2008)
11. Dodis, Y., Vadhan, S., Wichs, D.: Proofs of retrievability via hardness amplification. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 109–127. Springer, Heidelberg (2009)
12. Dwork, C., Naor, M., Rothblum, G.N., Vaikuntanathan, V.: How efficient can memory checking be? In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 503–520. Springer, Heidelberg (2009)
13. Erway, C.C., Küpçü, A., Papamanthou, C., Tamassia, R.: Dynamic provable data possession. In: Al-Shaer, E., Jha, S., Keromytis, A.D. (eds.) ACM CCS 2009, pp. 213–222. ACM Press (November 2009)
14. Goldreich, O., Ostrovsky, R.: Software protection and simulation on oblivious RAMs. Journal of the ACM 43(3), 431–473 (1996)
15. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM Journal on Computing 18(1), 186–208 (1989)
16. Goodrich, M.T., Mitzenmacher, M.: Privacy-preserving access of outsourced data via oblivious RAM simulation. In: Aceto, L., Henzinger, M., Sgall, J. (eds.) ICALP 2011, Part II. LNCS, vol. 6756, pp. 576–587. Springer, Heidelberg (2011)
17. Goodrich, M.T., Mitzenmacher, M., Ohrimenko, O., Tamassia, R.: Oblivious RAM simulation with efficient worst-case access overhead. In: CCSW, pp. 95–100 (2011)
18. Juels, A., Kaliski Jr., B.S.: Pors: proofs of retrievability for large files. In: Ning, P., di Vimercati, S.D.C., Syverson, P.F. (eds.) ACM CCS 2007, pp. 584–597. ACM Press (October 2007)
19. Küpçü, A.: Efficient Cryptography for the Next Generation Secure Cloud. PhD thesis, Brown University (2010)
20. Küpçü, A.: Efficient Cryptography for the Next Generation Secure Cloud: Protocols, Proofs, and Implementation. Lambert Academic Publishing (2010)
21. Naor, M., Rothblum, G.N.: The complexity of online memory checking. In: 46th FOCS, pp. 573–584. IEEE Computer Society Press (October 2005)
22. Ostrovsky, R., Shoup, V.: Private information storage (extended abstract). In: 29th ACM STOC, pp. 294–303. ACM Press (May 1997)
23. Pinkas, B., Reinman, T.: Oblivious RAM revisited. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 502–519. Springer, Heidelberg (2010)
24. Shacham, H., Waters, B.: Compact proofs of retrievability. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 90–107. Springer, Heidelberg (2008)
25. Shi, E., Chan, T.-H.H., Stefanov, E., Li, M.: Oblivious RAM with $o((\log n)^3)$ worst-case cost. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 197–214. Springer, Heidelberg (2011)
26. Stefanov, E., van Dijk, M., Oprea, A., Juels, A.: Iris: A scalable cloud file system with efficient integrity checks. Cryptology ePrint Archive, Report 2011/585 (2011)
27. Wang, Q., Wang, C., Li, J., Ren, K., Lou, W.: Enabling public verifiability and data dynamics for storage security in cloud computing. In: Backes, M., Ning, P. (eds.) ESORICS 2009. LNCS, vol. 5789, pp. 355–370. Springer, Heidelberg (2009)

Message-Locked Encryption and Secure Deduplication

Mihir Bellare[1], Sriram Keelveedhi[1], and Thomas Ristenpart[2]

[1] Department of Computer Science & Engineering, University of California San Diego
`http://cseweb.ucsd.edu/~mihir/`, `http://cseweb.ucsd.edu/~skeelvee/`
[2] Department of Computer Sciences, University of Wisconsin–Madison
`http://pages.cs.wisc.edu/~rist/`

Abstract. We formalize a new cryptographic primitive that we call Message-Locked Encryption (MLE), where the key under which encryption and decryption are performed is itself derived from the message. MLE provides a way to achieve secure deduplication (space-efficient secure outsourced storage), a goal currently targeted by numerous cloud-storage providers. We provide definitions both for privacy and for a form of integrity that we call tag consistency. Based on this foundation, we make both practical and theoretical contributions. On the practical side, we provide ROM security analyses of a natural family of MLE schemes that includes deployed schemes. On the theoretical side the challenge is standard model solutions, and we make connections with deterministic encryption, hash functions secure on correlated inputs and the sample-then-extract paradigm to deliver schemes under different assumptions and for different classes of message sources. Our work shows that MLE is a primitive of both practical and theoretical interest.

1 Introduction

We introduce an intriguing new primitive that we call Message-Locked Encryption (MLE). An MLE scheme is a symmetric encryption scheme in which the key used for encryption and decryption is itself derived from the message. Instances of this primitive are seeing widespread deployment and application for the purpose of secure deduplication [1, 2, 4, 5, 7, 8, 10, 22, 23, 31, 35, 39, 43], but in the absence of a theoretical treatment, we have no precise indication of what these methods do or do not accomplish.

We provide definitions of privacy and integrity peculiar to this domain. Now having created a clear, strong target for designs, we make contributions that may broadly be divided into two parts: (1) practical and (2) theoretical. In the first category we analyze existing schemes and new variants, breaking some and justifying others with proofs in the random-oracle-model (ROM) [16]. In the second category we address the challenging question of finding a standard-model MLE scheme, making connections with deterministic public-key encryption [11], correlated-input-secure hash functions [27] and locally-computable extractors [9, 30, 40] to provide schemes exhibiting different trade-offs between assumptions made and the message distributions for which security is proven. From our treatment MLE emerges as a primitive that combines practical impact with theoretical depth and challenges, making it well worthy of further study and a place in the cryptographic pantheon. Below we begin with some background and then look more closely at our contributions.

1.1 Background

To save space, commercial cloud storage services such as Google Drive [6], Dropbox [3] and bitcasa [1] perform file-level deduplication across all their users. Say a user Alice stores a file M and Bob requests to store the same file M. Observing that M is already stored, the server, instead of storing a second copy of M, simply updates metadata associated to M to indicate that Bob and Alice both stored M. In this way, no file is stored more than once, moving storage costs for a file stored by u users from $\mathcal{O}(u \cdot |M|)$ to $\mathcal{O}(u + |M|)$ where the big-O notation hides implementation-dependent constants.

However, as users we may want our files to be encrypted. We may not want the storage provider to see our data. Even if we did trust the provider, we may legitimately worry about errant employees or the risk of server compromise by an external adversary. When users themselves are corporations outsourcing their data storage, policy or government regulation may mandate encryption.

Conventional encryption, however, makes deduplication impossible. Say Alice stores not her file M but its encryption C_A under her password pw_A. Bob would store C_B, the encryption of M under his password pw_B. Two issues arise: (1) how the server is to detect that the data underlying the two ciphertexts is the same, and (2) even if it can so detect, what can it store short of (C_A, C_B) that allows both parties, based on their separate respective passwords, to recover the data from what is stored. Standard IND-CPA encryption means even (1) is not possible. We might use some kind of searchable encryption [11,20,38] but it is still not clear how to solve (2). Just storing Alice's ciphertext, for example, does not work because Bob cannot later decrypt it to recover the file, and visa versa.

Douceur et. al. (DABST) [24] proposed a clever solution called convergent encryption (CE). Alice derives a key $K = H(M)$ from her message M and then encrypts the message as $C = E(K, M) = E(H(M), M)$, where H is a cryptographic hash function and E is a block cipher. (They assume the message is one block long.) The ciphertext is given to the server and the user retains K. Since encryption is deterministic, if Bob starts from the same message he would produce the same key and ciphertext. The server can now perform deduplication on the ciphertext C, checking, when it receives C, whether or not it is already stored, and, if the latter, as before, not re-storing but instead updating metadata to indicate an additional owner. Both Alice and Bob can decrypt C since both have the same key K.

These ideas have been attractive enough to see significant usage, with CE or variants deployed in [1,2,4,5,8,31,35,39,43]. It is not however clear what precisely is the underlying security goal and whether deployed schemes achieve it.

1.2 Definitions and Relations

We introduce Message-Locked Encryption (MLE) —so named because the message is locked, as it were, under itself— with the goal of providing an encryption primitive that provably enables secure deduplication.

SYNTAX. As depicted in Fig. 2, the key generation algorithm of an MLE scheme $\mathcal{K}$ maps a message M to a key K. The encryption algorithm $\mathcal{E}$ takes input the key K and a message M and produces a ciphertext C. The decryption algorithm $\mathcal{D}$ allows recovery of M from C given the key K. The tagging algorithm $\mathcal{T}$ maps the ciphertext C to a tag T used by the server to detect duplicates. (Tag correctness requires that tags corresponding to messages M_1, M_2 are likely to be the same iff M_1, M_2 are the same.) All algorithms may depend on a parameter P but the latter is public and common to all parties including the adversary, and thus is not a key.

Any MLE scheme enables deduplication of ciphertexts. CE is captured by our syntax as the MLE scheme that lets $K = H(M)$, $C = E(K, M)$ and tag $T = H(C)$.

MLE is trivially achieved by letting the key K equal the message M. (Set $C = T = \varepsilon$ to the empty string and have decryption simply return the key.) This degenerate solution is however useless for deduplication since the client stores as K the entire file and no storage savings result. We rule it out by requiring that keys be shorter than messages, ideally keys are of a fixed, short length.

PRIVACY. No MLE scheme can achieve semantic-security-style privacy in the spirit of [13,26]. Indeed, if the target message M is drawn from a space S of size s then an adversary, given an encryption C of M, can recover M in $\mathcal{O}(s)$ trials. (For each candidate $M' \in S$ test whether $\mathcal{D}(\mathcal{K}(M'), C) = M'$ and if so return M'.) As with deterministic public-key encryption [11], we therefore ask for the best possible privacy, namely semantic security when messages are unpredictable (have high min-entropy). Adapting definitions from [11, 12, 14, 21] we formalize a PRV-CDA notion where encryptions of two unpredictable messages should be indistinguishable. ("cda" stands for "chosen-distribution attack" [12].) We also formalize a stronger PRV\$-CDA notion where the encryption of an unpredictable message must be indistinguishable from a random string of the same length (cf. [37]).

These basic notions are for non-adaptive adversaries. The corresponding adaptive versions are PRV-CDA-A and PRV\$-CDA-A. We show that PRV-CDA does not imply PRV-CDA-A but, interestingly, that PRV\$-CDA does imply PRV\$-CDA-A. (See the right hand side of Fig. 1 for a comprehensive relations summary.) Thus PRV\$-CDA emerges as the preferred target for designs because non-adaptive security is easier to prove yet adaptive security is implied.

TAG CONSISTENCY. Suppose client Alice has a message M_A and client Bob has a different message M_B. Alice is malicious and uploads not an honest encryption of M_A but a maliciously-generated ciphertext C_A such that, when Bob tries to upload C_B, the server sees a tag match $\mathcal{T}(C_A) = \mathcal{T}(C_B)$. (This does not contradict the correctness requirement that tags are usually equal iff the messages are equal because that holds for honestly-generated ciphertexts.) The server thus keeps only C_A, deleting C_B. Yet later, when Bob downloads to get C_A, the decryption is M_A, not M_B, meaning the integrity of his data has been compromised.

This is a serious concern, and not mere speculation, for such "duplicate-faking" attacks have been found on some CE variants [39]. We define tag consistency to rule out these types of integrity violations. Notion TC asks that it be hard to create (M, C) such that $\mathcal{T}(C) = \mathcal{T}(\mathcal{E}(\mathcal{K}(M), M))$ but $\mathcal{D}(\mathcal{K}(M), C)$ is a string different from M. In words, an adversary cannot make an honest client recover an incorrect message, meaning one different from the one it uploaded. Notion STC ("S" for "strong") asks that it additionally be hard to create (M, C) such that $\mathcal{T}(C) = \mathcal{T}(\mathcal{E}(\mathcal{K}(M), M))$ but $\mathcal{D}(\mathcal{K}(M), C) = \perp$, meaning an adversary cannot erase an honest client's message. STC is strictly stronger than TC; we define both because, as we will see, some schemes meet only the weaker, but still meaningful, TC version.

1.3 Practical Contributions

The definitional framework outlined above puts us in a position to rigorously assess —a decade after its inception in [24]— the security of convergent encryption (CE). The task is complicated by the presence and deployment of numerous variants of the basic CE idea. We address this by formulating two MLE schemes, that we call CE and $\mathsf{HCE1}$, that represent two major variants of CE and between them capture the prominent existing schemes. They each make use of a RO hash function H and a deterministic symmetric encryption scheme SE. CE with SE set to a blockcipher, for example, is the scheme of [24] and $\mathsf{HCE1}$ with SE as a blockcipher in counter mode with fixed IV is used within the Tahoe FileSystem (TahoeFS) [43].

CE sets $K = H(M)$, $C = \mathsf{SE}(K, M)$ and tag $T = H(C)$, while $\mathsf{HCE1}$ sets $K = H(M)$, $C = \mathsf{SE}(K, M) \| H(K)$ and $T = H(K)$. The rationale for $\mathsf{HCE1}$ is to offer better performance for the server who can simply read the tag as the second part of the ciphertext rather than needing to compute it by hashing the possibly long ciphertext. But we observe that $\mathsf{HCE1}$ is vulnerable to duplicate faking attacks, meaning it does not even achieve TC security. We discuss the implications for the security of TahoeFS in Section 4.

We ask whether performance gains of the type offered by $\mathsf{HCE1}$ over CE can be obtained without loss in consistency, and offer as answers two new schemes, $\mathsf{HCE2}$ and RCE. The former is as efficient as $\mathsf{HCE1}$. RCE however is even more efficient, needing just one concerted pass over the data to generate the key, encrypt the message and produce the tag. On the other hand, $\mathsf{HCE2}$ needs two passes, one pass to generate the key and a second for encryption, while CE needs a third pass for producing the tag. RCE achieves this via a novel use of randomization (all previous schemes were deterministic). Roughly, encryption picks a fresh random key L and then computes $\mathsf{SE}(L, M)$ and $K = H(M)$ in the same pass, finally placing an encryption of L under K, together with an appropriate tag, in the ciphertext. We have implemented all three schemes and the results [15] show that RCE does indeed outperform the other two.

Fig. 1 (table, first four rows) summarizes the findings of our security analysis of the four schemes. Under standard assumptions on the deterministic symmetric encryption scheme SE (one-time real-or-random ciphertext, or ROR, security as well as key-recovery security) and with H a RO, we show that all four MLE schemes meet our strong privacy notion PRV\$-CDA. The consistency findings are more involved. As mentioned, HCE1 provides no tag consistency. The good news is that CE, HCE2 and RCE all achieve TC security, so that an adversary cannot make a client recover a file different from the one she uploaded. But only CE offers STC security, implying that the reduction in server cost offered by HCE1, HCE2 and RCE comes at a price, namely loss of STC-security. The conclusion is that designers will need to trade performance for strong tag consistency. Whether this is fundamental or if better schemes exist is an interesting open question.

1.4 Theoretical Contributions

Is MLE possible in the standard-model? This emerges as the natural and most basic theoretical question in this domain. Another question is, how does MLE relate to other (existing) primitives? MLE has in common with Deterministic Public-Key Encryption (D-PKE) [11] and Correlated-input-secure Hash Functions (CI-H) [27] a goal of privacy on unpredictable but possibly related inputs, so it is in particular natural to ask about the relation of MLE to these primitives. The two questions are related, for showing that a primitive X implies MLE yields a construction of an MLE scheme based on X. In exploring these questions it is instructive to distinguish between D-MLE (where encryption is deterministic) and R-MLE (where encryption may be randomized). The connections we now discuss are summarized by the picture on the right side of Fig. 1:

- D-PKE $\Rightarrow$ D-MLE: We show how to construct an MLE scheme from any D-PKE scheme that is PRIV-secure in the sense of [11]. The first idea that may come to mind is to make public a public key pk for the D-PKE scheme DE and MLE-encrypt M as $\mathsf{DE}(pk, M)$. But this does not make sense because sk is needed to decrypt and the latter is not derived from M. Our XtDPKE ("e$\underline{\text{x}}$tract-$\underline{\text{t}}$hen-$\underline{\text{D}}$-$\underline{\text{PKE}}$") solution, described in Section 5, is quite different and does not exploit the decryptability of DE at all. We apply a strong randomness extractor to M to get the MLE key K and then encrypt M bit-by-bit, the encryption of the i-th bit $M[i]$ being $C[i] = \mathsf{DE}(pk, K\|i\|M[i])$. Decryption, given K, is done by re-encrypting, for each i, both possible values of the i-th message bit and seeing which ciphertext matches $C[i]$. We assume a trusted generation of pk in which nobody retains sk. XtDPKE has PRV-CDA privacy and provides STC (strong) tag consistency.
- CI-H $\Leftrightarrow$ D-MLE: Our XtCIH ("e$\underline{\text{x}}$tract-$\underline{\text{t}}$hen-$\underline{\text{CI}}$-$\underline{\text{H}}$ash") scheme derives a D-MLE scheme from any CI-H hash function [27] by using the latter in place of the D-PKE scheme in the above. XtCIH is PRV\$-CDA private while retaining STC consistency. Conversely, any PRV\$-CDA D-MLE scheme can be used to construct a CI-H hash function, making the primitives equivalent.

Scheme	Model	D/R	Privacy		Integrity	
			PRV-CDA	PRV$-CDA	TC	STC
CE	RO	D	✓	✓	✓	✓
HCE1	RO	D	✓	✓	✗	✗
HCE2	RO	D	✓	✓	✓	✗
RCE	RO	R	✓	✓	✓	✗
XtCIH	STD	D	✓	✓	✓	✓
XtDPKE	STD	D	✓	✗	✓	✓
XtESPKE	STD	R	✓	✗	✓	✓
SXE	STD	D	✓	✓	✓	✓

$$\begin{tikzcd}
\text{CI-H} \arrow[d, shift left] & & \\
\begin{array}{c}\text{D-MLE}\\ \text{(PRV\$-CDA)}\end{array} \arrow[u, shift left] \arrow[rr, bend left] \arrow[d] & \text{D-PKE} \arrow[dl] \arrow[d] & \begin{array}{c}\text{R-MLE}\\ \text{(PRV\$-CDA)}\end{array} \arrow[d] \\
\begin{array}{c}\text{D-MLE}\\ \text{(PRV-CDA)}\end{array} \arrow[rr, bend right] & \text{ES-PKE} \arrow[r] & \begin{array}{c}\text{R-MLE}\\ \text{(PRV-CDA)}\end{array}
\end{tikzcd}$$

Fig. 1. Left: For each MLE scheme that we construct, we indicate whether it is in the RO or standard model; whether it is deterministic or randomized; and which security properties it is proven to possess. The assumptions for XtCIH, XtDPKE and XtESPKE are, respectively, a CI-H function, a D-PKE scheme and an ES-PKE scheme, while the others assume only a symmetric encryption scheme. **Right:** An arrow X → Y means we can construct primitive Y from primitive X. Dark arrows are our results while light arrows indicate trivial or known implications.

We believe these results are interesting as connections between prominent primitives. However, they do not, right now, yield MLE schemes under standard assumptions because providing the required D-PKE schemes or CI-H functions under such assumptions is still open and deemed challenging. Indeed, Wichs [41] shows that secure D-PKE schemes or CI-H functions may not be obtained via blackbox reductions from any assumption that may be modeled as a game between an adversary and a challenger. We note that his result applies to D-MLE as well but, as far as we can tell, not to R-MLE. One potential route to MLE with standard assumptions may thus be to exploit randomization but we are unaware of how to do this beyond noting that XtDPKE extends to a R-MLE scheme XtESPKE based on any ES-PKE (Efficiently Searchable PKE) scheme [11], a weaker primitive than D-PKE.

In the D-PKE domain, progress was made by restricting attention to special message distributions. In particular D-PKE under standard assumptions have been achieved for independent messages or block sources [14, 19, 21, 25]. CI-H functions have been built for messages given by polynomials evaluated at the same random point [27]. It is thus natural to ask whether we can obtain MLE under standard assumptions for special message distributions. One might think that this follows from our D-PKE ⇒ D-MLE and CI-H ⇒ D-MLE constructions and the known results on D-PKE and CI-H, but this is not the case because our constructions do not preserve the message distribution.

The final contribution we mention here is MLE schemes under standard assumptions for certain classes of message distributions . Our SXE (Sample-extract-encrypt) MLE scheme is inspired by locally-computable extractors [9,30,40] and the sample-then-extract paradigm [33, 40]. The idea is to put a random subset of the message bits through an extractor to get a key used to encrypt the rest of the bits, and the only assumption made is a standard, ROR-secure symmetric encryption scheme.

1.5 Further Remarks and Related Work

There are folklore suggestions along the lines of CE predating [24]. See [34].

Recall that we have introduced an indistinguishability-from-random notion (PRV\$-CDA) for MLE and showed that it implied its adaptive counterpart. This is of broader interest for the parent settings of deterministic and hedged encryption. Here achieving adaptive security has been challenging [12]. We suggest that progress can be made by defining and then targeting indistinguishability-from-random style definitions.

Mironov, Pandey, Reingold and Segev [32] suggest deduplication as a potential application of their incremental deterministic public-key encryption scheme. But this will only work with a single client. It won't allow deduplication across clients, since they would all have to share the secret key.

Recent work showed that client-side deduplication gives rise to side-channel attacks because users are told if another user already uploaded a file [29]. MLE is compatible with either client- or server-side deduplication (the latter prevents such side-channels). We note that one of our new schemes, RCE, gives rise to such a side-channel (see Section 4). MLE targets a different class of threats than proofs of ownership [28], which were proposed for deduplication systems in order to mitigate abuse of services for surreptitious content distribution.

In independent and concurrent work, Xu, Chang and Zhou [44] consider leakage resilience in the deduplication setting. They provide a randomized construction similar to RCE.

2 Preliminaries

Notations and Conventions. The empty string is denoted by ε. If $\mathbf{x}$ is a vector then $|\mathbf{x}|$ denotes the number of components in $\mathbf{x}$, $\mathbf{x}[i]$ denotes the i-th component, and $\mathbf{x}[i,j] = \mathbf{x}[i]\ldots\mathbf{x}[j]$ for $1 \le i \le j \le |\mathbf{x}|$. A (binary) string x is identified with a vector over $\{0,1\}$ so that $|x|$ is its length, $x[i]$ is its i-th bit and $x[i,j] = x[i]\ldots x[j]$ for $1 \le i \le j \le |x|$. If S is a finite set then $|S|$ denotes its size and $s \leftarrow\$ S$ denotes picking an element uniformly from S and assigning it to s. For $i \in \mathbb{N}$ we let $[i] = \{1,\ldots,i\}$. We denote by $\lambda \in \mathbb{N}$ the security parameter and by 1^λ its unary representation.

"PT" stands for "polynomial-time." Algorithms are randomized unless otherwise indicated. By $y \leftarrow A(x_1,\ldots;R)$, we denote the operation of running algorithm A on inputs $x_1,\ldots$ and coins R and letting y denote the output. By $y \leftarrow\$ A(x_1,\ldots)$, we denote the operation of letting $y \leftarrow A(x_1,\ldots;R)$ with R chosen at random. We denote by $[A(x_1,\ldots)]$ the set of points that have positive probability of being output by A on inputs $x_1,\ldots$. Adversaries are algorithms or tuples of algorithms. In the latter case, the running time of the adversary is the sum of the running times of all the algorithms in the tuple.

The guessing probability $\mathbf{GP}(X)$ and min-entropy $\mathbf{H}_\infty(X)$ of a random variable X are defined via $\mathbf{GP}(X) = \max_x \Pr[X = x] = 2^{-\mathbf{H}_\infty(X)}$. The conditional guessing probability $\mathbf{GP}(X\,|\,Y)$ and conditional min-entropy $\mathbf{H}_\infty(X\,|\,Y)$ of a random variable X given a random variable Y are defined via

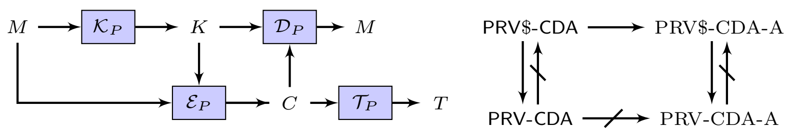


Fig. 2. Left: Depiction of syntax of MLE scheme $\mathsf{MLE} = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{T})$. The parameter generation algorithm is not shown. **Right:** Relations between notions of privacy for MLE schemes. An arrow from A to B means that any A-secure MLE scheme is also B-secure. A barred arrow means there is an A-secure MLE scheme that is not B-secure.

$\mathbf{GP}(X \mid Y) = \sum_y \Pr[Y = y] \cdot \max_x \Pr[X = x|Y = y] = 2^{-\mathbf{H}_\infty(X \mid Y)}$. By $\mathsf{SD}(X;Y)$ we denote the statistical distance between random variables X and Y. For our security definitions and proofs we use the code-based game playing framework of [17], though adopting some of the syntax and semantics of [36].

3 Message-Locked Encryption

Syntax and correctness. An MLE scheme $\mathsf{MLE} = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{T})$ is a five-tuple of PT algorithms, the last two deterministic — see Fig. 2. On input 1^λ the parameter generation algorithm $\mathcal{P}$ returns a public parameter P. On input P and a message M, the key-generation algorithm $\mathcal{K}$ returns a message-derived key $K \leftarrow\!\!\$\, \mathcal{K}_P(M)$. On inputs P, K, M the encryption algorithm $\mathcal{E}$ returns a ciphertext $C \leftarrow\!\!\$\, \mathcal{E}_P(K, M)$. On inputs P, K and a ciphertext C, the decryption algorithm $\mathcal{D}$ returns $\mathcal{D}_P(K, C) \in \{0,1\}^* \cup \{\bot\}$. On inputs P, C the tag generation algorithm returns a tag $T \leftarrow \mathcal{T}_P(C)$. Associated to the scheme is a *message space* $\mathrm{MsgSp}_{\mathsf{MLE}}$ that associates to any $\lambda \in \mathbb{N}$ a set $\mathrm{MsgSp}_{\mathsf{MLE}}(\lambda) \subseteq \{0,1\}^*$. We require that there is a function Cl such that, for all $\lambda \in \mathbb{N}$, all $P \in [\mathcal{P}(1^\lambda)]$ and all $M \in \{0,1\}^*$, any output of $\mathcal{E}_P(\mathcal{K}_P(M), M)$ has length $\mathsf{Cl}(P, \lambda, |M|)$, meaning the length of a ciphertext depends on nothing about the message other than its length. The *decryption correctness* condition requires that $\mathcal{D}_P(K, C) = M$ for all $\lambda \in \mathbb{N}$, all $P \in [\mathcal{P}(1^\lambda)]$, all $M \in \mathrm{MsgSp}_{\mathsf{MLE}}(\lambda)$, all $K \in [\mathcal{K}_P(M)]$ and all $C \in [\mathcal{E}_P(K, M)]$. The *tag correctness* condition requires that there is a negligible function $\delta\colon \mathbb{N} \to [0,1]$, called the false negative rate, such that $\Pr[\mathcal{T}_P(C) \neq \mathcal{T}_P(C')] \leq \delta(\lambda)$ for all $\lambda \in \mathbb{N}$, all $P \in [\mathcal{P}(1^\lambda)]$ and all $M \in \mathrm{MsgSp}_{\mathsf{MLE}}(\lambda)$, where the probability is over $C \leftarrow\!\!\$\, \mathcal{E}_P(\mathcal{K}_P(M), M)$ and $C' \leftarrow\!\!\$\, \mathcal{E}_P(\mathcal{K}_P(M), M)$. We say that MLE is deterministic if $\mathcal{K}$ and $\mathcal{E}$ are deterministic. We observe that if MLE is deterministic then it has perfect tag correctness, meaning a false negative rate of 0.

Discussion. In the application to secure deduplication, the server publishes P and maintains a database that we view as a table Da, initially everywhere $\bot$. In the UPLOAD protocol, the client, having P, M, computes $K \leftarrow\!\!\$\, \mathcal{K}_P(M)$ and $C \leftarrow\!\!\$\, \mathcal{E}_P(K, M)$. The client stores K securely. (It may do so locally or store K encrypted under its password on the server, but the implementation is not relevant here.) It sends C to the server. The latter computes $T \leftarrow \mathcal{T}_P(C)$. If $\mathsf{Da}[T] = \bot$

then it lets $\mathsf{Da}[T] \leftarrow C$. The server provides the client with a filename or pointer that we may, for simplicity, just view as the tag T. In the DOWNLOAD protocol, the client sends the server a tag T and the server returns $\mathsf{Da}[T]$. If Alice uploads M and Bob later does the same, tag correctness means that their tags will most likely be equal and the server will store a single ciphertext on their behalf. Downloads will return to both this common ciphertext C, and decryption correctness guarantees that both can decrypt C under their respective (although possibly different) keys to recover M.

A trivial construction of an MLE scheme $\mathsf{MLE} = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{T})$ may be obtained by setting the key to the message. In more detail, let $\mathcal{P}(1^\lambda) = \varepsilon$; let $\mathcal{K}_\varepsilon(M) = M$; let $\mathcal{E}_\varepsilon(M, M) = \mathcal{T}_\varepsilon(C) = \varepsilon$; let $\mathcal{D}_\varepsilon(M, C) = M$. This will meet the decryption and tag correctness conditions besides meeting the security requirements (privacy and tag consistency) we will formalize below. However, this scheme is of no use for deduplication because the client stores the entire file as the key and no storage savings are gleaned. To avoid this kind of degenerate scheme, we insist that an MLE scheme have keys that are shorter than the message. Formally, there must be a constants $c, d < 1$ such that the function that on input $\lambda \in \mathbb{N}$ returns $\max_{P,M} \Pr[|\mathcal{K}_P(M)| > d \cdot |M|^c]$ is negligible where the probability is over the choices of $\mathcal{K}$ and the maximum is over all $P \in [\mathcal{P}(1^\lambda)]$ and all $M \in \mathrm{MsgSp}_{\mathsf{MLE}}(\lambda)$. Particular schemes we construct or analyze, however, do much better, with the key-length for most of them depending only on the security parameter.

Our formulation of search via tag comparison enables fast search: the server can use the tag to index directly into a table or perform a logarithmic-time binary search as in [11]. These requirements could be relaxed to define MLE variants where search was allowed linear time (cf. [20]) or search ability was not even provided. MLE does not appear easy to achieve even in the last case.

Privacy. A *source* is a PT algorithm $\mathcal{M}$ that on input 1^λ returns $(\mathbf{M}_0, \ldots, \mathbf{M}_{n-1}, Z)$ where $\mathbf{M}_0, \ldots, \mathbf{M}_{n-1}$ are vectors over $\{0,1\}^*$ and $Z \in \{0,1\}^*$. Here $n \geq 1$ is a constant called the arity of the source. (We will only consider $n \in \{1, 2\}$.) We require that all the vectors have the same length $m(\lambda)$ for some function m called the number of messages of the source. We require that there is a function len, called the message length of the source, such that the string $\mathbf{M}_j[i]$ has length $\mathsf{len}(\lambda, i)$ for all $i \in [m(\lambda)]$ and all $j \in \{0, \ldots, n-1\}$. We require that $\mathbf{M}_j[i_1] \neq \mathbf{M}_j[i_2]$ for all distinct $i_1, i_2 \in [m(\lambda)]$ and all $j \in \{0, \ldots, n-1\}$, meaning the entries of each vector are distinct. We refer to Z as the auxiliary information. The guessing probability $\mathbf{GP}_\mathcal{M}$ of source $\mathcal{M}$ is defined as the function which on input $\lambda \in \mathbb{N}$ returns $\max_{i,j} \mathbf{GP}(\mathbf{M}_j[i] \mid Z)$ where the probability is over $(\mathbf{M}_0, \ldots, \mathbf{M}_{n-1}, Z) \leftarrow\!\!{\scriptstyle\$}\; \mathcal{M}(1^\lambda)$ and the maximum is over all $i \in [m(\lambda)]$ and all $j \in \{0, \ldots, n-1\}$. We say that $\mathcal{M}$ is *unpredictable* if $\mathbf{GP}_\mathcal{M}(\cdot)$ is negligible. (Meaning, messages are unpredictable given the auxiliary information. We do *not* require that the components $\mathbf{M}_j[1], \ldots, \mathbf{M}_j[m(\lambda)]$ of a vector are independent, just that each, individually, is unpredictable.) We refer to $-\log(\mathbf{GP}_\mathcal{M}(\cdot))$ as the min-entropy of the source. We say that $\mathcal{M}$ is MLE-valid

```
main $\mathrm{PRV\text{-}CDA}^A_{\mathsf{MLE},\mathcal{M}}(\lambda)$
$P \leftarrow\!\!\$\ \mathcal{P}(1^\lambda)$
$b \leftarrow\!\!\$\ \{0,1\}$
$(\mathbf{M}_0, \mathbf{M}_1, Z) \leftarrow\!\!\$\ \mathcal{M}(1^\lambda)$
For $i = 1, \ldots, |\mathbf{M}_b|$ do
    $\mathbf{K}[i] \leftarrow\!\!\$\ \mathcal{K}_P(\mathbf{M}_b[i])$
    $\mathbf{C}[i] \leftarrow\!\!\$\ \mathcal{E}_P(\mathbf{K}[i], \mathbf{M}_b[i])$
$b' \leftarrow\!\!\$\ A(P, \mathbf{C}, Z)$
Ret $(b = b')$
```

```
main $\mathrm{PRV\$\text{-}CDA}^A_{\mathsf{MLE},\mathcal{M}}(\lambda)$
$P \leftarrow\!\!\$\ \mathcal{P}(1^\lambda)$
$b \leftarrow\!\!\$\ \{0,1\}$
$(\mathbf{M}, Z) \leftarrow\!\!\$\ \mathcal{M}(1^\lambda)$
For $i = 1, \ldots, |\mathbf{M}|$ do
    $\mathbf{K}[i] \leftarrow\!\!\$\ \mathcal{K}_P(\mathbf{M}[i])$
    $\mathbf{C}_1[i] \leftarrow\!\!\$\ \mathcal{E}_P(\mathbf{K}[i], \mathbf{M}[i])$
    $\mathbf{C}_0[i] \leftarrow\!\!\$\ \{0,1\}^{|\mathbf{C}_1[i]|}$
$b' \leftarrow\!\!\$\ A(P, \mathbf{C}_b, Z)$
Ret $(b = b')$
```

```
main $\boxed{\mathrm{TC}^A_{\mathsf{MLE}}(\lambda)}$ $\mathrm{STC}^A_{\mathsf{MLE}}(\lambda)$
$P \leftarrow\!\!\$\ \mathcal{P}(1^\lambda)$; $(M, C') \leftarrow\!\!\$\ A(P)$
If $(M = \bot)$ or $(C' = \bot)$ then
    Ret false
$T \leftarrow\!\!\$\ \mathcal{T}_P(\mathcal{E}_P(\mathcal{K}_P(M), M))$
$T' \leftarrow\!\!\$\ \mathcal{T}_P(C')$
$M' \leftarrow\!\!\$\ \mathcal{D}_P(\mathcal{K}_P(M), C')$
If $(M = M')$ then Ret false
If $(T \neq T')$ then Ret false
$\boxed{\text{If } (M' = \bot) \text{ then Ret false}}$
Ret true
```

Fig. 3. Games defining PRV-CDA, PRV\$-CDA privacy and TC, STC tag consistency security of MLE scheme $\mathsf{MLE} = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{T})$

if $\mathbf{M}_j[i] \in \mathrm{MsgSp}_{\mathsf{MLE}}(\lambda)$ for all $\lambda \in \mathbb{N}$, all $(\mathbf{M}_0, \ldots, \mathbf{M}_{n-1}, Z) \in [\mathcal{M}(1^\lambda)]$, all $i \in [m(\lambda)]$ and all $j \in \{0, \ldots, n-1\}$.

In the games of Fig. 3, "CDA" stands for "Chosen-Distribution Attack," referring to the distribution on messages imposed by the MLE-valid source $\mathcal{M}$, which in game PRV-CDA has arity 2 and in game PRV\$-CDA has arity 1. If A is an adversary we let $\mathbf{Adv}^{\mathsf{prv\text{-}cda}}_{\mathsf{MLE},\mathcal{M},A}(\lambda) = 2 \cdot \Pr[\mathrm{PRV\text{-}CDA}^A_{\mathsf{MLE},\mathcal{M}}(\lambda)] - 1$ and $\mathbf{Adv}^{\mathsf{prv\$\text{-}cda}}_{\mathsf{MLE},\mathcal{M},A}(\lambda) = 2 \cdot \Pr[\mathrm{PRV\$\text{-}CDA}^A_{\mathsf{MLE},\mathcal{M}}(\lambda)] - 1$. We say that MLE is PRV-CDA (resp. PRV\$-CDA) secure over a class $\overline{\mathcal{M}}$ of PT, MLE-valid sources if $\mathbf{Adv}^{\mathsf{prv\text{-}cda}}_{\mathsf{MLE},\mathcal{M},A}(\cdot)$ (resp. $\mathbf{Adv}^{\mathsf{prv\$\text{-}cda}}_{\mathsf{MLE},\mathcal{M},A}(\cdot)$) is negligible for all PT A and all $\mathcal{M} \in \overline{\mathcal{M}}$. We say that MLE is PRV-CDA (resp. PRV\$-CDA) secure if it is PRV-CDA (resp. PRV\$-CDA) secure over the class of all PT, unpredictable MLE-valid sources. PRV-CDA asks for indistinguishability of encryptions of two unpredictable messages and is based on formalizations of deterministic [11,14,21] and hedged [12] PKE. PRV\$-CDA is a new variant, asking for the stronger property that encryptions of unpredictable messages are indistinguishable from random strings, an adaption to this setting of the corresponding notion for symmetric encryption from [37].

The source is not given the parameter P as input, meaning privacy is only assured for messages that do not depend on the parameter. This is analogous to the restriction that messages do not depend on the public key in D-PKE [11], and without this restriction, privacy is not possible. However, the adversary A does get the parameter.

The notions here are non-adaptive in the sense that the distribution of the next message does not depend on the previous ciphertext. In the full version [15], we give corresponding adaptive definitions PRV-CDA-A and PRV\$-CDA-A, and prove the relations summarized in Fig. 2. The one we highlight is that non-adaptive PRV\$-CDA implies its adaptive counterpart. This is not true for PRV-CDA and makes PRV\$-CDA preferable to achieve.

Tag consistency. Consider the games of Fig. 3 and let A be an adversary. Game $\mathrm{TC}_{\mathsf{MLE}}$ includes the boxed statement, while $\mathrm{STC}_{\mathsf{MLE}}$ does not. We let $\mathbf{Adv}^{\mathsf{TC}}_{\mathsf{MLE},A}(\lambda) = \Pr[\mathrm{TC}^{A}_{\mathsf{MLE}}(\lambda)]$ and $\mathbf{Adv}^{\mathsf{STC}}_{\mathsf{MLE},A}(\lambda) = \Pr[\mathrm{STC}^{A}_{\mathsf{MLE}}(\lambda)]$. We say that MLE is TC (resp. STC) secure if $\mathbf{Adv}^{\mathsf{TC}}_{\mathsf{MLE},A}(\cdot)$ (resp. $\mathbf{Adv}^{\mathsf{STC}}_{\mathsf{MLE},A}(\cdot)$) is negligible.

Tag consistency (TC) aims to provide security against duplicate faking attacks in which a legitimate message is undetectably replaced by a fake one. In such an attack we imagine the adversary A creating and uploading C'. Later, an honest client, holding M (the formalism allows A to pick M) computes $K \leftarrow\!\!{\scriptstyle\$}\, \mathcal{K}_P(M)$ and uploads $C \leftarrow\!\!{\scriptstyle\$}\, \mathcal{E}_P(K, M)$. The server finds that the tags of C and C' are equal and thus continues to store only C'. Later, the honest client downloads C' and decrypts under K. It expects to recover M, but in a successful duplicate-faking attack it recovers instead some message $M' \neq M$. The integrity of its data has thus been violated. TC security protects against this. Note that TC explicitly excludes an attack in which $M' = \perp$. Thus TC secure schemes may still admit duplicate faking attacks that lead to erasures: a client can detect corruption but no longer be able to recover their message. STC (strong tag consistency) aims to additionally provide security against such erasure attacks. In terms of implications, STC implies TC but TC does not imply STC.

Duplicate faking attacks are not just a theoretical concern. They were first discussed in [39], yet currently deployed schemes are still vulnerable, as we'll see in the next section. Discussions with practitioners suggest that security against them is viewed as an important requirement in practice.

Given any TC secure scheme, we can prevent all of the attacks above by having a client, upon being informed that her ciphertext is already stored, download it immediately and check that decryption yields her message. If not, she complains. This however is not optimal, being expensive and complex and leading to deduplication side-channels (cf. [29]).

If an MLE scheme is deterministic, letting the tag equal the ciphertext will result in a scheme that is STC secure. This provides a relatively easy way to ensure resistance to duplicate faking attacks, but the price paid is that the tag is as long as the ciphertext. CR-hashing the ciphertext (still for a D-MLE scheme) preserves STC, but for efficiency other, less effective options have been employed in practice, as we will see.

ROM. An RO [16] is a game procedure H that maintains a table $\mathtt{H}[\cdot,\cdot]$, initially everywhere $\perp$. Given a query x, k with $x \in \{0,1\}^*$ and $k \in \mathbb{N}$, it executes: If $\mathtt{H}[x,k] = \perp$ then $\mathtt{H}[x,k] \leftarrow\!\!{\scriptstyle\$}\, \{0,1\}^k$. It then returns $\mathtt{H}[x,k]$. We will omit the length k when it is clear from context. In the ROM, both scheme algorithms and adversary algorithms will have access to H.

In lifting the privacy definitions to the ROM, we do not give the source access to H. This is to simplify our proofs. Our methods and proofs can be extended to handle sources with access to H under an extension of the definition of unpredictability to this setting in which, following [12, 36], the unpredictability of the source is independent of the coins underlying H.

4 The Security of Fast MLE Schemes

We investigate four MLE schemes, two that correspond to in-use schemes and two new schemes.

Ingredients. The schemes are built from a one-time symmetric encryption scheme and a hash function family $\mathsf{H} = (\mathcal{HK}, \mathcal{H})$. The former is a tuple of algorithms $\mathsf{SE} = (\mathcal{SK}, \mathcal{SE}, \mathcal{SD})$: key generation $\mathcal{SK}$, on input 1^λ, outputs a key K of length $k(\lambda)$; deterministic encryption $\mathcal{SE}$ maps a key K and plaintext M to a ciphertext C; and deterministic decryption $\mathcal{SD}$ maps a key K and ciphertext C to a message M. We require that $\Pr[\mathcal{SD}(K, \mathcal{SE}(K, M)) = M] = 1$ for all $\lambda \in \mathbb{N}$, all $K \in [\mathcal{SK}(1^\lambda)]$, and all $M \in \{0,1\}^*$. We assume that there exists a function $\mathrm{cl}_{\mathsf{SE}}$ such that for all $\lambda \in \mathbb{N}$ and all $M \in \{0,1\}^*$ any output of $\mathcal{SE}(K, M)$ has length $\mathrm{cl}_{\mathsf{SE}}(\lambda, |M|)$. For simplicity we assume that $\mathcal{H}$ and SE are compatible: $\mathcal{H}(K_h, M)$ outputs a message of length $k(\lambda)$ for any $K_h \in [\mathcal{HK}(1^\lambda)]$. We require schemes that provide both key recovery security (KR) and one-time real-or-random security (ROR) [37]. Formal definitions are recalled in [15].

The four schemes. Let $\mathsf{SE} = (\mathcal{SK}, \mathcal{SE}, \mathcal{SD})$ be a symmetric encryption scheme and $\mathsf{H} = (\mathcal{HK}, \mathcal{H})$ be a hash function family. All schemes inherit their message space from SE (typically $\{0,1\}^*$), use as parameter generation $\mathcal{HK}$, and share a common key generation algorithm $\mathcal{K}$ which derives keys as $K \leftarrow \mathcal{H}(P, M)$.

The first scheme, that we simply call convergent encryption (CE), generalizes the original scheme of DABST [24]. CE encrypts the message as $C \leftarrow \mathcal{SE}(K, M)$. Tags are computed as $T \leftarrow \mathcal{H}(P, C)$. (One could alternatively use the ciphertext itself as the tag, but this is typically not practical.) Decryption returns $M \leftarrow \mathcal{SD}(K, C)$ on input K, C. The second scheme, $\mathsf{HCE1}$ (Hash-and-CE 1), is a popular variant of the CE scheme used in a number of systems [4, 22, 23, 43]. Compared to CE, $\mathsf{HCE1}$ computes tags during encryption by hashing the per-message key ($T \leftarrow \mathcal{H}(P, K)$) and including the result in the ciphertext. Tag generation just extracts this embedded tag. This offloads work from the server to the client and reduces the number of passes needed to encrypt and generate a tag from three to two.

$\mathsf{HCE1}$ is vulnerable to attacks that break TC security, as first discussed in [39]. The attack is straightforward: adversary A chooses two messages $M \neq M'$, computes $C \leftarrow \mathsf{SE}(\mathcal{H}(P, M), M')$ and $T \leftarrow \mathcal{H}(P, \mathcal{H}(P, M))$, and finally outputs $(M, C \,\|\, T)$. This means an adversary, given knowledge of a user's to-be-stored message, can undetectably replace it with any arbitrary message. In TahoeFS's use of $\mathsf{HCE1}$, the client additionally stores a message authentication code (MAC) computed over the message, and checks this MAC during decryption. This means that the TC attack against $\mathsf{HCE1}$ would be detected. TahoeFS is, however, still vulnerable to erasure attacks. We have reported this to the developers, and are discussing possible fixes with them.

We suggest a new scheme, $\mathsf{HCE2}$, that modifies $\mathsf{HCE1}$ to directly include a mechanism, called guarded decryption, that helps it to achieve TC security. The decryption routine now additionally checks the tag embedded in the ciphertext by recomputing the tag using the just-decrypted message. If the check fails, then $\perp$ is returned.

For performance in practice, the important operation is deriving the ciphertext and tag from the message. This involves generating the key, followed by encryption and tag generation. CE requires three full passes to perform encryption and tag generation, while HCE1 and HCE2 require two. Using at least two passes here is fundamental: deterministic MLE schemes that output bits of ciphertext before processing most of the message will not achieve PRV-CDA security.

The fourth scheme, Randomized Convergent Encryption (RCE), takes advantage of randomization to give a version of HCE2 that can generate the key, encrypt the message, and produce the tag, all together, in a single pass. RCE accomplishes this by first picking a random symmetric encryption key L and then encrypting the message with L, and deriving the MLE key K in a single pass. Finally it encrypts L using K as a one-time pad, and derives the tag from K. Like HCE2 it uses guarded decryption.

We note that RCE does admit a side-channel attack similar to those arising in client-side deduplication systems [29]. A user can infer whether she is the first to upload a file, by first storing its encryption under RCE and then immediately downloading to check if the recovered ciphertext is the one just uploaded.

For all the schemes, it is easy to verify decryption correctness. Tag correctness follows as the tags are all deterministic.

Privacy. We prove the following theorem, which establishes the PRV\$-CDA security of the four schemes when modeling H as a RO, in the full version [15]. The key-recovery (KR) and one-time real-or-random (ROR) security notions referred to below are recalled in [15].

Theorem 1. *Let* H *be a RO and let* $\mathsf{SE} = (\mathcal{SK}, \mathcal{SE}, \mathcal{SD})$ *be a one-time symmetric encryption scheme with key length* $k(\cdot)$. *Then if* SE *is both* KR*-secure and* ROR*-secure, the scheme* $\mathsf{XXX}[\mathsf{SE}, \mathrm{H}]$ *for* $\mathsf{XXX} \in \{\mathsf{CE}, \mathsf{HCE1}, \mathsf{HCE2}, \mathsf{RCE}\}$ *is* PRV\$-CDA*-secure.* □

Tag consistency. As discussed in Section 3, any deterministic scheme is STC-secure when tags are CR-hashes of the ciphertext. So too with CE. For HCE2 and RCE, a straightforward reduction establishes the following theorem. Here CR refers to standard collision resistance, the definition being recalled in [15].

Theorem 2. *Let* $\mathsf{SE} = (\mathcal{SK}, \mathcal{SE}, \mathcal{SD})$ *be a one-time symmetric encryption scheme and let* $\mathsf{H} = (\mathcal{HK}, \mathcal{H})$ *be a hash function family. If* H *is* CR*-secure then* $\mathsf{HCE2}[\mathsf{SE}, \mathsf{H}]$ *and* $\mathsf{RCE}[\mathsf{SE}, \mathsf{H}]$ *are* TC*-secure.* □

HCE2 and RCE are not STC-secure, by the same attack as used against the TC security of HCE1. (The tag check makes it so that decryption outputs $M' = \bot$.) One could in theory achieve STC security using non-interactive zero-knowledge proofs [18], but this would obviate the speedups offered by the schemes compared to CE. We conclude that finding fast, STC-secure schemes with $\mathcal{O}(1)$ tag generation is an interesting open problem, surfaced by our definitions and results above.

Discussion. The above schemes use a hash function family H. In practice, we might use SHA-256 or SHA-3, and key them appropriately by choosing a uniform bit string to prepend to messages. In [15] we explore various instantiations of the MLE schemes, including ones that are entirely built from AES. We also report on performance there.

5 Constructions without ROs

We overview Extract-Hash-Check (which yields standard model MLE from D-PKE or CI-H hash functions) and Sample-Extract-Encrypt (which yields the same from weaker assumptions but for particular classes of sources). Refer to [15] for full construction descriptions and security proofs.

Extract-Hash-Check. It is natural to aim to build MLE from a D-PKE scheme or a CI-H function because the latter primitives already provide privacy on unpredictable messages. However, in attempting to build MLE from these primitives, several problems arise. One is that neither of the base primitives derives the decryption key from the message. Indeed, in both, keys must be generated upfront and independently of the data. A related problem is that it is not clear how an MLE scheme might decrypt. CI-H functions are not required to be efficiently invertible. D-PKE does provide decryption, but it requires the secret key, and it is not clear how this can yield message-based decryption.

Our solution will in fact not use the decryptability of the D-PKE scheme, but rather view the latter as providing a CI-H function keyed by the public key. We apply an extractor (its seed S will be in the parameters of the MLE scheme) to the message M to get the MLE key K. Given S, M, this operation is deterministic. The scheme encrypts the message bit by bit, creating from $M = M[1] \dots M[|M|]$ the ciphertext $C = C[1] \dots C[|M|]$ in which $C[i]$ is a hash of $K \| \langle i \rangle \| M[i]$. (The key for the hash function is also in the parameters.) To decrypt $C[i]$ given K, hash both $K \| \langle i \rangle \| 1$ and $K \| \langle i \rangle \| 0$ and see which equals $C[i]$. (This is the "check" part.) The proof of privacy relies on the fact that each input to each application of the hash function will have a negligible guessing probability even given the parameters. The reduction will take an MLE source and build a source for the hash function that itself computes K and produces the inputs to the hash function. Details may be found in [15].

Sample-Extract-Encrypt. This MLE scheme relies only on a standard and weak assumption, namely a one-time symmetric encryption scheme, which can be built from any one-way function. The tradeoff is that it is only secure for a limited class of sources.

Stepping back, if we are to consider special sources, the obvious starting point is uniform and independent messages. Achieving MLE here is easy because we can use part of the message as the key to encrypt the other part. The next obvious target is block sources, where each message is assumed to have negligible guessing probability given the previous ones. D-PKE for such sources was achieved in [19]. We might hope, via the above XHC construction, to thus automatically obtain MLE for the same sources, but XHC does not preserve the block source restriction because the inputs to the hash function for different bits of the same message are highly correlated.

Sample-Extract-Encrypt (SXE) builds an MLE scheme for classes of block sources where a random subset of the bits of each message remains unpredictable even given the rest of the bits and previous messages. For example, if a message has some subset of uniform bits embedded within it. The scheme then uses a random subset of the message bits as a key, applies an extractor, and then symmetrically encrypts the rest of the message. Refer to [15] for details.

Acknowledgments. We thank James Lentini for pointing out the STC attack against RCE, Ananth Raghunathan for early discussions about convergent encryption, the TahoeFS developers for discussions regarding duplicate faking attacks in TahoeFS, and the anonymous reviewers for their helpful comments. Bellare and Keelveedhi were supported in part by NSF grants CNS-0904380, CCF-0915675 and CNS-1116800. Ristenpart was supported in part by NSF grant CNS-1065134 and gifts from RSA Labs and Microsoft.

References

1. Bitcasa, inifinite storage, `http://www.bitcasa.com/`
2. Ciphertite data backup, `http://www.ciphertite.com/`
3. Dropbox, a file-storage and sharing service, `http://www.dropbox.com/`
4. The Flud backup system, `http://flud.org/wiki/Architecture`
5. GNUnet, a framework for secure peer-to-peer networking, `https://gnunet.org/`
6. Google Drive, `http://drive.google.com/`
7. Adya, A., Bolosky, W., Castro, M., Cermak, G., Chaiken, R., Douceur, J., Howell, J., Lorch, J., Theimer, M., Wattenhofer, R.: Farsite: Federated, available, and reliable storage for an incompletely trusted environment. ACM SIGOPS Operating Systems Review 36(SI), 1–14 (2002)
8. Anderson, P., Zhang, L.: Fast and secure laptop backups with encrypted de-duplication. In: Proc. of USENIX LISA (2010)
9. Bar-Yossef, Z., Reingold, O., Shaltiel, R., Trevisan, L.: Streaming computation of combinatorial objects. In: Proceedings of the 17th IEEE Annual Conference on Computational Complexity, pp. 133–142. IEEE (2002)
10. Batten, C., Barr, K., Saraf, A., Trepetin, S.: pStore: A secure peer-to-peer backup system. Unpublished report, MIT Laboratory for Computer Science (2001)
11. Bellare, M., Boldyreva, A., O'Neill, A.: Deterministic and efficiently searchable encryption. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 535–552. Springer, Heidelberg (2007)
12. Bellare, M., Brakerski, Z., Naor, M., Ristenpart, T., Segev, G., Shacham, H., Yilek, S.: Hedged public-key encryption: How to protect against bad randomness. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 232–249. Springer, Heidelberg (2009)
13. Bellare, M., Desai, A., Jokipii, E., Rogaway, P.: A concrete security treatment of symmetric encryption. In: 38th FOCS, pp. 394–403. IEEE Computer Society Press (October 1997)
14. Bellare, M., Fischlin, M., O'Neill, A., Ristenpart, T.: Deterministic encryption: Definitional equivalences and constructions without random oracles. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 360–378. Springer, Heidelberg (2008)

15. Bellare, M., Keelveedhi, S., Ristenpart, T.: Message-locked encryption and secure deduplication. Cryptology ePrint Archive, Report 2012/631 (2012), `http://eprint.iacr.org/`
16. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Ashby, V. (ed.) ACM CCS 1993, pp. 62–73. ACM Press (November 1993)
17. Bellare, M., Rogaway, P.: The security of triple encryption and a framework for code-based game-playing proofs. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 409–426. Springer, Heidelberg (2006)
18. Blum, M., Feldman, P., Micali, S.: Non-interactive zero-knowledge and its applications. In: Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing, pp. 103–112. ACM (1988)
19. Boldyreva, A., Fehr, S., O'Neill, A.: On notions of security for deterministic encryption, and efficient constructions without random oracles. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 335–359. Springer, Heidelberg (2008)
20. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 506–522. Springer, Heidelberg (2004)
21. Brakerski, Z., Segev, G.: Better security for deterministic public-key encryption: The auxiliary-input setting. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 543–560. Springer, Heidelberg (2011)
22. Cooley, J., Taylor, C., Peacock, A.: ABS: the apportioned backup system. MIT Laboratory for Computer Science (2004)
23. Cox, L.P., Murray, C.D., Noble, B.D.: Pastiche: making backup cheap and easy. SIGOPS Oper. Syst. Rev. 36, 285–298 (2002)
24. Douceur, J., Adya, A., Bolosky, W., Simon, D., Theimer, M.: Reclaiming space from duplicate files in a serverless distributed file system. In: Proceedings of the 22nd International Conference on Distributed Computing Systems, pp. 617–624. IEEE (2002)
25. Fuller, B., O'Neill, A., Reyzin, L.: A unified approach to deterministic encryption: New constructions and a connection to computational entropy. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 582–599. Springer, Heidelberg (2012)
26. Goldwasser, S., Micali, S.: Probabilistic encryption. Journal of Computer and System Sciences 28(2), 270–299 (1984)
27. Goyal, V., O'Neill, A., Rao, V.: Correlated-input secure hash functions. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 182–200. Springer, Heidelberg (2011)
28. Halevi, S., Harnik, D., Pinkas, B., Shulman-Peleg, A.: Proofs of ownership in remote storage systems. In: Chen, Y., Danezis, G., Shmatikov, V. (eds.) ACM CCS 2011, pp. 491–500. ACM Press (October 2011)
29. Harnik, D., Pinkas, B., Shulman-Peleg, A.: Side channels in cloud services: Deduplication in cloud storage. IEEE Security & Privacy 8(6), 40–47 (2010)
30. Lu, C.-J.: Hyper-encryption against space-bounded adversaries from on-line strong extractors. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 257–271. Springer, Heidelberg (2002)
31. Marques, L., Costa, C.: Secure deduplication on mobile devices. In: Proceedings of the 2011 Workshop on Open Source and Design of Communication, pp. 19–26. ACM (2011)
32. Mironov, I., Pandey, O., Reingold, O., Segev, G.: Incremental deterministic public-key encryption. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 628–644. Springer, Heidelberg (2012)

33. Nisan, N., Zuckerman, D.: Randomness is linear in space. Journal of Computer and System Sciences 52(1), 43–52 (1996)
34. Pettitt, J.: Content based hashing, http://cypherpunks.venona.com/date/1996/02/msg02013.html
35. Rahumed, A., Chen, H., Tang, Y., Lee, P., Lui, J.: A secure cloud backup system with assured deletion and version control. In: 2011 40th International Conference on Parallel Processing Workshops (ICPPW), pp. 160–167. IEEE (2011)
36. Ristenpart, T., Shacham, H., Shrimpton, T.: Careful with composition: Limitations of the indifferentiability framework. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 487–506. Springer, Heidelberg (2011)
37. Rogaway, P., Bellare, M., Black, J., Krovetz, T.: OCB: A block-cipher mode of operation for efficient authenticated encryption. In: ACM CCS 2001, pp. 196–205. ACM Press (November 2001)
38. Song, D.X., Wagner, D., Perrig, A.: Practical techniques for searches on encrypted data. In: 2000 IEEE Symposium on Security and Privacy, pp. 44–55. IEEE Computer Society Press (May 2000)
39. Storer, M., Greenan, K., Long, D., Miller, E.: Secure data deduplication. In: Proceedings of the 4th ACM International Workshop on Storage Security and Survivability, pp. 1–10. ACM (2008)
40. Vadhan, S.P.: On constructing locally computable extractors and cryptosystems in the bounded storage model. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 61–77. Springer, Heidelberg (2003)
41. Wichs, D.: Barriers in cryptography with weak, correlated and leaky sources. In: Proceedings of the 4th Conference on Innovations in Theoretical Computer Science, ITCS 2013, pp. 111–126. ACM, New York (2013)
42. Wilcox-O'Hearn, Z.: Convergent encryption reconsidered (2011), http://www.mail-archive.com/cryptography@metzdowd.com/msg08949.html
43. Wilcox-O'Hearn, Z., Warner, B.: Tahoe: The least-authority filesystem. In: Proceedings of the 4th ACM International Workshop on Storage Security and Survivability, pp. 21–26. ACM (2008)
44. Xu, J., Chang, E.-C., Zhou, J.: Leakage-resilient client-side deduplication of encrypted data in cloud storage. Cryptology ePrint Archive, Report 2011/538 (2011), http://eprint.iacr.org/

Keccak

Guido Bertoni[1], Joan Daemen[1], Michaël Peeters[2], and Gilles Van Assche[1]

[1] STMicroelectronics
[2] NXP Semiconductors

In October 2012, the American National Institute of Standards and Technology (NIST) announced the selection of Keccak as the winner of the SHA-3 Cryptographic Hash Algorithm Competition [10,11]. This concluded an open competition that was remarkable both for its magnitude and the involvement of the cryptographic community. Public review is of paramount importance to increase the confidence in the new standard and to favor its quick adoption. The SHA-3 competition explicitly took this into account by giving open access to the candidate algorithms and everyone in the cryptographic community could try to break them, compare their performance, or simply give comments.

While preparing for the SHA-3 competition, we developed and presented the *sponge construction* [1]. Our initial goal of this effort was to solve the problem of compactly expressing a comprehensive security claim. It turned out to be a powerful tool for building a hash function and we adopted it for our SHA-3 candidate Keccak. Additionally, with its variable output length it can be used as a mask generating function, a stream cipher or a MAC computation function. To support more sophisticated modes such as single-pass authenticated encryption and reseedable pseudorandom sequence generation, we additionally introduced the *duplex construction* [3]. We have proven both sponge and duplex constructions sound in the indifferentiability framework [8,2,3]. Our permutation-based modes can be seen as an alternative to the block-cipher based modes that have dominated mainstream symmetric cryptography in the last decades. They are simpler than the traditional block cipher modes and offer at the same time more flexibility by allowing to trade in security strength level for speed and vice versa.

At the core of Keccak is a set of seven permutations called Keccak-$f[b]$, with width b chosen between 25 and 1600 by multiplicative steps of 2 [4]. Depending on b, the resulting function ranges from a toy cipher to a wide function. The instances proposed for SHA-3 use exclusively Keccak-$f[1600]$ for all security levels [5], whereas lightweight alternatives can use for instance Keccak-$f[200]$ or Keccak-$f[400]$, leaving Keccak-$f[800]$ as an intermediate choice [6]. Inside Keccak-f, the state to process is organized in 5×5 *lanes* of $b/25$ bits each, or alternatively as $b/25$ *slices* of 25 bits each. The round function processes the state using a non-linear layer of algebraic degree two (χ), a linear mixing layer (θ), inter- and intra-slice dispersion steps (ρ, π) and the addition of round constants (ι). The choice of operations in Keccak-f makes it very different from the SHA-2 family or even Rijndael (AES) [9,7]. On the implementation side, these operations are efficiently translated into bitwise Boolean operations and circular shifts, they lead to short critical paths in hardware implementations and they are well suited for protections against side-channel attacks.

References

1. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Sponge functions. In: Ecrypt Hash Workshop 2007 (May 2007); available as public comment to NIST, http://www.csrc.nist.gov/pki/HashWorkshop/Public_Comments/2007_May.html
2. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: On the indifferentiability of the sponge construction. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 181–197. Springer, Heidelberg (2008), http://sponge.noekeon.org/
3. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Duplexing the sponge: single-pass authenticated encryption and other applications. In: Miri, A., Vaudenay, S. (eds.) SAC 2011. LNCS, vol. 7118, pp. 320–337. Springer, Heidelberg (2012)
4. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: The Keccak reference (January 2011), http://keccak.noekeon.org/
5. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: The Keccak SHA-3 submission (January 2011), http://keccak.noekeon.org/
6. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G., Van Keer, R.: Keccak implementation overview (May 2012), http://keccak.noekeon.org/
7. Daemen, J., Rijmen, V.: The design of Rijndael — AES, the advanced encryption standard. Springer (2002)
8. Maurer, U., Renner, R., Holenstein, C.: Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 21–39. Springer, Heidelberg (2004)
9. NIST, Federal information processing standard 180-2, secure hash standard (August 2002)
10. NIST, selects winner of secure hash algorithm (SHA-3) competition (October 2012), http://www.nist.gov/itl/csd/sha-100212.cfm
11. NIST, Third-round report of the SHA-3 cryptographic hash algorithm competition (November 2012), http://dx.doi.org/10.6028/NIST.IR.7896

Batch Fully Homomorphic Encryption over the Integers[⋆]

Jung Hee Cheon[1], Jean-Sébastien Coron[2], Jinsu Kim[1], Moon Sung Lee[1], Tancrède Lepoint[3,4], Mehdi Tibouchi[5], and Aaram Yun[6]

[1] Seoul National University (SNU), Republic of Korea
{jhcheon,kjs2002,moolee}@snu.ac.kr
[2] Tranef, France
jscoron@tranef.com
[3] CryptoExperts, France
[4] École Normale Supérieure, France
tancrede.lepoint@cryptoexperts.com
[5] NTT Secure Platform Laboratories, Japan
tibouchi.mehdi@lab.ntt.co.jp
[6] Ulsan National Institute of Science and Technology (UNIST), Republic of Korea
aaramyun@unist.ac.kr

Abstract. We extend the fully homomorphic encryption scheme over the integers of van Dijk *et al.* (DGHV) into a batch fully homomorphic encryption scheme, *i.e.* to a scheme that supports encrypting and homomorphically processing a vector of plaintexts as a single ciphertext.

We present two variants in which the semantic security is based on different assumptions. The first variant is based on a new decisional problem, the Decisional Approximate-GCD problem, whereas the second variant is based on the more classical computational Error-Free Approximate-GCD problem but requires additional public key elements.

We also show how to perform arbitrary permutations on the underlying plaintext vector given the ciphertext and the public key. Our scheme offers competitive performance even with the bootstrapping procedure: we describe an implementation of the homomorphic evaluation of AES, with an amortized cost of about 12 minutes per AES ciphertext on a standard desktop computer; this is comparable to the timings presented by Gentry *et al.* at Crypto 2012 for their implementation of a Ring-LWE based fully homomorphic encryption scheme.

Keywords: Fully Homomorphic Encryption, Batch Encryption, Chinese Remainder Theorem, Approximate GCD, Homomorphic AES..

1 Introduction

Fully Homomorphic Encryption (FHE). Fully homomorphic encryption allows a worker to perform implicit additions and multiplications on plaintext

[⋆] This paper is a merger of two independent works [CLT13, KLYC13] built on the same basic idea but with different contributions. The respective full versions are posted on ePrint.

values while exclusively manipulating encrypted data. The first construction of a fully homomorphic scheme (based on ideal lattices) was described by Gentry in [Gen09], and proceeds in several steps. First, one constructs a *somewhat homomorphic* encryption scheme, which only supports a limited number of multiplications: ciphertexts contain some noise that becomes larger with successive homomorphic multiplications, and only ciphertexts whose noise size remains below a certain threshold can be decrypted correctly. The second step is to *squash* the decryption procedure associated with an arbitrary ciphertext so that it can be expressed as a low degree polynomial in the secret key bits. Then, Gentry's key idea, called *bootstrapping*, consists in homomorphically evaluating this decryption polynomial on encryptions of the secret key bits, resulting in a different ciphertext associated with the same plaintext, but with possibly reduced noise. This *refreshed* ciphertext can then be used in subsequent homomorphic operations. By repeatedly refreshing ciphertexts, the number of homomorphic operations becomes unlimited, resulting in a fully homomorphic encryption scheme.

Since Gentry's breakthrough result, many improvements have been made, introducing new variants, improving efficiency, and providing new features. Recently, Brakerski, Gentry and Vaikuntanathan described a different framework where the ciphertext noise grows only linearly with the multiplicative level instead of exponentially [BGV12], so that bootstrapping is no longer necessary to obtain a scheme supporting the homomorphic evaluation of any given polynomial size circuit. Currently three main families of fully homomorphic encryption schemes are known:

1. Gentry's original scheme [Gen09] based on ideal lattices. An implementation of Gentry's scheme was proposed by Gentry and Halevi in [GH11] with a public key of 2.3 GB and a ciphertext refresh procedure of 30 minutes; the implementation is based on many interesting algorithmic optimizations, including some borrowed from Smart and Vercauteren [SV10].
2. van Dijk, Gentry, Halevi and Vaikuntanathan's (DGHV) scheme over the integers [DGHV10]. It was recently shown how to significantly reduce the public key size in DGHV, yielding a 10.3 MB public key and an 11-minute refresh procedure [CNT12].
3. Brakerski and Vaikuntanathan's scheme based on the Learning with Errors (LWE) and Ring Learning with Errors (RLWE) problems [BV11a, BV11b], and follow-up works (*e.g.* the scale-free variant of Brakerski [Bra12] and the NTRU-variant [LATV12]). An implementation is described in [GHS12b] with an efficient (given the current state of knowledge) homomorphic evaluation of the full AES encryption circuit. The authors use the *batch* RLWE-based scheme proposed in [BGV12, GHS12a], that allows one to encrypt vectors of plaintexts in a single ciphertext and to perform any permutation on the underlying plaintext vector while manipulating only the ciphertext [SV11].

Our Contributions. In this paper we focus on the DGHV scheme. Our goal is to extend DGHV to support the same batching capability as in RLWE-based schemes [BGV12, GHS12a], and to homomorphically evaluate a full AES circuit with roughly the same level of efficiency as [GHS12b], in order to obtain an implementation of a FHE scheme with similar features but based on different techniques and assumptions.

In the original DGHV scheme, a ciphertext has the form

$$c = q \cdot p + 2r + m$$

where p is the secret key, q is a large random integer, and r is a small random integer (noise); the bit message $m \in \{0, 1\}$ is recovered by computing $m = [c \bmod p] \bmod 2$. The scheme is clearly homomorphic for both addition and multiplication, since addition and multiplication of ciphertexts correspond to addition and multiplication of plaintexts modulo 2.

To encrypt multiple bits m_i into a single ciphertext c, we use the Chinese Remainder Theorem with respect to a tuple of $(\ell+1)$ coprime integers $q_0, p_0, \ldots, p_{\ell-1}$. The batch ciphertext has the form

$$c = \mathsf{CRT}_{q_0, p_0, \ldots, p_{\ell-1}}(q, 2r_0 + m_0, \ldots, 2r_{\ell-1} + m_{\ell-1}),$$

and correctly decrypts to the bit vector (m_i) given by $m_i = [c \bmod p_i] \bmod 2$ for all $0 \leqslant i < \ell$.[1] Modulo each of the p_i's the ciphertext c behaves as in the original DGHV scheme. Accordingly, the addition or multiplication of two ciphertexts yields a new ciphertext that decrypts to the componentwise sum or product modulo 2 of the original plaintexts.

The main challenge, however, was to prove the semantic security of our new scheme. In the original DGHV scheme, public-key encryption is performed by masking the message m with a random subset sum of the public key elements $x_j = q_j \cdot p + r_j$ as

$$c = \left[m + 2r + 2 \sum_{j \in S} x_j \right]_{x_0} . \tag{1}$$

The semantic security is proved by applying the Leftover Hash Lemma on the subset sum modulo q_0, and using the random $2r$ in (1) to further randomize the ciphertext modulo p.

To prove semantic security for the batch scheme our first technique is to rely on a new assumption, namely the Decisional Approximate-GCD assumption. Under this assumption the integers x_j in the subset-sum from (1) are assumed to be indistinguishable from random modulo x_0; semantic security is then proved by applying the Leftover Hash Lemma modulo x_0; the additional random $2r$ in (1) becomes unnecessary. Extending DGHV public-key encryption to the batch setting is then straightforward; namely one can use the same random subset sum technique with public key elements x_j having a small residue modulo each of the p_i's instead of only modulo p. We show that our batch DGHV scheme can encrypt $\ell = \tilde{\mathcal{O}}(\lambda^3)$ bits in a single ciphertext; therefore the ciphertext expansion ratio becomes $\tilde{\mathcal{O}}(\lambda^2)$ instead of $\tilde{\mathcal{O}}(\lambda^5)$ in the original scheme.

[1] We denote by $\mathsf{CRT}_{q_0, p_0, \ldots, p_{\ell-1}}(q, a_0, \ldots, a_{\ell-1})$ the unique integer u with $0 \leqslant u < q_0 \cdot \prod_{i=0}^{\ell-1} p_i$ such that $u \equiv q \pmod{q_0}$ and $u \equiv a_i \pmod{p_i}$ for all $0 \leqslant i < \ell$.

In Section 4 we describe a different technique that does not rely on a new decisional assumption but on the known (computational) Error-Free Approximate-GCD assumption used in previous work [DGHV10, CMNT11, CNT12]. For the proof of semantic security to go through in the batch setting, the ciphertext c should be independently randomized modulo each of the p_i's, which is not easy to achieve without knowing the p_i's. Indeed, if we only use a single additive term $2r$ as in Equation (1), then the *same* random term $2r = 2r \bmod p_i$ is added modulo each of the p_i, which breaks the security proof. Our technique is to replace the term $2r$ in (1) by another subset sum of public key elements which, taken modulo each of the p_i's, generate a lattice with special properties. We then apply the Leftover Hash Lemma modulo this lattice instead of only modulo q_0, which proves semantic security.

In addition to componentwise addition and multiplication, we also show how to perform any permutation on plaintext bits publicly. As opposed to RLWE based schemes [BGV12, GHS12a], we cannot use an underlying algebraic structure to perform rotations over plaintext bits (clearly, the automorphisms of $\mathbb{Z}$ do not provide any useful action on ciphertexts). Instead we show how to perform arbitrary permutations on the plaintext vector during the ciphertext refresh operation at no additional cost (but with a slight increase of the public key size). Our ciphertext refresh operation $\mathsf{Recrypt}$ is done in parallel over the ℓ slots, with the same complexity as a single $\mathsf{Recrypt}$ operation in the original scheme.

Finally, we describe an implementation of our batch DGHV scheme of Section 4, with concrete parameters. We perform an homomorphic evaluation of the full AES encryption circuit. For the "Large" parameters with 72 bits of security, our implementation homomorphically encrypts up to 531 AES ciphertexts in parallel in an amortized time of 12 minutes per AES ciphertext on a desktop computer. This is comparable to the timings presented by Gentry *et al.* at Crypto 2012 for their implementation of an RLWE-based scheme [GHS12b].[2]

While our batch variant of DGHV does not provide additional features nor significantly improved efficiency over the RLWE-based scheme of [GHS12a], we believe it is interesting to obtain FHE schemes with similar properties but based on different techniques and assumptions.

2 The Somewhat Homomorphic DGHV Scheme

We first recall the somewhat homomorphic encryption scheme over the integers of van Dijk, Gentry, Halevi and Vaikuntanathan (DGHV) in [DGHV10]. Let λ be the security parameter, τ be the number of elements in the public key, γ their bit-length, η the bit-length of the secret key p and ρ (resp. ρ') the bit-length of the noise in the public key (resp. in a fresh ciphertext).

For a real number x, we denote by $\lceil x \rceil$, $\lfloor x \rfloor$, and $\lceil x \rfloor$ the upper, lower, and nearest integer part of x. For integers z, p we denote the reduction of z modulo p by $(z \bmod p)$ or $[z]_p$ with $-p/2 < [z]_p \leqslant p/2$.

[2] Notice that our implementation uses bootstrapping whereas the implementation of [GHS12b] used a leveled homomorphic encryption scheme without bootstrapping.

For a specific η-bit odd integer p, we use the following distribution over γ-bit integers:

$$\mathcal{D}_{\gamma,\rho}(p) = \left\{ \begin{array}{c} \textsf{Choose } q \leftarrow \mathbb{Z} \cap [0, 2^\gamma/p),\ r \leftarrow \mathbb{Z} \cap (-2^\rho, 2^\rho) : \\ \textsf{Output } x = q \cdot p + r \end{array} \right\}.$$

$\textsf{DGHV.KeyGen}(1^\lambda)$. Generate an η-bit random prime integer p. For $0 \leqslant i \leqslant \tau$, sample $x_i \leftarrow \mathcal{D}_{\gamma,\rho}(p)$. Relabel the x_i's so that x_0 is the largest. Restart unless x_0 is odd and $[x_0]_p$ is even. Let $\textsf{pk} = (x_0, x_1, \ldots x_\tau)$ and $\textsf{sk} = p$.

$\textsf{DGHV.Encrypt}(\textsf{pk}, m \in \{0,1\})$. Choose a random subset $S \subseteq \{1, 2, \ldots, \tau\}$ and a random integer r in $(-2^{\rho'}, 2^{\rho'})$, and output the ciphertext:

$$c = \left[m + 2r + 2 \sum_{i \in S} x_i \right]_{x_0}. \tag{2}$$

$\textsf{DGHV.Evaluate}(\textsf{pk}, C, c_1, \ldots, c_t)$. Given the circuit C with t input bits and t ciphertexts c_i, apply the addition and multiplication gates of C to the ciphertexts, performing all the additions and multiplications over the integers, and return the resulting integer.

$\textsf{DGHV.Decrypt}(\textsf{sk}, c)$. Output $m \leftarrow [c \bmod p]_2$.

As shown in [DGHV10] the scheme is somewhat homomorphic, *i.e.*, a limited number of homomorphic operations can be performed on ciphertexts. More precisely, given two ciphertexts $c = q \cdot p + 2r + m$ and $c' = q' \cdot p + 2r' + m'$ where r and r' are ρ'-bit integers, the ciphertext $c + c'$ is an encryption of $m + m'$ mod 2 under a $(\rho' + 1)$-bit noise and the ciphertext $c \cdot c'$ is an encryption of $m \cdot m'$ with noise bit-length $\simeq 2\rho'$. Since the ciphertext noise must remain smaller than p to maintain correctness, the scheme roughly allows η/ρ' successive multiplications on ciphertexts. The scheme is semantically secure under the Approximate-GCD assumption (see [DGHV10]):

Definition 1 (Approximate GCD). *The (ρ, η, γ)-approximate GCD problem consists, given a random η-bit odd integer p and given polynomially many samples from $\mathcal{D}_{\gamma,\rho}(p)$, in outputting p.*

3 Batch DGHV Scheme Based on a New Decisional Assumption[3]

We describe our first extension of the DGHV scheme for the batch setting. We extend the DGHV scheme by packing ℓ plaintexts m_0, ..., $m_{\ell-1}$ into a single ciphertext, using the Chinese Remainder Theorem. Moreover, for somewhat homomorphic encryption, this allows us to encrypt not only bits but elements from rings of form $\mathbb{Z}_Q$.

[3] A part of this section was made public through [Mem12].

Therefore, for public parameters $Q_0, \ldots, Q_{\ell-1}$, we encrypt $(m_0, \ldots, m_{\ell-1}) \in \mathbb{Z}_{Q_0} \times \cdots \times \mathbb{Z}_{Q_{\ell-1}}$ into a ciphertext of the following form:

$$c = \mathsf{CRT}_{q_0,p_0,\ldots,p_{\ell-1}}(q, Q_0 r_0 + m_0, \ldots, Q_{\ell-1} r_{\ell-1} + m_{\ell-1}),$$

where q is uniform random modulo q_0 and r_i's are small noises. Decryption can be done by

$$m_i = [c \bmod p_i]_{Q_i}.$$

Homomorphic addition and multiplication is done by the corresponding arithmetic operations on ciphertexts.

This scheme can be considered as encrypting vectors $(m_0, \ldots, m_{\ell-1})$ and supporting parallel, componentwise additions and multiplications. Or, if we choose the Q_i's to be pairwise coprime, then using the isomorphism $\mathbb{Z}_{\prod Q_i} \cong \prod \mathbb{Z}_{Q_i}$, we may regard this as a somewhat homomorphic encryption supporting arithmetic operations on $\mathbb{Z}_Q$, with $Q = \prod Q_i$.

In order to allow public-key encryption, we provide integers x'_i and x_i in the public key, such that $x'_i \bmod p_j = Q_j r'_{i,j} + \delta_{i,j}$, and $x_i \bmod p_j = Q_j r_{i,j}$ for all i, j where $\delta_{i,j}$ is the Kronecker delta. Then, a plaintext vector $\boldsymbol{m} = (m_0, \ldots, m_{\ell-1})$ is encrypted as follows:

$$c = \left[\sum_{i=0}^{\ell-1} m_i \cdot x'_i + \sum_{i \in S} x_i \right]_{x_0}.$$

As explained in the introduction, the additional, larger noise $2r$ from the DGHV scheme is not used in this version. This simplifies the construction, with the trade-off of a new decisional assumption and a larger security loss. In Section 4, we present another version which handles this issue differently. In Section 5, we describe a transformation to fully homomorphic encryption schemes where $Q_0 = \cdots = Q_{\ell-1} = 2$.

3.1 Description

$\mathsf{IDGHV.KeyGen}(1^\lambda, (Q_j)_{0 \leqslant j < \ell})$. Choose η-bit distinct primes p_j, $0 \leqslant j < \ell$, and denote π their product. Let us define the error-free public key element $x_0 = q_0 \cdot \pi$, where $q_0 \leftarrow \mathbb{Z} \cap [0, 2^\gamma/\pi)$ is a 2^{λ^2}-rough integer.[4] Make sure that $\gcd(Q_j, x_0) = 1$ for $0 \leqslant j < \ell$, and abort otherwise.[5]

Choose the following integers x_i and x'_i with a quotient by π uniformly and independently distributed in $\mathbb{Z} \cap [0, q_0)$, and with the following distribution modulo p_j for $0 \leqslant j < \ell$:

$$\begin{array}{lll} 1 \leqslant i \leqslant \tau, & x_i \bmod p_j = Q_j r_{i,j}, & r_{i,j} \leftarrow \mathbb{Z} \cap (-2^\rho, 2^\rho), \\ 0 \leqslant i \leqslant \ell - 1, & x'_i \bmod p_j = Q_j r'_{i,j} + \delta_{i,j}, & r'_{i,j} \leftarrow \mathbb{Z} \cap (-2^\rho, 2^\rho). \end{array}$$

[4] An integer a is b-rough when it does not contain prime factors smaller than b. As in [CMNT11] one can generate q_0 as a product of 2^{λ^2}-bit primes.

[5] Note that the abort case happens with negligible probability.

Finally, let $\mathsf{pk} = \left\{x_0, (Q_i)_{0\leqslant i\leqslant \ell-1}, (x_i)_{1\leqslant i\leqslant \tau}, (x'_i)_{0\leqslant i\leqslant \ell-1}\right\}$ and let $\mathsf{sk} = (p_j)_{0\leqslant j\leqslant \ell-1}$.

$\mathsf{IDGHV.Encrypt}(\mathsf{pk}, \boldsymbol{m})$**.** For any $\boldsymbol{m} = (m_0, \ldots, m_{\ell-1})$ with $m_i \in \mathbb{Z}_{Q_i}$, choose a random binary vector $\boldsymbol{b} = (b_i)_{1\leqslant i\leqslant\tau} \in \{0,1\}^\tau$ and output the ciphertext:

$$c = \left[\sum_{i=0}^{\ell-1} m_i \cdot x'_i + \sum_{i=1}^{\tau} b_i \cdot x_i\right]_{x_0} . \tag{3}$$

$\mathsf{IDGHV.Decrypt}(\mathsf{sk}, c)$**.** Output $\boldsymbol{m} = (m_0, \ldots, m_{\ell-1})$ where $m_j \leftarrow [c \bmod p_j]_{Q_j}$.

$\mathsf{IDGHV.Add}(\mathsf{pk}, c_1, c_2)$**.** Output $c_1 + c_2 \bmod x_0$.

$\mathsf{IDGHV.Mult}(\mathsf{pk}, c_1, c_2)$**.** Output $c_1 \cdot c_2 \bmod x_0$.

3.2 Parameters and Correctness

The size of the message space is determined by ℓ and the binary length of the Q_j's, which can be an integer from 2 to $\eta/8$ depending on the multiplicative depth of the scheme. The parameters should satisfy the following constraints:

- $\rho = \tilde{\Omega}(\lambda)$, to be secure against Chen-Nguyen's attack [CN12] and Howgrave-Graham's attack [HG01],
- $\eta = \tilde{\Omega}(\lambda^2 + \rho \cdot \lambda)$, to resist the factoring attack using the elliptic curve method [Len87], and to permit enough multiplicative depth,
- $\gamma = \omega(\eta^2 \log \lambda)$, to resist Cohn and Heninger's attack [CH11] and the attack using Lagarias algorithm [Lag85] on the approximate GCD problem,
- $\tau = \gamma + \omega(\log \lambda)$, in order to use leftover hash lemma (see Section 3.3).

We choose $\gamma = \tilde{\mathcal{O}}(\lambda^5)$, $\eta = \tilde{\mathcal{O}}(\lambda^2)$, $\rho = 2\lambda$, $\tau = \gamma + \lambda$ which is similar to the DGHV's convenient parameter setting [DGHV10]. We remark that b_i can be chosen in a much larger interval so as to reduce the public key size as in Section 4.

Notice that the above scheme is correct as is proved in the full version [KLYC13] with the definition of correctness for homomorphic encryption schemes, with respect to a set of permitted circuits.

3.3 Semantic Security

Here we show the semantic security of the IDGHV scheme, based on a new assumption called ℓ-$\mathbf{DACD}_Q$. In fact, this rather complicated assumption is given only as an intermediate step for the proof, and in Section 3.4, we prove this assumption from a simpler decisional assumption, called the Decisional Approximate GCD assumption (**DACD**). So the security of our scheme is eventually based on this assumption.

Given two integer vectors $\boldsymbol{p} = (p_0, \ldots, p_{\ell-1})$, $\boldsymbol{Q} = (Q_0, \ldots, Q_{\ell-1})$ of length ℓ and an integer q_0, let us define the distribution $\mathcal{D}_\rho(\boldsymbol{p}; \boldsymbol{Q}; q_0)$ as follows. The output of the distribution is

$$x = \mathsf{CRT}_{q_0, p_0, \ldots, p_{\ell-1}}(q, Q_0 r_0, \ldots, Q_{\ell-1} r_{\ell-1}),$$

where $q \leftarrow \mathbb{Z} \cap [0, q_0)$ and $r_i \leftarrow \mathbb{Z} \cap (-2^\rho, 2^\rho)$ for $i = 0, \ldots, \ell - 1$.

Definition 2 (ℓ-Decisional Approximate GCD$_Q$ Problem: ℓ-DACD$_Q$). *The $(\rho, \eta, \gamma, \mu)$-$\ell$-decisional approximate GCD$_Q$ problem is: for η-bit distinct primes $p_0, \dots, p_{\ell-1}$ and μ-bit integers $Q_0, \dots, Q_{\ell-1}$, given a γ-bit integer $x_0 := q_0 p_0 \cdots p_{\ell-1}$, with $\gcd(x_0, Q_i) = 1$ for $i = 0, \dots, \ell-1$, and polynomially many samples from $\mathcal{D} := \mathcal{D}_\rho(\boldsymbol{p}; \boldsymbol{Q}; q_0)$ and a set X consisting of ℓ integers $x'_i = \mathsf{CRT}_{q_0, p_0, \dots, p_{\ell-1}}(q_i, Q_0 r'_{i,0} + \delta_{i,0}, \dots, Q_{\ell-1} r'_{i,\ell-1} + \delta_{i,\ell-1})$ where q_i, $r'_{i,j}$ are chosen as $q_i \leftarrow \mathbb{Z} \cap [0, q_0)$, $r'_{i,j} \leftarrow \mathbb{Z} \cap (-2^\rho, 2^\rho)$ for all $i, j \in \{0, \dots, \ell-1\}$, determine $b \in \{0,1\}$ from $z = x + r \cdot b \mod x_0$ where $x \leftarrow \mathcal{D}$ and $r \leftarrow \mathbb{Z} \cap [0, x_0)$.*

The $(\rho, \eta, \gamma, \mu)$-$\ell$-decisional approximate GCD$_Q$ assumption then states that this problem is hard for any polynomial time distinguisher.

Theorem 1. *The* IDGHV *scheme is semantically secure under the ℓ-decisional approximate* GCD$_Q$ *assumption.*

Proof. We provide a sketch of the proof. See the complete proof in the full version [KLYC13].

Essentially, the idea of the proof is that the IDGHV scheme is a lossy encryption [BHY09]. Both the correctly generated public key pk and the lossy key pk' have the following form

$$\left\{x_0, (Q_i)_{0 \leqslant i \leqslant \ell-1}, (x_i)_{1 \leqslant i \leqslant \tau}, (x'_i)_{0 \leqslant i \leqslant \ell-1}\right\},$$

but note that, for the real public key pk, x_i are chosen as $x_i \leftarrow \mathcal{D}_\rho(\boldsymbol{p}; \boldsymbol{Q}; q_0)$, and for the lossy public key pk', $x_i \leftarrow \mathbb{Z} \cap [0, x_0)$.

Now we may rely on the standard hybrid argument to show that pk and pk' are computationally indistinguishable, under the $(\rho, \eta, \gamma, \mu)$-$\ell$-decisional approximate GCD$_Q$ assumption: for each $\hat{\imath} \in \{1, \dots, \tau\}$, we define $\mathsf{pk}^{(\hat{\imath})}$, where

$$x_i \leftarrow \begin{cases} \mathbb{Z} \cap [0, x_0) & \text{for } 1 \leqslant i \leqslant \hat{\imath}, \\ \mathcal{D}_\rho(\boldsymbol{p}; \boldsymbol{Q}; q_0) & \text{for } \hat{\imath} < i \leqslant \tau. \end{cases}$$

Then $\mathsf{pk}^{(0)}$ is identical to pk, and $\mathsf{pk}^{(\tau)}$ is identical to pk', and using the $(\rho, \eta, \gamma, \mu)$-$\ell$-decisional approximate GCD$_Q$ assumption, we may show that $\mathsf{pk}^{(\hat{\imath})}$ is indistinguishable from $\mathsf{pk}^{(\hat{\imath}+1)}$ for $\hat{\imath} = 0, \dots, \tau - 1$, proving that pk and pk' are computationally indistinguishable.

Next, we show that under the lossy key pk', the IDGHV scheme is semantically secure: for any two plaintexts $\boldsymbol{m}$ and $\boldsymbol{m}'$, the distributions of their corresponding ciphertexts c, c' under the lossy key pk' are statistically close. Namely, recall that from Equation (3), we have

$$c = \left[\sum_{i=0}^{\ell-1} m_i \cdot x'_i + \sum_{i=1}^{\tau} b_i \cdot x_i\right]_{x_0}.$$

Now, when we choose x_i from the lossy public key pk', x_i are uniform on $\mathbb{Z} \cap [0, x_0)$, and we may use the Leftover Hash Lemma. In fact we may use the simplified version shown in Lemma 1 of [DGHV10] to conclude that the distribution of $[\sum_{i=1}^{\tau} b_i \cdot x_i]_{x_0}$ is statistically close to the uniform distribution on $\mathbb{Z} \cap [0, x_0)$, when the parameters are chosen according to Section 3.2. Therefore, the distribution of c is uniformly random, regardless of the plaintext vector $\boldsymbol{m}$. This shows that IDGHV is a lossy encryption scheme, and the semantic security directly follows from that. $\square$

3.4 Hardness Assumption

We show that the semantic security of our scheme can be based on a simpler decisional assumption with a single prime p. For two specific integers p and q_0, we use the following distribution over γ-bit integers:

$$\mathcal{D}_\rho(p, q_0) := \{\mathsf{Choose}\ q \leftarrow [0, q_0), r \leftarrow \mathbb{Z} \cap (-2^\rho, 2^\rho) : \mathsf{Output}\ y = q \cdot p + r\}.$$

Definition 3 (Decisional Approximate GCD Problem: DACD). *The (ρ, η, γ)-decisional approximate GCD problem is: for a random η-bit prime p, given a γ-bit integer $x_0 = q_0 \cdot p$ and polynomially many samples from $\mathcal{D}_\rho(p, q_0)$, determine $b \in \{0, 1\}$ from $z = x + r \cdot b \mod x_0$ where $x \leftarrow \mathcal{D}_\rho(p, q_0)$ and $r \leftarrow \mathbb{Z} \cap [0, x_0)$.*

The (ρ, η, γ)-decisional approximate GCD assumption then states that this problem is hard for any polynomial time distinguisher.

We now give a sketch of the proof of the following lemma. For the detailed proof, see the full version [KLYC13].

Lemma 1. *The ℓ-decisional approximate GCD$_Q$ problem is hard under the decisional approximate GCD assumption.*

Proof (Sketch). The main difference between **DACD** and 1-**DACD**$_Q$ is the existence of Q_0 in the latter problem. When Q_0 is coprime to x_0, it is easy to see that both problems are equivalent. Namely, multiplying by Q_0 modulo x_0 efficiently converts samples from $\mathcal{D}_\rho(p, q_0)$ to $\mathcal{D}_\rho((p); (Q_0); q_0)$.

Now, we need to show that the ℓ-**DACD**$_Q$ problem is hard under the 1-**DACD**$_Q$ assumption. We use a hybrid argument. From the 1-**DACD**$_Q$ problem instance $x_0 = q_0 \cdot p$ and samples from $\mathcal{D}_\rho((p); (Q_0); q_0)$, we choose $\ell - 1$ primes ourselves. Putting a prime p from 1-**DACD**$_Q$ at a random position i_0 among the p_i's, we can construct samples from $\mathcal{D}_\rho(\boldsymbol{p}; \boldsymbol{Q}; q_0)$ using the Chinese Remainder Theorem with samples from $\mathcal{D}_\rho((p); (Q_0); q_0)$. And a set X also can be efficiently constructed. For the challenge z, we construct a ℓ-**DACD**$_Q$ challenge z' such that

$$z' = \mathsf{CRT}_{x_0, (p_i)_{i \neq i_0}}(z, r'_0 Q_0, \ldots, r'_{i_0-1} Q_{i_0-1}, e'_{i_0+1}, \ldots, e'_{\ell-1})$$

where $r'_i \leftarrow \mathbb{Z} \cap (-2^\rho, 2^\rho)$ and $e'_i \leftarrow \mathbb{Z} \cap [0, p_i)$. By the hybrid argument, it can be shown that any ℓ-**DACD**$_Q$ distinguisher can be efficiently converted to a 1-**DACD**$_Q$ distinguisher. This terminates the proof of Lemma 1. $\square$

Corollary 1. *The* IDGHV *scheme is semantically secure under the decisional approximate GCD assumption.*

3.5 Application to Secure Large Integer Arithmetic

Secure integer arithmetic is one of the most important applications of homomorphic encryption schemes. It includes frequently used statistical functions such as mean, standard deviation, logistical regression, and secure evaluation of a multivariate function over the integers. Some applications may require very large integer inputs in the computation of these functions. For the homomorphic computation of these functions, one may use FHE supporting homomorphic bit operations. However, the large ciphertext expansion and rather high cost of bootstrapping make this cumbersome and inefficient. In fact, even an addition of two λ-bit integers using bit operations needs computing degree-$O(\lambda)$ polynomial over $\mathbb{Z}_2$ due to the carry computation. For this reason, it is very important to construct an efficient somewhat homomorphic scheme supporting large integer arithmetic on encrypted data.

As mentioned earlier in this section, IDGHV scheme supports arithmetic operations on $\mathbb{Z}_Q$ with $Q = \prod_{i=0}^{\ell-1} Q_i$ when all Q_i's are pairwise coprime. We can freely choose ℓ up to $\tilde{\mathcal{O}}(\lambda^3)$ depending on the applications. And the advantage of our scheme in the overhead stands out, as the plaintext space gets larger.

4 Batch DGHV Scheme Based on the Error-Free Approximate-GCD

In this section, we present a variant of the previous IDGHV scheme but based on a (weaker) *computational* assumption instead of the decisional assumption from Def. 3. This is made possible by adding a new set of elements in the public key but yields a more intricate scheme.[6]

Ideally we would like to base the security of the new batch DGHV scheme on the same assumption as the original single-bit DGHV scheme, *i.e.* the Approximate-GCD assumption from Definition 1. However we can only show its security under the (stronger) Error-Free Approximate-GCD assumption already considered in [DGHV10, CMNT11, CNT12]. For two specific integers p and q_0, we use the following distribution over γ-bit integers:

$$\mathcal{D}_\rho(p, q_0) = \{\textsf{Choose } q \leftarrow [0, q_0), r \leftarrow \mathbb{Z} \cap (-2^\rho, 2^\rho) : \textsf{Output } y = q \cdot p + r\}.$$

Definition 4 (Error-free approximate GCD). *The (ρ, η, γ)-error-free approximate-GCD problem is: For a random η-bit prime p, given $y_0 = q_0 \cdot p$ where q_0 is a random integer in $[0, 2^\gamma/p)$, and polynomially many samples from $\mathcal{D}_\rho(p, q_0)$, output p.*

[6] For the sake of simplicity, we use $Q_0 = \cdots = Q_{\ell-1} = 2$ throughout the rest of the paper; however the security proof extends to general Q_i's as in Section 3.

In the following we briefly explain our proof strategy. In the original DGHV scheme, public-key encryption is performed by masking the message m with a random subset sum of the public key elements $x_j = q_j \cdot p + r_j$ as

$$c = \left[m + 2r + 2 \sum_{j \in S} x_j \right]_{x_0} . \tag{4}$$

The semantic security is proved by applying the Leftover Hash Lemma on the subset sum modulo q_0, and using the random $2r$ in (4) to further randomize the ciphertext modulo p.

However in the batch scheme it would not be sufficient to add such random term $2r$; namely the *same* random $2r = 2r \bmod p_i$ would be added modulo each of the p_i's, whereas for the security proof to go through these random terms should be independently distributed modulo each of the p_i's. Therefore a new technique is required to extend DGHV to a semantically secure batch encryption scheme, whose security can be based on the Error-Free Approximate-GCD problem.

In the following, we start by describing a variant of DGHV still for a single bit message m only, but which *does* extend naturally to the batch setting. We first consider the DGHV scheme without the additional random $2r$, since this term is of no use in the batch setting. A single message bit m is then encrypted as

$$c = \left[m + 2 \sum_{i \in S} x_i \right]_{x_0}$$

where $x_i = q_i \cdot p + r_i$. In order to prove semantic security as in [DGHV10], one should prove that the values q and r' given by $c = q \cdot p + 2r' + m$ are essentially random and independently distributed. The randomness of $q = 2 \sum_{i \in S} q_i \bmod q_0$ follows from the Leftover Hash Lemma (LHL) modulo q_0. However we cannot apply the LHL to $r' = \sum_{i \in S} r_i$ because it is distributed over $\mathbb{Z}$ instead of modulo an integer. Note that in the original scheme the randomness of r' followed from adding a random $2r$ in (4), much larger than the r_i's.

Let us assume that we could somehow reduce the integer variable $r' = \sum_{i \in S} r_i$ modulo some integer ϖ. Then we could apply the LHL simultaneously modulo q_0 and modulo ϖ, and the distributions of $q \bmod q_0$ and $r' \bmod \varpi$ would be independently random as required. However, during public-key encryption we certainly do not have access to the variable $r' = \sum_{i \in S} r_i$, so we cannot *a priori* reduce it modulo an integer ϖ in the encryption phase.

Our technique is the following: instead of reducing the variable r' modulo ϖ, we add a large random multiple of ϖ to r'. This can be done by extending the public key with a new element Π such that $\Pi \bmod p = \varpi$. Encryption would then be performed as

$$c = \left[m + 2b \cdot \Pi + 2 \sum_{i \in S} x_i \right]_{x_0} \tag{5}$$

for some large random integer b. Modulo p this gives a new integer $r'' = r' + b \cdot \varpi$, and we argue that this enables to proceed as if r' was actually reduced modulo ϖ.

Namely, if we generate the r_i's such that the sum $r' = \sum_{i \in S} r_i$ is not much larger than ϖ, then reducing r' modulo ϖ would just subtract a small multiple of ϖ, which is negligible compared to the large random multiple $b \cdot \varpi$ obtained through (5). Formally the distribution of $r' + b \cdot \varpi$ is statistically close to $(r' \bmod \varpi) + b \cdot \varpi$, which enables us to apply the LHL to $r' \bmod \varpi$ and eventually obtain a security proof.

Now the advantage of (5) is that it can be easily extended to the batch setting. Instead of using a single random multiple of Π, we use a subset sum of ℓ such multiples Π_i, where $\Pi_i \bmod p_j = \varpi_{i,j}$. The Leftover Hash Lemma is then applied modulo the lattice generated by the $\varpi_{i,j}$. This shows that the random noise values modulo the p_i's follow essentially independent distributions, and eventually leads to a security proof based on the Error-Free Approximate-GCD problem above.

4.1 Description

$\mathsf{BDGHV.KeyGen}(1^\lambda)$**.** Generate a collection of ℓ random η-bit primes p_j, $0 \leqslant j < \ell$, and denote π their product. Let us define the error-free public key element $x_0 = q_0 \cdot \pi$, where $q_0 \leftarrow \mathbb{Z} \cap [0, 2^\gamma/\pi)$ is a 2^{λ^2}-rough integer.
Generate the following integers x_i, x'_i and Π_i with a quotient by π uniformly and independently distributed in $\mathbb{Z} \cap [0, q_0)$, and with the following distribution modulo p_j for $0 \leqslant j < \ell$:

$$\begin{aligned}
&1 \leqslant i \leqslant \tau, && x_i \bmod p_j = 2r_{i,j},\\
&0 \leqslant i \leqslant \ell - 1, && x'_i \bmod p_j = 2r'_{i,j} + \delta_{i,j},\\
&0 \leqslant i \leqslant \ell - 1, && \Pi_i \bmod p_j = 2\varpi_{i,j} + \delta_{i,j} \cdot 2^{\rho'+1},
\end{aligned}$$

with $r_{i,j} \leftarrow \mathbb{Z} \cap (-2^{\rho'-1}, 2^{\rho'-1})$ and $r'_{i,j}, \varpi_{i,j} \leftarrow \mathbb{Z} \cap (-2^\rho, 2^\rho)$. Finally, let

$$\mathsf{pk} = \left\{ x_0, (x_i)_{1 \leqslant i \leqslant \tau}, (x'_i)_{0 \leqslant i \leqslant \ell-1}, (\Pi_i)_{0 \leqslant i \leqslant \ell-1} \right\}$$

and $\mathsf{sk} = (p_j)_{0 \leqslant j \leqslant \ell-1}$.

$\mathsf{BDGHV.Encrypt}(\mathsf{pk}, \boldsymbol{m} \in \{0,1\}^\ell)$**.** Choose random integer vectors $\boldsymbol{b} = (b_i) \in (-2^\alpha, 2^\alpha)^\tau$ and $\boldsymbol{b}' = (b'_i)_{0 \leqslant i \leqslant \ell-1} \in (-2^{\alpha'}, 2^{\alpha'})^\ell$ and output the ciphertext:

$$c = \left[\sum_{i=0}^{\ell-1} m_i \cdot x'_i + \sum_{i=0}^{\ell-1} b'_i \cdot \Pi_i + \sum_{i=1}^{\tau} b_i \cdot x_i \right]_{x_0}. \tag{6}$$

$\mathsf{BDGHV.Decrypt}(\mathsf{sk}, c)$**.** Output $\boldsymbol{m} = (m_0, \ldots, m_{\ell-1})$ where $m_j \leftarrow [c]_{p_j} \bmod 2$.

$\mathsf{BDGHV.Add}(\mathsf{pk}, c_1, c_2)$**.** Output $c_1 + c_2 \bmod x_0$

$\mathsf{BDGHV.Mult}(\mathsf{pk}, c_1, c_2)$**.** Output $c_1 \cdot c_2 \bmod x_0$.

4.2 Parameters and Correctness

The parameters must meet the following constraints (where λ is the security parameter):

- $\rho = \Omega(\lambda)$ to avoid brute force attack on the noise [CN12, CNT12],
- $\eta \geqslant \alpha' + \rho' + 1 + \log_2(\ell)$ for correct decryption,
- $\eta \geqslant \rho \cdot \Theta(\lambda \log^2 \lambda)$ for homomorphically evaluating the "squashed decryption" circuit
- $\gamma = \omega(\eta^2 \cdot \log \lambda)$ in order to thwart lattice-based attacks [DGHV10, CMNT11];
- $\rho' \geqslant \rho + \lambda$ and $\alpha' \geqslant \alpha + \lambda$ for the proof of semantic security,
- $\alpha \cdot \tau \geqslant \gamma + \lambda$ and $\tau \geqslant \ell \cdot (\rho' + 2) + \lambda$ in order to apply the leftover hash lemma.

To satisfy the above constraints one can take $\rho = 2\lambda$, $\eta = \tilde{\mathcal{O}}(\lambda^2)$, $\gamma = \tilde{\mathcal{O}}(\lambda^5)$, $\alpha = \tilde{\mathcal{O}}(\lambda^2)$, $\tau = \tilde{\mathcal{O}}(\lambda^3)$ as in [CNT12], with $\rho' = \tilde{\mathcal{O}}(\lambda)$, $\alpha' = \tilde{\mathcal{O}}(\lambda^2)$ and $\ell = \tilde{\mathcal{O}}(\lambda^2)$. We refer to Section 5.4 for concrete parameters and timings. We show in [CLT13, Appendix A] that the above scheme is correct for a set of permitted circuits.

4.3 Semantic Security

To prove the semantic security of our scheme, we first introduce a temporary decisional assumption that is implied by the Error-Free Approximate-GCD assumption.

Given integers q_0 and $p_0, \ldots, p_{\ell-1}$, we define the oracle $\mathcal{O}_{q_0,(p_i)}(\boldsymbol{v})$ which, given as input a vector $\boldsymbol{v} \in \mathbb{Z}^\ell$, outputs x with

$$x = \mathsf{CRT}_{q_0,(p_i)}(q, v_0 + 2r_0, \ldots, v_{\ell-1} + 2r_{\ell-1})$$

where $q \leftarrow [0, q_0)$ and $r_i \leftarrow (-2^\rho, 2^\rho)$. Therefore $\mathcal{O}_{q_0,(p_i)}(\boldsymbol{v})$ outputs a ciphertext for the plaintext $\boldsymbol{v}$. Note that the components v_i can be any integer, not only $0, 1$.

Definition 5 (O-ℓ-dAGCD$_{\lambda,\gamma,\eta}$). *The oracle ℓ-decisional-approximate-GCD problem is as follows. Pick random η-bit integers $p_0, \ldots, p_{\ell-1}$ of product π, a random 2^{λ^2}-rough $q_0 \leftarrow \mathbb{Z} \cap [0, 2^\gamma/\pi)$, a random bit b, set $\boldsymbol{v}_0 = (0, \ldots, 0)$ and $\boldsymbol{v}_1 \leftarrow \{0,1\}^\ell$. Given $x_0 = q_0 p_0 \cdots p_{\ell-1}$, $z = \mathcal{O}_{q_0,(p_i)}(\boldsymbol{v}_b)$ and oracle access to $\mathcal{O}_{q_0,(p_i)}$, guess b.*

This decisional problem is somehow to distinguish between an encryption of 0 and an encryption of a random message. To prove the semantic security of our scheme, we must show that this still holds when using the public-key encryption procedure instead of the oracle $\mathcal{O}_{q_0,(p_i)}$; this essentially amounts to applying a variant of the Leftover Hash Lemma. We refer to the full version [CLT13] for the proof.

Theorem 2. *The batch DGHV scheme is semantically secure under the oracle ℓ-decisional-approximate-GCD assumption.*

Lemma 2. *The oracle-ℓ-decisional-approximate-GCD problem is hard if the error-free-approximate-GCD problem is hard.*

Corollary 2. *The batch DGHV scheme is semantically secure under the error-free-approximate-GCD assumption.*

5 Making the Scheme Fully Homomorphic

In this section, we follow Gentry's blueprint [Gen09] to transform a somewhat homomorphic encryption scheme into a fully homomorphic encryption scheme. This technique applies directly to both schemes described in Section 3 and 4.

5.1 The Squashed Scheme

As mentioned in the introduction, to follow Gentry's blueprint and make our somewhat homomorphic schemes amenable to bootstrapping, we first need to squash the decryption circuit, *i.e.* change the decryption procedure so as to express it as a low degree polynomial in the bits of the secret key.

We use the same technique as in the original DGHV scheme [DGHV10] but generalize it to the batch setting. We add to the public key a set $\boldsymbol{y} = \{y_0, \ldots, y_{\Theta-1}\}$ of rational numbers in $[0, 2)$ with κ bits of precision after the binary point, such that for all $0 \leqslant j \leqslant \ell - 1$ there exists a sparse subset $S_j \subset [0, \Theta - 1]$ of size θ with $\sum_{i \in S_j} y_i \simeq 1/p_j \bmod 2$. The secret-key is replaced by the indicator vector of the subsets S_j. Formally the scheme is modified as follows:

$\mathsf{BDGHV.KeyGen}(1^\lambda)$. Generate $\mathsf{sk}^* = (p_0, \ldots, p_{\ell-1})$ and pk^* as before. Set $x_{p_j} \leftarrow \lfloor 2^\kappa / p_j \rceil$ for $j = 0, \ldots, \ell - 1$. Choose at random Θ-bit vectors $\boldsymbol{s}_j = (s_{j,0}, \ldots, s_{j,\Theta-1})$, each of Hamming weight θ, for $0 \leqslant j < \ell$. Choose at random Θ integers $u_i \in [0, 2^{\kappa+1})$ for $0 \leqslant i < \Theta$, fulfilling the condition that $x_{p_j} = \sum_{i=0}^{\Theta-1} s_{j,i} \cdot u_i \bmod 2^{\kappa+1}$ for all j. Set $y_i = u_i / 2^\kappa$ and $\boldsymbol{y} = (y_0, \ldots, y_{\Theta-1})$. Hence, each y_i is a positive number smaller than two, with κ bits of precision after the binary point, and verifies

$$\frac{1}{p_j} = \sum_{i=0}^{\Theta-1} s_{j,i} \cdot y_i + \varepsilon_j \bmod 2 \tag{7}$$

for some $|\varepsilon_j| < 2^{-\kappa}$. Finally, output the key pair

$$\mathsf{sk} = (\boldsymbol{s}_0, \ldots, \boldsymbol{s}_{\ell-1}) \quad \text{and} \quad \mathsf{pk} = (\mathsf{pk}^*, y_0, \ldots, y_{\Theta-1}) \, .$$

$\mathsf{BDGHV.Expand}(\mathsf{pk}, c)$. The ciphertext expansion procedure takes as input a ciphertext c and computes an expanded ciphertext: for every $0 \leqslant i \leqslant \Theta - 1$, compute z_i given by $z_i = \lfloor c \cdot y_i \rceil \bmod 2$ with n bits of precision after the binary point. Define the vector $\boldsymbol{z} = (z_i)_{i=0,\dots,\Theta-1}$ and output the expanded ciphertext $(c, \boldsymbol{z})$.

$\mathsf{BDGHV.Decrypt}(\mathsf{sk}, c, \boldsymbol{z})$. Output $\boldsymbol{m} = (m_0, \dots, m_{\ell-1})$ with

$$m_j \leftarrow \left[\left\lfloor \sum_{i=0}^{\Theta-1} s_{j,i} \cdot z_i \right\rceil \right]_2 \oplus (c \bmod 2) \; . \tag{8}$$

This completes the description of the scheme. We use $n = \lceil \log_2(\theta + 1) \rceil$ as in [CMNT11].

5.2 Bootstrapping

As in [DGHV10], we get that the BDGHV scheme is bootstrappable. Moreover, the Recrypt procedures works naturally in *parallel* over the plaintext bits.

Namely the decryption equation (8) for the batch scheme can be evaluated homomorphically by providing for all $0 \leqslant i < \Theta$ an encryption σ_i of the ℓ secret-key bits $s_{j,i}$, with:

$$\sigma_i = \mathsf{Encrypt}(s_{0,i}, \dots, s_{\ell-1,i}) \; .$$

This gives a new ciphertext that encrypts the same ℓ-bit plaintext vector, but with a (possibly) reduced noise. Notice that the ℓ equations in (8) are homomorphically evaluated in parallel, one in each of the ℓ plaintext slots of the ciphertext. Therefore, with the same complexity as a single Recrypt operation in the original scheme, the batch Recrypt operation is performed in parallel over the ℓ slots.

From Gentry's theorem, we obtain a homomorphic encryption scheme for circuits of any depth. The proof of the following theorem is identical to the proof of [CMNT11, Theorem 5.1].

Theorem 3. *Let $\mathcal{E}$ be the above scheme, and let $D_{\mathcal{E}}$ be the set of augmented (squashed) decryption circuits. Then $D_{\mathcal{E}} \subset C(\mathcal{P}_{\mathcal{E}})$.*

5.3 Complete Set of Operations for Plaintext Vectors

From what precedes, we can implement homomorphic SIMD-type operations on our packed ciphertexts, where the Add and Mult operations are applied to ℓ different input bits at once. However, a desired feature when dealing with packed ciphertexts is the ability to move values between plaintext slots with a public Permute operation. As opposed to [GHS12a] we cannot rely on an underlying algebraic structure. Instead we show how to perform such Permute at ciphertext refresh time, *i.e.* when performing a Recrypt. This feature is therefore supported at no extra cost assuming a ciphertext refresh operation has to be carried out anyway (*i.e.* after each Mult gate). Notice that a similar technique was independently described in [BGH13] for the RLWE-based fully homomorphic schemes [BV11a, BV11b, GHS12a].

For any permutation ζ over $\{0, \ldots, \ell-1\}$, we want to homomorphically evaluate the function

$$\ell\text{-}\mathsf{Permute}\left(\zeta, (u_0, \ldots, u_{\ell-1})\right) = \left(u_{\zeta(0)}, \ldots, u_{\zeta(\ell-1)}\right).$$

Let ζ be a permutation to be applied homomorphically on the plaintext bits. During the KeyGen operation, the authority can define for each $i \in [0, \Theta-1]$

$$\sigma_i^{\zeta} = \mathsf{Encrypt}(s_{\zeta(0),i}, \ldots, s_{\zeta(\ell-1),i}).$$

Now, performing the ciphertext refresh operation ("recryption") with the σ_i^{ζ}'s instead of the σ_i's gives a ciphertext of the plaintext vector $(m_{\zeta(0)}, \ldots, m_{\zeta(\ell-1)})$ which is exactly the desired result. Therefore any permutation ζ can be implemented by putting the corresponding σ_i^{ζ}'s in the public key.

To be able to perform arbitrary permutations on the plaintext vector, one can augment the public key by a minimal set of permutations ζ's that generates the whole permutation group $\mathfrak{S}_\ell$ over $\{0, \ldots, \ell-1\}$, such as the transposition $(1, 2)$ and the cycle $(1, 2, \ldots, \ell)$. In that case the impact on the public key is small (as only $2 \cdot \Theta \cdot \gamma$ bits are added), but the performance overhead is significant, since as many as $\mathcal{O}(\ell)$ ciphertext refresh operations may be needed to carry out a desired permutation.

A more practical solution is to use a Beneš network [Ben64] of permutations as in [GHS12a]. In that case it suffices to add $2\log_2(\ell)$ permuting elements to the public key to enable circular rotations by $\pm 2^i$ bit position. Then any permutation can be obtained in $(2\log(\ell) - 1)$ steps. At each step, at most two rotations and two Select operations are performed, where the Select operation on c_1 and c_2 constructs a ciphertext where each of the ℓ plaintext slot is chosen either from c_1 or c_2; such Select operation is easily obtained with two Mult (and two recryptions) and one Add, see [GHS12a]. This approach has a limited impact on the public key ($2\log_2(\ell) \cdot \Theta \cdot \gamma$ more bits), and any permutation can then be performed with at most $6 \cdot (2\log_2 \ell - 1)$ recryptions.

In practice, however, the circuit to be homomorphically evaluated is likely to be known in advance, so it is possible to put a set of distinguished permutations in the public key that provides an optimal time-memory trade-off. In the next section, we describe two variants of homomorphic evaluations of the full AES circuit that require respectively only *four* permutations and no permutation at all.

5.4 Implementation Results

We provide in Table 1 concrete key sizes and timings for the batch DGHV scheme, based on a C++ implementation using the GMP library. We use essentially the same parameters as in [CNT12, CT12]; in particular, the parameters take into account the attack from [CN12]. We use the same compressed public-key variant as in [CNT12]; a complete description of the scheme is given in [CLT13]. As in [CMNT11, CNT12], we take $n = 4$ and $\theta = 15$ for all security levels.

We obtain essentially the same running times as in [CNT12]. The main difference is that the Recrypt operation is now performed in parallel over $\ell = 531$ bits (for the "Large" setting) instead of a single bit.

Table 1. Benchmarking for our Batch DGHV with a compressed public key on a desktop computer (Intel Core i7 at 3.4Ghz, 32GB RAM)

Instance	λ	ℓ	ρ	η	$\gamma/10^6$	τ	Θ	pk size	KeyGen	Encrypt	Decrypt	Expand	Recrypt
Small	52	37	41	1558	0.90	661	555	13 MB	1.74s	0.23s	0.02s	0.08s	1.10s
Medium	62	138	56	2128	4.6	2410	2070	304 MB	73s	3.67s	0.45s	1.60s	11.9s
Large	72	531	71	2698	21	8713	7965	5.6 GB	3493s	61s	9.8s	28s	172s

6 Homomorphic Evaluation of the AES Circuit

In this section, we show how to homomorphically evaluate the AES-128 encryption circuit using the batch encryption scheme of Section 4 with compressed public key elements (see [CLT13]), and provide concrete timings. A similar implementation with the RLWE-based fully-homomorphic encryption scheme [BV11a, BV11b, GHS12a] was already described in [GHS12b]. As mentioned in [SV11, NLV11, GHS12b], such an implementation can be used to optimize the communication cost in cloud-based applications. Indeed, since the ciphertext expansion ratio in most fully-homomorphic encryption schemes is huge, data can rather be sent encrypted under AES with a ciphertext expansion equal to 1, along with the public key $\mathsf{pk}_{\mathsf{FHE}}$ of the FHE scheme as well as the AES secret-key encrypted under $\mathsf{pk}_{\mathsf{FHE}}$. Then, before the cloud performs homomorphic operations on the data, it can first run the AES decryption algorithm homomorphically to obtain the plaintext data encrypted under $\mathsf{pk}_{\mathsf{FHE}}$.

We consider our BDGHV scheme with ℓ slots. We describe two variants of our implementation which we call *byte-wise bitslicing* and *state-wise bitslicing* .

Byte-Wise Bitslicing. In this representation, the 16-byte AES state is viewed as a matrix of 16 rows of 8 bits each (one row for every byte). Each of the 8 columns is then stored on a different ciphertext. Therefore an AES state is stored in 8 ciphertexts, and one can perform $k = \ell/16$ AES encryptions in parallel using these 8 ciphertexts. Formally the AES state is composed of the ciphertexts $c_0, \ldots, c_7$, where the underlying plaintexts $\boldsymbol{m}_0, \ldots, \boldsymbol{m}_7$ are such that $\boldsymbol{m}_i[k \cdot 16 + j]$ is the i-th bit of the j-th element of the AES state of the k-th AES (see Figure 1).[7] We briefly describe how to implement the AES stages; full details on the implementation are given in [CLT13].

The AddRoundKey stage performs a XOR between the AES state and the current round key. This operation only consists of 8 Add operations. To minimize

[7] Thus, $\boldsymbol{m}_0$ represents the LSBs of the AES states of the k AES-plaintexts, and $\boldsymbol{m}_7$ the MSBs. This construction is similar to general-purpose bitslicing [Bih97, KS09].

Column 0																···	Column 3								
Row 0				Row 1				Row 2				Row 3				···	Row 0				···	Row 3			
AES 1	AES 2	···	AES k	AES 1	AES 2	···	AES k	AES 1	AES 2	···	AES k	AES 1	AES 2	···	AES k	···	AES 1	AES 2	···	AES k	···	AES 1	AES 2	···	AES k

Fig. 1. Bit ordering in $\boldsymbol{m}_i$ in the byte-wise bitslicing representation

the number of Recrypt during the SubBytes stage, we used the 115 gates circuit of Boyar and Peralta [BP10] to compute the Sbox.[8] Thus, this step needs 17 Recrypt operations on 9 of the temporary variables and on the 8 outputs. In total, this stage costs 83 Add, 32 Mult and 17 Recrypt.

The ShiftRows stage consists in performing a permutation of the state. For this we add the σ_i^ζ's of the associated permutation ζ in the public key, and the rotation is performed at no additional cost during the final Recrypt of the SubBytes stages. Finally the MixColumns stage requires 3 permutations of the AES state; this yields a total of $3 \times 8 = 24$ Recrypt and 38 Add, and the addition of the σ_i^ζ's of three permutations ζ to the public key.

In total, our byte-wise implementation of AES requires 1260 Add, 320 Mult, and 377 Recrypt.

State-Wise Bitslicing. In this representation, each of the 128 bits of the AES state is stored in a different ciphertext. One can then perform $k = \ell$ AES encryptions in parallel. This corresponds to a full bitslice implementation of AES. More precisely the AES state is composed of 128 ciphertexts $c_0, \ldots, c_{127}$, where the underlying plaintexts $\boldsymbol{m}_0, \ldots, \boldsymbol{m}_{127}$ are such that $\boldsymbol{m}_{i+j\cdot 8}[k]$ is the i-th bit of the j-th byte of the state of the k-th AES.

The AddRoundKey stage requires 128 Add operations. The SubBytes stage is implemented using the same circuit as above. Since the circuit needs to be evaluated on each of the 16 bytes of the AES state, the stage costs $16 \times 83 = 1328$ Add, $16 \times 32 = 512$ Mult, and $16 \times 17 = 272$ Recrypt. The ShiftRows stage consists in performing a permutation of the state, and this is done by permuting the indices of bits in the homomorphic AES state at no additional cost. The MixColumns stage requires 608 Add. The total cost the AES evaluation is then 14688 Add, 5120 Mult and 2448 Recrypt.

Implementation Results. We implemented both variants using the concrete parameters from Table 1; our results are summarized in Table 2. The relative time is the total time of AES evaluation divided by the number of encryptions processed in parallel. Notice that the state-wise bitslicing variant yields better relative times.

Our timings are comparable to [GHS12b] for the RLWE-based scheme, where a relative time of 5 minutes per block is reported; the authors used a 24-core server with 256GB of RAM, while our program runs on a more modest desktop computer with 4 hyper-threaded cores and 32GB of RAM (the whole public key

[8] To minimize the number of bootstrappings in a given circuit we refer to [LP13].

Table 2. Timings of byte-wise and state-wise homomorphic AES developed in C++ with GMP, running on a desktop computer (Intel Core i7 at 3.4Ghz, 32GB RAM)

(a) Timings for byte-wise representation

Instance	λ	ℓ	# of enc. in parallel	Add-RoundKey	ShiftRows & SubBytes	Mix-Columns	**Total AES** (in hours)	**Relative time**
Small	52	48	3	0.04s	21s	29s	**0.125**	2min 30s
Medium	62	144	9	0.3s	210s	290s	**1.25**	8min 20s
Large	72	528	33	1.6s	2970s	4165s	**18.3**	33min

(b) Timings for state-wise representation

Instance	λ	ℓ	# of enc. in parallel	Add-RoundKey	Sub-Bytes	Shift-Rows	Mix-Columns	**Total AES** (in hours)	**Relative time**
Small	52	37	37	0.06s	309s	0s	0.09s	**0.74**	1min 12s
Medium	62	138	138	4.5s	3299s	0s	0.44s	**7.86**	3min 25s
Large	72	531	531	27s	47656s	0.04s	2.8s	**113**	12min 46s

fits in RAM). We claim a slightly lower security level, however: 72 bits versus 80 bits for the implementation from [GHS12b].

Acknowledgments. We would like to thank Taekyoung Kwon and Hyung Tae Lee for valuable comments. The first, third, and fourth authors were supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MEST) (No. 2012-0001243). The fourth author was also supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (No. 2012R1A1A2039129). The last author was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MEST) (No. 2011–0025127).

References

[Ben64] Beneš, V.E.: Optimal rearrangeable multistage connecting networks. Bell Systems Technical Journal 43(7), 1641–1656 (1964)

[BGH13] Brakerski, Z., Gentry, C., Halevi, S.: Packed ciphertexts in LWE-based homomorphic encryption. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 1–13. Springer, Heidelberg (2013)

[BGV12] Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. In: Goldwasser, S. (ed.) ITCS 2012, pp. 309–325. ACM (2012)

[BHY09] Bellare, M., Hofheinz, D., Yilek, S.: Possibility and impossibility results for encryption and commitment secure under selective opening. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 1–35. Springer, Heidelberg (2009)

[Bih97] Biham, E.: A fast new DES implementation in software. In: Biham, E. (ed.) FSE 1997. LNCS, vol. 1267, pp. 260–272. Springer, Heidelberg (1997)

[BP10] Boyar, J., Peralta, R.: A new combinational logic minimization technique with applications to cryptology. In: Festa, P. (ed.) SEA 2010. LNCS, vol. 6049, pp. 178–189. Springer, Heidelberg (2010)

[Bra12] Brakerski, Z.: Fully homomorphic encryption without modulus switching from classical GapSVP. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 868–886. Springer, Heidelberg (2012)

[BV11a] Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) LWE. In: Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, pp. 97–106. IEEE Computer Society (2011)

[BV11b] Brakerski, Z., Vaikuntanathan, V.: Fully homomorphic encryption from Ring-LWE and security for key dependent messages. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 505–524. Springer, Heidelberg (2011)

[CH11] Cohn, H., Heninger, N.: Approximate common divisors via lattices. Cryptology ePrint Archive, Report 2011/437 (2011), http://eprint.iacr.org

[CLT13] Coron, J.-S., Lepoint, T., Tibouchi, M.: Batch fully homomorphic encryption over the integers. Cryptology ePrint Archive, Report 2013/036 (2013), http://eprint.iacr.org

[CMNT11] Coron, J.-S., Mandal, A., Naccache, D., Tibouchi, M.: Fully homomorphic encryption over the integers with shorter public keys. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 487–504. Springer, Heidelberg (2011)

[CN12] Chen, Y., Nguyen, P.Q.: Faster algorithms for approximate common divisors: Breaking fully-homomorphic-encryption challenges over the integers. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 502–519. Springer, Heidelberg (2012)

[CNT12] Coron, J.-S., Naccache, D., Tibouchi, M.: Public key compression and modulus switching for fully homomorphic encryption over the integers. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 446–464. Springer, Heidelberg (2012)

[CT12] Coron, J.-S., Tibouchi, M.: Implementation of the fully homomorphic encryption scheme over the integers with compressed public keys in sage (2012), https://github.com/coron/fhe

[DGHV10] van Dijk, M., Gentry, C., Halevi, S., Vaikuntanathan, V.: Fully homomorphic encryption over the integers. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 24–43. Springer, Heidelberg (2010)

[Gen09] Gentry, C.: A fully homomorphic encryption scheme. PhD thesis, Stanford University (2009), http://crypto.stanford.edu/craig

[GH11] Gentry, C., Halevi, S.: Implementing Gentry's fully-homomorphic encryption scheme. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 129–148. Springer, Heidelberg (2011)

[GHS12a] Gentry, C., Halevi, S., Smart, N.P.: Fully homomorphic encryption with polylog overhead. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 465–482. Springer, Heidelberg (2012)

[GHS12b] Gentry, C., Halevi, S., Smart, N.P.: Homomorphic evaluation of the AES circuit. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 850–867. Springer, Heidelberg (2012)

[HG01] Howgrave-Graham, N.: Approximate integer common divisors. In: Silverman, J.H. (ed.) CaLC 2001. LNCS, vol. 2146, pp. 51–66. Springer, Heidelberg (2001)

[KLYC13] Kim, J., Lee, M.S., Yun, A., Cheon, J.H.: CRT-based fully homomorphic encryption over the integers. Cryptology ePrint Archive, Report 2013/057 (2013), `http://eprint.iacr.org`

[KS09] Käsper, E., Schwabe, P.: Faster and timing-attack resistant AES-GCM. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 1–17. Springer, Heidelberg (2009)

[Lag85] Lagarias, J.C.: The computational complexity of simultaneous diophantine approximation problems. SIAM J. Comput. 14(1), 196–209 (1985)

[LATV12] López-Alt, A., Tromer, E., Vaikuntanathan, V.: On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In: Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, pp. 1219–1234. ACM (2012)

[Len87] Lenstra Jr., H.W.: Factoring integers with elliptic curves. The Annals of Mathematics 126(3), 649–673 (1987)

[LP13] Lepoint, T., Paillier, P.: On the minimal number of bootstrappings in homomorphic circuits. In: WAHC 2013. LNCS. Springer, Heidelberg (to appear, 2013)

[Mem12] Memoirs of the 6th Cryptology Paper Contest, arranged by Korea Communications Commission (2012)

[NLV11] Naehrig, M., Lauter, K., Vaikuntanathan, V.: Can homomorphic encryption be practical? In: Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop, CCSW 2011, pp. 113–124. ACM (2011)

[SV10] Smart, N.P., Vercauteren, F.: Fully homomorphic encryption with relatively small key and ciphertext sizes. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 420–443. Springer, Heidelberg (2010)

[SV11] Smart, N.P., Vercauteren, F.: Fully homomorphic SIMD operations. To appear in Designs, Codes and Cryptography (2011)

Practical Homomorphic MACs for Arithmetic Circuits

Dario Catalano[1] and Dario Fiore[2,⋆]

[1] Dipartimento di Matematica e Informatica, Università di Catania, Italy
catalano@dmi.unict.it
[2] Max Planck Institute for Software Systems (MPI-SWS), Germany
fiore@mpi-sws.org

Abstract. Homomorphic message authenticators allow the holder of a (public) evaluation key to perform computations over previously authenticated data, in such a way that the produced tag σ can be used to certify the authenticity of the computation. More precisely, a user knowing the secret key sk used to authenticate the original data, can verify that σ authenticates the correct output of the computation. This primitive has been recently formalized by Gennaro and Wichs, who also showed how to realize it from fully homomorphic encryption. In this paper, we show new constructions of this primitive that, while supporting a smaller set of functionalities (i.e., polynomially-bounded arithmetic circuits as opposite to boolean ones), are much more efficient and easy to implement. Moreover, our schemes can tolerate any number of (malicious) verification queries. Our first construction relies on the sole assumption that one way functions exist, allows for arbitrary composition (i.e., outputs of previously authenticated computations can be used as inputs for new ones) but has the drawback that the size of the produced tags grows with the degree of the circuit. Our second solution, relying on the D-Diffie-Hellman Inversion assumption, offers somewhat orthogonal features as it allows for very short tags (one single group element!) but poses some restrictions on the composition side.

1 Introduction

Cloud Computing allows a user to outsource his data to remote service providers in such a way that he can later access the data from multiple platforms (e.g., his desktop at work, his laptop, his smartphone, etc.), and virtually from everywhere. Moreover, using this paradigm, even clients with very limited storage capacity (e.g., smart phones) can have access "on demand" to very large amounts of data. Having access to the outsourced data does not necessarily mean only to retrieve such data. Indeed, a user may wish to perform a computation on (a subset of) the outsourced data, and this too can be delegated to the service provider. These and other benefits are the key success of Cloud Computing. The paradigm, however, raises security concerns essentially because cloud providers

⋆ Work done while Postdoctoral researcher at NYU.

cannot always be trusted. One problem is related to preserving the privacy of the outsourced data. This question has been successfully addressed by the recent work on fully homomorphic encryption [24]. The second question deals with enforcing the *authenticity* of the computations performed on the outsourced data, and is the focus of this work. In a nutshell, this problem can be described as follows. Assume that a client outsources a collection of data $m_1, \ldots, m_n$ to a server, and later asks the server to run a program $\mathcal{P}$ over $(m_1, \ldots, m_n)$. The server computes $m \leftarrow \mathcal{P}(m_1, \ldots, m_n)$ and sends m to the client. The problem here is that the client wants to be sure that m is the value obtained by running $\mathcal{P}$ on its own data. A trivial solution would be to have the server send $m_1, \ldots, m_n$ to the client, who can then compute/check $m = \mathcal{P}(m_1, \ldots, m_n)$ by itself. This however vanishes the advantages of the outsourcing and is too costly in terms of bandwidth. Therefore, the main goal here is to find solutions in which the server can authenticate the output of the computation by sending some value whose size is much shorter than $m_1, \ldots, m_n$. Such property is also motivated by the fact that, in spite of the continuous progress in increasing the computational power of small devices, bandwidth (especially in mobile data connections) seems to remain the most serious and expensive bottleneck.

The research community has recently put a notable effort in developing new cryptographic tools that can help in solving this and related problems. It is the case, for instance, for works on verifiable computation [28,29,26,21,17,3] and memory delegation [18].

Another line of research has explored the idea of enabling computation on authenticated data [2] by means of homomorphic authentication primitives.

In the public key setting Boneh and Freeman introduced the notion of *(fully) homomorphic signatures* [11]. Roughly speaking, a homomorphic signature allows a user to generate signatures $\sigma_1, \ldots, \sigma_n$ on messages $m_1, \ldots, m_n$ so that later anyone (without knowledge of the signing key) can compute a signature σ that is valid for the value $m = f(m_1, \ldots, m_n)$. Boneh and Freeman also showed a realization of homomorphic signatures for bounded (constant) degree polynomials, from ideal lattices.

Very recently, Gennaro and Wichs proposed, formally defined and constructed the secret-key analogue of homomorphic signatures, that is *homomorphic message authenticators* (homomorphic MACs, for short) [23]. Their construction makes use of fully homomorphic encryption and allows to evaluate every circuit.

In this work, we continue the study of homomorphic MACs and propose new constructions which, while less general than that given in [23], are much more efficient.

Homomorphic Message Authenticators. Informally, a homomorphic MAC scheme enables a user to use his secret key for generating a tag σ which authenticates a message m so that later, given a set of tags $\sigma_1, \ldots, \sigma_n$ authenticating messages $m_1, \ldots, m_n$ respectively, anyone can homomorphically execute a program $\mathcal{P}$ over $(\sigma_1, \ldots, \sigma_n)$ to generate a short tag σ that authenticates m as the output of $\mathcal{P}(m_1, \ldots, m_n)$. Given such a primitive, it is not hard to imagine how it can be employed to solve the problem of verifying computations on outsourced data. However, the above description needs some refinements, in particular to explain what means to authenticate a message as the output of a program. To do this Gennaro and Wichs introduce the notion of *labeled data and programs.* The *label* τ of a data m is some binary string τ chosen by the user to authenticate m, i.e., $\sigma \leftarrow \mathsf{Auth}(\mathsf{sk}, \tau, m)$. One can think of labels as some indexing of the data. For example, assume that a company outsources a database with informations on its customers, in which each column contains a different attribute (e.g., age, expended amount, etc.). Then, to authenticate the "age" column of the database the user can define a label "(age, i)" for the age value in record i. On the other hand, a *labeled program* $\mathcal{P}$ is defined by a circuit f and a set of labels $\tau_1, \ldots, \tau_n$, one for each input wire of f. This can be seen as a way to specify on which inputs the circuit should be evaluated upon, without knowing the input values themselves. So, given a labeled program $\mathcal{P} = (f, \tau_1, \ldots, \tau_n)$ and a set of tags $\sigma_1, \ldots, \sigma_n$ that authenticate messages m_i under label τ_i, anyone can run the homomorphic evaluation algorithm $\sigma \leftarrow \mathsf{Eval}(\mathcal{P}, \sigma_1, \ldots, \sigma_n)$ whose output σ will authenticate $m = \mathcal{P}(m_1, \ldots, m_n)$. Precisely, the secret-key verification algorithm takes as input a triple $(m, \mathcal{P}, \sigma)$ and verifies that m is the output of the program $\mathcal{P}$ run on some previously authenticated and labeled messages, without knowing such messages themselves.

Informally, homomorphic MACs are secure if any adversary who can adaptively query tags for messages of its choice cannot produce a valid tag σ that authenticates m as the output of $\mathcal{P}$ unless σ can be honestly computed by applying Eval on the queried tags.

Homomorphic MACs are also required to be *succinct.* Informally, succinctness requires that the output of $\mathcal{P}$ run over (previously) authenticated data can be certified with significantly less communication than that of sending the original inputs. Another property one might want from homomorphic MACs is *composability*, which allows to combine tags authenticating previous computations to create a tag that authenticates a composition of such computations. More precisely, given tags $\sigma_1, \ldots, \sigma_t$ that authenticate $m_1, \ldots, m_t$ as the outputs of $\mathcal{P}_1, \ldots, \mathcal{P}_t$ respectively, composability allows to further compute $\sigma \leftarrow \mathsf{Eval}(\mathcal{P}, \sigma_1, \ldots, \sigma_t)$ which authenticates $m = \mathcal{P}(m_1, \ldots, m_t)$ as the output of $\mathcal{P}^*$, the composed program obtained by running $\mathcal{P}$ on the outputs of $\mathcal{P}_1, \ldots, \mathcal{P}_t$.

1.1 Our Contribution

In this paper we propose the first practically efficient constructions of homomorphic MACs. The most attractive feature of our schemes is that they are efficient, simple to implement and rely on well studied assumptions. Moreover, they are secure against PPT adversaries that can make an unbounded number of verification queries, as opposite to the construction in [23] that supports only an a-priori bounded number of verification queries (see next section for more details about this). On the negative side our solution works only for functionalities that can be expressed as arithmetic circuits with certain additional restrictions that we describe below.

Our first construction is surprisingly simple and relies only on the existence of pseudorandom functions. While it offers arbitrary composition, it does not achieve full succinctness. More precisely, the size of the authentication tags grows with the degree d of the circuit[1], and thus we are able to guarantee succinct authenticators only when d is smaller than the input size n.

Our second construction enjoys succinct, constant-size tags (just one group element!) but only supports a limited form of composition . More precisely, for a fixed bound D (polynomial in the security parameter) the scheme allows to evaluate any arithmetic circuit of degree $d \leq D$. In general, the evaluation has to be done in a "single shot", that is the authentication tags obtained from the Eval algorithm cannot be used again to be composed with other tags. However, we interestingly show that the scheme achieves what we call *local composition*. The idea is that one can keep locally a non-succinct version of the tag that allows for arbitrary composition. Next, when it comes to send an authentication tag to the verifier, one can securely compress such large tag in a very compact one of constant-size. We prove the security of our second construction under the D-Diffie Hellman Inversion assumption [13,30] (where D is the bound on the maximal circuit's degree supported by the scheme).

Succinct Tags and Composition. Even though our solutions do not achieve succinctness and composition at the same time, we argue that these limitations might not be too relevant in many real life scenarios. First, we notice that several interesting functions and statistics (e.g., the standard deviation function) can be represented by constant-degree polynomials. In such a case, our first construction perfectly fits the bill as it is efficient, simple to implement and produces constant-size tags (and, of course, it only requires the existence of a PRF to be proved secure). For the case of polynomials of large degree d (i.e., d polynomial in the security parameter), our scheme fits well in those applications where composition is not needed. Think for example of the application described at the beginning of this section. There, if the server just runs $m \leftarrow \mathcal{P}(m_1, \ldots, m_n)$ on the client's data, using our second construction it can produce a succinct tag that authenticates m as $\mathcal{P}$'s output, and this tag is only one group element.

Finally, in applications where composition is needed but does not involve different parties, the notion of local composition achieved by our second scheme still allows to save in bandwidth and to (locally) compose tags of partial computations.

Overview of Our Techniques. The main idea behind our construction is a "re-interpretation" of some classical techniques for information-theoretic MACs. The authentication tag of a message $m \in \mathbb{Z}_p$ with label τ is a degree-1 polynomial $y(z) \in \mathbb{Z}_p[z]$ that evaluates to m on the point 0, and to r_τ on a random point x (i.e., $y(0) = m$ and $y(x) = r_\tau$). Here $r_\tau = F_K(\tau)$ is a pseudorandom value, unique per each label, defined by the PRF, while x is the secret key. If we do not care about the homomorphic property and we assume that each r_τ is truly random,

[1] Informally, the degree of an arithmetic circuit is related to the degree of the polynomial computed by the circuit (see next section for more details).

then this is a secure information-theoretic MAC. Now, the basic observation that allows to show the homomorphic property is the following. Let f be an arithmetic circuit and assume to evaluate the circuit over the tags (i.e., over these polynomials $y(z)$) as follows: for every additive gate we compute the addition of the two input polynomials, and for every multiplicative gate we compute the multiplication of them (i.e., the convolution of their coefficients). Now, we observe that these operations are naturally *homomorphic* with respect to the evaluation of the polynomial in every point. In particular, if we have two tags $y^{(1)}$ and $y^{(2)}$ (i.e., we are given only the coefficients of these polynomials) such that $y^{(1)}(0) = m_1$ and $y^{(2)}(0) = m_2$, then for $y = y^{(1)} + y^{(2)}$ (resp. $y = y^{(1)} * y^{(2)}$) we clearly obtain $y(0) = m_1 + m_2$ (resp. $y(0) = m_1 \cdot m_2$). The same holds for its evaluation at the random point x, i.e., $y(x) = r_{\tau_1} + r_{\tau_2}$ (resp. $y(x) = r_{\tau_1} \cdot r_{\tau_2}$). By extending this argument to the evaluation of the entire circuit f, this allows to verify a tag y for a labeled program $\mathcal{P} = (f, \tau_1, \dots, \tau_n)$ and a message m, by simply checking that $m = y(0)$ and $f(r_{\tau_1}, \dots, r_{\tau_n}) = y(x)$, where $r_{\tau_i} = F_K(\tau_i)$.

A drawback of this construction is that the tag's size grows linearly with the degree of the evaluated circuit f. The reason is that the above homomorphic evaluation increases the degree of the "tag polynomial" y at every multiplication gate. This is why this MAC fails in achieving the succinctness property when the degree d becomes greater than the input size n of the circuit.

Our second construction overcomes this drawback as follows. First, the evaluation algorithm computes a tag $y = (y_0, \dots, y_d)$ as before, and then it "accumulates" these coefficients in a single group element $\Lambda = \prod_{i=1}^{d} (g^{x^i})^{y_i}$. Verification will check that $g^{f(r_{\tau_1}, \dots, r_{\tau_n})} = g^m \cdot \Lambda$. If Λ is computed correctly, then $\Lambda = g^{y(x) - y(0)}$, and thus one can easily see why correctness holds. The need to resort to the $(D-1)$-Diffie Hellman Inversion assumption[2], comes from the fact that, in order to perform the evaluation procedure correctly, the values $g^x, g^{x^2}, \dots, g^{x^D}$ need to be published as part of the evaluation key ek. Once a tag of the Λ form is created, it can be composed with other tags of the same form only for additions but not for multiplications. To satisfy partial composition, the idea is that one can keep locally the large version of the tag consisting of the coefficients $y_0, \dots, y_d$, and always send to the verifier its compact version $\Lambda = \prod_{i=1}^{d} (g^{x^i})^{y_i}$. In the full version of this paper we also show an extension of this scheme that, by using bilinear pairings, allows to further compute an additional level of multiplications and unbounded additions on tags of the Λ form.

1.2 Related Work

Homomorphic Message Authenticators and Signatures. Recently, many papers considered the problem of realizing homomorphic (mostly linear) authenticators either in the symmetric setting (MAC) or in the asymmetric one (signatures). This line of research has been initiated by the work of Johnson et al. [27]

[2] Very briefly, this assumption states that it is computationally infeasible to compute $g^{1/x}$, given $g, g^x, g^{x^2}, \dots, g^{x^{D-1}}$

and became very popular in recent years because of the important application to linear network coding. Efficient solutions for this latter application have been proposed both in the random oracle [10,22,12,14] and in the standard model [1,4,15,16,20,5,6]. Linearly-homomorphic message authenticators have been considered also for proofs of retrievability for outsourced storage [32]. Only two works, however, consider the problem of realizing solutions for more complex functionalities (i.e., beyond linear).

Boneh and Freeman defined the notion of (fully) homomorphic signatures and showed a realization for bounded (constant) degree polynomials, from ideal lattices [11]. With respect to our work this solution has the obvious advantage of allowing for public verifiability. On the negative side it is not truly practical and the bound on the degree of the supported polynomials is more stringent than in our case (as they can support only polynomials of constant degree).

Closer to our setting is the recent work of Gennaro and Wichs [23] where fully homomorphic MACs are introduced, formally defined and constructed. The solution given there supports a wider class of functionalities with respect to ours, and it allows to achieve succinct tags and composability at the same time. Their tags have size $\mu(\lambda) = \mathsf{poly}(\lambda)$ where λ is the security parameter, and thus they are asymptotically succinct as long as the circuit's input size n is greater than $\mu(\lambda)$. Despite its nice properties, the proposed construction seems unfortunately far from being truly practical as it relies on fully homomorphic encryption. Moreover, it is proven secure only for a bounded and a-priori fixed number of verification queries[3], meaning with this that the scheme becomes insecure if the verifier leaks information on whether it accepts/rejects tags.

Succinct Non-interactive Arguments of Knowledge. The problem of realizing homomorphic signatures can be solved in theory using *Succinct Non-interactive Arguments of Knowledge* (SNARKs) [8]. In a nutshell, given any NP statement a SNARK allows to construct a succinct argument that can be used to prove knowledge of the corresponding witness. The nice feature of SNARKs is that the size of the argument is independent of the size of both the statement and the witness. A drawback of SNARKs is that they are not very efficient (or at least not nearly as practical as we require) and require either the random oracle model [29] or non-standard, non-falsifiable assumptions [25]. Moreover, SNARK-based solutions seem to allow for only very limited composability [34,9].

Other Related Work. The notion of homomorphic authenticators is also (somewhat) related to the notion of verifiable computation [28,29,26,21,17,3,7,31,19]. There, one wants to delegate a computationally heavy task to a remote server while keeping the ability to verify the result in a very efficient way. While the two primitives might seem quite different at first, one can reinterpret some of the results on verifiable computation in our setting. The resulting solutions however present several limitations that make them of limited practical interest compared to

[3] More precisely, their basic construction cannot support verification queries at all. This can be extended to allow for some fixed a-priori number of queries q at the cost of increasing by $O(q)$ the size of the tag.

homomorphic authenticators. We refer the reader to [23] for a nice discussion about this.

Homomorphic authenticators are also related to memory delegation [18]. This primitive allows a client to outsource large amounts of data to a server so that he can later verify computations on the data. The advantage of this approach over ours is that it offers an efficient verification procedure, and it supports a dynamic memory in which the client can update the outsourced data. However, current (non-interactive) realizations of memory delegation, in the standard model, are rather inefficient and require the user to keep a state. Moreover, in known constructions, efficient verification comes at the price of an offline phase where the runtime of both the delegator and the server depends polynomially on the size of the memory.

Organization. The paper is organized as follows. In Section 2 we provide a background and relevant definitions of arithmetic circuits and homomorphic authenticators. Section 3 describes our first construction from PRFs while our second compact construction is given in Section 4. For lack of space, all proofs will appear in the full version of this paper.

2 Background and Definitions

Arithmetic Circuits. Here we provide a very brief overview of arithmetic circuits. The interested reader is referred to [33] for a more detailed treatment of the subject.

An arithmetic circuit over a field $\mathbb{F}$ and a set of variables $X = \{\tau_1 \dots \tau_n\}$, is a directed acyclic graph with the following properties. Each node in the graph is called *gate*. Gates with in-degree 0 are called *input* gates (or input nodes) while gates with out-degree 0 are called *output* gates. Each input gate is labeled by either a *variable* or a *constant*. Variable input nodes are labeled with binary strings $\tau_1, \dots, \tau_n$, and can take arbitrary values in $\mathbb{F}$. A constant input node instead is labeled with some constant c and it can take only some fixed value $c \in \mathbb{F}$. Gates with in-degree and out-degree greater than 0 are called *internal* gates. Each internal gate is labeled with an arithmetic operation symbol. Gates labeled with $\times$ are called product gates, while gates labeled with $+$ are called sum gates. In this paper, we consider circuits with a single output node and where the in-degree of each internal gate is 2. The *size* of the circuit is the number of its gates. The *depth* of the circuit is the length of the longest path from input to output.

Arithmetic circuits evaluate polynomials in the following way. Input gates compute the polynomial defined by their labels. Sum gates compute the polynomial obtained by the sum of the (two) polynomials on their incoming wires. Product gates compute the product of the two polynomials on their incoming wires. The output of the circuit is the value contained on the outgoing wire of the output gate. The *degree of a gate* is defined as the total degree of the polynomial computed by that gate. The *degree of a circuit* is defined as the maximal degree of the gates in the circuit.

We stress that arithmetic circuits should be seen as computing *specific* polynomials in $\mathbb{F}[X]$ rather than functions from $\mathbb{F}^{|X|}$ to $\mathbb{F}$. In other words, when studying arithmetic circuits one is interested in the formal computation of polynomials rather than the functions that these polynomials define[4].

In this paper we restrict our interest to families of polynomials $\{f_n\}$ over $\mathbb{F}$ which have *polynomially bounded degree*, meaning with this that both the number of variables and the degree of f_n are bounded by some polynomial $p(n)$. The class **VP** (also known as $\mathbf{AlgP_{/poly}}$) contains all such polynomials. More precisely it contains all polynomially bounded degree families of polynomials that are computable by arithmetic circuits of polynomial size and degree.

2.1 Homomorphic Message Authenticators

Labeled Programs. First, we recall the notion of labeled programs introduced by Gennaro and Wichs in [23]. A labeled program $\mathcal{P}$ consists of a tuple $(f, \tau_1, \ldots, \tau_n)$ where $f : \mathbb{F}^n \to \mathbb{F}$ is a circuit, and the binary strings $\tau_1, \ldots, \tau_n \in \{0,1\}^*$ are the *labels* of the input nodes of f. Given some labeled programs $\mathcal{P}_1, \ldots, \mathcal{P}_t$ and a function $g : \mathbb{F}^t \to \mathbb{F}$ it is possible to define the *composed program* $\mathcal{P}^* = g(\mathcal{P}_1, \ldots, \mathcal{P}_t)$ which consists in evaluating a circuit g on the outputs of $\mathcal{P}_1, \ldots, \mathcal{P}_t$ respectively. The labeled inputs of $\mathcal{P}^*$ are all distinct labeled inputs of $\mathcal{P}_1, \ldots, \mathcal{P}_t$, i.e., all inputs with the same label are put together in a single input of the new program. We denote with $\mathcal{I}_\tau = (g_{id}, \tau)$ the *identity program* with label τ where g_{id} is the canonical identity function and $\tau \in \{0,1\}^*$ is some input label. Finally, we notice that any program $\mathcal{P} = (f, \tau_1, \ldots, \tau_n)$ can be expressed as the composition of n identity programs $\mathcal{P} = f(\mathcal{I}_{\tau_1}, \ldots, \mathcal{I}_{\tau_n})$.

While Gennaro and Wichs [23] defined labeled programs for Boolean circuits (i.e., $f : \{0,1\}^n \to \{0,1\}$), here we consider its extension to the case of arithmetic circuits $f : \mathbb{F}^n \to \mathbb{F}$ where $\mathbb{F}$ is some finite field, e.g., $\mathbb{Z}_p$ for a prime p.

Homomorphic Authenticator Scheme. A homomorphic message authenticator scheme HomMAC is a 4-tuple of algorithms working as follows:

$\mathsf{KeyGen}(1^\lambda)$**:** on input the security parameter λ, the key generation algorithm outputs a secret key sk and a public evaluation key ek.

$\mathsf{Auth}(\mathsf{sk}, \tau, m)$**:** given the secret key sk, an input-label τ and a message $m \in \mathcal{M}$, it outputs a tag σ.

$\mathsf{Ver}(\mathsf{sk}, m, \mathcal{P}, \sigma)$**:** given the secret key sk, a message $m \in \mathcal{M}$, a program $\mathcal{P} = (f, \tau_1, \ldots, \tau_n)$ and a tag σ, the verification algorithm outputs 0 (reject) or 1 (accept).

$\mathsf{Eval}(\mathsf{ek}, f, \boldsymbol{\sigma})$**:** on input the evaluation key ek, a circuit $f : \mathcal{M}^n \to \mathcal{M}$ and a vector of tags $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_n)$, the evaluation algorithm outputs a new tag σ.

[4] While, in general, every polynomial defines a unique function the converse is not true as a function may be expressed as a polynomial in several ways.

Authentication Correctness. Intuitively, a homomorphic MAC satisfies this property if any tag σ generated by the algorithm $\mathsf{Auth}(\mathsf{sk}, \tau, m)$ authenticates with respect to the identity program $\mathcal{I}_\tau$. Formally, we require that for any message $m \in \mathcal{M}$, all keys $(\mathsf{sk}, \mathsf{ek}) \overset{\$}{\leftarrow} \mathsf{KeyGen}(1^\lambda)$, any label $\tau \in \{0,1\}^*$, and any tag $\sigma \overset{\$}{\leftarrow} \mathsf{Auth}(\mathsf{sk}, \tau, m)$, it holds: $\Pr[\mathsf{Ver}(\mathsf{sk}, m, \mathcal{I}_\tau, \sigma) = 1] = 1$.

Evaluation Correctness. Informally, this property states that if the evaluation algorithm is given a vector of tags $\boldsymbol{\sigma} = (\sigma_1, \dots, \sigma_n)$ such that each σ_i authenticates some message m_i as the output of a labeled program $\mathcal{P}_i$, then the tag σ produced by Eval must authenticate $f(m_1, \dots, m_n)$ as the output of the composed program $f(\mathcal{P}_1, \dots, \mathcal{P}_n)$.

More formally, let us fix a pair of keys $(\mathsf{sk}, \mathsf{ek}) \overset{\$}{\leftarrow} \mathsf{KeyGen}(1^\lambda)$, a function $g : \mathcal{M}^t \to \mathcal{M}$ and any set of message/program/tag triples $\{(m_i, \mathcal{P}_i, \sigma_i)\}_{i=1}^t$ such that $\mathsf{Ver}(\mathsf{sk}, m_i, \mathcal{P}_i, \sigma_i) = 1$. If $m^* = g(m_1, \dots, m_t)$, $\mathcal{P}^* = g(\mathcal{P}_1, \dots, \mathcal{P}_t)$, and $\sigma^* = \mathsf{Eval}(\mathsf{ek}, g, (\sigma_1, \dots, \sigma_t))$, then it must hold: $\mathsf{Ver}(\mathsf{sk}, m^*, \mathcal{P}^*, \sigma^*) = 1$.

Succinctness. The size of a tag is bounded by some fixed polynomial in the security parameter, that is independent of the number of inputs taken by the evaluated circuit.

Security. Let HomMAC be a homomorphic MAC scheme as defined above. Consider the following experiment $\mathsf{HomUF{-}CMA}_{\mathcal{A},\mathsf{HomMAC}}(\lambda)$ between a challenger and an adversary $\mathcal{A}$ against HomMAC:

Setup. The challenger generates $(\mathsf{sk}, \mathsf{ek}) \overset{\$}{\leftarrow} \mathsf{KeyGen}(1^\lambda)$ and gives ek to $\mathcal{A}$. It also initializes a list $T = \emptyset$.

Authentication queries. The adversary can adaptively ask for tags on label-message pairs of its choice. Given a query (τ, m), if there is some $(\tau, \cdot) \in T$ (i.e., the label was already queried), then the challenger ignores the query. Otherwise, it computes $\sigma \overset{\$}{\leftarrow} \mathsf{Auth}(\mathsf{sk}, \tau, m)$, returns σ to $\mathcal{A}$ and updates the list $T = T \cup (\tau, m)$. If $(\tau, m) \in T$ (i.e., the query was previously made), then the challenger replies with the same tag generated before.

Verification queries. The adversary is also given access to a verification oracle. Namely, $\mathcal{A}$ can submit a query $(m, \mathcal{P}, \sigma)$ and the challenger replies with the output of $\mathsf{Ver}(\mathsf{sk}, m, \mathcal{P}, \sigma)$.

Forgery. At some point the adversary is supposed to output a forgery $(m^*, \mathcal{P}^* = (f^*, \tau_1^*, \dots, \tau_n^*), \sigma^*)$. Notice that such tuple can be returned by $\mathcal{A}$ also as a verification query $(m^*, \mathcal{P}^*, \sigma^*)$.

Before describing the outcome of this experiment, we define the notion of well defined program with respect to a list T. Informally, there are two ways for a program $\mathcal{P}^* = (f^*, \tau_1^*, \dots, \tau_n^*)$ to be well defined. Either all the τ_i^*s are in T or, if there are labels τ_i^* not in T, then the inputs associated with such labels are somewhat "ignored" by f^* when computing the output. In other words input corresponding to labels not in T do not affect the behavior of f^* in any way.

More formally, we say that a labeled program $\mathcal{P}^* = (f^*, \tau_1^*, \dots, \tau_n^*)$ is *well defined on* T if either one of the following two cases occurs:

1. there exists $i \in \{1, \ldots, n\}$ such that $(\tau_i^*, \cdot) \notin T$ (i.e., $\mathcal{A}$ never asked an authentication query with label τ_i^*), and $f^*(\{m_j\}_{(\tau_j, m_j) \in T} \cup \{\tilde{m}_j\}_{(\tau_j, \cdot) \notin T})$ outputs the same value for all possible choices of $\tilde{m}_j \in \mathcal{M}$;
2. T contains tuples $(\tau_1^*, m_1), \ldots, (\tau_n^*, m_n)$, for some messages $m_1, \ldots, m_n$.

The experiment $\mathsf{HomUF{-}CMA}$ outputs 1 if and only if $\mathsf{Ver}(\mathsf{sk}, m^*, \mathcal{P}^*, \sigma^*) = 1$ and one of the following conditions holds:

– *Type 1 Forgery:* $\mathcal{P}^*$ is not well-defined on T.
– *Type 2 Forgery:* $\mathcal{P}^*$ is well defined on T and $m^* \neq f^*(\{m_j\}_{(\tau_j, m_j) \in T})$, i.e., m^* is not the correct output of the labeled program $\mathcal{P}^*$ when executed on previously authenticated messages $(m_1, \ldots, m_n)$.

We say that a homomorphic MAC scheme HomMAC is secure if for every PPT adversary $\mathcal{A}$ we have that $\Pr[\mathsf{HomUF{-}CMA}_{\mathcal{A},\mathsf{HomMAC}}(\lambda) = 1]$ is negligible.

Remark 1 (Comments on our definition). First, we observe that our definition explicitly disallow the possibility of re-using a label to authenticate more than one value. Essentially, this is a way to uniquely keep track of the authenticated inputs. We notice that such restriction is implicitly present in the Gennaro-Wichs construction as well as in all previous works on homomorphic signatures.

Second, the notion of well defined programs aims at capturing, in a formal way, which tuples generated by the adversary should be considered as forgeries. The catch here is that, since we are dealing with a homomorphic primitive, we should be able to differentiate MACs produced by Eval from MACs generated in some other, possibly malicious, way. Notice, however, that even maliciously generated MACs should not necessarily be considered as forgeries. This is because, in our setting, the adversary can trivially modify a circuit C she is allowed to evaluate by adding dummy gates and inputs that are simply ignored in the evaluation of the modified circuit (i.e., the new circuit is semantically equivalent to C). This last case does not constitute an infringement of our security requirements. Our notion of well defined program $\mathcal{P}$ captures exactly this: either $\mathcal{P}$ is run on legal (i.e. in T) inputs only, or, if this is not the case, those inputs not in T do not affect the computation in any way.

Finally, we observe that for arbitrary computations checking whether a program is well defined may not be efficiently computable. In particular, the difficult task is to check the first condition, i.e., whether a program always outputs the same value for all possible choices of the inputs that are not in T. However, for the case of arithmetic circuits in (exponentially) large fields and of polynomial degree this check can be efficiently performed as follows: by fixing all inputs in T one writes the computation as a new multivariate polynomial whose variables are only the inputs not in T. Then, one checks whether this polynomial is a constant function.

Remark 2 (Relations with previous definitions). Our definition is very similar to that proposed by Gennaro and Wichs in [23] except for two modifications. First, we explicitly allow the adversary to query the verification oracle. Second, we adopt a definition of forgery slightly weaker than that in [23]. More precisely, Gennaro and Wichs define Type 1 forgeries as ones where at least one new label is present. Type 2 forgeries, on the other hand, contain only labels that have been already queried, but m^* is not the correct output of the program when executed on the previously queried inputs.

Notice that our notion becomes equivalent to that given in [23] by simply changing the definition of "well defined program" so that $\mathcal{P}^* = (f^*, \tau_1^*, \ldots, \tau_n^*)$ is said well defined on T if $(\tau_i, m_i) \in T \ \forall i = 1, \ldots n$. The difference between the two definitions is that, as we explained above, we do not consider forgeries all those tuples where "fresh" labels (i.e. labels not in T) do not contribute to the output of the program.

Even though our security definition is weaker than the one in [23], we stress that it is perfectly meaningful for the notion of homomorphic MAC. Indeed, we are still excluding from forgeries all those MACs that can be trivially computed by the adversary from what it queried during the game.

On a technical level, our definition of forgery is inspired by the security definition recently proposed by Freeman for homomorphic signatures [20], except that in our case we do not consider the notion of data set.

3 Our Homomorphic MAC from OWFs

In this section we propose our first construction of homomorphic MACs whose security relies only on a pseudo-random function (and thus on one-way functions). The scheme is simple and efficient and allows to homomorphically evaluate arithmetic circuits $f : \mathbb{Z}_p^n \to \mathbb{Z}_p$ for a prime p of roughly λ bits, where λ is the security parameter.

Our Scheme. In our construction we restrict to circuits whose additive gates do not get inputs labeled by constants. This can be done without loss of generality as, when needed, one can use an equivalent circuit in which there is a special variable/label for the value 1, and can publish the MAC of 1. The description of our scheme follows.

$\mathsf{KeyGen}(1^\lambda)$. Let p be a prime of roughly λ bits. Choose a seed K of a pseudo-random function $F_K : \{0,1\}^* \to \mathbb{Z}_p$ and a random value $x \xleftarrow{\$} \mathbb{Z}_p$. Output $\mathsf{sk} = (K, x), \mathsf{ek} = p$ and let the message space $\mathcal{M}$ be $\mathbb{Z}_p$.

$\mathsf{Auth}(\mathsf{sk}, \tau, m)$. To authenticate a message $m \in \mathbb{Z}_p$ with label $\tau \in \{0,1\}^\lambda$, compute $r_\tau = F_K(\tau)$, set $y_0 = m$, $y_1 = (r_\tau - m)/x \bmod p$ and output $\sigma = (y_0, y_1)$. Basically, y_0, y_1 are the coefficients of a degree-1 polynomial $y(z)$ with the special property that it evaluates to m on the point 0 ($y(0) = m$), and it evaluates to r_τ on a hidden random point x ($y(x) = r_\tau$).

In our construction we will interpret tags σ as polynomials $y \in \mathbb{Z}_p[z]$ of degree $d \geq 1$ in some (unknown) variable z, i.e., $y(z) = \sum_i y_i z^i$.

$\mathsf{Eval}(\mathsf{ek}, f, \boldsymbol{\sigma})$. The homomorphic evaluation algorithm takes as input the evaluation key $\mathsf{ek} = p$, an arithmetic circuit $f : \mathbb{Z}_p^n \to \mathbb{Z}_p$, and a vector $\boldsymbol{\sigma}$ of tags $(\sigma_1, \ldots, \sigma_n)$.

Intuitively, Eval consists in evaluating the circuit f on the tags $\sigma_1, \ldots, \sigma_n$ instead of evaluating it on messages. However, since the values σ_i's are not messages in $\mathbb{Z}_p$, but rather are polynomials $y^{(i)} \in \mathbb{Z}_p[z]$, we need to specify how this evaluation is carried through.

Eval proceeds gate-by-gate as follows. At each gate g, given two tags σ_1, σ_2 (or a tag σ_1 and a constant $c \in \mathbb{Z}_p$), it runs the algorithm $\sigma \leftarrow \mathsf{GateEval}(\mathsf{ek}, g, \sigma_1, \sigma_2)$ described below that returns a new tag σ, which is in turn passed on as input to the next gate in the circuit.

When the computation reaches the last gate of the circuit f, Eval outputs the tag vector σ obtained by running GateEval on such last gate.

To complete the description of Eval we describe the subroutine GateEval.

- $\mathsf{GateEval}(\mathsf{ek}, g, \sigma_1, \sigma_2)$. Let $\sigma_i = \boldsymbol{y}^{(i)} = (y_0^{(i)}, \ldots, y_{d_i}^{(i)})$ for $i = 1, 2$ and $d_i \geq 1$ (see below for the special case when one of the two inputs is a constant $c \in \mathbb{Z}_p$).
  If $g = +$, then:
  - let $d = \max(d_1, d_2)$. Here we assume without loss of generality that $d_1 \geq d_2$ (i.e., $d = d_1$).
  - Compute the coefficients $(y_0, \ldots, y_d)$ of the polynomial $y(z) = y^{(1)}(z) + y^{(2)}(z)$. This can be efficiently done by adding the two vectors of coefficients, $\boldsymbol{y} = \boldsymbol{y}^{(1)} + \boldsymbol{y}^{(2)}$ ($\boldsymbol{y}^{(2)}$ is eventually padded with zeroes in positions $d_1 ... d_2$).

  If $g = \times$, then:
  - let $d = d_1 + d_2$.
  - Compute the coefficients $(y_0, \ldots, y_d)$ of the polynomial $y(z) = y^{(1)}(z) * y^{(2)}(z)$ using the convolution operator $*$, i.e., $\forall k = 0, \ldots, d$, define $y_k = \sum_{i=0}^{k} y_i^{(1)} \cdot y_{k-i}^{(2)}$.

  If $g = \times$ and one of the two inputs, say σ_2, is a constant $c \in \mathbb{Z}_p$, then:
  - let $d = d_1$.
  - Compute the coefficients $(y_0, \ldots, y_d)$ of the polynomial $y(z) = c \cdot y^{(1)}(z)$.

  Return $\sigma = (y_0, \ldots, y_d)$.

As one can notice, the size of a tag grows only after the evaluation of a multiplication gate (where both inputs are not constants). It is not hard to see that after the homomorphic evaluation of a circuit f, it holds $|\sigma| = d + 1$, where d is the degree of f.

$\mathsf{Ver}(\mathsf{sk}, m, \mathcal{P}, \sigma)$. Let $\mathcal{P} = (f, \tau_1, \ldots, \tau_n)$ be a labeled program, $m \in \mathbb{Z}_p$ and $\sigma = (y_0, \ldots, y_d)$ be a tag for some $d \geq 1$. Verification proceeds as follows:
- If $y_0 \neq m$, then output 0 (reject). Otherwise continue as follows.
- For every input wire of f with label τ compute $r_\tau = F_K(\tau)$.
- Next, evaluate the circuit on $r_{\tau_1}, \ldots, r_{\tau_n}$, i.e., compute $\rho \leftarrow f(r_{\tau_1}, \ldots, r_{\tau_n})$, and use x to check whether the following equation holds:

$$\rho = \sum_{k=0}^{d} y_k x^k \tag{1}$$

  If this is true, then output 1. Otherwise output 0.

Observe that the above applies also to identity programs $\mathcal{I}_\tau$, in which case the algorithm just checks that $r_\tau = y_0 + y_1 \cdot x$ and $y_0 = m$.

Efficiency. Our scheme is extremely efficient in generating a tag using the Auth algorithm: just one PRF evaluation (e.g., one AES evaluation, in practice).

If we analyze the Eval algorithm, its complexity is dominated by the cost of evaluating the circuit f with an additional overhead due to the modified gate evaluation and to that the tag's size grows with the degree of the circuit. If the circuit has degree d, in the worst case, this overhead is going to be $O(d)$ for addition gates, and $O(d \log d)$ for multiplication gates[5].

The cost of verification is basically the cost of computing $\rho = f(r_{\tau_1}, \dots, r_{\tau_n})$, that is $O(|f|)$, plus the cost of computing $\sum_{i=0}^{d} y_i x^i$, that is $O(d)$.

Correctness. Very roughly, correctness follows from the special property of the polynomials y generated by Auth, i.e., that $y(0) = m$ and $y(x) = r_\tau$. In particular, this property is preserved when evaluating the circuit f over tags $y^{(1)}, \dots, y^{(n)}$. We give a formal proof of correctness in the full version of this paper.

Security. The security of our scheme is established by the following theorem (again the proof is deferred to the full version of this paper).

Theorem 1. *If F is a PRF, then the homomorphic MAC scheme described in Section 3 is secure.*

4 A Compact Homomorphic MAC for Circuits of Bounded Polynomial Degree

As we mentioned earlier, the homomorphic MAC of Section 3 has the drawback that the tags' size grows linearly with the degree of the evaluated circuit. While this may be acceptable in some cases, e.g., circuits evaluating constant-degree polynomials, it may become impractical in other situations, e.g., when the degree is greater than the input size of the circuit. In this section, we propose a second scheme that solves this issue and enjoys tags of constant size. The scheme keeps almost the same efficiency of the previous one, even though constant-size tags come at the price of a couple of restrictions. First, we have to fix an a-priori bound D on the degree of the circuits that can be evaluated. Second, the homomorphic evaluation has to be done in a "single shot", that is the authentication tags obtained from the Eval algorithm cannot be used again to be composed with other tags. Nevertheless, we show that the scheme achieves an interesting property that we call *local composition.* The idea is that one can keep locally a non-succinct version of the tag that allows for arbitrary composition. Later, when it comes to send an authentication tag to the verifier, one can securely compress such large tag in a very compact one of constant-size.

[5] This bound follows from that one can use optimized algorithms based on FFT to compute the convolution.

For security, in addition to a PRF we need to rely on a computational assumption that says that one cannot compute g given values $g^x, \ldots, g^{x^D}$. This problem is basically a re-writing of a problem already considered in the past: the ℓ-Diffie-Hellman Inversion. We recall its definition below.

Definition 1 (ℓ-DHI [13,30]). *Let $\lambda \in \mathbb{N}$ be the security parameter, and $\mathbb{G}$ be a group of order $p > 2^\lambda$. For a generator $g \in \mathbb{G}$ and a randomly chosen $x \xleftarrow{\$} \mathbb{Z}_p$ we define the advantage of an adversary $\mathcal{A}$ in solving the ℓ-DHI problem as $\mathbf{Adv}^{DHI}_{\mathcal{A}}(\lambda) = \Pr[\mathcal{A}(g, g^x, \ldots, g^{x^\ell}) = g^{1/x}]$ and we say that the ℓ-DHI assumption holds in $\mathbb{G}$ if for every PPT $\mathcal{A}$ and for $\ell = \mathsf{poly}(\lambda)$, the advantage $\mathbf{Adv}^{DHI}_{\mathcal{A}}(\lambda)$ is at most negligible in λ.*

Our Construction. The description of our scheme follows.

$\mathsf{KeyGen}(1^\lambda, D)$. Let λ be the security parameter and $D = \mathsf{poly}(\lambda)$ be an upper bound so that the scheme can support the homomorphic evaluation of circuits of degree at most D. The key generation works as follows.
Generate a group $\mathbb{G}$ of order p where p is a prime of roughly λ bits, and choose a random generator $g \xleftarrow{\$} \mathbb{G}$. Choose a seed K of a pseudorandom function $F_K : \{0,1\}^* \to \mathbb{Z}_p$ and a random value $x \xleftarrow{\$} \mathbb{Z}_p$. For $i = 1$ to D compute $h_i = g^{x^i}$. Output $\mathsf{sk} = (K, g, x), \mathsf{ek} = (h_1, \ldots, h_D)$ and let the message space $\mathcal{M}$ be $\mathbb{Z}_p$.

$\mathsf{Auth}(\mathsf{sk}, \tau, m)$. The tagging algorithm is the same as the one of the construction in Section 3. To authenticate a message $m \in \mathbb{Z}_p$ with label $\tau \in \{0,1\}^\lambda$, compute $r_\tau = F_K(\tau)$, set $y_0 = m$, $y_1 = (r_\tau - m)/x \bmod p$, and output $\sigma = (y_0, y_1)$.

$\mathsf{Eval}(\mathsf{ek}, f, \boldsymbol{\sigma})$. The homomorphic evaluation algorithm takes as input the evaluation key ek, an arithmetic circuit $f : \mathbb{Z}_p^n \to \mathbb{Z}_p$, and a vector $\boldsymbol{\sigma}$ of tags $(\sigma_1, \ldots, \sigma_n)$ so that $\sigma_i \in \mathbb{Z}_p^2$ (i.e., it is a tag for a degree-1 polynomial).
First, proceed exactly as in the construction of Section 3 to compute the coefficients $(y_0, \ldots, y_d)$. If $d = 1$ (i.e., the circuit f computes a degree-1 polynomial), then return $\sigma = (y_0, y_1)$. Otherwise, compute $\Lambda = \prod_{i=1}^d h_i^{y_i}$ and return $\sigma = \Lambda$.

$\mathsf{Ver}(\mathsf{sk}, m, \mathcal{P}, \sigma)$. Let $\mathcal{P} = (f, \tau_1, \ldots, \tau_n)$ be a labeled program, $m \in \mathbb{Z}_p$ and σ be a tag of either the form $(y_0, y_1) \in \mathbb{Z}_p^2$ or $\Lambda \in \mathbb{G}$. First, proceed as in the construction of Section 3 to compute $\rho = f(r_{\tau_1}, \ldots, r_{\tau_n})$. If the program $\mathcal{P}$ computes a polynomial of degree 1, then proceed exactly as in the construction of Section 3 and check that $\rho = y_0 + y_1 \cdot x$ and $y_0 = m$. Otherwise, use g to check whether the following equation holds:

$$g^\rho = g^m \cdot \Lambda \tag{2}$$

If the checks are satisfied, then output 1. Otherwise output 0.

Correctness. The correctness easily follows from the correctness of the scheme described in Section 3 and by observing that equation (2) is essentially equivalent to checking that $\rho = \sum_{i=0}^{d} y_i x^i$, which is the verification equation (1) in the scheme of Section 3.

Local Composition. The above scheme satisfies an interesting property that we call *local composition.* The idea is that one can keep locally the large version of the tag, i.e., the polynomial y with its $d+1$ coefficients $y_0, \ldots, y_d$, but still send its compact version $\Lambda = \prod_{i=1}^{d} (g^{x^i})^{y_i}$ to the verifier. Keeping y allows for arbitrary composition as in the scheme of Section 3. In applications where composition does not involve many parties, this property allows to achieve succinct tags and local composition of partial computations at the same time.

Extension. In the full version of this paper we show an extension of this scheme that, by using pairings, allows to further compute an additional level of multiplications and unbounded additions on tags of the Λ form.

Security. Security follows from the following theorem (whose proof is postponed to the full version of this paper).

Theorem 2. *If F is a PRF and the $(D-1)$-Diffie Hellman Inversion Assumption holds in $\mathbb{G}$, then the homomorphic MAC scheme described in Section 4 is secure.*

Acknowledgements. The authors would like to thank Valerio Pastro and Daniel Wichs for helpful discussions on this work.

References

1. Agrawal, S., Boneh, D.: MACs: MAC-based integrity for network coding. In: Abdalla, M., Pointcheval, D., Fouque, P.-A., Vergnaud, D. (eds.) ACNS 2009. LNCS, vol. 5536, pp. 292–305. Springer, Heidelberg (2009)
2. Ahn, J.H., Boneh, D., Camenisch, J., Hohenberger, S., Shelat, A., Waters, B.: Computing on authenticated data. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 1–20. Springer, Heidelberg (2012)
3. Applebaum, B., Ishai, Y., Kushilevitz, E.: From secrecy to soundness: Efficient verification via secure computation. In: Abramsky, S., Gavoille, C., Kirchner, C., Meyer auf der Heide, F., Spirakis, P.G. (eds.) ICALP 2010, Part I. LNCS, vol. 6198, pp. 152–163. Springer, Heidelberg (2010)
4. Attrapadung, N., Libert, B.: Homomorphic network coding signatures in the standard model. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 17–34. Springer, Heidelberg (2011)
5. Attrapadung, N., Libert, B., Peters, T.: Computing on authenticated data: New privacy definitions and constructions. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 367–385. Springer, Heidelberg (2012)
6. Attrapadung, N., Libert, B., Peters, T.: Efficient completely context-hiding quotable and linearly homomorphic signatures. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 386–404. Springer, Heidelberg (2013)
7. Benabbas, S., Gennaro, R., Vahlis, Y.: Verifiable delegation of computation over large datasets. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 111–131. Springer, Heidelberg (2011)

8. Bitansky, N., Canetti, R., Chiesa, A., Tromer, E.: From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In: ITCS 2012: Proceedings of the 3rd Symposium on Innovations in Theoretical Computer Science (2012)
9. Bitansky, N., Canetti, R., Chiesa, A., Tromer, E.: Recursive composition and bootstrapping for snarks and proof-carrying data. Cryptology ePrint Archive, Report 2012/095 (2012), `http://eprint.iacr.org`
10. Boneh, D., Freeman, D., Katz, J., Waters, B.: Signing a linear subspace: Signature schemes for network coding. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 68–87. Springer, Heidelberg (2009)
11. Boneh, D., Freeman, D.M.: Homomorphic signatures for polynomial functions. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 149–168. Springer, Heidelberg (2011)
12. Boneh, D., Freeman, D.M.: Linearly homomorphic signatures over binary fields and new tools for lattice-based signatures. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 1–16. Springer, Heidelberg (2011)
13. Boyen, X.: The uber-assumption family: A unified complexity framework for bilinear groups. In: Galbraith, S.D., Paterson, K.G. (eds.) Pairing 2008. LNCS, vol. 5209, pp. 39–56. Springer, Heidelberg (2008)
14. Catalano, D., Fiore, D., Gennaro, R., Vamvourellis, K.: Algebraic (trapdoor) one way functions and their applications. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 680–699. Springer, Heidelberg (2013)
15. Catalano, D., Fiore, D., Warinschi, B.: Adaptive pseudo-free groups and applications. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 207–223. Springer, Heidelberg (2011)
16. Catalano, D., Fiore, D., Warinschi, B.: Efficient network coding signatures in the standard model. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 680–696. Springer, Heidelberg (2012)
17. Chung, K.-M., Kalai, Y., Vadhan, S.: Improved delegation of computation using fully homomorphic encryption. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 483–501. Springer, Heidelberg (2010)
18. Chung, K.-M., Kalai, Y.T., Liu, F.-H., Raz, R.: Memory delegation. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 151–168. Springer, Heidelberg (2011)
19. Fiore, D., Gennaro, R.: Publicly verifiable delegation of large polynomials and matrix computations, with applications. In: 2012 ACM Conference on Computer and Communication Security. ACM Press (October 2012) Full version avaiable at, `http://eprint.iacr.org/2012/281`
20. Freeman, D.M.: Improved security for linearly homomorphic signatures: A generic framework. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 697–714. Springer, Heidelberg (2012)
21. Gennaro, R., Gentry, C., Parno, B.: Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 465–482. Springer, Heidelberg (2010)
22. Gennaro, R., Katz, J., Krawczyk, H., Rabin, T.: Secure network coding over the integers. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 142–160. Springer, Heidelberg (2010)
23. Gennaro, R., Wichs, D.: Fully homomorphic message authenticators. Cryptology ePrint Archive, Report 2012/290 (2012), `http://eprint.iacr.org`

24. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Mitzenmacher, M. (ed.) 41st ACM STOC, Bethesda, Maryland, USA, May 31-June2, pp. 169–178. ACM Press (2009)
25. Gentry, C., Wichs, D.: Separating succinct non-interactive arguments from all falsifiable assumptions. In: Fortnow, L., Vadhan, S.P. (eds.) 43rd ACM STOC, San Jose, California, USA, June 6-8, pp. 99–108. ACM Press (2011)
26. Goldwasser, S., Kalai, Y.T., Rothblum, G.N.: Delegating computation: interactive proofs for muggles. In: Ladner, R.E., Dwork, C. (eds.) 40th ACM STOC, Victoria, British Columbia, Canada, May 17-20, pp. 113–122. ACM Press (2008)
27. Johnson, R., Molnar, D., Song, D., Wagner, D.: Homomorphic signature schemes. In: Preneel, B. (ed.) CT-RSA 2002. LNCS, vol. 2271, pp. 244–262. Springer, Heidelberg (2002)
28. Kilian, J.: A note on efficient zero-knowledge proofs and arguments. In: 24th ACM STOC, Victoria, British Columbia, Canada, May 4-6, pp. 723–732. ACM Press (1992)
29. Micali, S.: Cs proofs. In: 35th FOCS, Santa Fe, New Mexico, November 20-22 (1994)
30. Mitsunari, S., Sakai, R., Kasahara, M.: A new traitor tracing. IEICE Transactions on Fundamentals E85-A(2), 481–484 (2002)
31. Parno, B., Raykova, M., Vaikuntanathan, V.: How to delegate and verify in public: Verifiable computation from attribute-based encryption. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 422–439. Springer, Heidelberg (2012)
32. Shacham, H., Waters, B.: Compact proofs of retrievability. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 90–107. Springer, Heidelberg (2008)
33. Shpilka, A., Yehudayoff, A.: Arithmetic circuits: A survey of recent results and open questions. Foundations and Trends in Theoretical Computer Science 5(3-4), 207–388 (2010)
34. Valiant, P.: Incrementally verifiable computation or proofs of knowledge imply time/Space efficiency. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 1–18. Springer, Heidelberg (2008)

Streaming Authenticated Data Structures

Charalampos Papamanthou[1], Elaine Shi[2], Roberto Tamassia[3], and Ke Yi[4]

[1] UC Berkeley
cpap@cs.berkeley.edu
[2] University of Maryland
elaine@cs.umd.edu
[3] Brown University
rt@cs.brown.edu
[4] The Hong Kong University of Science and Technology
yike@cse.ust.hk

Abstract. We consider the problem of streaming verifiable computation, where both a verifier and a prover observe a stream of n elements $x_1, x_2, \ldots, x_n$ and the verifier can later delegate some computation over the stream to the prover. The prover must return the output of the computation, along with a cryptographic proof to be used for verifying the correctness of the output. Due to the nature of the streaming setting, the verifier can only keep *small local state* (e.g., logarithmic) which must be updatable in a streaming manner and with *no interaction* with the prover. Such constraints make the problem particularly challenging and rule out applying existing verifiable computation schemes.

We propose *streaming authenticated data structures*, a model that enables efficient verification of *data structure queries* on a stream. Compared to previous work, we achieve an *exponential improvement* in the prover's running time: While previous solutions have linear prover complexity (in the size of the stream), even for queries executing in sublinear time (e.g., set membership), we propose a scheme with $O(\log M \log n)$ prover complexity, where n is the size of the stream and M is the size of the universe of elements. Our schemes support a series of expressive queries, such as (non-)membership, successor, range search and frequency queries, over an ordered universe and even in higher dimensions. The central idea of our construction is a new authentication tree, called *generalized hash tree*. We instantiate our generalized hash tree with a hash function based on lattices assumptions, showing that it enjoys suitable algebraic properties that traditional Merkle trees lack. We exploit such properties to achieve our results.

1 Introduction

With the growing market of cloud computing, it is crucial to construct protocols that enable the verification of computation performed by untrusted servers. For example, when searching over our remotely stored Gmail inbox, it would be desirable if the search results could be accompanied by a cryptographic proof vouching for their correctness, e.g., that no email was omitted (either deliberately or not) from the answer.

We consider verifiable computation in a *streaming* setting, where the dataset outsourced is rapidly evolving (e.g., stock quotes, network flows, sensor streams), and the verifier can only store a small state as the stream goes by (which should be efficiently updatable). Many prior verifiable computation schemes are unsuitable in the streaming setting: Some schemes require that the verifier (client) has access to all the data ahead of time, and performs some preprocessing (e.g., [17,40]) before outsourcing it to the prover (server). Other existing schemes allow a client to update the dataset through an interactive protocol between the client and the server (e.g., [35]). Particularly, since the client does not have sufficient local storage to store all the data, the client needs the server's help to update its local state. Unfortunately, requiring an interactive protocol for every update may be too expensive in a streaming setting. For example, consider a network traffic accounting application [14], where an ISP charges a customer based on the type and duration of its network flows. To enforce that the ISP is performing the accounting correctly, the ISP logs a customer's network flows such that the customer can later make queries to the logs to perform auditing (typically the customer does not have sufficient local storage to log all the flows). In such high link-speed settings, performing an interactive protocol with every packet or flow sent is very expensive.

Our Contributions. We introduce *streaming authenticated data structures*, and design expressive and efficient schemes that *do not require any interaction between the client and server while the stream is observed.* In our scenario, streaming elements $x_1, x_2, \dots, x_n$ are inserted into a data structure that is stored by the prover and the verifier stores and updates small local state of size $O(\log n)$. At any point of time, the verifier can send a data structure query to the prover, e.g., "return the predecessor of y" or "return the elements in the range $[a, b]$". Subsequently the prover can compute an answer and a proof in time $O(\log M \log n)$, where n is the size of the stream and M is the size of the ordered universe from where the elements are drawn. Our protocols are based on the difficulty of solving the *small integer solution problem* [30], and are the first ones to achieve the following properties simultaneously:

1. **Independence of prover and verifier**. While elements $x_1, x_2, \dots, x_n$ are streaming, the verifier and the prover update their states *independently* and with *no interaction* (there is not even unidirectional communication[1]). The only eligible interaction occurs in the querying phase (which is inherent anyways). Such interaction consists only of one round as opposed to existing schemes in the statistical setting that can have up to a logarithmic number of interactions (e.g., [11,12]).
2. **Efficiency**. The running times of the verifier and the prover (for both updates and queries) as well as the proof size are all logarithmic in n and M, where n is the size of the stream and M is the size of the elements universe. In comparison, existing schemes in this setting (e.g., [10,11,12]) incur a linear proof generation overhead on the server, even for sublinear computations (see Section 1.1). We thus achieve an exponential improvement for many common queries in the prover's running time.
3. **Expressiveness**. Our construction supports a wide range of queries over an ordered universe, such as (non-)membership, successor, range search and frequencies. Our results can also be extended for d-dimensional elements, by applying well-known techniques from authenticated data structures [27]. To the best of our knowledge, our construction is the first streaming verifiable protocol to support such an extensive suite of queries with logarithmic prover *and* verifier complexity.

[1] Such a property is not achieved in the recent work of Schröder and Schröder [42].

1.1 Related Work

The research community has introduced *streaming verifiable computation*, both in the statistical [11,12] and the cryptographic setting [10,42], where a verifier and a prover observe a stream of n elements $x_1, x_2, \ldots, x_n$ and the verifier can delegate some computation over the stream to the prover. The protocols in [11,12] are probabilistic and use multiple interactions for verification, which reveal the secret randomness. Thus they support one-shot computation tasks, whereas we allow any number of queries.

Although most of the existing streaming verifiable protocols [10,11,12] are particularly efficient in terms of verifier complexity (e.g., (poly-)logarithmic in the size of the stream), the main shortcoming of all previous work (except for the work of Schröder and Schröder [42], see next paragraph) is the fact that the prover complexity is *linear* in the size of the stream, *even for sublinear (logarithmic) computations*, e.g., membership and range search queries on a stream of elements drawn from an ordered universe.[2] This significantly limits the applicability of these protocols since such an overhead introduces a large amount of latency, making them impractical for real-world deployment. Indeed, as Cormode *et al.* [12] point out in their experimental results, *"the chief bottleneck of these protocols seems to be $\mathcal{P}$'s time to make the proof"*. In this paper, we address the prover's complexity bottleneck for queries of practical importance (e.g., range search) and we design constructions supporting logarithmic prover *and* verifier complexity. Apart from a major theoretical improvement, we believe our protocols comprise a significant step towards practical verifiable streaming protocols.

The only efficient verifiable streaming protocol (with logarithmic prover complexity) was recently introduced by Schröder and Schröder [42]. Their construction can be applied only to *sequential* streams and hence it does not support data structures like dictionaries, where the relative order is decided depending on the element that is being streamed. Therefore it cannot be used to verify range queries efficiently. Moreover, there is unidirectional communication from the verifier to the prover per stream update.

Practical streaming verifiable computation has been also studied by the database community, with often increased worst-case complexities. Li *et al.* [25] considered verifying queries on a data stream with sliding windows, hence the verifier's space is proportional to the window size. The protocol of Papadopoulos *et al.* [33] verifies continuous queries over a stream, again requiring linear verifier space in the worst case.

Other related works such as *verifiable computation* [2,5,15,16,17,18,34,40] and *authenticated data structures* [13,20,21,31,35,36,37,38] are not directly applicable to the streaming setting or their application yields high complexities or interactive protocols.

Why Common Solutions Fail. Traditional Merkle trees [28] (using collision resistant hash functions like SHA-2) can be used to provide very efficient proofs for membership and range search queries in logarithmic time. However, since the client cannot keep linear amount of local state, in order to update the digest when a new item is streamed, the client needs to interact with the server, where the server returns and proves the correctness of the path of the Merkle tree that is "touched" by the update (e.g., see [35]).

To avoid interaction with the prover, one could use accumulator-based solutions (e.g., [9]). Indeed, accumulators have the attractive property that a set of elements can

[2] For linear-time computations, linear complexity at the prover is acceptable, e.g., see [11].

be represented with a small digest that can be updated in a very straightforward way, e.g., by performing an exponentiation and with no interaction. However, this property comes at some significant cost, since proofs of membership can be computed in linear time (or time $O(n^\epsilon)$, e.g., see [36]) which translates into increased prover complexity.

Our work combines the merits of the above paradigms, enabling flexible updates with no interaction and at the same time achieving logarithmic prover complexity.

1.2 Our Techniques

The core idea of our scheme is a new primitive called *generalized hash tree*, which is a generalization of traditional Merkle trees [28]. Generalized hash trees can be instantiated with various collision resistant hash functions. In our construction we choose the hash function $h_n(\mathbf{x}, \mathbf{y}) = \mathbf{Lx} + \mathbf{Ry} \mod q$ (originally introduced in [1]), where $\mathbf{L}, \mathbf{R}$ are picked at random from $\mathbb{Z}_q^{k\times m}$ and $\mathbf{x}, \mathbf{y}$ are m-dimensional vectors with entries bounded by $n < q$. As in Merkle trees, we hierarchically apply this hash function over a binary tree—however this creates a problem, since the output of the above hash function is a vector of different dimensions and larger entries than its inputs. To overcome this problem, we devise a way to map the outputs back to the input domain. Although many mappings could be used, we choose one that maintains the function's homomorphic properties (Figure 4.2), allowing us to express the label of the root (i.e., the roothash) as the sum of well-defined functions of the leaves called *partial labels* (see Definition 16).

For example if the stream contains elements $\{3, 4, 6, 7\}$, we can simply express the label of the root of our hash tree as $\mathcal{L}(3) + \mathcal{L}(4) + \mathcal{L}(6) + \mathcal{L}(7)$, where $\mathcal{L}(x)$ is the partial label that depends only on x and on the public matrices $\mathbf{L}$ and $\mathbf{R}$, and which can be computed in logarithmic time. Clearly, such representation allows for efficient streaming updates of the verifier's state (label of the root), just like accumulators constructions [9]. More importantly (and unlike accumulators constructions), a proof for any element can still be computed in logarithmic time, by having the prover maintain an appropriate Merkle-tree-like authenticated data structure.

2 Definitions

We now present definitions for streaming authenticated data structures (SADS[3]). Our definitions are similar to the ones given by Chung *et al.* [10] for streaming delegation, adjusted to the data structures setting. We denote with k the security parameter and with $n = \mathsf{poly}(k)$ an upper bound on the size of the stream.[4] PPT stands for *probabilistic polynomial-time* and $\mathsf{neg}(k)$ is a negligible function, i.e., a function less than $1/p(k)$, for all polynomials $p(k)$. Finally we define $[n] = \{0, 1, \dots, n\}$.

Definition 1 (SADS scheme). *Let D be* any *data structure that supports queries q and updates* upd. *An SADS (streaming authenticated data structure) scheme $\mathcal{A}$ is a collection of the following six PPT algorithms:*

1. $\mathsf{pk} \leftarrow \mathsf{genkey}(1^k, n)$: *On input the security parameter k and an upper bound n on the size of the stream, it outputs a public key* pk*;*

[3] This acronym has also been used by Pappas *et al.* [39] to denote a private search system.

[4] Otherwise (i.e., if n is not $\mathsf{poly}(k)$) the server might need exponential space.

2. $\{\mathsf{auth}(D_0), d_0\} \leftarrow \mathsf{initialize}(D_0, \mathsf{pk})$*: On input an empty data structure* D_0 *and the public key* pk*, it computes the authenticated data structure* $\mathsf{auth}(D_0)$ *and the respective state* d_0 *of it;*
3. $d_{h+1} \leftarrow \mathsf{updateVerifier}(\mathsf{upd}, d_h, \mathsf{pk})$*: On input an update* upd *to data structure* D_h*, the current state* d_h *and the public key* pk*, it outputs the updated state* d_{h+1} *(run by verifier);*
4. $\{D_{h+1}, \mathsf{auth}(D_{h+1})\} \leftarrow \mathsf{updateProver}(\mathsf{upd}, D_h, \mathsf{auth}(D_h), \mathsf{pk})$*: On input an update* upd *to data structure* D_h*, the authenticated data structure* $\mathsf{auth}(D_h)$ *and the public key* pk*, it outputs the updated data structure* D_{h+1} *along with the updated authenticated data structure* $\mathsf{auth}(D_{h+1})$ *(run by prover);*
5. $\{\alpha(q), \Pi(q)\} \leftarrow \mathsf{query}(q, D_h, \mathsf{auth}(D_h), \mathsf{pk})$*: On input a query* q *on data structure* D_h*, the authenticated data structure* $\mathsf{auth}(D_h)$ *and the public key* pk*, it returns the answer* $\alpha(q)$ *to the query, along with a proof* $\Pi(q)$ *(run by prover);*
6. $\{1, 0\} \leftarrow \mathsf{verify}(q, \alpha(q), \Pi(q), d_h, \mathsf{pk})$*: On input a query* q*, an answer* $\alpha(q)$*, a proof* $\Pi(q)$ *for query* q*, a digest* d_h *and the public key* pk*, it outputs either* 1 *(accepts) or* 0 *(rejects) (run by verifier);*

As part of the data structure specification (and not of the above definition), we also define the algorithm $\{0, 1\} \leftarrow \mathsf{check}(q, \alpha, D_h)$ such that it outputs 1 if and only if α is the correct answer to query q on data structure D_h (otherwise it outputs 0).

Note that there is *no secret key in our definition*, supporting in this way a stronger definition with *public verifiability*, as opposed to other verifiable streaming constructions that appear in the literature [10,42], where the verifier's state needs to be secret.

There are two properties that an SADS scheme should satisfy, namely *correctness* and *security* (as in signature schemes definitions).

Definition 2 (Correctness). *Let* $\mathcal{A}$ *be an SADS scheme consisting of the set of algorithms* {genkey, initialize, updateVerifier, updateProver, query, verify}*. We say that the SADS scheme* $\mathcal{A}$ *is* correct *if, for all* $k \in \mathbb{N}$*, for all* pk *output by algorithm* genkey*, for all* $D_h, \mathsf{auth}(D_h), d_h$ *output by one invocation of* initialize *followed by polynomially-many invocations of* updateVerifier *and* updateProver*, where* $h \geq 0$*, for all queries* q *and for all* $\Pi(q), \alpha(q)$ *output by* $\mathsf{query}(q, D_h, \mathsf{auth}(D_h), \mathsf{pk})$*, with all but negligible probability* $\mathsf{neg}(k)$*, it holds that* $1 \leftarrow \mathsf{verify}(q, \Pi(q), \alpha(q), d_h, \mathsf{pk})$.

Definition 3 (Security). *Let* $\mathcal{A}$ *be an SADS scheme consisting of the set of algorithms* {genkey, initialize, updateVerifier, updateProver, query, verify}*,* k *be the security parameter,* D_0 *be the empty data structure and* $\mathsf{pk} \leftarrow \mathsf{genkey}(1^k)$*. Let also* Adv *be a PPT adversary and let* d_0 *be the state output by* $\mathsf{initialize}(D_0, \mathsf{pk})$.

- *(Update) For* $i = 0, \ldots, h-1 = \mathsf{poly}(k)$*,* Adv *picks the update* upd_i *to data structure* D_i*. Let* $d_{i+1} \leftarrow \mathsf{updateVerifier}(\mathsf{upd}_i, d_i, \mathsf{pk})$ *be the new state corresponding to the updated data structure* D_{i+1}.
- *(Forge)* Adv *outputs a query* q*, an answer* α *and a proof* Π.

We say that the SADS scheme $\mathcal{A}$ *is* secure *if for all* $k \in \mathbb{N}$*, for all* pk *output by algorithm* genkey*, and for any PPT adversary* Adv *it holds that*

$$\Pr\left[\begin{array}{r}\{q, \Pi, \alpha\} \leftarrow \mathsf{Adv}(1^k, \mathsf{pk});\ 1 \leftarrow \mathsf{verify}(q, \alpha, \Pi, d_h, \mathsf{pk});\\ 0 \leftarrow \mathsf{check}(q, \alpha, D_h).\end{array}\right] \leq \mathsf{neg}(k)\,. \quad (2.1)$$

3 Small Integer Solution Problem

The security of our constructions is based on the hardness of the *small integer solution* problem, as given in the following definition:

Definition 4 (Problem $\mathsf{SIS}_{q,\mu,\beta}$). *Given an integer q, a matrix $\mathbf{M} \in \mathbb{Z}_q^{k\times\mu}$ picked uniformly at random (where $\mu \geq k$) and a real β, find an integer vector $\mathbf{z} \in \mathbb{Z}^{\mu}\backslash\{\mathbf{0}\}$ such that $\mathbf{Mz} = \mathbf{0} \mod q$ and $||\mathbf{z}|| \leq \beta$.*

For certain parameters, Micciancio and Peikert [29] proved that SIVP_γ (shortest independent vector problem [41]), a hard problem in lattices, reduces to $\mathsf{SIS}_{q,\mu,\beta}$ for $\gamma = \mathsf{poly}(k)$. In the following we state an immediate corollary of Theorem 1.1 in [29]:

Corollary 1 (Reducing SIVP_γ to $\mathsf{SIS}_{q,\mu,\beta}$ [29]). *Let $\mathsf{SIS}_{q,\mu,\beta}$ be an instance of the small integer solution problem. Let also β, μ, q be $\mathsf{poly}(k)$, where q is a prime such that $q \geq \beta \cdot k^\delta$ for some $\delta > 0$. $\mathsf{SIS}_{q,\mu,\beta}$ is as hard as approximating the problem SIVP_γ in the worst case to within certain $\gamma = \frac{\beta}{k^\delta} \cdot O(\beta\sqrt{k} \cdot \mathsf{poly}(\log k))$.*

For exponential values of γ, i.e., $\gamma = 2^{O(k)}$, one can use the LLL algorithm [24] and solve the SIVP_γ problem in polynomial time. However, for polynomial γ, no efficient algorithm is known to date, even for factors slightly smaller than exponential [41]. Therefore, for the parameters of Corollary 1, $\mathsf{SIS}_{q,\mu,\beta}$ is also hard, leading to the following assumption:

Assumption 1 (Hardness of $\mathsf{SIS}_{q,\mu,\beta}$). *Let k be the security parameter and $\mathsf{SIS}_{q,\mu,\beta}$ be an instance of the small integer solution problem. Let also β, μ, q be $\mathsf{poly}(k)$, where q is a prime such that $q \geq \beta \cdot k^\delta$ for some $\delta > 0$. There is no PPT algorithm for solving $\mathsf{SIS}_{q,\mu,\beta}$, except with negligible probability $\mathsf{neg}(k)$.*

3.1 Setting the Parameters q, μ and β

For the application we are considering in this paper, we are using an instance of the problem $\mathsf{SIS}_{q,\mu,\beta}$ where β (i.e., the norm of the solution vector) takes polynomially-large values depending on a polynomially-bounded application parameter n (n will be the size of the stream). Specifically we are going to use the parameters q, μ, β as set by the algorithm $\mathsf{parameters}(1^k, n)$ in Figure 3.1.

We note here that the parameters in Figure 3.1 comply with Corollary 1: First, as $n = \mathsf{poly}(k)$, all q, μ, β are $\mathsf{poly}(k)$. Second, $q/\sqrt{\lceil\log q\rceil} \geq \sqrt{2} \cdot n \cdot k^{0.5+\delta} \Leftrightarrow q \geq \beta \cdot k^\delta$, since $\beta = n\sqrt{\mu}$ and $\mu = 2k\lceil\log q\rceil$. Also note that there is always a prime $q = \Theta(n \cdot k^{0.5+\delta} \cdot \sqrt{\log k})$ satisfying the inequality above, for some $\delta > 0$.

3.2 The Hash Function

Our construction uses a hash function that is a syntactic modification (it accepts two inputs instead of one) of the collision resistant hash function presented by Micciancio and Regev [30], following seminal work by Ajtai [1] and Goldreich *et al.* [19]. The security of our function is based on the hardness of $\mathsf{SIS}_{q,\mu,\beta}$, using the parameters by Micciancio and Peikert [29], as shown above. We note here that a similar two-input hash function was also used to build a string commitment scheme by Kawachi *et al.* [23].

Algorithm $\{q,\mu,\beta\} \leftarrow \mathsf{parameters}(1^k,n)$*:* For $n = \mathsf{poly}(k)$, let q be the smallest prime satis- fying $q/\sqrt{\lceil \log q \rceil} \geq \sqrt{2}\cdot n \cdot k^{0.5+\delta}$ for some $\delta > 0$. Set $\mu = 2k\lceil \log q\rceil$ and $\beta = n\sqrt{\mu}$.

Fig. 3.1. Setting the parameters of $\mathsf{SIS}_{q,\mu,\beta}$ as a function of the application parameter n

Definition 5 (Hash function [29,30]). *Let k be the security parameter, $n = \mathsf{poly}(k)$ and q, μ, β be the parameters output by algorithm* $\mathsf{parameters}(1^k, n)$*. Set $m = \frac{\mu}{2}$. Let also $\mathbf{L}, \mathbf{R} \in \mathbb{Z}_q^{k\times m}$ be two $k \times m$ matrices picked uniformly at random. We define the function $h_n : [n]^m \times [n]^m \to \mathbb{Z}_q^k$ as $h_n(\mathbf{x}, \mathbf{y}) = \mathbf{L}\cdot\mathbf{x} + \mathbf{R}\cdot\mathbf{y} \mod q$.*

Theorem 1 (Collision resistance [29,30]). *Let k be the security parameter, $n = \mathsf{poly}(k)$ and $\{q,\mu,\beta\} \leftarrow \mathsf{parameters}(1^k, n)$. Set $m = \frac{\mu}{2}$. Let also $\mathbf{L}, \mathbf{R} \in \mathbb{Z}_q^{k\times m}$ be matrices picked uniformly at random. Assuming hardness of $\mathsf{SIS}_{q,\mu,\beta}$ (see Assumption 1), there is no PPT algorithm that outputs two distinct pairs of vectors $(\mathbf{x}_1, \mathbf{y}_1) \in [n]^m \times [n]^m$ and $(\mathbf{x}_2, \mathbf{y}_2) \in [n]^m \times [n]^m$ such that $\mathbf{L}\cdot\mathbf{x}_1 + \mathbf{R}\cdot\mathbf{y}_1 = \mathbf{L}\cdot\mathbf{x}_2 + \mathbf{R}\cdot\mathbf{y}_2 \mod q$, except with negligible probability $\mathsf{neg}(k)$.*

3.3 Binary Representations

For our constructions, we are going to need *binary representations* of vectors:

Definition 6 (Binary representation of scalars). *Let $\tau = \lceil \log q\rceil$. Denote with $\mathbf{b}(a) = [\mathbf{b}_0, \mathbf{b}_1, \ldots, \mathbf{b}_{\tau-1}]^\mathsf{T} \in \{0,1\}^\tau$ the binary representation of $a \in \mathbb{Z}_q$, i.e., $a = \sum_{i=0}^{\tau-1}\mathbf{b}_i 2^i$.*

Note now that Definition 6 can be naturally extended for vectors $\mathbf{a} \in \mathbb{Z}_q^k$: For $i = 0,\ldots,k-1$, $\mathbf{a}_i$ is mapped to the respective τ entries $\mathbf{b}(\mathbf{a}_i)$ in the resulting vector $\mathbf{b}(\mathbf{a})$:

Definition 7 (Binary representation of vectors). *Let $\mathbf{a} = [\mathbf{a}_0, \mathbf{a}_1, \ldots, \mathbf{a}_{k-1}]^\mathsf{T} \in \mathbb{Z}_q^k$. We denote with $\mathbf{b}(\mathbf{a}) = [\mathbf{b}(\mathbf{a}_0), \mathbf{b}(\mathbf{a}_1), \ldots, \mathbf{b}(\mathbf{a}_{k-1})]^\mathsf{T} \in \{0,1\}^{k\cdot\tau}$ ($\tau = \lceil \log q\rceil$) the binary representation of $\mathbf{a} \in \mathbb{Z}_q^k$, where $\mathbf{b}(\mathbf{a}_i)$ is defined in Definition 6.*

For example, if $k = 2$, $q = 8$ and $\mathbf{a} = [6,3]^\mathsf{T} \in \mathbb{Z}_8^2$, then $\mathbf{b}(\mathbf{a}) = [0,1,1,1,1,0]^\mathsf{T}$, since $\mathbf{b}(6) = [0,1,1]^\mathsf{T}$ and $\mathbf{b}(3) = [1,1,0]^\mathsf{T}$.

4 Generalized Hash Trees

The main primitive of our construction is what we call a *generalized hash tree*. A generalized hash tree has several differences from the traditional Merkle hash tree [28].

First we recall that a Merkle hash tree is a labeled binary tree T where the label $\lambda(w)$ of every node w is the collision resistant hash (e.g., a SHA-2 hash) of the labels $\lambda(u)$ and $\lambda(v)$ and of its children u and v, i.e., $\lambda(w) = h(\lambda(u), \lambda(v))$. When function h is applied recursively on all the nodes of the tree, the label $\lambda(r)$ of the root r has the following property: A PPT adversary cannot find two different data sets at the leaves that produce the same label at the root of a Merkle tree.

In our work, instead of using a hash function such as SHA-2 that lacks algebraic structure, we employ the hash function h_n described in Section 3. However, we cannot directly apply this function since its domain (vectors in $[n]^m$) is different from its range (vectors in $\mathbb{Z}_q^k$). Generalized hash trees, introduced in the next section, provide a way to overcome this domain-range discrepancy problem.

4.1 Defining Generalized Hash Trees

Let $h : \mathcal{D} \times \mathcal{D} \to \mathcal{R}$ be a collision resistant hash function accepting two inputs that take values from domain $\mathcal{D}$ and outputting a value in a *different* range $\mathcal{R}$. Generalized hash trees solve the domain-range discrepancy problem (and at the same time maintain the authentication and algorithmic properties of traditional Merkle hash trees [28]) as follows: They require that the labels $\lambda(u) \in \mathcal{D}$ and $\lambda(v) \in \mathcal{D}$ of the children u and v hash to a *deterministic* and *easily computable* projection function $f : \mathcal{D} \to \mathcal{R}$ of the label $\lambda(w) \in \mathcal{D}$ of the parent w, i.e., $f(\lambda(w)) = h(\lambda(u), \lambda(v))$.

An immediate implication of this property is that the labels of a generalized hash tree are generally *not* uniquely determined by the labels of the leaves: In the above example, $\lambda(w)$ can be any f-preimage of $h(\lambda(u), \lambda(v))$. However, the collision resistant property of Merkle trees is still true: Any two valid hash trees representing different data sets at the leaves but with the same root label yield a collision to the underlying hash function. We now continue with defining generalized hash trees formally. We first need the following definition for representing binary trees.

Definition 8 (Full binary tree). *A* full binary tree T *is a non-empty tree where every internal node has two children. It is represented with set of binary strings, where ϵ is the empty string representing the root of T and $w0$ and $w1$ are the string representations of the left and right children of a node having string representation w.*

For example, a full binary tree with five nodes is $T = \{\epsilon, 0, 1, 00, 01\}$. Note that full binary trees need not be complete, i.e., not all leaves must lie at the same level.

Definition 9 (Labeled binary tree). *A* labeled binary tree (T, λ) *is a full binary tree T along with labels $\lambda(w)$ for all $w \in T$.*

Definition 10 (Generalized hash tree). *Given functions $h : \mathcal{D} \times \mathcal{D} \to \mathcal{R}$ and $f : \mathcal{D} \to \mathcal{R}$, a* generalized hash tree (T, λ, f, h) *is a labeled binary tree (T, λ) such that **(a)** for all $w \in T$, $\lambda(w) \in \mathcal{D}$; **(b)** for all internal nodes $w \in T$, $f(\lambda(w)) = h(\lambda(w0), \lambda(w1))$, where $w0$ and $w1$ are the left and right children of w respectively.*

Definition 11 (Tree collision). *A* tree collision *is a pair of two distinct generalized hash trees (T, λ, f, h) and (T, l, f, h) such that $\lambda(\epsilon) = l(\epsilon)$.*

We now give our main security theorem, establishing collision resistance for generalized hash trees. The proof is in the Appendix (see Section 6).

Theorem 2 (Collision resistance). *Let k be the security parameter, T be a full binary tree of* $\mathsf{poly}(k)$ *depth. If h is collision resistant, there is no PPT algorithm that can output a tree collision (T, λ, f, h) and (T, l, f, h), except with probability* $\mathsf{neg}(k)$.

Function $\mathbf{y} = f(\mathbf{x})$: Let $\tau = \lceil \log q \rceil$. On input a vector $\mathbf{x} \in [n]^m$, where $m = k \cdot \tau$, output a vector $\mathbf{y}$ of k entries such that each $\mathbf{y}_i$ ($i = 0, \dots, k-1$) is the number in $\mathbb{Z}_q$ represented by the radix-2 representation $[\mathbf{x}_{i\tau}, \mathbf{x}_{i\tau+1}, \dots, \mathbf{x}_{(i+1)\tau-1}]^\mathsf{T}$, namely

$$\mathbf{y}_i = \sum_{j=0}^{\tau-1} \mathbf{x}_{i\tau+j} 2^j \mod q\,, \text{ for } i = 0, \dots, k-1\,.$$

Fig. 4.2. The projection function f. It parses the input $\mathbf{x}$ as a vector of radix-2 representations and convers it to a vector $\mathbf{y}$ (of smaller dimension) storing the respective numbers in $\mathbb{Z}_q$.

4.2 An Instantiation of Generalized Hash Trees

In our application setting, we are using a *structured binary tree* which is a special case of the full binary tree from Definition 8:

Definition 12 (Structured binary tree). *Let M be a power of two. A* structured binary tree *$T_\mathcal{C}$ is a full binary tree T of $\log M$ levels where all the leaves lie at the last level of the tree, storing values $\mathcal{C} = [c_0, c_1, \dots, c_{M-1}]$, where $c_i \in \mathbb{Z}_q$.*

Definition 13 (Range of a node). *Let w be a node of a structured binary tree $T_\mathcal{C}$. The set $\mathsf{range}(w)$ contains the leaves of the subtree of $T_\mathcal{C}$ rooted on w.*

In the following sections, we instantiate the generalized hash tree for a structured binary tree using the lattice-based hash function $h_n(\mathbf{x}, \mathbf{y}) = \mathbf{L} \cdot \mathbf{x} + \mathbf{R} \cdot \mathbf{y}$ from Definition 5, where $\mathcal{D} = [n]^m$ and $\mathcal{R} = \mathbb{Z}_q^k$—see Section 3 for the definition of all parameters k, n, m, q. We will also show which projection function f to use and how to compute the labels λ so that Definition 10 is satisfied.

4.3 The Projection Function f

The projection function $f : [n]^m \rightarrow \mathbb{Z}_q^k$ we use is very simple. It *parses* the input vector $\mathbf{x}$ as a radix-2 representation (i.e., a base-2 representation but not necessarily of binary coefficients) and converts it to the respective vector in $\mathbb{Z}_q^k$. We give the code of the function in Figure 4.2. We now have the following corollary for function f:

Corollary 2 (Applying function f to binary representations). *Let $\mathbf{a} \in \mathbb{Z}_q^k$. Then $f(\mathbf{b}(\mathbf{a})) = \mathbf{a}$, where $\mathbf{b}(\mathbf{a})$ is the binary representation of $\mathbf{a}$ defined in Definition 6.*

Clearly, function f is a linear function. This property (stated below) is crucial for proving that the labels (defined in Section 4.4) comply with Definition 10:

Corollary 3 (Linearity of function f). *Let $\mathbf{x} \in [n]^m$ and $\mathbf{y} \in [n]^m$ such that $\mathbf{x} + \mathbf{y} \in [n]^m$. Then $f(\mathbf{x} + \mathbf{y}) = f(\mathbf{x}) + f(\mathbf{y})$.*

4.4 Computing the Labels

We now continue with defining the labels of the generalized hash tree (see Definition 16). Before that, we give some necessary definitions:

Definition 14. *Define the functions* $g_0 : [n]^m \rightarrow [n]^m$ *and* $g_1 : [n]^m \rightarrow [n]^m$ *such that* $g_0(\mathbf{x}) = \mathbf{b}(\mathbf{L} \cdot \mathbf{x})$ *and* $g_1(\mathbf{x}) = \mathbf{b}(\mathbf{R} \cdot \mathbf{x})$*. Also, for a bitstring* $w = b_1 b_2 \ldots b_e$*, define the function* $g_w : [n]^m \rightarrow [n]^m$ *as the composition* $g_w(\mathbf{x}) = g_{b_1} \circ g_{b_2} \circ \ldots \circ g_{b_e}(\mathbf{x})$*.*

Definition 15 (Partial labels of a node w**).** *Let* $T_\mathcal{C}$ *be a structured binary tree. The* partial label *of a leaf node* v *with respect to itself is defined as* $\mathcal{L}_v(v) = \mathbf{1}$*, where* $\mathbf{1} = [1, 1, \ldots, 1]^\mathsf{T} \in [n]^m$*. For every other node* w *of* $T_\mathcal{C}$*, and for every leaf* $v \in \mathsf{range}(w)$*, the* partial label $\mathcal{L}_w(v)$ *of* w *with respect to* v *is defined as* $\mathcal{L}_w(v) = g_{v-w}(\mathbf{1})$*, where* $v - w$ *is the result of removing prefix* w *from bitstring* v*.*

E.g., for a structured binary tree of 8 leaves, the partial label of the root wrt leaves 2 and 3 are $\mathcal{L}_\epsilon(2) = \mathbf{b}(\mathbf{L} \cdot \mathbf{b}(\mathbf{R} \cdot \mathbf{b}(\mathbf{L} \cdot \mathbf{1})))$ and $\mathcal{L}_\epsilon(3) = \mathbf{b}(\mathbf{L} \cdot \mathbf{b}(\mathbf{R} \cdot \mathbf{b}(\mathbf{R} \cdot \mathbf{1})))$ respectively.

Definition 16. *Let* $T_\mathcal{C}$ *be a structured binary tree, where* $\mathcal{C} = [c_0, c_1, \ldots, c_{M-1}]$*. For every node* $w \in T_\mathcal{C}$ *we define a function* $\lambda(w) = \sum_{v \in \mathsf{range}(w)} c_v \cdot \mathcal{L}_w(v)$*.*

Lemma 1. *Let* $T_\mathcal{C}$ *be a structured binary tree. If* $\sum_{i=0}^{M-1} c_i \leq n$*, then for all nodes* $w \in T_\mathcal{C}$ *it holds that* $\lambda(w) \in [n]^m$*, where* $\lambda(w)$ *is the function defined in Definition 16.*

Proof. Write $\lambda(w)$ as in Definition 16. Since $\sum_{i=0}^{M-1} c_i \leq n$ and the entries of each partial label $\mathcal{L}_w(v)$ are in $\{0, 1\}$, it follows that $\lambda(w) \in [n]^m$. □

Lemma 2. *Let* $T_\mathcal{C}$ *be a structured binary tree. If* $\sum_{i=0}^{M-1} c_i \leq n$*, then* $f(\lambda(w)) = \mathbf{L} \cdot \lambda(w0) + \mathbf{R} \cdot \lambda(w1)$*, where* $\lambda(w)$ *is the function defined in Definition 16 and* w *is any internal node of* $T_\mathcal{C}$*.*

Proof. Let w be an internal node of the structured binary tree $T_\mathcal{C}$. Let $w0$ be its left child and $w1$ be its right child. Since $\sum c_i \leq n$, by Lemma 1, it is $\lambda(w) \in [n]^m$, so we can apply function f. Therefore we have

$$f(\lambda(w)) = f\left(\sum_{v \in \mathsf{range}(w)} c_v \cdot \mathcal{L}_w(v)\right) \quad \text{(Def. 16)}$$

$$= \sum_{v \in \mathsf{range}(w0)} c_v \cdot f\left(\mathcal{L}_w(v)\right) + \sum_{v \in \mathsf{range}(w1)} c_v \cdot f\left(\mathcal{L}_w(v)\right) \quad \text{(Cor. 3)}$$

$$= \sum_{v \in \mathsf{range}(w0)} c_v \cdot f\left(g_{w-v}(\mathbf{1})\right) + \sum_{v \in \mathsf{range}(w1)} c_v \cdot f\left(g_{w-v}(\mathbf{1})\right) \quad \text{(Def. 15)}$$

$$= \sum_{v \in \mathsf{range}(w0)} c_v \cdot f\left(g_0(g_{w0-v}(\mathbf{1}))\right) + \sum_{v \in \mathsf{range}(w1)} c_v \cdot f\left(g_1(g_{w1-v}(\mathbf{1}))\right) \quad \text{(Def. 14)}$$

$$= \sum_{v \in \mathsf{range}(w0)} c_v \cdot f\left(g_0(\mathcal{L}_{w0}(v))\right) + \sum_{v \in \mathsf{range}(w1)} c_v \cdot f\left(g_1(\mathcal{L}_{w1}(v))\right) \quad \text{(Def. 15)}$$

$$= \sum_{v \in \mathsf{range}(w0)} c_v \cdot f\left(\mathbf{b}(\mathbf{L} \cdot \mathcal{L}_{w0}(v))\right) + \sum_{v \in \mathsf{range}(w1)} c_v \cdot f\left(\mathbf{b}(\mathbf{R} \cdot \mathcal{L}_{w1}(v))\right) \quad \text{(Def. 14)}$$

$$= \sum_{v \in \mathsf{range}(w0)} c_v \cdot \mathbf{L} \cdot \mathcal{L}_{w0}(v) + \sum_{v \in \mathsf{range}(w1)} c_v \cdot \mathbf{R} \cdot \mathcal{L}_{w1}(v) \quad \text{(Cor. 2)}$$

$= \mathbf{L} \cdot \lambda(w0) + \mathbf{R} \cdot \lambda(w1)$ (Def. 16). This completes the proof. □

Theorem 3. *Let $T_\mathcal{C}$ be a structured binary tree. If $\sum_{i=0}^{M-1} c_i \leq n$, then $(T_\mathcal{C}, \lambda, f, h_n)$ is a generalized hash tree, where $h_n(\mathbf{x}, \mathbf{y}) = \mathbf{L} \cdot \mathbf{x} + \mathbf{R} \cdot \mathbf{y}$ is the function from Definition 5, λ is defined in Definition 16 and f is the function in Figure 4.2.*

Proof. It follows from Lemmas 1 and 2 and by Definition 10. □

4.5 Efficient Updates of the Labels

Note that Definition 16 enables very efficient updates of the label of any node, whenever a leaf value changes. For example, if $\lambda(\epsilon)$ is the label of the root of a generalized hash tree $(T_\mathcal{C}, \lambda, f, h_n)$ with eight leaves $\{0, 1, 2, \ldots, 7\}$ where $c_3 = 2$, $c_4 = c_6 = c_7 = 1$ and $c_0 = c_1 = c_2 = c_5 = 0$, then the root label $\lambda(\epsilon)$ can be expressed as $2\mathcal{L}_\epsilon(3) + \mathcal{L}_\epsilon(4) + \mathcal{L}_\epsilon(6) + \mathcal{L}_\epsilon(7)$. Particularly, each occurrence of an element i contributes $\mathcal{L}_\epsilon(i)$ (i.e., partial label of the root ϵ with respect to i) to the root label. Adding (or removing) an element x to the set is equivalent to adding $\mathcal{L}_\epsilon(x)$ (or $-\mathcal{L}_\epsilon(x)$) to $\lambda(\epsilon)$. It is also important to note that the partial labels (defined in Definition 15) required for such updates can be easily computed in polylogarithmic time:

Lemma 3. *The partial label $\mathcal{L}_w(v)$ can be computed in time $O(\log M \log^2 n)$.*

Proof. Computing $\mathcal{L}_w(v)$, by Definition 15, requires $O(\log M)$ recursive calls, each one of which involves: (a) computing a binary representation of k $O(\log q)$-bit numbers, which takes time $O(k \log q)$; (b) multiplying a $k \times O(k \log q)$ matrix with a vector of $O(k \log q)$ bits, which takes time $O(k \log^2 q)$. This completes the proof. □

5 Our SADS Construction

Let $T_\mathcal{C}$ be a structured binary tree with M leaves corresponding to the universe of integer values $\mathcal{U} = \{0, 1, \ldots, M-1\}$. For our construction, we are using a generalized hash tree $(T_\mathcal{C}, \lambda, f, h_n)$ as described in the previous section, where λ is defined in Definition 16, f is the function in Figure 4.2, h_n is the hash function from Definition 5 and $\{c_0, c_1, \ldots, c_{M-1}\}$ correspond to the frequency of elements $\{0, 1, \ldots, M-1\}$ appearing in the stream. Note that even for an exponential value of M, the condition $\sum_{i=0}^{M-1} c_i \leq n$ of Theorem 3 still holds since for the elements x that do not appear in the stream it is $c_x = 0$. To store the generalized hash tree, we store only the labels that are defined on the paths from non-zero leaves to the root (all other labels are zero). This requires space proportional to $O(\nu \log M)$, where ν is the number of distinct element appearing in the stream. In this way, we avoid storing $O(M)$ space, which is prohibitive given the potential exponential universe size M.

Figure 5.3 presents our SADS scheme for frequency queries. We note that algorithms query and verify are the same for all generalized hash trees, unlike the update algorithms that are specific for the algebraic hash function h_n.

5.1 Range Search Queries

In this section we show how to support range search queries. The proof for a range search query $[x, y]$ simply contains the two proofs $\Pi(x)$ and $\Pi(y)$ as output by

Algorithm $\mathsf{pk} \leftarrow \mathsf{genkey}(1^k, n)$: Call $\{q, \mu, \beta\} \leftarrow \mathsf{parameters}(1^k, n)$ from Figure 3.1, on input the security parameter k and a bound n on the size of the stream. Set $\mathsf{pk} = \{\mathbf{L}, \mathbf{R}, q, \mathcal{U}\}$, where $\mathcal{U}$ is a universe such that $|\mathcal{U}| = M$ and $\mathbf{L}, \mathbf{R}$ are picked uniformly at random from $\mathbb{Z}_q^m$ for $m = \frac{\mu}{2}$.

Algorithm $\{\mathsf{auth}(D_0), d_0\} \leftarrow \mathsf{initialize}(D_0, \mathsf{pk})$: Let D_0 be a structured binary tree $T_\mathcal{C}$ where $c_i = 0\,(i = 0, \ldots, M-1)$. The algorithm outputs the generalized hash tree $(T_\mathcal{C}, \lambda, f, h_n)$ as $\mathsf{auth}(D_0)$, where $\lambda(v) = \mathbf{0} \in [n]^m$ for all nodes v in $T_\mathcal{C}$. Also it outputs $d_0 = \mathbf{0} \in [n]^m$.

Algorithm $d_{h+1} \leftarrow \mathsf{updateVerifier}(x, d_h, \mathsf{pk})$: Let $x \in \mathcal{U}$ be the current element of the stream. The algorithm updates the local state by setting $d_{h+1} = d_h + \mathcal{L}_\epsilon(x)$, where ϵ is the root of $T_\mathcal{C}$ and $\mathcal{L}_\epsilon(x)$ is defined in Definition 15.

Algorithm $\{D_{h+1}, \mathsf{auth}(D_{h+1})\} \leftarrow \mathsf{updateProver}(x, D_h, \mathsf{auth}(D_h), \mathsf{pk})$: Let $x \in \mathcal{U}$ be the current element of the stream. The algorithm sets $c_x = c_x + 1$, outputting the updated tree $T_\mathcal{C}$. Let $v_\ell, \ldots, v_1$ be the path in $T_\mathcal{C}$ from node v_ℓ (v_ℓ stores c_x) to the child v_1 of the root ϵ of $T_\mathcal{C}$. Set

$$\lambda(v_i) = \lambda(v_i) + \mathcal{L}_{v_i}(x) \text{ for } i = \ell, \ell-1, \ldots, 1\,, \tag{5.2}$$

where $\mathcal{L}_{v_i}(x)$ is defined in Definition 15. The new authenticated data structure $\mathsf{auth}(D_{h+1})$ is the new generalized hash tree with the updated labels as computed in Equation 5.2.

Algorithm $\{\alpha(q), \Pi(q)\} \leftarrow \mathsf{query}(q, D_h, \mathsf{auth}(D_h), \mathsf{pk})$: Let q be a frequency query for el-ement $x \in \mathcal{U}$. Set $\alpha(q) = c_x$ (note that if $c_x = 0$, x is not contained in the collection). Let $v_\ell, \ldots, v_1$ be the path in the structured binary tree $T_\mathcal{C}$ from node v_ℓ (v_ℓ stores the value c_x) to the child v_1 of the root ϵ of $T_\mathcal{C}$. Let also $w_\ell, \ldots, w_1$ be the *sibling nodes* of $v_\ell, \ldots, v_1$. Proof $\Pi(q)$ contains the ordered sequence of the pairs of labels belonging to the tree path from leaf v_ℓ to the root ϵ of the tree, i.e., the pairs $\{(\lambda(v_\ell), \lambda(w_\ell)), (\lambda(v_{\ell-1}), \lambda(w_{\ell-1})), \ldots, (\lambda(v_1), \lambda(w_1))\}$.

Algorithm $\{1, 0\} \leftarrow \mathsf{verify}(q, \alpha(q), \Pi(q), d_h, \mathsf{pk})$: Let q be a frequency query for element $x \in \mathcal{U}$. Parse $\Pi(q)$ as $\{(\lambda(v_\ell), \lambda(w_\ell)), \ldots, (\lambda(v_1), \lambda(w_1))\}$ and $\alpha(q)$ as c_x.
If $\lambda(v_\ell) \neq c_x\mathbf{1}$ or $\lambda(v_\ell), \lambda(w_\ell) \neq [n]^m$, output 0. Compute values $y_{\ell-1}, y_{\ell-2}, \ldots, y_0$ as $y_i = \mathbf{L} \cdot \lambda(v_{i+1}) + \mathbf{R} \cdot \lambda(w_{i+1})$ (if v_{i+1} is v_i's left child) or $y_i = \mathbf{R} \cdot \lambda(v_{i+1}) + \mathbf{L} \cdot \lambda(w_{i+1})$ (if v_{i+1} is v_i's right child). For $i = \ell - 1, \ldots, 1$, if $f(\lambda(v_i)) \neq y_i$ **or** $\lambda(v_i), \lambda(w_i) \notin [n]^m$ output 0. If $f(d_h) \neq y_0$, output 0. Output 1.

Fig. 5.3. Algorithms of the SADS scheme for verifying frequency queries

algorithms $\mathsf{query}(x, D_h, \mathsf{auth}(D_h), \mathsf{pk})$ and $\mathsf{query}(y, D_h, \mathsf{auth}(D_h), \mathsf{pk})$ respectively from Figure 5.3. It also contains the frequencies $\mathcal{C}_{xy} = \{c_{a_1}, c_{a_2}, \ldots, c_{a_s}\}$ of the reported range as an answer. Let now $\mathcal{R}_{xy} = \{a_1, a_2, \ldots, a_s\}$ denote the respective reported range that corresponds to $\mathcal{C}_{xy}$.

For verification, the proofs $\Pi(x)$ and $\Pi(y)$ are verified first by using algorithm verify from Figure 5.3. If this verification is successful, perform the following test (else reject): If for all labels $\lambda(v) \in \Pi(x) \cup \Pi(y)$ such that $\mathsf{range}(v) \cap \mathcal{R}_{xy}$ is not empty, the following relation (as in Definition 16)

$$\lambda(v) = \sum_{i \in \mathsf{range}(v) \cap \mathcal{R}_{xy}} c_i \cdot \mathcal{L}_v(i) \tag{5.3}$$

is true, output $\mathtt{1}$ (i.e., accept), else output $\mathtt{0}$ (i.e., reject). The above relation ensures that all the range (with the correct frequencies) has been reported, or otherwise, the adversary could find a collision. The above technique can be also used for verifying successor queries, where the reported range is empty.

We now give our final result stating the formal security guarantee of our algorithms, along with their detailed asymptotic performance. The correctness of our scheme follows easily by inspecting the algorithms, therefore its proof is omitted. The security proof and the proof of asymptotic performance are in the Appendix.

Theorem 4 (Streaming authenticated frequency with range search). *Let k be the security parameter, $n = \mathsf{poly}(k)$ be an upper bound on the size of a stream containing elements from an ordered universe $\mathcal{U}$ of size M, $\{q, \mu, \beta\} \leftarrow \mathsf{parameters}(1^k, n)$ and ν be the number of unique elements that have appeared in the stream. There exists a streaming authenticated data structure scheme for one-dimensional frequency queries and one-dimensional range queries (outputting the respective frequencies) such that: **(a)** It is correct according to Definition 2 and secure according to Definition 3 and assuming hardness of $\mathsf{SIS}_{q,\mu,\beta}$ (Assumption 1); **(b)** Algorithms $\mathsf{updateVerifier}$ and $\mathsf{updateProver}$ run in $O(\log M \log^2 n)$ time; **(c)** Algorithm query (both for frequency and range search queries) runs in $O(\log M \log n)$ time, outputting a proof of size $O(\log M \log n)$; **(d)** A frequency query can be verified in $O(\log M \log^2 n)$ time and a range search query can be verified in $O(s \log M \log^2 n)$ time, where s is the size of the output range; **(e)** The space required at the verifier is $O(\log n)$ and the space required at the prover is $O(\nu \log M \log n)$.*

Our algorithms can be extended to two (or multiple) dimensions by leveraging existing methods for multidimensional range queries [27], carefully adjusted in our framework. Due to space limitations, we defer such extensions to the full version of our paper.

6 Applications

In this section we present three applications of our construction.

Cryptographic Accumulator with Efficient Witness Generation. A cryptographic accumulator [4,6] allows one to hash a set of inputs into one short *accumulation value*, such that there is a witness that a given input was incorporated into the accumulator, and at the same time, it is infeasible to find a witness for a value that was not accumulated. In CRYPTO 2002, Camenisch and Lysyanskaya [9] introduced *dynamic accumulators*, that enable updating the accumulation value when inputs are dynamically added or deleted, such that the cost of an update is independent of the number of accumulated inputs. However, all dynamic accumulator constructions that appeared since then (e.g., [3,8,9,26,32]) share one common limitation: Computing a witness, in absence of the trapdoor information (which has many practical applications, e.g., [36]), takes at least linear time. We observe that our construction comprises a dynamic accumulator that does not have this limitation: Specifically, for a set of elements $\mathcal{X} \subseteq \{0, 1, \ldots, M - 1\}$, our accumulation value, from Definition 16, is $\mathsf{acc}(\mathcal{X}) = \sum_{i \in \mathcal{X}} \mathcal{L}_\epsilon(i)$. To update the accumulation value with element y, one has to set $\mathsf{acc}(\mathcal{X}) = \mathsf{acc}(\mathcal{X}) + \gamma \cdot \mathcal{L}_\epsilon(y)$, where $\gamma \in \{1, -1\}$ depending on whether we add or remove y from the set. Our construction satisfies basic accumulator properties such as quasi-commutativity and efficient updates [9]. Moreover, one can use the generalized hash tree and compute witnesses in *logarithmic* time (see Theorem 4), as opposed to *linear* time.

Parallel Online Memory Checking in the Public Key Setting. Memory checking [7] studies the problem of cryptographically verifying the correctness of untrusted indexed storage by only storing small local memory. Many checkers with logarithmic *sequential* query complexity (number of reads and writes to the *untrusted* memory), e.g., [7,31,20], have appeared in the literature. However, *parallelizing* existing checker constructions can only be achieved in the secret key setting (e.g., see [22]). Checkers in the public key setting (e.g., [20]) cannot be naturally parallelized because they are traditionally implemented with Merkle trees [28]: Whenever a leaf value of the checker tree is written, the roothash can be updated only after the value of its child has been updated, which is an inherently sequential process. Our generalized hash tree can be used to overcome this barrier, yielding a *parallel* memory checker in the public key setting (recall we do not use any secret keys in our construction). This is because in our construction, whenever a leaf value i is written, changing its value from c to c', we can execute algorithm updateProver from Section 5 in parallel (note the loop described in Relation 5.2 is fully parallelizable) and by accessing *only* the old value c, thus issuing $O(1)$ queries to the untrusted memory. Therefore our construction yields the first parallel memory checker in the public key setting with $O(1)$ query complexity using $O(\log M)$ processors.

Authenticated Data Structure with Logarithmic Space at the Trusted Source. Our construction can be used for implementing an authenticated dictionary with improved space bounds in the three-party model—the traditional model of authenticated data structures [43]. Specifically, we can reduce the space of the trusted source from $O(n)$ to $O(\log n)$. This is because in the three-party model of authenticated data structures, the only goal of the trusted source is to update the publish the latest digest d_h, which, in our construction can be achieved in a streaming fashion (by storing only the previous digest d_{h-1}) and without having access to all the elements of the dictionary (only access to the element of the update is required, see Algorithm updateVerifier). In previous implementations however (e.g., [20,36]), the source keeps all the Merkle tree locally (otherwise the digest cannot be updated), therefore requiring $O(n)$ local space.

Acknowledgments. This work was supported by the Center for Geometric Computing and the Kanellakis fellowship at Brown University, by a NetApp Faculty Fellowship, by Intel through the ISTC for Secure Computing and by the NSF grant CNS-1228485. Part of the work was performed while the first author was at Brown University.

We are grateful to Chris Peikert for his continuous feedback and support during the early stages of this work and to Daniele Micciancio for his insightful feedback on improving the presentation of our result. We particularly thank Bobby Bhattacharjee for suggesting the bandwidth sharing application and the idea of composing multiple streams. We also thank Aris Anagnostopoulos, Hubert Chan, Michael Goodrich, Christopher Imbriano, Jonathan Katz, Behzad Koosha, Dave Levin, Anna Lysyanskaya, Shay Mozes, Franco Preparata, Nikos Triandopoulos and Vinod Vaikuntanathan for useful discussions, and the EUROCRYPT 2013 program committee for their helpful reviews.

References

1. Ajtai, M.: Generating hard instances of lattice problems. In: STOC, pp. 99–108 (1996)
2. Applebaum, B., Ishai, Y., Kushilevitz, E.: From secrecy to soundness: Efficient verification via secure computation. In: Abramsky, S., Gavoille, C., Kirchner, C., Meyer auf der Heide, F., Spirakis, P.G. (eds.) ICALP 2010, Part I. LNCS, vol. 6198, pp. 152–163. Springer, Heidelberg (2010)
3. Au, M.H., Tsang, P.P., Susilo, W., Mu, Y.: Dynamic universal accumulators for DDH groups and their application to attribute-based anonymous credential systems. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 295–308. Springer, Heidelberg (2009)
4. Barić, N., Pfitzmann, B.: Collision-free accumulators and fail-stop signature schemes without trees. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 480–494. Springer, Heidelberg (1997)
5. Benabbas, S., Gennaro, R., Vahlis, Y.: Verifiable delegation of computation over large datasets. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 111–131. Springer, Heidelberg (2011)
6. Benaloh, J.C., de Mare, M.: One-way accumulators: A decentralized alternative to digital signatures. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 274–285. Springer, Heidelberg (1994)
7. Blum, M., Evans, W.S., Gemmell, P., Kannan, S., Naor, M.: Checking the correctness of memories. Algorithmica 12(2/3), 225–244 (1994)
8. Camenisch, J., Kohlweiss, M., Soriente, C.: An accumulator based on bilinear maps and efficient revocation for anonymous credentials. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 481–500. Springer, Heidelberg (2009)
9. Camenisch, J., Lysyanskaya, A.: Dynamic accumulators and application to efficient revocation of anonymous credentials. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 61–76. Springer, Heidelberg (2002)
10. Chung, K.-M., Kalai, Y.T., Liu, F.-H., Raz, R.: Memory delegation. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 151–168. Springer, Heidelberg (2011)
11. Cormode, G., Mitzenmacher, M., Thaler, J.: Practical verified computation with streaming interactive proofs. In: ITCS 2012, pp. 90–112 (2012)
12. Cormode, G., Thaler, J., Yi, K.: Verifying computations with streaming interactive proofs. PVLDB 5(1), 25–36 (2011)
13. Devanbu, P., Gertz, M., Kwong, A., Martel, C., Nuckolls, G., Stubblebine, S.: Flexible authentication of XML documents. Journal of Computer Security 6, 841–864 (2004)
14. Estan, C., Varghese, G.: New directions in traffic measurement and accounting: Focusing on the elephants, ignoring the mice. ACM Trans. Comput. Syst. 21(3), 270–313 (2003)
15. Fiore, D., Gennaro, R.: Improved publicly verifiable delegation of large polynomials and matrix computations. Cryptology ePrint Archive, Report 2012/434 (2012)
16. Fiore, D., Gennaro, R.: Publicly verifiable delegation of large polynomials and matrix computations, with applications. In: CCS, pp. 501–512 (2012)
17. Gennaro, R., Gentry, C., Parno, B.: Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 465–482. Springer, Heidelberg (2010)
18. Gennaro, R., Gentry, C., Parno, B., Raykova, M.: Quadratic span programs and succinct NIZKs without PCPs. In: Johansson, T., Nguyen, P. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 627–646. Springer, Heidelberg (2013)
19. Goldreich, O., Goldwasser, S., Halevi, S.: Collision-free hashing from lattice problems. Electronic Colloquium on Computational Complexity 3(42) (1996)
20. Goodrich, M.T., Tamassia, R., Schwerin, A.: Implementation of an authenticated dictionary with skip lists and commutative hashing. In: DISCEX II, pp. 68–82 (2001)

21. Goodrich, M.T., Tamassia, R., Triandopoulos, N.: Efficient authenticated data structures for graph connectivity and geometric search problems. Algorithmica 60(3), 505–552 (2011)
22. Eric Hall, W., Jutla, C.S.: Parallelizable authentication trees. In: Preneel, B., Tavares, S. (eds.) SAC 2005. LNCS, vol. 3897, pp. 95–109. Springer, Heidelberg (2006)
23. Kawachi, A., Tanaka, K., Xagawa, K.: Concurrently secure identification schemes based on the worst-case hardness of lattice problems. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 372–389. Springer, Heidelberg (2008)
24. Lenstra, A.K., Lenstra Jr., H.W., Lovasz, L.: Factoring polynomials with rational coefficients. Math. Ann. (261), 515–534 (1982)
25. Li, F., Yi, K., Hadjieleftheriou, M., Kollios, G.: Proof-infused streams: enabling authentication of sliding window queries on streams. In: VLDB, pp. 147–158 (2007)
26. Li, J., Li, N., Xue, R.: Universal accumulators with efficient nonmembership proofs. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 253–269. Springer, Heidelberg (2007)
27. Martel, C.U., Nuckolls, G., Devanbu, P.T., Gertz, M., Kwong, A., Stubblebine, S.G.: A general model for authenticated data structures. Algorithmica 39(1), 21–41 (2004)
28. Merkle, R.C.: A certified digital signature. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 218–238. Springer, Heidelberg (1990)
29. Micciancio, D., Peikert, C.: Hardness of SIS and LWE with small parameters. Cryptology ePrint Archive, Report 2013/069 (2013)
30. Micciancio, D., Regev, O.: Worst-case to average-case reductions based on gaussian measures. SIAM J. Comput. 37(1), 267–302 (2007)
31. Naor, M., Nissim, K.: Certificate revocation and certificate update. In: USENIX Security, pp. 217–228 (1998)
32. Nguyen, L.: Accumulators from bilinear pairings and applications. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 275–292. Springer, Heidelberg (2005)
33. Papadopoulos, S., Yang, Y., Papadias, D.: Continuous authentication on relational streams. VLDB Journal 19(2), 161–180 (2010)
34. Papamanthou, C., Shi, E., Tamassia, R.: Signatures of correct computation. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 222–242. Springer, Heidelberg (2013)
35. Papamanthou, C., Tamassia, R.: Time and space efficient algorithms for two-party authenticated data structures. In: Qing, S., Imai, H., Wang, G. (eds.) ICICS 2007. LNCS, vol. 4861, pp. 1–15. Springer, Heidelberg (2007)
36. Papamanthou, C., Tamassia, R., Triandopoulos, N.: Authenticated hash tables. In: CCS, pp. 437–448 (2008)
37. Papamanthou, C., Tamassia, R., Triandopoulos, N.: Optimal authenticated data structures with multilinear forms. In: Joye, M., Miyaji, A., Otsuka, A. (eds.) Pairing 2010. LNCS, vol. 6487, pp. 246–264. Springer, Heidelberg (2010)
38. Papamanthou, C., Tamassia, R., Triandopoulos, N.: Optimal verification of operations on dynamic sets. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 91–110. Springer, Heidelberg (2011)
39. Pappas, V., Raykova, M., Vo, B., Bellovin, S.M., Malkin, T.: Private search in the real world. In: ACSAC, pp. 83–92 (2011)
40. Parno, B., Raykova, M., Vaikuntanathan, V.: How to delegate and verify in public: Verifiable computation from attribute-based encryption. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 422–439. Springer, Heidelberg (2012)
41. Regev, O.: On the complexity of lattice problems with polynomial approximation factors. In: The LLL Algorithm, pp. 475–496 (2010)
42. Shroeder, D., Shroeder, H.: Verifiable data streaming. In: CCS, pp. 953–964 (2012)
43. Tamassia, R.: Authenticated data structures. In: Di Battista, G., Zwick, U. (eds.) ESA 2003. LNCS, vol. 2832, pp. 2–5. Springer, Heidelberg (2003)

Appendix

Proof of Collision Resistance (Proof of Theorem 2)

Since generalized hash trees (T, λ, f, h) and (T, l, f, h) comprise a collision, it is $\lambda(\epsilon) = l(\epsilon)$ and there exists $v_\ell \in T$ such that $\lambda(v_\ell) \neq l(v_\ell)$—see Definition 11. Consider now the path of nodes $v_\ell, v_{\ell-1}, \ldots, v_1, v_0 = \epsilon$ from node v_ℓ to the root $v_0 = \epsilon$ of T. Let also $w_\ell, w_{\ell-1}, \ldots, w_1$ be the siblings of the nodes $v_\ell, v_{\ell-1}, \ldots, v_1$ respectively. We define the following events: **(1)** $\mathcal{E}_{\ell,0}$: $l(v_\ell) \neq \lambda(v_\ell)$; **(2)** $\mathcal{E}_{i,0}$: $l(v_i) \neq \lambda(v_i)$ for $i = \ell-1, \ldots, 1$; **(3)** $\mathcal{E}_{i,0}$: $l(v_i) = \lambda(v_i)$ for $i = \ell - 1, \ldots, 1$; **(4)**: $\mathcal{E}_{0,1}$: $\lambda(\epsilon) = l(\epsilon)$. The probability that a PPT algorithm can output a collision (T, λ, f, h) and (T, l, f, h) is at most

$$\begin{aligned}&\Pr[\mathcal{E}_{\ell,0} \cap (\mathcal{E}_{\ell-1,0} \cup \mathcal{E}_{\ell-1,1}) \cap (\mathcal{E}_{\ell-2,0} \cup \mathcal{E}_{\ell-2,1}) \cap \ldots \cap \mathcal{E}_{0,1}]\\ &\leq \Pr[\mathcal{E}_{\ell,0} \cap \mathcal{E}_{\ell-1,1}] + \ldots + \Pr[\mathcal{E}_{1,0} \cap \mathcal{E}_{0,1}] = \sum_{0 \leq i \leq \ell-1} \Pr[\mathcal{E}_{i+1,0} \cap \mathcal{E}_{i,1}] \,.\end{aligned}$$

Note that the event $\mathcal{E}_{i+1,0} \cap \mathcal{E}_{i,1}$ is equivalent to the set of conditions: **(i)** $l(v_{i+1}) \neq \lambda(v_{i+1})$; **(ii)** $l(v_i) = \lambda(v_i)$ (recall $v_0 = \epsilon$).

It is easy to see that if h is collision resistant, the probability $\Pr[\mathcal{E}_{i+1,0} \cap \mathcal{E}_{i,1}]$ is $\mathsf{neg}(k)$ since it is equivalent with outputting a collision to function h: Since both (T, λ, f, h) and (T, l, f, h) are generalized hash trees it is $f(l(v_i)) = h(l(v_{i+1}), l(w_{i+1}))$ and $f(\lambda(v_i)) = h(\lambda(v_{i+1}), \lambda(w_{i+1}))$. Since now $\lambda(v_i) = l(v_i)$ we have $f(\lambda(v_i)) = f(l(v_i))$. But $l(v_{i+1}) \neq \lambda(v_{i+1})$ and therefore $(\lambda(v_{i+1}), \lambda(w_{i+1}))$ is a collision with the pair $(l(v_{i+1}), l(w_{i+1}))$. Therefore the probability $\Pr[\mathcal{E}_{i,0}|\mathcal{E}_{i-1,1}]$ is $\mathsf{neg}(k)$, implying that the sum $\sum_{i=0}^{\ell-1} \Pr[\mathcal{E}_{i,0}|\mathcal{E}_{i-1,1}]$ is also $\mathsf{neg}(k)$, as T has polynomial depth and ℓ is no greater than T's depth. $\square$

Proof of Security (Stated in Theorem 4)

Fix the security parameter k and output $\mathsf{pk} = (\mathbf{L}, \mathbf{R}, q, \mathcal{U})$ by calling algorithm genkey. Let Adv be a PPT adversary. Let D_0 an initial structured binary tree $T_\mathcal{C}$ where $c_i = 0$ for $i = 0, \ldots, M - 1$ and let d_0 be the state output by initialize(D_0, pk).

1. *Update.* For $t = 1, \ldots, h \leq n$, the adversary Adv picks an element $x_t \in \mathcal{U}$. Let d_t be the final state output by calling updateVerifier for every element x_t and let $T_\mathcal{C}$ be the final structured binary tree D_t after all the updates have been performed, where $\mathcal{C} = [c_0, c_1, \ldots, c_{M-1}]$.
2. *Forge.* Let $x \in \mathcal{U}$ be a query element and $\alpha \neq c_x$ be an *incorrect* value for index x picked by Adv (as in Definition 3). The adversary Adv outputs

$$\Pi(x) = \{(l(v_\ell), l(w_\ell)), (l(v_{\ell-1}), l(w_{\ell-1})), \ldots, (l(v_1), l(w_1))\}$$

as the proof for element x, where v_ℓ is the node corresponding to index x.

We prove that the probability that $1 \leftarrow$ verify$(x, \alpha, \Pi(x), d_h, \mathsf{pk})$ while $\alpha \neq c_x$ is negligible. To do that we consider the full binary tree T (see Definition 8) defined by the nodes $(v_\ell, w_\ell), (v_{\ell-1}, w_{\ell-1}), \ldots, (v_1, w_1)$ and the root node ϵ. It is easy to see that since the verification algorithm accepts, (T, l, f, h_n) is a generalized hash tree as defined in Definition 10, and where l is the labeling in $\Pi(x)$.

Consider now the structured binary tree $T_{\mathcal{C}}$ (where $\mathcal{C} = [c_0, c_1, \ldots, c_{M-1}]$) as defined in Definition 12. Let $T'_{\mathcal{C}}$ be the subtree of $T_{\mathcal{C}}$ that has the same nodes as T. By Theorem 3, the adversary can compute (T, λ, f, h_n), which is also a generalized hash tree. However, since $\alpha \neq c_x$ this means that $l(v_\ell) \neq \lambda(v_\ell)$. Note now that $\lambda(\epsilon) = l(\epsilon) = d_h$ and therefore the adversary has output a tree collision, which, by Theorem 2, happens with probability $\mathsf{neg}(k)$ since h_n is collision resistant (see Theorem 1). A same argument applies for the range search query. This completes the proof. □

Proof of Asymptotic Performance (Stated in Theorem 4)

Algorithm updateVerifier requires computing the partial label $\mathcal{L}_\epsilon(x)$, where x is the element of the update and ϵ is the root of $T_{\mathcal{C}}$. Computing $\mathcal{L}_\epsilon(x)$ can be achieved in $O(\log M \log^2 n)$ time, by Lemma 3.

Algorithm updateProver needs to compute the partial labels $\mathcal{L}_{v_j}(x)$ $(j = \ell, \ldots, 0)$, where $v_\ell, v_{\ell-1}, \ldots, v_0$ are the nodes of the structured binary tree from element x to the root of the tree. By Definition 15, all these labels can be computed in $O(\log M \log^2 n)$ time during the computation of $\mathcal{L}_\epsilon(x)$ (i.e., in one pass). However, to update a label $\lambda(v_i)$, one needs to retrieve it from the underlying data structure that stores the "useful" portion of the generalized hash tree (and either store it back or delete if the label becomes **0**). Therefore updateProver needs to spend an extra $O(\log \nu)$ time in the worst case, where ν is the number of the currently stored elements. Since however $\nu \leq n$, it follows that the time required is $O(\log M \log^2 n)$.

Algorithm query for membership and successor queries needs to retrieve $O(\log M)$ binary representations of $O(\log n)$ bits each, spending $O(\log \nu)$ time to retrieve each one of them. Since $\nu \leq n$, it follows that query runs in $O(\log M \log n)$ time. The proof has also size $O(\log M \log n)$, since it contains $O(\log M)$ binary representations of $O(\log n)$ bits each. Since range search is implemented via s successor queries, the same bounds apply multiplied with s, where s is the size of the output range.

Finally, for the space at the client, it is required that the client store d_h, which consists of k $O(\log n)$-bit numbers, therefore the space at the client is $O(\log n)$. For the space at the prover, we recall that we only store labels $\lambda(v)$ that lie on tree paths starting from leaves x such that $c_x > 0$ (all these labels are also non-zero and have $O(\log n)$ bits). Since at every point in time there are ν elements stored in the data structure, it follows that the space at the server is $O(\nu \log M \log n)$.

Improved Key Recovery Attacks on Reduced-Round AES in the Single-Key Setting

Patrick Derbez[1], Pierre-Alain Fouque[1,2], and Jérémy Jean[1]

[1] École Normale Supérieure, 45 Rue d'Ulm, 75005 Paris, France
{Patrick.Derbez,Pierre-Alain.Fouque,Jeremy.Jean}@ens.fr
[2] Université de Rennes, France

Abstract. In this paper, we revisit meet-in-the-middle attacks on AES in the single-key model and improve on Dunkelman, Keller and Shamir attacks at Asiacrypt 2010. We present the best attack on 7 rounds of AES-128 where data/time/memory complexities are below 2^{100}. Moreover, we are able to extend the number of rounds to reach attacks on 8 rounds for both AES-192 and AES-256. This gives the best attacks on those two versions with a data complexity of 2^{107} chosen-plaintexts, a memory complexity of 2^{96} and a time complexity of 2^{172} for AES-192 and 2^{196} for AES-256. Finally, we also describe the best attack on 9 rounds of AES-256 with 2^{120} chosen plaintexts and time and memory complexities of 2^{203}. All these attacks have been found by carefully studying the number of reachable multisets in Dunkelman et al. attacks.

1 Introduction

The Rijndael block cipher has been designed by Daemen and Rijmen in 1997 and accepted as the AES (Advanced Encryption Standard) standard since October 2000 by the NIST. Nowadays, it is probably the most used block cipher. It has very good performances in both software and hardware on various platforms and it is provably resistant against differential and linear attacks.

However, new attacks have been recently developed using many ideas from the cryptanalysis of hash functions. The first analysis studies AES in the strong adversarial model where the adversary can ask the encryption of a message under a related-key by specifying the relation. Biryukov, Khovratovich and Nikolic show some drawbacks of the key-schedule algorithms and how to exploit it to mount an attack on the full versions of AES-192 and AES-256 in [4,5,6]. In a second analysis, Dunkelman, Keller and Shamir show in [15] more efficient meet-in-the-middle attacks using ideas from rebound attacks on hash functions [22]. Finally, the biclique attack [8] also uses meet-in-the-middle ideas for preimage attacks on hash functions by Sasaki et al. [1]. It has been developed by Bogdanov, Khovratovich and Rechberger in [8] and allows to mount an attack on the full AES for all versions with a marginal time complexity over exhaustive search.

Overview of the Attacks on AES. The first attack on AES is the SQUARE attack, proposed by Daemen, Knudsen and Rijmen on the SQUARE block cipher [10]. In [11], Daemen and Rijmen remark that if we encrypt a δ-set, i.e. a set of 256 plaintexts where a byte (called *active* byte) can take all values and the 15 other bytes are constant, after 3 rounds of Rijndael, the sum of each byte of the 256 ciphertexts equals zero. This distinguishing property can be used to mount efficient attacks up to 6 rounds. The first attack has a time complexity of 2^{72} encryptions and requires 2^{32} messages, and it has been improved by Ferguson et al. to 2^{46} operations in [16].

Then, Gilbert and Minier show in [18] that this property can be made more precise using functions of the active byte, which allows to build a distinguisher on 3 rounds. The main idea is to consider the set of functions mapping one active byte to one byte after 3 rounds. This set depends on 9 one-byte parameters so that the whole set can be described using a table of 2^{72} entries of a 256-byte sequence $(f(0), \ldots, f(255))$. Their attack allows to break 7 rounds of AES with a marginal time complexity over exhaustive search. This idea has been generalized at FSE 2008 by Demirci and Selçuk in [12] using meet-in-the-middle techniques, whereas Gilbert and Minier used collision between the functions. More specifically, they show that on 4 rounds, the value of each byte of the ciphertext can be described by a function of the active byte parameterized by 25 in [12] and 24 8-bit parameters in [13]. The last improvement is due to the observation that the 25th parameter is a key byte which is constant for all functions. Consequently, by considering $(f(0) - f(0), f(1) - f(0), \ldots, f(255) - f(0))$ we can use only 24 parameters. The main drawback of the meet-in-the-middle attack is the memory requirement. Indeed, the basic attack only works for the 256-bit version and then Demirci and Selçuk have to use a time/memory tradeoff to extend the attack for the 192-bit AES version.

Another idea has been developed by Biham and Keller [3] and is based on a 4-round impossible differential, as well as the work of Bahrak and Aref in [2]. Later, this idea has been refined by Lu, Dunkelman, Keller and Kim in [20]. At the present time, it is the most efficient attack on 7-round AES-128.

At ASIACRYPT 2010, Dunkelman, Keller and Shamir develop many new ideas to solve the memory problems of the Demirci and Selçuk attacks. First of all, they show that instead of storing the whole sequence, we can only store the associated multiset, i.e. the unordered sequence with multiplicity rather than the ordered sequence. This reduces the table by a factor 4 and avoids the need to guess one key byte during the attack. The second and main idea is the differential enumeration which allows to reduce the number of parameters that describes the set of functions from 24 to 16. However, to reduce this number, they rely on a special property on a truncated differential characteristic. The idea consists in using a differential truncated characteristic whose probability is not too small. The property of this characteristic is that the set of functions from one state to the state after 4 rounds can only take a restricted number of values, which is much smaller than the number of all functions. The direct consequence is an increase of the amount of needed data, but the memory requirement is reduced to 2^{128} and the same analysis also applies to the 128-bit version. However, this attack is not better than the impossible differential attack even though many tradeoffs could be used.

Finally, at CRYPTO 2011, Bouillaguet, Derbez and Fouque describe in [9] new meet-in-the-middle attacks that allow to efficiently break a small number of rounds of AES using a very small amount of data. These attacks have been found automatically. Similar attacks have been developed against other symmetric schemes that reuse AES component such as the Pelican-MAC message authentication code or the LEX stream cipher. However, the complexity of the algorithm that looks for the best attack is exponential in the number of variables and if we try to take into account more rounds or more plaintext/ciphertext pairs, then the program does not find anything of interest. This tool looks promising since improvements on existing attacks usually consider a small number of rounds. For instance, if we want to improve on Dunkelman et al. attacks, we need to study 4 rounds of AES.

Dunkelman, Keller and Shamir's Attack. In [15], a new attack is developed using ideas from differential and meet-in-the-middle attacks. In the first stage, differential attacks find a differential characteristic with high or low probability covering many rounds. Then, in the online stage, the adversary asks for the encryption of many pairs: for each pair, the adversary tries to decrypt by guessing the last subkey and if the differential characteristic is followed, then the adversary increases the counter of the associated subkey. If the probability of the characteristic is high enough, then the counter corresponding to the right secret-key would be among the higher counters. In some case, it is also possible to add some rounds at the beginning by guessing part of the first subkeys.

Here, Dunkelman et al. propose a novel differential attack. Instead of increasing a counter once a pair is found, the adversary uses another test to eliminate the wrong guesses of the first or last subkeys. This test decides with probability one whether the middle rounds are covered with the differential. The idea is that the middle rounds follow a part of the differential and the function f that associates each byte of the input state to one byte of the output state can be stored efficiently. Demirci and Selçuk propose to store in a table the function with no differential characteristic, which turns out to be much larger that this one. Consequently, in Dunkelman et al.'s attack, the adversary guesses the first and last subkeys and looks for a pair that follows the beginning and last rounds of the differential characteristic. Once such a pair is found, the adversary takes one of the messages that follows the characteristic and constructs a structure to encrypt which is related to a δ-set for the intermediate rounds. From the encryption of this set, the adversary can decrypt the last rounds and check whether the encryption of this δ-set belongs to the table. If this is the case, then the part of the first and last subkeys are correct and an exhaustive search on the other parts of the key allows to find the whole key.

To construct the table, the idea is similar to the attack. We need to find a pair of messages that satisfies the truncated differential characteristic. Then, we take one message in the pair and we compute the function f. Dunkelman et al. use a rebound technique to find the pair that follows the characteristic.

Our Results. Dunkelman et al. show that by using a particular 4-round differential characteristic with a not too small probability, the active states in the middle of the characteristic can only take 2^{64} values. In their characteristic, they also need to consider the same 8 key bytes as Demirci and Selçuk. They claim that *"In order to reduce the size of the precomputed table, we would like to choose the δ-set such that several of these parameters will equal to predetermined constants. Of course, the key bytes are not known to the adversary and thus cannot be "replaced" by such constants"*. Here, we show that it is possible to enumerate the whole set of solutions more efficiently than by taking all the values for the key bytes such that every value of these bytes are possible. We show that the whole set can take only 2^{80} values with this efficient enumeration technique. Of course, it might be possible to improve this result to 2^{64} but not any further since the key bytes may take all the 2^{64} possible values. Using the same ideas, we show that it is possible to have an efficient enumeration for a 5-round differential characteristic which allows us to mount an attack on 9 rounds for AES-256. The bottleneck of the attack is no longer the memory, but the time and data complexities.

In this paper, we show that the number of parameters describing the functions can be further reduced to 10 and that this attack is now more efficient than the impossible differential attack [20]. We describe the best key-recovery attacks on 7 rounds of all versions of AES with all complexities below 2^{100}, as the related-key attack of Biryukov and Nikolič in [7]. We also show improved key-recovery attacks on 8 rounds of AES-192 and on 8 and 9 rounds of AES-256. Additionally, we show that all our attacks are optimal in the class of attacks from [15] that we revisit. We also show that it allows us to attack one more round on AES-256, and for the AES-192 the attack is comparable even though some improvements can be made. To this end, we use several tradeoffs proposed by Dunkelman et al. and we use a more careful analysis of the enumeration technique.

Organization of the Paper. In Section 2, we describe the AES and some properties used by the previous attacks on this block cipher. In Section 3, we present our basic attack on 7 rounds for all AES versions. Then, in section Section 4, we show that we can also attack 8 rounds for AES-192 and AES-256 and 9 rounds for AES-256.

2 AES and Previous Work

2.1 Description of the AES

The Advanced Encryption Standard (AES) [23] is a Substitution-Permutation Network that can be instantiated using three different key bit-lengths: 128, 192, and 256. The 128-bit plaintext initializes the internal state viewed as a 4×4 matrix of bytes as values in the finite field $GF(2^8)$, which is defined via the irreducible polynomial $x^8 + x^4 + x^3 + x + 1$ over $GF(2)$. Depending on the version of the AES, N_r rounds are applied to that state: $N_r = 10$ for AES-128, $N_r = 12$ for AES-192 and $N_r = 14$ for AES-256. Each of the N_r AES round (Figure 1) applies four operations to the state matrix (except the last one where we omit the **MixColumns**):

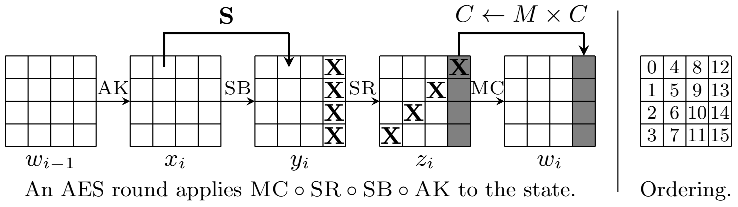


Fig. 1. Description of one AES round and the ordering of bytes in an internal state

- **AddRoundKey** (AK) adds a 128-bit subkey to the state.
- **SubBytes** (SB) applies the same 8-bit to 8-bit invertible S-Box S 16 times in parallel on each byte of the state,
- **ShiftRows** (SR) shifts the i-th row left by i positions,
- **MixColumns** (MC) replaces each of the four column C of the state by $M \times C$ where M is a constant 4×4 maximum distance separable matrix over $GF(2^8)$,

After the N_r-th round has been applied, a final subkey is added to the internal state to produce the ciphertext. We refer to official specification document [23] for the key expansion algorithms to produce the $N_r + 1$ subkeys.

Proposition 1 (Differential Property of S). *Given Δ_i and Δ_o two non-zero differences in $\mathbb{F}_{256}$, the equation $S(x) + S(x + \Delta_i) = \Delta_o$, has one solution in average. This property also applies to S^{-1}.*

Notations and Units. In this paper, we count the AES rounds from 0 and we refer to a particular byte of an internal state x by $x[i]$, as depicted in Figure 1, or $x[i, \ldots, j]$ for bytes as positions between i and j. Moreover, in the ith round, we denote the internal state after **AddRoundKey** by x_i, after **SubBytes** by y_i, after **ShiftRows** by z_i and after **MixColumns** by w_i. To refer to the difference in a state x, we use the notation Δx. The first added subkey is the master key k_{-1}, and the one added after round i is denoted by k_i. In some cases, we are interested in swapping the order of the **MixColumns** and **AddRoundKey** operations. As these operations are linear they can be interchanged, by first XORing the data with an equivalent key and applying the **MixColumns** operation afterwards. We denote the equivalent subkey for this new round-function description by: $u_i = \mathsf{MC}^{-1}(k_i)$. We measure memory complexities of our attacks in number of 128-bit AES blocks and time complexities in terms of AES encryptions.

In the following sections, we use particular structures of messages captured by Definition 1 and Definition 2.

Definition 1 (δ-set, [11]). *Let a δ-set be a set of 256 AES-states that are* all *different in one state bytes (the active byte) and all equal in the other state bytes (the inactive bytes).*

Definition 2 (Multisets of bytes). *A multiset generalizes the set concept by allowing elements to appear more than once. Here, a multiset of* 256 *bytes can* *take as many as* $\binom{2^8+2^8-1}{2^8} \approx 2^{506.17}$ *different values. From the point of view of information theory, we can represent such a multiset on* 512 *bits.*

2.2 Attack Scheme

In this section, we present a unified view of the previously known meet-in-the-middle (MITM) attacks on AES [12, 15, 18], where n rounds of the block cipher can be split into three consecutive parts of n_1, n_2 and n_3 rounds, $n = n_1+n_2+n_3$, such that a particular set of messages may verify a certain property that we denote $\bigstar$ in the sequel in the middle n_2 rounds (Figure 2).

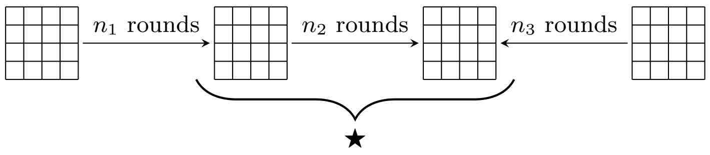


Fig. 2. General scheme of the meet-in-the-middle attack on AES, where some messages in the middle rounds may verify a certain $\bigstar$ property used to perform the meet-in-the-middle

The general attack uses three successive steps:

Precomputation phase

1. In this phase, we build a lookup table T containing all the possible sequences constructed from a δ-set such that one message verifies the $\bigstar$ property.

Online phase

2. Then, in the online phase, we need to identify a δ-set containing a message m verifying the desired property.
3. Finally, we partially decrypt the associated δ-set through the last n_3 rounds and check whether it belongs to T.

The two steps of the online phase require to guess some key bytes while the goal of this attack is to filter some of their values. In the best case, only the right ones should pass the test.

Demirci and Selçuk Attack. The starting point is to consider the set of functions $f : \{0,1\}^8 \to \{0,1\}^8$ that maps a byte of a δ-set to another byte of the state after four AES rounds. A convenient way is to view f as an ordered byte sequence $(f(0), \ldots, f(255))$ so that it can be represented by 256 bytes. The crucial observation made by the generalizing Gilbert and Minier attack is that this set is tiny since it can be described using 25 byte-parameters ($2^{25\cdot 8} = 2^{200}$) compared with the set of all functions of this type which counts as many as $2^{8\cdot 2^8} = 2^{2048}$ elements. Considering the differences $(f(0)-f(0), f(1)-f(0), \ldots, f(255)-f(0))$ rather than values, the set of functions can be described by 24 parameters. Dunkelman et al. identify these parameters as: the full state x_3 of message 0, four bytes of state x_2 of message 0, four bytes of subkey k_3. The four bytes of the state x_2 only depend on the column of z_1 where the active byte of the δ-set is located; for instance, if it is column 0, then those bytes are $x_2[0,1,2,3]$. Similarly, the four bytes of k_3 depend on the column of x_5 where the byte we want to determine is located; as an example, if it is column 0, then those bytes are $k_3[0,5,10,15]$.

In their attacks [12], Demirci and Selçuk use the ★ property that does not filter any message. Consequently, they do not require to identify a particular message m. The data complexity of their basic attack is very small and around 2^{32}. However, since there is no particular property, the size of the table T is very large and the basic attack only works for the AES-256. To mount an attack on the AES-192, they consider some time/memory tradeoff. More precisely, the table T does not contain all the possible states, but only a fraction α. Consequently, a specific δ-set may not be in the table T, so that we have to wait for this event and redo the attack $O(1/\alpha)$ times on average. The attack becomes probabilistic and the memory requirement makes the attack possible for AES-192. The consequence of this advanced version of the attack, which also works for AES-256, is that the amount of data increases a lot. The time and memory requirement of the precomputation phase is due to the construction of table T that contains messages for the $n_2 = 4$ middle rounds, which counts as many as $2^{8\cdot 24} = 2^{192}$ ordered sequences of 256 bytes.

Finally, it is possible to remove from each function some output values. Since we know that these functions can be described by the key of 24 or 32 bytes, one can reduce T by a factor 10 or 8 by storing only the first differences. Such an observation has been used by Wei et al. in [24].

Dunkelman et al. Attack. In [15], Dunkelman, Keller and Shamir introduced two new improvements to further reduce the memory complexity of [12]. The first one uses multisets, behaving as unordered sequences, and the authors show that there is still enough information so that the attack succeeds. The second improvement uses a particular 4-round differential characteristic (Figure 3) to reduce the size of the precomputed lookup table T, at the expense of trying more pairs of messages to expect at least one to conform to the truncated characteristic. The main idea of the differential characteristic is to fix the values of as many state-bytes as possible to a constant. Assume now we have a message

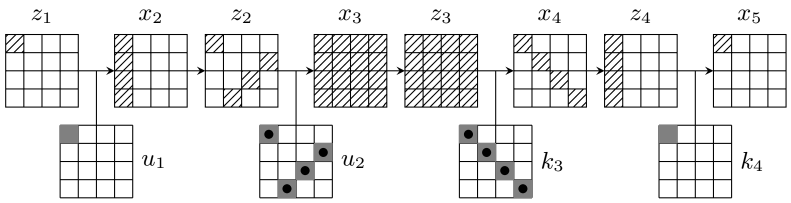


Fig. 3. Truncated differential characteristic used in the middle of the 7-round attacks on AES. A hatched byte denotes a non-zero difference, whereas a while cell has no difference.

m such that we have a pair (m, m') that satisfies the whole 7-round differential characteristic and our goal is to recover the key. Contrary to classical differential attacks, where the adversary guesses some bytes of the last subkey and eliminates the wrong guess, the smart idea of Dunkelman et al. is to use a table to recover the right key more efficiently. Usually, differential attacks do not use memory to recover the key or to find the right pair. The attack principle consists in constructing the δ-set from m which can be made since we already have to guess some key bytes to check if the pair (m, m') has followed the right differential characteristic. Then, the table allows to identify the right key from the encryption of the δ-set.

It is now easy to see that the differential characteristic can be described using only 16 bytes. The states x_3 and y_3 can only take 2^{32} possible differences each, so that the number of solutions for these two states is 2^{64}. We also have the 4 key-bytes of u_2 and the 4 key-bytes of k_3 corresponding to the active bytes of Figure 3 in states z_2 and x_4.

The following Table 1 shows the best cryptanalysis of AES variants, including our new results detailed in this article.

Table 1. Current cryptanalysis of reduced AES variants in the secret-key model

Version	Rounds	Data (CP)	Time	Memory	Technique	Reference
128	7	$2^{112.2}$	$2^{117.2}$	$2^{112.2}$	ID	[20]
	7	$2^{106.2}$	$2^{110.2}$	$2^{90.2}$	ID	[21]
	7	2^{116}	2^{116}	2^{116}	MITM	[15]
	7	$\mathbf{2^{105}}$	$\mathbf{2^{99}}$	$\mathbf{2^{90}}$	**MITM**	**Section 3**
	7	$\mathbf{2^{97}}$	$\mathbf{2^{99}}$	$\mathbf{2^{98}}$	**MITM**	**Section 3**
	8	2^{88}	$2^{125.3}$	2^{8}	Bicliques	[8]
	10 (full)	2^{88}	$2^{126.2}$	2^{8}	Bicliques	[8]
192	7	2^{116}	2^{116}	2^{116}	MITM	[15]
	7	$\mathbf{2^{99}}$	$\mathbf{2^{99}}$	$\mathbf{2^{96}}$	**MITM**	**Section 3**
	8	2^{113}	2^{172}	2^{129}	MITM	[15]
	8	$\mathbf{2^{113}}$	$\mathbf{2^{172}}$	$\mathbf{2^{82}}$	**MITM**	**Section 4.1**
	8	$\mathbf{2^{107}}$	$\mathbf{2^{172}}$	$\mathbf{2^{96}}$	**MITM**	**Section 4.1**
	9	2^{80}	$2^{188.8}$	2^{8}	Bicliques	[8]
	12 (full)	2^{80}	$2^{189.4}$	2^{8}	Bicliques	[8]
256	7	2^{116}	2^{116}	2^{116}	MITM	[15]
	7	$\mathbf{2^{99}}$	$\mathbf{2^{98}}$	$\mathbf{2^{96}}$	**MITM**	**Section 3**
	8	2^{113}	2^{196}	2^{129}	MITM	[15]
	8	$\mathbf{2^{113}}$	$\mathbf{2^{196}}$	$\mathbf{2^{82}}$	**MITM**	**Section 4.1**
	8	$\mathbf{2^{107}}$	$\mathbf{2^{196}}$	$\mathbf{2^{96}}$	**MITM**	**Section 4.1**
	9	2^{120}	$2^{251.9}$	2^{8}	Bicliques	[8]
	9	$\mathbf{2^{120}}$	$\mathbf{2^{203}}$	$\mathbf{2^{203}}$	**MITM**	**Section 4.2**
	14 (full)	2^{40}	$2^{254.4}$	2^{8}	Bicliques	[8]

CP: Chosen-plaintext. ID: Impossible Differential. MITM: Meet-in-the-Middle.

3 New Attack on AES

In this section, we describe our basic attack on AES, which is independent of the key schedule algorithms. We begin in Section 3.1 by describing an efficient way to enumerate and store all the possible multisets in the middle that are used to mount the meet-in-the-middle attack. We continue in Section 3.2 by applying the general scheme previously described to construct a key-recovery attack on all AES versions reduced to 7 rounds. Finally, in Section 3.3, we show that modifying slightly the property for the middle rounds allows to trade some memory for data and time.

3.1 Efficient Tabulation

As in the previous results, our attack also uses a large memory lookup table constructed in the precomputation phase, and used in the online phase. Dunkelman, Keller and Shamir showed that if a message m belongs to a pair of states conforming to the truncated differential characteristic of Figure 3, then the multiset of differences $\Delta x_5[0]$ obtained from the δ−set constructed from m in x_1 can only take 2^{128} values, because 16 of the 24 parameters used to build the multisets can take only 2^{64} values instead of 2^{128}. We make the following proposition that reduces the size of the table by a factor 2^{48}.

Proposition 2. *If a message m belongs to a pair of states conforming to the truncated differential characteristic of Figure 3, then the multiset of differences $\Delta x_5[0]$ obtained from the δ−set constructed from m in x_1 can only take 2^{80} values. More precisely, the 24 parameters (which are state bytes of m) can take only 2^{80} values in that case. Conversely, for each of these 2^{80} values there exists a tuple (m, m', k) such that m is set to the chosen value and, the pair (m, m') follows the truncated characteristic.*

Proof. The proof uses rebound-like arguments borrowed from the hash function cryptanalysis domain [22]. Let (m, m') be a right pair. We show in the following how the knowledge of 10 particular bytes restricts the values of the 24 parameters used to construct the multisets, namely:

$$x_2[0,1,2,3],\ x_3[0,\ldots,15],\ x_4[0,5,10,15]. \tag{1}$$

In the sequel, we use the state names mentioned in Figure 1. The 10 bytes

$$\Delta z_1[0],\ x_2[0,1,2,3],\quad \Delta w_4[0],\ z_4[0,1,2,3]. \tag{2}$$

can take as many as 2^{80} possible values, and for each of them, we can determine the values of all the differences shown on Figure 3: linearly in x_2, applying the SBox to reach y_2, linearly for x_3 and similarly in the other direction starting from z_4.

By the differential property of the AES SBox (Proposition 1), we get on average one value for each of the 16 bytes of state x_3[1]. From the known values around the two **AddRoundKey** layers of rounds 3 and 4, this suggests four bytes of

[1] In fact, only 2^{64} values of the 10 bytes lead to a solution for x_3 but for each value, there are 2^{16} solutions for x_3.

the equivalent subkey $u_2 = \mathsf{MC}^{-1}(k_2)$ and four others in subkey k_3: those are $u_2[0]$, $u_2[7]$, $u_2[10]$, $u_2[13]$ and $k_3[0]$, $k_3[5]$, $k_3[10]$, $k_3[15]$; they are marked by a bullet (●) in Figure 3.

The converse is now trivial: the only difficulty is to prove that for each value of the 8 key bytes, there exists a corresponding master key. This actually gives a chosen-key distinguisher for 7 rounds of AES, as it has been done in [14].

To construct the multiset for each of the 2^{80} possible choice for the 10 bytes from (2), we consider all the $2^8 - 1$ possible values for the difference $\Delta y_1[0]$, and propagate them until x_5. This leads to a multiset of $2^8 - 1$ differences in $\Delta x_5[0]$. Finally, as the AES SBox behaves as a permutation over $\mathbb{F}_{256}$, the sequence in $\Delta y_1[0]$ allows to derive the sequence in $\Delta x_1[0]$. Note that in the present case where there is a single byte of difference between m and m' in the state x_1, both messages belongs to the same δ-set. This does not hold if we consider more active bytes as we will see in Section 4. □

3.2 Simple Attack

Precomputation Phase. In the precomputation phase of the attack, we build the lookup table that contains the 2^{80} multisets for difference Δx_5 by following the proof of Proposition 2. This step is performed by first iterating on the 2^{80} possible values for the 10 bytes of (2) and for each of them, we deduce the possible values of the 24 original parameters. Then, for each of them, we construct the multiset of $2^8 - 1$ differences. Using the differential property of the AES SBox (Proposition 1), we can count exactly the number of multisets that are computed:

$$2^{80} \times \left(4 \times \frac{2^8-1}{(2^8-1)^2} + 2 \times \frac{(2^8-1)(2^7-1-1)}{(2^8-1)^2}\right)^{16} \approx 2^{80.09}. \tag{3}$$

Finally, the lookup table of the $2^{80.09}$ possible multisets that we simplify to 2^{80} requires about 2^{82} 128-bit blocks to be stored. To construct the table, we have to perform 2^{80} partial encryptions on 256 messages, which we estimate to be equivalent to 2^{84} encryptions.

Online Phase. The online phase splits into three parts: the first one finds pairs of messages that conform to the truncated differential characteristic of Figure 3, which embeds the previous 4-round characteristic in the middle rounds. The second step uses the found pairs to create a δ-set and test them against the precomputed table and retrieve the secret key in a final phase.

To generate one pair of messages conforming to the 7-full-round characteristic where there are only four active bytes in both the plaintext and the ciphertext differences, we prepare a structure of 2^{32} plaintexts where the diagonal takes all the possible 2^{32} values, and the remaining 12 bytes are fixed to some constants. Hence, each of the $2^{32} \times (2^{32} - 1)/2 \approx 2^{63}$ pairs we can generate satisfies the plaintext difference. Among the 2^{63} corresponding ciphertext pairs, we expect $2^{63} \cdot 2^{-96} = 2^{-33}$ to verify the truncated difference pattern. Finding *one* such pair then requires 2^{33} structures of 2^{32} messages and $2^{32+33} = 2^{65}$ encryptions under the secret key. Using this secret key, the probability that the whole truncated characteristic of Figure 3 is verified is $2^{-2\times3\times8} = 2^{-48}$ because of the two $4 \to 1$ transitions in the **MixColumns** of rounds 0 and 5. By repeating the previous procedure to find 2^{48} pairs, one is expected to verify the full 7-round characteristic. All in all, we ask the encryptions of $2^{48+65} = 2^{113}$ messages to find 2^{48} pairs of messages. Note that we do not have to examine each pair in order to find the right one. Indeed, if a pair verifies the full 7-round characteristic, then the ciphertext difference has only four active bytes. Thus, we can store the structures in a hash table indexed by the 12 inactive bytes to get the right pairs in average time of one.

For each of the 2^{48} pairs, we get $2^{8\times(8-2\times3)} \cdot 2^8 = 2^{24}$ suggestions for the 9 key bytes: $k_{-1}[0,5,10,15]$, $u_5[0]$ and $u_6[0,7,10,13]$. Indeed, there are 2^8 possibilities for the bytes from k_{-1} since the pair of diagonals in x_0 need to be active only in w_0 after the **MixColumns** operation. Among the 2^{32} possible values for those bytes, only $2^{32} \times 2^{-24} = 2^8$ verifies the truncated pattern. The same reasoning applies for $u_6[0,7,10,13]$, and the last byte $u_5[0]$ can take all the 2^8 values.

For all 2^{24} possibilities, we construct a δ-set to use the precomputed table. To do so, we partially decrypt the diagonal of one message, using the four known bytes from k_{-1} and consider the $2^8 - 1$ possible non-zero differences for $\Delta x_1[0]$. This gives one set of 2^8 plaintexts, whose corresponding ciphertexts may be partially decrypted using the four known bytes from u_6 and the one from u_5. Once decrypted, we can construct the multiset of differences for Δx_5 and check if it lies in the precomputed lookup table. If not, we can discard the subkey with certainty. On the other hand, the probability for a wrong guess to pass this test is smaller than $2^{80} \cdot 2^{-467.6} = 2^{-387.6}$ so, as we try $2^{48} \cdot 2^{24} = 2^{72}$ multisets, only the right subkey should verify the test. Note that the probability is $2^{-467.6}$ (and not $2^{-506.17}$) because the number of ordered sequences associated to a multiset is not constant.

To evaluate the complexity of the online phase of the simple attack, we count the number of AES encryptions. First, we ask the encryption of 2^{113} chosen-plaintexts, so that the time complexity for that step is already 2^{113} encryptions. Then, for each of the 2^{48} found pairs, we perform 2^{24} partial encryptions/decryptions of a δ-set. We evaluate the time complexity of this part to $2^{48+24+8} \cdot 2^{-5} = 2^{75}$ encryptions since we can do the computations in a good ordering. All in all, the time complexity is dominated by 2^{113} encryptions, the data complexity is 2^{113} chosen-plaintexts, and the memory complexity is 2^{82} since it requires to store 2^{80} multisets.

3.3 Efficient Attack: New Property ★

Unlike the previous attacks where the bottleneck complexity is the memory, our attack uses a smaller table which makes the time complexity to find the pairs the dominating one. Therefore, we would like to decrease the time spent in that phase. The natural idea is to find a new property ★ for the four middle rounds that can be checked more efficiently. To do so, we reuse the idea of Dunkelman et al. from [15], which adds an active byte in the second round of the differential

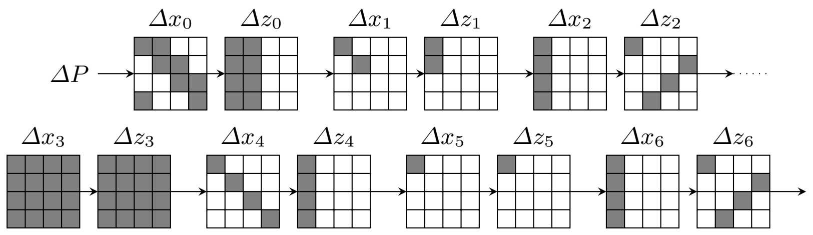


Fig. 4. Example of a truncated differential characteristic used in the efficient attack on 7 rounds

characteristic. The sequence of active bytes becomes:

$$8 \xrightarrow{R_0} 2 \xrightarrow{R_1} 4 \xrightarrow{R_2} 16 \xrightarrow{R_3} 4 \xrightarrow{R_4} 1 \xrightarrow{R_5} 4 \xrightarrow{R_6} 16, \qquad (4)$$

with the constraint that the two active bytes of the second round belong to the same diagonal to be transformed in a column in the next round.

As a consequence, it is now easier to find pairs conforming to that truncated differential characteristic. Indeed, the size of the structures of plaintexts may take as many as 2^{64} different values, so that we can construct at most $2^{64} \cdot (2^{64}-1)/2 = 2^{127}$ pairs from each structure. Therefore, it is enough to ask the encryption of $2^{8\cdot3\cdot3}/2^{127-8\cdot12} = 2^{41}$ structures to get 2^{72} pairs with the desired output difference pattern, and expect one to conform to the 7-round characteristic of Figure 4. Consequently in this new setting, we only need 2^{105} chosen plaintexts. In return, the number of pairs that the adversary has to consider is increased by a factor 2^{24} and so is the time complexity. Furthermore, we now need 11 parameters to generate the 24 parameters of the precomputed table, increasing the memory requirement by a factor 2^8. These parameters are the previous 10 ones and the difference in the second active byte of z_2. All in all, the time complexity of this attack is $2^{75+24} = 2^{99}$ encryptions, the data complexity is 2^{105} chosen plaintexts and the memory requirement is $2^{82+8} = 2^{90}$ 128-bit blocks.

Note that the time spent on one pair is the same for both the simple attack and the new one. Indeed, let K be the key bytes needed to construct the multiset. We suppose that we have a set of pairs such that one follows the differential. To find it, and incidentally some key-byte values, we proceed as follows: for each pair (m, m'), enumerate all possible values of K such that (m, m', K) have a non-zero probability to follow the differential. For each of them, construct the corresponding multiset from m or m'. If it belongs to the table, then we expect that it follows the differential characteristic since the table has been constructed that way. Otherwise, we know with probability 1 that either the pair (m, m') does not satisfy the characteristic, or the guessed value from K is wrong.

Assuming that the bytes of diagonals 0 and 2 of the structure of plaintexts takes all the values[2], the two differences in the first state of the second round can take four different positions: $(0, 10)$, $(1, 11)$, $(2, 8)$ and $(3, 9)$. Similarly, the

[2] Those are bytes 0, 2, 5, 7, 8, 10, 13 and 15.

position of the active byte in the penultimate round is not constrained; it can be placed anywhere on the 16 positions. We can also consiser the opposite: one active byte at the beginning, and two active bytes in the end. These possibilities actually define tweaked versions of the property ★ and allows to trade some time for memory: with less data, we can check more tables for the same final probability of success. Namely, by storing $4 \times 16 + \binom{4}{2} \times 4 = 2^8$ tables to cover all the cases by adapting the proof of Proposition 2, the encryption of $2^{41}/2^8 = 2^{33}$ structures of 2^{64} plaintexts suffices to expect a hit in one of the 2^8 tables. Therefore, the memory complexity reaches 2^{98} AES blocks and the time complexity remains unchanged since we analyze 2^8 times less pairs, but the quantity of work to check *one* pair is multiplied by the same factor.

3.4 Turning the Distinguisher into a Key Recovery Attack

In this section, we present an efficient way to turn this distinguisher into a key recovery attack. First, let us summarize what the adversary has in his possession at the end of the efficient attack: a pair (m, m') following the truncated differential characteristic, a δ-set containing m, the knowledge of 9 key bytes and the corresponding multiset for which we found a match in the precomputed table. Thus, there are still 2^{56}, 2^{120} or 2^{184} possible keys, if we consider AES-128, AES-192 or AES-256 respectively. As a consequence, performing an exhaustive search to find the missing key bytes would drastically increase the complexity of the whole attack, except for the 128-bit version. Even in that case, it seems nontrivial to recover the 2^{56} possible keys in less than 2^{96}, as the 9 key bytes do not belong to the same subkey.

A natural way to recover the missing bytes would be to replay the efficient attack by using different positions for the input and output differences. Unfortunately, this increases the complexity, and it would also interfere with the trade-off since we could not look for several positions of the differences anymore.

We propose a method that recovers the two last subkeys in a negligible time compared to the 2^{99} encryptions of the efficient attack. First the adversary guesses the 11 parameters used to build the table of multisets, computes the value the corresponding 24 parameters and keeps the only one used to build the checked multiset. In particular, he obtains the value of the all intermediate state x_3 and one column of x_2. As a consequence, and for any position of the active byte of x_5, the Demerci and Selçuk original attack may be performed really quickly. Indeed, among the 9 (resp. 24) bytes to guess to perform the online (resp. offline) phase, at least 4 (resp. 20) are already known and the data needed is also in his possession. Finally, the adversary do this attack for each position of the active byte of x_5 and thus retrieves the two last subkeys.

4 Extension to More Rounds

4.1 8-Round Attacks on AES-192 and AES-256

We can extend the simple attack on the AES presented Section 3.2 to an 8-round attack for both 192- and 256-bit versions by adding one additional round at the

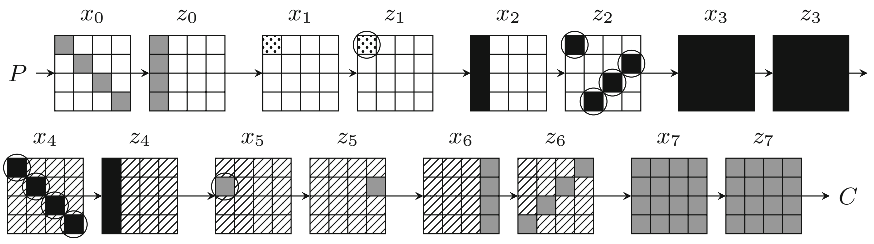


Fig. 5. Scheme of the attack on 8 rounds. Gray bytes are needed to identify a δ-set and to build the multiset. Black bytes are needed to construct the table. White bytes are constant for a δ-set. If differences in hashed bytes are null then black bytes can be derived from the difference in circled bytes.

end. This attack is schematized on Figure 5. The main difficulty compared to the previous attack is that we cannot apply a first step to the structure to filter the wrong pairs. Indeed, now for each pair from the structure, there exists at least one key such that the pair follows the differential characteristic. Then our goal is to enumerate, for each pair and as fast as possible, the key bytes needed to identify a δ-set and construct the associated multiset assuming that the pair is a right one.

The main idea to do so is the following: if there is a single non-zero difference in a column of a state before (resp. after) the **MixColumns** operation, then the difference on same column in the state after (resp. before) can only assume 2^8-1 values among all the $(2^8-1)^4$ possible ones. Combining this with the key schedule equations and to the differential property of the AES SBox (Proposition 1), this leads to an attack requiring 2^{113} chosen plaintexts, 2^{82} 128-bit blocks of storage and a time complexity equivalent to 2^{172} (resp. 2^{196}) encryptions on AES-192 (resp. AES-256).

To reach this time complexity, the position of the output active byte must be chosen carefully. The position of the input active byte for both the pair and the δ-set must be identical, as well as the output active byte of the pair and the byte that is to be checked. Then, the output difference must be located at position 1, 6, 11 or 12 in the case of AES-192. As for the AES-256, it can be located anywhere, except on bytes 0, 5, 10 and 15. Finally, in both cases, the position of the input difference does not matter.

Assume that the positions of the input and output active bytes are respectively 0 and 1. In the first stage of the attack, we ask for the encryption of 2^{81} structures of 2^{32} plaintexts. Then, the following procedure applied on each of the $2^{81} \cdot 2^{32+31} = 2^{144}$ pairs allows to enumerate the 2^{24} possible values for the needed key bytes in about 2^{24} simple operations for the 192-bit version:

1. (a) Guess the difference in $x_1[0]$ then deduce bytes 0, 5, 10 and 15 of k_{-1}.
   (b) Store all these values in a hash table T_{-1} indexed by $k_{-1}[15]$.
2. Guess the difference in $z_5[13]$.
3. (a) Guess the difference in $z_6[3]$ and $z_6[6]$, and deduce the actual values of the two first columns of x_7 and bytes $x_6[14]$ and $x_6[15]$.

   (b) Deduce $u_6[3]$, $u_6[6]$ and bytes 0, 1, 4, 7, 10, 11, 13 and 14 of k_7 (or u_7 if we do not omit the last **MixColumns**) and store all these values in a hash table T_7.
4. (a) Similarly, guess the difference in the two other active bytes of z_6 and deduce $u_6[9]$, $u_6[12]$ and the 8 others bytes of the last subkey.
   (b) Retrieve $u_6[3]$, $u_6[6]$ and bytes 0, 1, 4, 7, 10, 11, 13 and 14 of k_7 (resp. u_7) using T_7 since $u_6[3]$ and $u_6[6]$ are linearly dependent of k_7 (and also of u_7).
   (c) Deduce $u_5[13]$ and $k_{-1}[15]$ from k_7.
   (d) Get bytes 0, 5 and 10 of k_{-1} using T_{-1}.

The fact we can deduce $u_5[13]$, $u_6[3]$, $u_6[6]$ comes from the following observation.

Proposition 3. *By the key schedule of* AES-192*, knowledge of the subkey k_7 allows to linearly deduce columns* 0 *and* 1 *of k_6 and column* 3 *of k_5.*

In contrast, to deduce $k_{-1}[15]$ from k_7, we need a more complicated observation made by Dunkelman et al. in [15].

Proposition 4 (Key bridging, [15]). *By the key schedule of* AES-192*, the knowledge of columns* 0, 1, 3 *of the subkey k_7 allows to deduce column* 3 *of the whitening key k_{-1}.*

Note that it is now easy to see why the choice of the input active byte does not affect the complexity and why only four positions for the output active byte lead to the minimal complexity.

Finally, for each of the 2^{144} pairs and for each of the 2^{24} subkeys corresponding to one pair, the adversary identifies the δ-set and verifies whether the corresponding multiset belongs to the precomputed table. Thus, the time complexity of this part is equivalent to $2^{144} \cdot 2^{24} \cdot 2^{8} \cdot 2^{-4} = 2^{172}$ encryptions.

In the case of the 256-bit version, k_6 and k_7 are independent and the only key schedule property we can use is the following one.

Proposition 5. *By the key schedule of* AES-256*, knowledge of the subkey k_7 allows to linearly deduce columns 1, 2 and 3 of k_5.*

Then, there are 2^{48} possible values for the required key bytes and a procedure like the previous one enumerates them in 2^{48} simple operations.

It is possible to save some data in exchange for memory by considering several differentials in parallel. We can bypass the fact that all the positions for the output active byte does not lead in the same complexity by performing the check on y_5 instead of x_5. This is done by just adding one parameter to the precomputed table and increases its size by a factor 2^8. Then, we can look for all the $4 \cdot 16 = 2^6$ differentials in parallel on the same structure. All in all, the data complexity and the memory requirement become respectively 2^{107} chosen plaintexts and 2^{96} 128-bit blocks.

4.2 9-Round Attack on AES-256

The 8-round attack on AES-256 can be extended to an attack on 9-round by adding one round right in the middle. This only increases the memory requirements: the time and data complexities remain unchanged. More precisely, the number of parameters needed to construct the precomputed table turns out to be $24 + 16 = 40$, but they can only assume $2^{8\times(10+16)} = 2^{208}$ different values. All in all, the data complexity of the attack stays at 2^{113} chosen-plaintexts, the time complexity remains 2^{196} encryptions and the memory requirement reaches about 2^{210} 128-bit blocks. To reduce its complexity, we can cover only a fraction 2^{-7} of the possible multisets stored in the precomputed table. In return, the data and time complexities are increased by a factor 2^7 by replaying the attack several times. This way, we reach the complexities mentioned in Table 1. This attack is schematized on Figure 6.

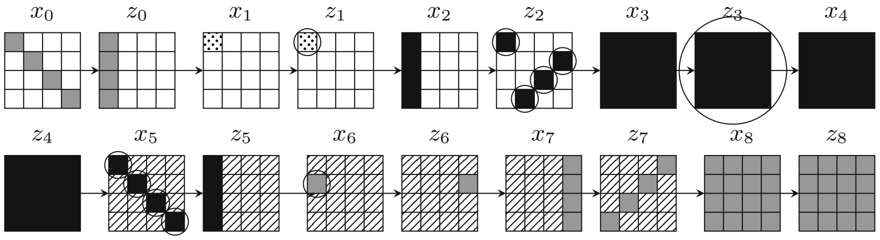


Fig. 6. Scheme of the attack on 9 rounds. Gray bytes are needed to identify a δ-set and to build the multiset. Black bytes are needed to construct the table. White bytes are constant for a δ-set. If differences in hashed bytes are null then black bytes can be derived from the difference in circled bytes.

5 Conclusion

In this article, we have provided improved cryptanalysis of reduced round variants of all the AES versions in the standard single-key model, where the adversary wants to recover the secret key. In particular, we present the best attack on 7 rounds of all AES versions that runs in less than 2^{100} encryptions of chosen-plaintexts, and an attack on 9 rounds of AES-256 in about 2^{203} encryptions.

References

1. Aoki, K., Sasaki, Y.: Meet-in-the-Middle Preimage Attacks Against Reduced SHA-0 and SHA-1. In: Halevi: [19], pp. 70–89
2. Bahrak, B., Aref, M.R.: A Novel Impossible Differential Cryptanalysis of AES. In: WEWoRc (2007)
3. Biham, E., Keller, N.: Cryptanalysis of Reduced Variants of Rijndael. Tech. rep., Computer Science Department, Technion – Israel Institute of Technology (2000)
4. Biryukov, A., Dunkelman, O., Keller, N., Khovratovich, D., Shamir, A.: Key Recovery Attacks of Practical Complexity on AES-256 Variants with up to 10 Rounds. In: Gilbert:[17], pp. 299–319

5. Biryukov, A., Khovratovich, D.: Related-Key Cryptanalysis of the Full AES-192 and AES-256. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 1–18. Springer, Heidelberg (2009)
6. Biryukov, A., Khovratovich, D., Nikolic, I.: Distinguisher and Related-Key Attack on the Full AES-256. In: Halevi: [19], pp. 231–249
7. Biryukov, A., Nikolic, I.: Automatic Search for Related-Key Differential Characteristics in Byte-Oriented Block Ciphers: Application to AES, Camellia, Khazad and Others. In: Gilbert: [17], pp. 322–344
8. Bogdanov, A., Khovratovich, D., Rechberger, C.: Biclique Cryptanalysis of the Full AES. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 344–371. Springer, Heidelberg (2011)
9. Bouillaguet, C., Derbez, P., Fouque, P.-A.: Automatic Search of Attacks on Round-Reduced AES and Applications. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 169–187. Springer, Heidelberg (2011)
10. Daemen, J., Knudsen, L.R., Rijmen, V.: The Block Cipher SQUARE. In: Biham, E. (ed.) FSE 1997. LNCS, vol. 1267, pp. 149–165. Springer, Heidelberg (1997)
11. Daemen, J., Rijmen, V.: AESproposal: Rijndael (1998)
12. Demirci, H., Selçuk, A.A.: A Meet-in-the-Middle Attack on 8-Round AES. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 116–126. Springer, Heidelberg (2008)
13. Demirci, H., Taşkın, İ., Çoban, M., Baysal, A.: Improved Meet-in-the-Middle Attacks on AES. In: Roy, B., Sendrier, N. (eds.) INDOCRYPT 2009. LNCS, vol. 5922, pp. 144–156. Springer, Heidelberg (2009)
14. Derbez, P., Fouque, P.-A., Jean, J.: Faster Chosen-Key Distinguishers on Reduced-Round AES. In: Galbraith, S., Nandi, M. (eds.) INDOCRYPT 2012. LNCS, vol. 7668, pp. 225–243. Springer, Heidelberg (2012)
15. Dunkelman, O., Keller, N., Shamir, A.: Improved Single-Key Attacks on 8-Round AES-192 and AES-256. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 158–176. Springer, Heidelberg (2010)
16. Ferguson, N., Kelsey, J., Lucks, S., Schneier, B., Stay, M., Wagner, D., Whiting, D.L.: Improved Cryptanalysis of Rijndael. In: Schneier, B. (ed.) FSE 2000. LNCS, vol. 1978, pp. 213–230. Springer, Heidelberg (2001)
17. Gilbert, H. (ed.): EUROCRYPT 2010. LNCS, vol. 6110. Springer, Heidelberg (2010)
18. Gilbert, H., Minier, M.: A Collision Attack on 7 Rounds of Rijndael. In: AES Candidate Conference, pp. 230–241 (2000)
19. Halevi, S. (ed.): CRYPTO 2009. LNCS, vol. 5677. Springer, Heidelberg (2009)
20. Lu, J., Dunkelman, O., Keller, N., Kim, J.: New Impossible Differential Attacks on AES. In: Chowdhury, D.R., Rijmen, V., Das, A. (eds.) INDOCRYPT 2008. LNCS, vol. 5365, pp. 279–293. Springer, Heidelberg (2008)
21. Mala, H., Dakhilalian, M., Rijmen, V., Modarres-Hashemi, M.: Improved Impossible Differential Cryptanalysis of 7-Round AES-128. In: Gong, G., Gupta, K.C. (eds.) INDOCRYPT 2010. LNCS, vol. 6498, pp. 282–291. Springer, Heidelberg (2010)
22. Mendel, F., Rechberger, C., Schläffer, M., Thomsen, S.S.: The Rebound Attack: Cryptanalysis of Reduced Whirlpool and Grøstl. In: Dunkelman, O. (ed.) FSE 2009. LNCS, vol. 5665, pp. 260–276. Springer, Heidelberg (2009)
23. NIST: Advanced Encryption Standard (AES), FIPS 197. Tech. Rep., NIST (November 2001)
24. Wei, Y., Lu, J., Hu, Y.: Meet-in-the-Middle Attack on 8 Rounds of the AES Block Cipher under 192 Key Bits. In: Bao, F., Weng, J. (eds.) ISPEC 2011. LNCS, vol. 6672, pp. 222–232. Springer, Heidelberg (2011)

New Links between Differential and Linear Cryptanalysis

Céline Blondeau and Kaisa Nyberg

Aalto University, School of Science,
Department of Information and Computer Science
{celine.blondeau,kaisa.nyberg}@aalto.fi

Abstract. Recently, a number of relations have been established among previously known statistical attacks on block ciphers. Leander showed in 2011 that statistical saturation distinguishers are on average equivalent to multidimensional linear distinguishers. Further relations between these two types of distinguishers and the integral and zero-correlation distinguishers were established by Bogdanov et al. [6]. Knowledge about such relations is useful for classification of statistical attacks in order to determine those that give essentially complementary information about the security of block ciphers. The purpose of the work presented in this paper is to explore relations between differential and linear attacks. The mathematical link between linear and differential attacks was discovered by Chabaud and Vaudenay already in 1994, but it has never been used in practice. We will show how to use it for computing accurate estimates of truncated differential probabilities from accurate estimates of correlations of linear approximations. We demonstrate this method in practice and give the first instantiation of multiple differential cryptanalysis using the LLR statistical test on PRESENT. On a more theoretical side, we establish equivalence between a multidimensional linear distinguisher and a truncated differential distinguisher, and show that certain zero-correlation linear distinguishers exist if and only if certain impossible differentials exist.

Keywords: statistical cryptanalysis, block cipher, key-alternating block cipher, multiple differential attack, truncated differential, multidimensional linear attack, zero-correlation, impossible differential.

1 Introduction

Block ciphers are used as building blocks for many symmetric cryptographic primitives for encryption, authentication, pseudo-random number generation, and hash functions. Security of these primitives is evaluated in regard to known attacks against block ciphers. Among the different types of attacks, the statistical ones exploit non-uniform behaviour of the data extracted from the cipher to find information about the secret key. Linear cryptanalysis [25] and differential cryptanalysis [4] are the most prominent statistical attacks against block ciphers.

Recently, a number of relations have been established among some previously known statistical attacks on block ciphers. Leander [23] observed that the statistical saturation distinguishers [15] are on average equivalent to multidimensional linear distinguishers [20]. Further relations between these two types of distinguishers and the integral and zero-correlation distinguishers were established by Bogdanov et al. [6]. The goal of the work presented in this paper is to explore relations between linear and differential attacks. The strength of linear distinguishers relies on exceptionally high correlation, or a complete lack of it, while differential distinguishers are measured based on their probabilities. In the latter case also impossible differentials can be meaningful. The mathematical link between differential probability and linear correlation was presented by Chabaud and Vaudenay already in 1994 [12], but has never been used in practice due to its large computational complexity. In spite of this link, it is well known that resistance against differential cryptanalysis does not imply resistance against linear cryptanalysis. Also examples of the converse situation are known in the classical setting of distinguishers based on single differentials and single linear approximations [28]. In this paper, we will see that the situation changes when the distinguishers involve multiple differentials and linear approximations. Indeed, we will establish relations between multidimensional linear distinguishers and truncated differential distinguishers, and show, in particular, that existence of a zero-correlation relation is equivalent to existence of an impossible differential property.

The second major goal of the current paper is to apply the Chabaud-Vaudenay link in practice. The main motivation is due to the fact that, for some ciphers, it may be easier to evaluate probabilities of differentials than correlations of linear approximations, and for some other ciphers, the other way round. The block cipher PRESENT [5] is known to have a clear structure of linear approximations and their correlations have previously been evaluated accurately in [14] and [23]. On the other hand, the differentials over PRESENT split to numerous differential trails and their probabilities are hard to evaluate directly using traditional methods such as branch-and-bound algorithms [26,7] or transition matrices [16].

As computation of the exact formula of the Chabaud-Vaudenay link between differential probabilities and squared correlations is not feasible, we develop a method based on theoretical arguments and assumptions to reduce the time complexity of the computation. The validity of these assumptions is then tested on a reduced-round version of PRESENT and PUFFIN [13].

Recently, an attack called multiple differential cryptanalysis (MDC) was proposed as an "all-in-one" generalisation of differential cryptanalysis [2,9]. In these papers, distributions of differences for small block ciphers were evaluated to provide attacks using `LLR` and χ^2 scoring functions. This model, which improves and generalises differential, truncated differential, and impossible differential cryptanalysis methods remained, however, to be completed. To apply the `LLR` statistical test to actual block ciphers, cryptanalysts must be able to provide an upstream evaluation of the differential probabilities [9]. Up to now, computation of differential probabilities has been challenging for many ciphers. Given the method described above, we compute accurate estimates of truncated differential probabilities and give the first practical instantiation of multiple differential cryptanalysis using the LLR statistical test on PRESENT.

The rest of the paper is organised as follows. In Section 2, we first recall the basic definitions, the link between differential probabilities and linear correlations, and present the theoretical foundations for reducing the time complexity of using the link in practice. We then establish two new links between linear and differential cryptanalysis. The first one expresses the capacity of a multidimensional linear approximation in terms of a truncated differential probability, and the second one shows a relation between zero-correlation approximations and impossible differentials. In Section 3 we present the method for computing squared correlations for key-alternating block ciphers. Section 4 is devoted to the MDC method, the related LLR test, and its data complexity. In Section 5, parameters, like time complexity of the computation and time complexity of the MDC are described. In Section 6 we present the results from practical experiments and conclude in Section 7.

2 Links between Differential and Linear Cryptanalysis

2.1 Differential Probabilities and Correlations of Linear Approximations

In differential cryptanalysis [4], the attacker is interested in finding and exploiting non-uniformity in occurrences of plaintext and ciphertext differences. Given a vectorial Boolean function $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$, a differential is a pair (δ, Δ) where $\delta \in \mathbb{F}_2^n$ and $\Delta \in \mathbb{F}_2^m$ and its probability is defined as

$$\mathbf{P}[\delta \xrightarrow{F} \Delta] = \mathbf{P}_{\mathbf{X}}\left[F(\mathbf{X}) \oplus F(\mathbf{X} \oplus \delta) = \Delta\right],$$

where the probability is taken over the distribution of $\mathbf{X}$. Throughout this paper, it will be assumed that $\mathbf{X}$ is uniformly distributed in $\mathbb{F}_2^n$ in which case

$$\mathbf{P}[\delta \xrightarrow{F} \Delta] = 2^{-n}\#\{x \in \mathbb{F}_2^n \mid F(x) \oplus F(x \oplus \delta) = \Delta\}.$$

Linear cryptanalysis [25] uses a linear relation between bits from plaintexts, corresponding ciphertext and encryption key. Linear relations are expressed as Boolean functions of the plaintext and the key. The strength of the linear relation is measured by its correlation.

Let $f : \mathbb{F}_2^n \to \mathbb{F}_2$ be a Boolean function. Its correlation is defined as its correlation with the all-zero function as

$$\mathbf{cor}_x(f) = 2^{-n}\Big[\#\{x \in \mathbb{F}_2^n | f(x) = 0\} - \#\{x \in \mathbb{F}_2^n | f(x) \neq 0\}\Big],$$

where the quantity within brackets can be computed as the Walsh transform of f evaluated at zero, see e.g. [11].

Given a vectorial Boolean function $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ we are interested in Boolean functions $f(x) = a \cdot x \oplus b \cdot F(x)$ defined by linear relations where $a \in \mathbb{F}_2^n$ and

$b \in \mathbb{F}_2^m$ are called linear input and output masks. Chabaud and Vaudenay showed that differential probabilities and squared linear correlations are linked to each other by the Walsh transform.

Theorem 1 ([12]). *Let $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ be a vectorial Boolean function. The probability of the differential (δ, Δ) over F can be given as*

$$\mathbf{P}[\delta \xrightarrow{F} \Delta] = 2^{-m} \sum_{a \in \mathbb{F}_2^n} \sum_{b \in \mathbb{F}_2^m} (-1)^{a\cdot\delta \oplus b\cdot\Delta} \mathbf{cor}_x^2 \left(a \cdot x \oplus b \cdot F(x)\right). \tag{1}$$

This formula has not been used before to compute differential probabilities of block cipher in practice. Indeed, the direct application of it would require computation and summing up 2^{n+m} squared correlations where n is the length of the input and m is the length of the output in bits of the function F. Later we will see that restricting attention to truncated differentials of a block cipher would allow us to reduce the size of the output space. Still, the problem with the large input space remains. Next, let us recall an important result of correlations of restrictions of Boolean functions.

Theorem 2. *Let $F : \mathbb{F}_2^s \times \mathbb{F}_2^t \to \mathbb{F}_2^m$ be a vectorial Boolean function, and let $x_t \in \mathbb{F}_2^t$ be uniformly distributed. Then*

$$\sum_{x_t \in \mathbb{F}_2^t} \mathbf{cor}_{x_s}^2 \left(a \cdot x_s + b \cdot F(x_s, x_t)\right) = 2^t \sum_{c \in \mathbb{F}_2^t} \mathbf{cor}_{x_s, x_t}^2 \left(a \cdot x_s + c \cdot x_t + b \cdot F(x_s, x_t)\right),$$

for all $a \in \mathbb{F}_2^s$ and all $b \in \mathbb{F}_2^m$.

This fact appeared in the context of Boolean functions as Lemma 4 of [24], see also [11], and was named as Fundamental Theorem in [27]. It describes the underlying principle for computing the average squared linear correlation, see Theorem 4 below, as well as for demonstrating the existence of the link between statistical saturation attack and the multidimensional attack [23]. We will use it now to derive the first result for reducing computations of differential probabilities according to Formula (1).

In our experiments, we observed that for SPN type block ciphers the number of active Sboxes at the first round influences the probability of the differential. Large probabilities can be found only with small number of active S-boxes. Hence, from the cryptanalyst's point of view, it seems reasonable to select the input difference δ to have a small Hamming weight. In such a situation we can apply Theorem 2 and reduce the space over which the correlations are computed.

Lemma 1. *Let $\mathbb{F}_2^n = \mathbb{F}_2^s \times \mathbb{F}_2^t$ and $\delta \in \mathbb{F}_2^n$ be such that $\delta = (\delta_s, \delta_t)$ where $\delta_s \in \mathbb{F}_2^s$ and $\delta_t \in \mathbb{F}_2^t$. If $\delta_t = 0$, then we have, for any fixed $b \in \mathbb{F}_2^m$,*

$$\begin{aligned} &\sum_{a \in \mathbb{F}_2^n} (-1)^{a\cdot\delta} \mathbf{cor}_x^2 \left(a \cdot x \oplus b \cdot F(x)\right) \\ &= 2^{-t} \sum_{x_t \in \mathbb{F}_2^t} \sum_{a_s \in \mathbb{F}_2^s} (-1)^{a_s\cdot\delta_s} \mathbf{cor}_{x_s}^2 (a_s \cdot x_s \oplus b \cdot F(x_s, x_t)), \end{aligned}$$

where $x = (x_s, x_t) \in \mathbb{F}_2^s \times \mathbb{F}_2^t$ and $a_s \in \mathbb{F}_2^s$.

This formula involves restricting the input space artificially by fixing a part of the input to $x_t \in \mathbb{F}_2^t$, and then taking the average over these fixations. According to our experiments this average can be accurately estimated in practice by restricting x_t to a small subset T of $\mathbb{F}_2^t$. How to choose T depends on the specific structure of the cipher under consideration and will help to reduce the time computation from 2^n to 2^s.

2.2 Links between Multidimensional Linear Approximations and Truncated Differentials

In this section, we present new links between multidimensional linear and truncated differential attacks. A multidimensional linear relation (approximation) of a vectorial Boolean function is a linear space formed by a number of linear relations. Without loss of generality, we can assume that the input space and output space is split into two subspaces so that $F : \mathbb{F}_2^s \times \mathbb{F}_2^t \to \mathbb{F}_2^q \times \mathbb{F}_2^r$. Let us consider linear approximations of the form

$$(a_s, 0) \cdot x \oplus (b_q, 0) \cdot F(x), \;\; a_s \in \mathbb{F}_2^s,\, b_q \in \mathbb{F}_2^q,$$

and truncated differentials of the form

$$(\delta_s, *) \xrightarrow{F} (\Delta_q, *), \;\; \delta_s \in \mathbb{F}_2^s,\, \Delta_q \in \mathbb{F}_2^q,$$

and define the probability of such a truncated differential as

$$\mathbf{P}((\delta_s, *) \xrightarrow{F} (\Delta_q, *)) = 2^{-t} \sum_{\delta_t \in \mathbb{F}_2^t} \sum_{\Delta_r \in \mathbb{F}_2^r} \mathbf{P}((\delta_s, \delta_t) \xrightarrow{F} (\Delta_q, \Delta_r)).$$

Then by summing up on both sides of Equation (1) over all $\delta_t \in \mathbb{F}_2^t$ and $\Delta_r \in \mathbb{F}_2^r$, we obtain the following link between truncated differentials and multidimensional linear approximations.

Theorem 3. *For all $\delta_s \in \mathbb{F}_2^s$ and $\Delta_q \in \mathbb{F}_2^q$ it holds that*

$$\mathbf{P}((\delta_s, *) \xrightarrow{F} (\Delta_q, *)) = 2^{-q} \sum_{a_s, b_q} (-1)^{a_s \cdot \delta_s \oplus b_q \cdot \Delta_q} \mathbf{cor}_x^2((a_s, 0) \cdot x \oplus (b_q, 0) \cdot F(x)).$$

As an application of this result, let us consider a function, which satisfies an integral [17], for which some part of the output is uniformly distributed if some part of the input is fixed to an arbitrary value. One example of such a function is a three-round Feistel network with a bijective round-function. Another example is a function formed by three rounds backward or four rounds forward of the AES encryption function [22,18]. As corollary of Theorem 3 we obtain the equivalence between such an integral condition and a condition on certain truncated differentials.

Corollary 1. *Let $F : \mathbb{F}_2^s \times \mathbb{F}_2^t \to \mathbb{F}_2^q \times \mathbb{F}_2^r$. Then the following are equivalent:*

(i) $\mathbf{cor}_{x_t}((b_q, 0) \cdot F(x_s, x_t)) = 0$ *for all* $x_s \in \mathbb{F}_2^s$ *and* $b_q \in \mathbb{F}_2^q \setminus \{0\}$,
(ii) $\mathbf{cor}_x((a_s, 0) \cdot x \oplus (b_q, 0) \cdot F(x)) = 0$ *for all* $a_s \in \mathbb{F}_2^s$ *and* $b_q \in \mathbb{F}_2^q \setminus \{0\}$,
(iii) $\mathbf{P}((\delta_s, *) \xrightarrow{F} (\Delta_q, *)) = 2^{-q}$ *for all* $\delta_s \in \mathbb{F}_2^s$ *and* $\Delta_q \in \mathbb{F}_2^q$,
(iv) $\mathbf{P}((0, *) \xrightarrow{F} (0, *)) = 2^{-q}$.

Proof. The equivalence of (i) and (ii) follows from Theorem 2. By Theorem 3, (ii) implies (iii). The implication from (iii) to (iv) is trivial, and finally, (iv) implies (ii) by Theorem 3.

The first condition means that the distribution of the first q bits of the output is uniform when taken over a fixed component x_s and variable component x_t in the input. Obviously, the conditions of Corollary 1 can hold only if $t \geq q$. In case $t = q$, we have the following equivalence between zero-correlation linear approximations and impossible differentials.

Corollary 2. *Let* $F : \mathbb{F}_2^s \times \mathbb{F}_2^t \to \mathbb{F}_2^t \times \mathbb{F}_2^r$ *be a vectorial Boolean function. Then all non-trivial linear relations* $(a_s, 0) \cdot x \oplus (b_q, 0) \cdot F(x)$, $a_s \in \mathbb{F}_2^s$, $b_q \in \mathbb{F}_2^q \setminus \{0\}$, *have correlation zero if and only if all non-trivial differentials* $(0, \delta_q) \xrightarrow{F} (0, \Delta_r)$, $\delta_q \in \mathbb{F}_2^q \setminus \{0\}$, $\Delta_r \in \mathbb{F}_2^r$, *are impossible.*

3 Key-Alternating Block Cipher

3.1 Linear Correlations

Let $\mathcal{E}_K : \mathbb{F}_2^n \to \mathbb{F}_2^n$ be a key-alternating block cipher, parametrised by a master key K, and comprising r' applications of the round function R_k, parametrised by the round key k. Let $(k_0, k_1, k_2, \cdots, k_{r'})$ be the round keys derived from the master key K. Without loss of generality, we assume that the key addition is the last component of the round function, that is, $R_{k_i}(x) = R(x) \oplus k_i$, for all $i = 1, \ldots, r'$. Then the block cipher E_K is defined as follows

$$\mathcal{E}_K(x) = R_{k_{r'}} \circ \cdots R_{k_2} \circ R_{k_1}(x \oplus k_0).$$

In the context of last rounds attacks, let us denote by F_K the first r rounds of the cipher. Then

$$\mathcal{E}_K(x) = R_{k_{r'}} \circ \cdots \circ R_{k_{r+1}} \circ F_K(x \oplus k_0).$$

By guessing (parts of) the keys $k_{r+1}, \ldots, k_{r'}$ the ciphertext can be (partially) decrypted over these rounds to achieve (partial) information about output data of F_K. Success of the attacks depends of many criteria. In the context of statistical attack, an evaluation of a non-uniform behaviour of r rounds of the cipher, allow the attacker to first build a distinguisher that will be used after to mount the attack.

As shown by Daemen [16] the correlation of a linear approximation $(a \cdot x \oplus b \cdot F_K(x))$ can be computed as a sum of key-dependent signed products of correlations of linear approximations that are chained over consecutive rounds. A chain of masks $U = (u_0, u_1 \cdots, u_r) \in (\mathbb{F}_2^n)^{r+1}$, where u_{i-1} and u_i

are the input and output masks over R at round i, is called a linear trail. If $k_0, \cdots, k_r$ are the round keys derived from a fixed master key K, then

$$\begin{aligned}&\mathbf{cor}_x\left(a \cdot x \oplus b \cdot F_{\mathrm{K}}(x)\right)\\ &= \sum_{U; u_0=a; u_r=b} (-1)^{u_0 \cdot k_0 \oplus \cdots \oplus u_r \cdot k_r} \prod_{i=0}^{r-1} \mathbf{cor}_x(u_i \cdot x \oplus u_{i+1} \cdot R(x)).\end{aligned} \tag{2}$$

Success and data complexity estimates in differential cryptanalysis are based on the average differential probabilities taken over all possible keys. We obtain a formula for this quantity by application of (1) for $F = F_K$, and then taking the average of both sides over K. It remains to compute the averages of the squared correlations. Next we recall the frequently used estimate of average squared correlations. This general form is obtained directly from Formula (2) by squaring both sides and taking the average over the round keys, or alternatively, by application of Theorem 2 by setting $y = K$ and $F(x, K) = F_K(x)$. By $\mathbf{E}_x(F(x))$ we denote the average value of F taken over x.

Theorem 4. *Using the notation given in this section and assuming that the round keys $k_0, \ldots, k_r$ are independent and uniformly distributed, we have*

$$\mathbf{E}_{k_0,\ldots,k_r}\left[\mathbf{cor}_x^2(a \cdot x \oplus b \cdot F_K(x))\right] = \sum_{\substack{U; u_0=u;\\ u_r=w}} \prod_{i=0}^{r-1} \mathbf{cor}_x^2(u_i \cdot x \oplus u_{i+1} \cdot R(x)). \tag{3}$$

3.2 Algorithm for Computing Average Squared Correlations

Daemen's formula (2) describes a way how to compute correlations round by round using correlation matrices. Similarly, Formula (3) can be implemented as a product of transition matrices corresponding to squared correlations of linear approximations over one round of the cipher.

In practice, as all correlations of one round linear approximations cannot be stored, a selection of the most significant linear approximations must be done, and only the squared correlations of the selected trails should be stored in the transition matrix. For instance, in the case of PRESENT, the single-bit linear trails are dominant, and a sharp estimate of the expected squared correlations of the cipher can be computed based only on these trails [14,23].

Let Ω be a $N \times N$ matrix consisting of the squared correlations of the dominant one round linear approximations. We denote

$$\Omega[i, j] = \mathbf{E}_k\left[\mathbf{cor}_x^2(u_i \cdot x \oplus u_j \cdot R_k(x))\right],$$

where w_i, $i = 1, \ldots, N$, are the selected masks and k is the round key. Then by (3), if z rounds of the cipher with master key K is denoted by $R_{k^z}^{(z)}$, we have

$$\mathbf{E}_{k^z}\left[\mathbf{cor}_x^2(w_i \cdot x \oplus w_j \cdot R_{k^z}^{(z)})\right] \approx \Omega^z[i, j],$$

where Ω^z is the z-th power of the matrix Ω.

As only the masks corresponding to the most dominant approximations can be reached using the transition matrix Ω, rounds at the beginning and at the end should be added to complete the computation of the expected squared correlation for other input and output masks.

4 Multiple Differential Cryptanalysis

In the context of linear cryptanalysis, generalisations using distribution vectors and `LLR` and χ^2 statistical tests were provided first by Baignères, Junod and Vaudenay [3], and more recently, with applications to practical ciphers, by Hermelin, Cho, Nyberg [19]. For differential cryptanalysis, such multidimensional extensions appeared not until 2012 [2,9]. In [2], a framework for such attacks was presented and tested for small block cipher. Cryptanalysis using multiple differentials on real ciphers, however, requires selection of suitable subsets of output differences, or grouping them in an appropriate way. In an attack model called *"unbalanced partitioning"* [9], a subspace of output differences is taken into consideration. In this model, the probability distributions involved ordinary differential probabilities, while the *"balanced partitioning"* model involves probability distributions of truncated differentials. The latter approach allows considering information from the whole output space. Advantages and disadvantages of both partitioning functions are discussed in [9]. In this article, we focus on MDC using balanced partitioning and probability distributions of truncated differentials, for the simple reason that those can be efficiently computed using the method of squared correlations.

4.1 Truncated Differentials

Let $F_K : \mathbb{F}_2^n \to \mathbb{F}_2^n$ be, as before, r rounds of the block cipher. We aim at computing the probability distribution of truncated differentials, where the input difference δ is fixed, and the output differences are truncated and vary over all possible values. More concretely, let Δ be an output difference in a vector space $V \subset \mathbb{F}_2^n$. Let $\bar{V}$ be a complementary subspace of V, that is $\bar{V} \oplus V = \mathbb{F}_2^n$. Then $S_\Delta = \Delta \oplus \bar{V}$ is a truncated output difference and $\cup_{\Delta \in V} S_\Delta = \mathbb{F}_2^n$.

The probability of the truncated differential (δ, S_Δ) is defined[1]

$$\mathbf{P}\left[\delta \overset{F}{\to} S_\Delta\right] = \sum_{\gamma \in S_\Delta} \mathbf{P}\left[\delta \overset{F}{\to} \gamma\right] = \mathbf{P}\left[\delta \overset{G}{\to} \Delta\right], \tag{4}$$

where $G_K = \pi \circ F_K$ and π is a projection from $\mathbb{F}_2^n$ to V.

In what follows in this paper, we assume that δ is fixed and Δ takes all possible values in V. We study the non-uniformity of the distribution vector $p = [\mathbf{P}(\delta \overset{G}{\to} v)]_{v \in V}$, and denote $p_v = \mathbf{P}(\delta \overset{G}{\to} v)$, for $v \in V$. Then

$$p_v = \frac{1}{|V|} \cdot \sum_{a \in \mathbb{F}_2^n} \sum_{b \in V} (-1)^{a \cdot \delta \oplus b \cdot v} \mathbf{E}_K \left(\mathbf{cor}_x^2 \left(a \cdot x \oplus b \cdot G_K(x)\right)\right). \tag{5}$$

[1] A more general definition is given in section 2.2.

Using the optimisations given in Lemma 1 and Section 3.2 we can efficiently compute estimates of the expected values of the squared correlations $\mathbf{E}_K(\mathbf{cor}_x^2(a \cdot x \oplus b \cdot G_K(x))$, for all $a \in \mathbb{F}_2^n$ and $b \in V$. In all our experiments we compute correlations over two rounds "by hand" without the transition matrix at the beginning and at the end, to obtain the following formula

$$\frac{1}{|T|} \sum_{x_t \in T} \sum_{i,j=1}^{N} \mathbf{cor}_{x_s}^2(a_s \cdot x_s \oplus w_i \cdot R^2(x_s, x_t)) \Omega^{r-4}[i,j] \mathbf{cor}_x^2(w_j \cdot x \oplus b \cdot \pi(R^2(x)), \quad (6)$$

where in the computation of the first correlation $x = (x_s, x_t) \in \mathbb{F}_2^s \times \mathbb{F}_2^t$ and $a_s \in \mathbb{F}_2^s$. Sometimes, depending of the cipher more that two rounds of correlations can be used before going to selected correlations represented by the matrix Ω.

4.2 LLR Statistical Test and Data Complexity

We adopt the classical model of statistical cryptanalysis and assume that the *Wrong-Key Randomisation Hypothesis* holds. It means that for a wrong key guess the corresponding distribution is assumed to be uniform. We will denote the uniform distribution vector by $\theta = [\theta_v]_{v \in V}$ where each $\theta_v = \frac{1}{|V|}$.

When evaluating the security of the cipher or the complexity of a statistical distinguisher, accurate estimates of the differential probabilities are important. In [9], the authors studied the complexities of MDC for the LLR and the χ^2 distinguishers. When a good estimate of the expected probabilities is available, then the LLR distinguisher provides better data and memory complexities than the one based using χ^2 statistics. Nevertheless, it is well known that a small deviation in the estimation of the expected probability distribution will not allow the construction of a distinguisher using the LLR test. We demonstrate the accuracy of the estimates computed using Equation (6) by performing simulated attacks using the LLR distinguisher and by comparing the theoretical and the observed data complexity. Next we recall results from [3,9] concerning the complexity of an attack using the LLR statistical test.

Definition 1. *Let $p = [p_v]_{v \in V}$ be the expected probability distribution vector, θ the uniform one and q^k the observed one for a key candidate k. For a given number of sample N_S, the optimal statistical test consists in comparing the following statistic to a fixed threshold.*

$$\mathtt{LLR}(q^k, p, \theta) \overset{\text{def}}{=} N_S \sum_{v \in V} q_v \log\left(\frac{p_v}{\theta_v}\right).$$

Definition 2. *Let p and p' be two probability distribution vectors over V. The* relative entropy *(aka.* Kullback-Leibler divergence*) between p and p' is*

$$D(p||p') \overset{\text{def}}{=} \sum_{v \in V} p_v \log\left(\frac{p_v}{p'_v}\right).$$

We also define the following metrics

$$D_2\left(p||p'\right) \overset{\text{def}}{=} \sum_{v \in V} p_v \log^2\left(\frac{p_v}{p'_v}\right), \quad \textit{and} \quad \varDelta D\left(p||p'\right) \overset{\text{def}}{=} D_2\left(p||p'\right) - D\left(p||p'\right)^2.$$

Theorem 5. *Let a be the advantage (see [31]) of an attack then the data complexity required to reach success probability P_S is*

$$N = 2 \cdot \frac{\left[\sqrt{\varDelta D\left(p||\theta\right)}\,\varPhi_{0,1}^{-1}(P_S) + \sqrt{\varDelta D\left(\theta||p\right)}\,\varPhi_{0,1}^{-1}\left(1 - 2^{-a}\right)\right]^2}{\left[D\left(p||\theta\right) + D\left(\theta||p\right)\right]^2}, \tag{7}$$

where $\varPhi_{0,1}$ is the cumulative function of the standard normal distribution.

5 Practical Applications

Computation of the truncated differential probabilities using (6) depends on the ciphers. To compose the transition matrix, cryptanalyst must identify the important linear trails of the cipher. We consider this problem for two SPN block ciphers PRESENT[5] and PUFFIN[13].

5.1 Description of the Ciphers

The block cipher PRESENT is designed as a lightweight primitive which operates on 64-bit blocks of data. Ciphertexts are obtained after 31 iterations of the round function. The 16 Sboxes of PRESENT are all identical and are defined as a 4-bit non-linear permutation. PRESENT is parametrised by a 80-bit or a 128-bit key. More details on the specification can be found in [5].

The lightweight block cipher PUFFIN was introduced in [13]. It is defined as a 64-bit SPN block cipher parametrised with a 128-bit key. The round function as described in [13] is applied 32 times[2]. The structure of this cipher is similar to the one of PRESENT. By choosing involution components, the designers aim at efficient implementation in hardware.
Even if PUFFIN might not be of general interest, we selected it as a reference cipher for our experiments on PRESENT. As the Sboxes and the linear diffusion of these ciphers are essentially different, the linear and differential attacks have different impact on these ciphers. For PRESENT, linear cryptanalysis is more powerful (26 rounds [14]) than differential cryptanalysis (18 rounds [8,34]). For PUFFIN, the best linear and differential types of attacks are about equally strong [10,23]. The observed differences are largely due to the fact that PRESENT has the particularity of having strong single-bit linear relations over the S-box. The Sbox of PUFFIN is built in such a way that differences of Hamming weight one have high probabilities but the single-bit linear relations are not among the strongest. In our experiments, the transition matrix for PRESENT is composed

[2] Later, the same authors propose a new version of this cipher called PUFFIN2 [32].

of correlations for single-bit masks, while for PUFFIN we use a matrix consisting of all single-bit and two-bit linear approximations of the Sbox. By this choice we also aim at showing that the estimation method (6) works also in case when the single-bit linear trails are not dominant.

5.2 Parameters in Practice

In the context of differential cryptanalysis the attacker builds a distinguisher over all but a small number of the last rounds of the cipher and wants to recover information on the subkeys used at these last rounds. As partial decryption (inversion of some Sboxes) over the last rounds is time consuming, the ratio between the number of guessed keys and the number of Sbox inversions is often maximised. With this aim in mind, it is reasonable to choose a projected output space V which corresponds to a group of active Sboxes in the following round. Since PRESENT and PUFFIN use 4-bit Sboxes, we conduct experiments with $|V| = 2^4, 2^8, 2^{12}, 2^{16}$.

The MDC attack described in Section 4 takes into consideration all ciphertexts in the computation of the observed probability distributions. Contrary to classical differential cryptanalysis, there is no sieving, which means that the time complexity of the attack is always larger than the data complexity [9]. For instance, for an SPN cipher, where only part of the last round key is guessed during the attack, the time complexity is of the order of $|V| \times N$, where $|V|$ is the size of the projected output difference space and N the data complexity as derived in Theorem 5. As stated in [9], the memory complexity of multiple differential attacks using the LLR statistical test is dominated by the storage of the expected distribution p and the storage of an array of counters for recording the observed frequencies. When only the last round subkey is guessed, the memory complexity is then the storage of $2 \times |V|$ counters.

5.3 Time Complexity of Computation of Differential Probability

In practice, difficulty of the computation of the truncated differential probabilities using square correlations depends of the structure of the cipher and of the number of square correlations to compute. Formula (6) shows that for a fixed truncated differential, this computation can be decomposed into three steps consisting of computing the correlations over the first rounds, the intermediate rounds and the last rounds. In the case of PRESENT and PUFFIN an efficient computation of r-round squared correlation can be done using transitional matrices on $r-4$ rounds and by adding two rounds at the beginning and the end. For other ciphers than PRESENT, larger transition matrices should be taken into consideration. In the case of PUFFIN, computation of the powers of this matrix remain easy and fast using the two-bit linear trails. Using Lemma 1, computation for the first two rounds is done by computing the squared correlation over $x_s \in \mathbb{F}_2^s$ for a certain small number of restrictions specified by a set T of randomly selected x_t. Experiments show that the distribution of output differences is less uniform if the fixed input difference δ is selected such that only one Sbox is active. Hence we can choose $x_s \in \mathbb{F}_2^4$. Computation over $T = 2^8$ random x_t has been seen to be enough for the ciphers studied in this paper. For the last two rounds, an average over 2^{20} random x, gives also a good estimate of the squared correlations $\mathbf{cor}_x(w_j \cdot x \oplus b \cdot \pi(R^2(x))$.

If we store values used many times, the time complexity of the computation of a truncated differential is dominated by the computation of the squared correlations over the first and and last rounds. It corresponds to a small number of encryptions. Using squared correlations and the transition matrix, computation of the expected differential probabilities can then be considered as independent of the number of rounds. In comparison, the complexity of the branch-and-bound algorithm is exponential in the number of rounds[8]. Hence, the computation of the truncated probabilities depends only on the size of V. For $V = 2^4$ this computation takes less than one minute on a standard computer.

In Section 4, we motivated why to use truncated differential probabilities in MDC. Truncated differential probabilities should be computed for all $v \in V$. As Formula (5) can be decomposed as

$$p_v = \frac{1}{|V|} \sum_{b \in V} (-1)^{b \cdot v} \cdot \sum_{a \in \mathbb{F}_2^n} (-1)^{a \cdot \delta} \mathbf{E}_K \left(\mathbf{cor}_x^2 \left(a \cdot x \oplus b \cdot G_K(x) \right) \right),$$

computation can be done efficiently by storing first the estimates of the sum over the input mask a computed using Lemma 1, for all $b \in V$. Then all p_v, $v \in V$, can be computed simultaneously using Fast Fourier Transform with time complexity $|V| \log |V|$.

6 Experiments and Attacks

6.1 Experiments

In this section we describe the experiments done with PRESENT and PUFFIN. We build an LLR distinguisher using the computed estimates of theoretical probability distributions over r rounds to attack $r+1$ rounds of the cipher. These experiments have been conducted in the following order: computation of the square correlation using transitional matrices, simulation of 100 multiple differential attacks using the `LLR` statistical test, and comparison between experimental data complexity and the theoretical one given by Theorem 5.

We conducted experiments on the ciphers PRESENT and PUFFIN with different numbers of rounds, different input differences, and different projected output spaces V of different sizes. Figure 1 illustrates the accuracy of the theoretical estimates in the case of the PRESENT cipher, for different sizes of V and different numbers of rounds. In this figure, we compare the differences in data complexity between the theoretical formula of Theorem 5 and the data requirements obtained using a mean over 100 simulated attacks. For the experiments presented in this figure, we selected $\delta =$ `0xf00000`. The advantage a is equal to 4 for $|V| \geq 2^8$, and equal to 2 for $|V| = 2^4$. The numbering of Sboxes corresponds to the one given in the specification [5]. Results of experiments on PUFFIN are

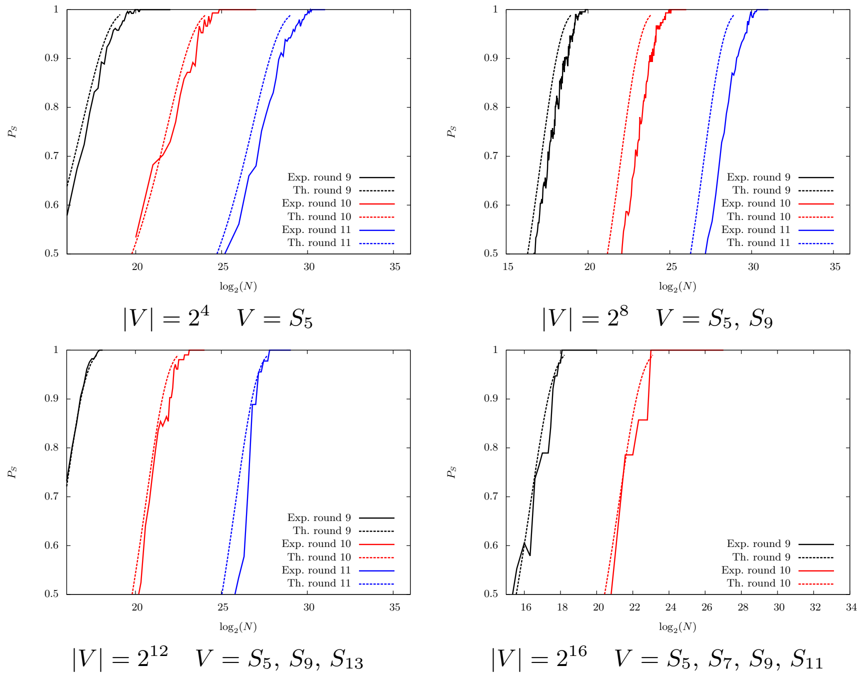


Fig. 1. Data complexity of attacks on 9, 10, 11 rounds of PRESENT

given in Appendix A. The obtained results from our simulations of MDC attacks are two-fold. First, it is well known that the `LLR` statistical test is efficient only if the analyst can provide a good estimate of the theoretical distribution. As the results of these experiments presented in Figure 1, 2 are tight, we can conclude that we were able to provide sufficiently accurate estimates of the differential probabilities using Formula (6). Secondly, we show for the first time an instantiation of an MDC attack on a full block size version of a state-of-the-art cipher.

6.2 Attacks on PRESENT

In the case of PRESENT with 80-bit key, the time complexity is bounded from above by 2^{80}. If the attack needs the full codebook then the size of the probability distribution must be less than 2^{16}. Different parameters are possible for the attack. As example, we propose an attack over 18 rounds using MDC distinguisher over 17 rounds. Parameters of this attack, with input difference $\delta =$ `0xf00000` and projected output difference space concentrated on Sboxes $S5$, $S9$ and $S13$, correspond to the ones used in attacks on a reduced-round version of the cipher (cf. Figure 1 with $|V| = 2^{12}$). Using the full codebook, this attack recovers 6-bits of the key with a success probability of 85% and has a time complexity of

Table 1. Parameters of attacks on PRESENT

#rounds	Key Length	Data Comp.	Adv.	Success Prob.	Time	Memory
18	80	2^{64}	6 bits	85%	$2^{76}+2^{74}$	2^{13}
18	80	2^{62}	2 bits	85%	$2^{74}+2^{78}$	2^{13}
19	128	2^{64}	6 bits	85%	$2^{124}+2^{122}$	2^{60}
19	128	2^{62}	2 bits	85%	$2^{122}+2^{126}$	2^{60}

Table 2. Summary of the attacks on PRESENT

#rounds	Version	Type of attack	Data	Time	Memory	Reference
16	80	Differential	$2^{64.0}$	$2^{64.0}$	$2^{32.0}$	[33]
18	80	Multiple Differential	$2^{64.0}$	$2^{64.0}$	$2^{32.0}$	[8]
18	80	Multiple Differential	$2^{64.0}$	$2^{64.0}$	$2^{32.0}$	[34]
18	**80**	**Multiple Differential (LLR)**	2^{62}	2^{78}	2^{13}	**This paper**
19	128	Algebraic Differential	$2^{62.0}$	$2^{113.0}$	n/r	[1]
19	**128**	**Multiple Differential (LLR)**	2^{62}	2^{126}	2^{60}	**This paper**
24	80	Linear	$2^{63.5}$	$2^{40.0}$	$2^{40.0}$	[30]
24	80	Statistical Saturation	$2^{57.0}$	$2^{57.0}$	$2^{32.0}$	[15]
25	128	Linear	$2^{64.0}$	$2^{96.7}$	$2^{40.0}$	[29]
26	80	Multiple Linear	$2^{64.0}$	$2^{72.0}$	$2^{32.0}$	[14]

2^{76} on which we add the exhaustive search of the 2^{74} remaining keys. Memory complexity of this attack, is defined by the storage of the expected distribution vector and the storage of the counter array, and is equal to 2^{13} counters.

For the 128-bit key, partial inversion of the last two rounds is possible. Therefore using the same distinguisher over 17 rounds, we can propose an attack on 19 rounds. The choices of parameters for this attack are resumed in Table 1.

We conclude that the MDC distinguisher using truncated differentials described in this paper is the best distinguisher on PRESENT in the context of differential cryptanalysis. On the other hand, the key recovery attacks presented in this paper do not significantly improve over the previous differential attack on this cipher (18 rounds for both the 80-bit key and the 128-bit key). Best attacks on PRESENT are summarised in Table 2. We present a comparison between the attacks in this paper and the ones in [8,34] which are based on simple differentials. Output differences of these simple differential focus on a small number of Sboxes. In this case, a sieving process can be applied, for both key schedules, and therefore one can invert two rounds of the cipher. Thus, using a 16-round distinguisher, 18 rounds can be attacked. In this paper, distribution of output differences over the the whole output space is taken into consideration. As no sieve is applied before guessing the key, the time complexity is larger and permits to invert only one round, for the 80-bit key. This explains why using the 17-round distinguisher, we are able to attack only 18 rounds of PRESENT-80 and 19 rounds for PRESENT-128.

Overall, the multiple differential attack presented in this paper corresponds quite well to the known differential properties of the PRESENT cipher. On the other hand, our simulations show that for PUFFIN truncated differentials do not provide better attacks than simple differential distribution.

7 Conclusion

Relations and dependencies between statistical attacks are of great importance when analysing the security of primitives based on block ciphers. In this paper, we extracted new relations between multiple differential and multidimensional linear distinguishers, and subsequently, between zero-correlation and impossible differential distinguishers. We used, for the first time, the relation between correlation of linear approximation and differential probability in practice to compute estimates of truncated differential probabilities of state-of-the-art ciphers from squared correlations of a selected set of linear approximations. We also derived a method to reduce the number of correlations needed to be computed, and in this manner, succeeded to speed up the computation of these correlations to make the computation possible on a standard computer. Time complexity of this method is immune to the number of rounds, while for branch-and-bound algorithm it increases exponentially with the number of rounds.

The method developed in this paper was tested experimentally on the block ciphers PRESENT and PUFFIN and was further developed to a multiple differential attack on PRESENT which improves the best known attack in the differential context.

An interesting topic left for further research is to instantiate Theorem 3 on some ciphers and investigate it more closely. In this theorem, the truncated differentials and the multidimensional linear approximations occupy disjoint parts of the cipher, while in the method described in this paper the truncated differentials are located in the areas covered by known strong linear approximations. Therefore it may lead to essentially different results.

Acknowledgments. We wish to thank Gregor Leander for useful discussions that led us to discovery of the results presented in Section 2.2. We also wish to thank the anonymous reviewers for helpful comments.

References

1. Albrecht, M., Cid, C.: Algebraic techniques in differential cryptanalysis. In: Dunkelman, O. (ed.) FSE 2009. LNCS, vol. 5665, pp. 193–208. Springer, Heidelberg (2009)
2. Albrecht, M.R., Leander, G.: An All-In-One Approach to Differential Cryptanalysis for Small Block Ciphers. In: Knudsen, L.R., Wu, H. (eds.) SAC 2012. LNCS, vol. 7707, pp. 1–15. Springer, Heidelberg (2013)
3. Baignères, T., Junod, P., Vaudenay, S.: How far can we go beyond linear cryptanalysis? In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 432–450. Springer, Heidelberg (2004)
4. Biham, E., Shamir, A.: Differential cryptanalysis of DES-like cryptosystems. In: Menezes, A., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 2–21. Springer, Heidelberg (1991)

5. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsoe, C.: PRESENT: An ultra-lightweight block cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 450–466. Springer, Heidelberg (2007)
6. Bogdanov, A., Leander, G., Nyberg, K., Wang, M.: Integral and Multidimensional Linear Distinguishers with Correlation Zero. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 244–261. Springer, Heidelberg (2012)
7. Blondeau, C., Gérard, B.: Links between theoretical and effective differential probabilities: Experiments on PRESENT. In: TOOLS 2010 (2010)
8. Blondeau, C., Gérard, B.: Multiple Differential Cryptanalysis: Theory and Practice. In: Joux, A. (ed.) FSE 2011. LNCS, vol. 6733, pp. 35–54. Springer, Heidelberg (2011)
9. Blondeau, C., Gérard, B., Nyberg, K.: Multiple Differential Cryptanalysis Using LLR and χ^2 Statistics. In: Visconti, I., De Prisco, R. (eds.) SCN 2012. LNCS, vol. 7485, pp. 343–360. Springer, Heidelberg (2012)
10. Blondeau, C., Gérard, B.: Differential Cryptanalysis of PUFFIN and PUFFIN2. In: ECRYPT Workshop on Lightweight Cryptography - LC 2011 (2011)
11. Carlet, C.: Boolean Functions for Cryptography and Error Correcting Codes. Cambridge University Press (to appear)
12. Chabaud, F., Vaudenay, S.: Links between Differential and Linear Cryptanalysis. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 356–365. Springer, Heidelberg (1995)
13. Cheng, H., Heys, M., Wang, C.: PUFFIN: A Novel Compact Block Cipher Targeted to Embedded. In: Fanucci, L. (ed.) DSD 2008, pp. 383–390. IEEE (2008)
14. Cho, J.Y.: Linear cryptanalysis of reduced-round PRESENT. In: Pieprzyk, J. (ed.) CT-RSA 2010. LNCS, vol. 5985, pp. 302–317. Springer, Heidelberg (2010)
15. Collard, B., Standaert, F.-X.: A statistical saturation attack against the block cipher PRESENT. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 195–210. Springer, Heidelberg (2009)
16. Daemen, J., Govaerts, R., Vandewalle, J.: Correlation matrices. In: Preneel, B. (ed.) FSE 1994. LNCS, vol. 1008, pp. 275–285. Springer, Heidelberg (1995)
17. Daemen, J., Knudsen, L.R., Rijmen, V.: The Block Cipher SQUARE. In: Biham, E. (ed.) FSE 1997. LNCS, vol. 1267, pp. 149–165. Springer, Heidelberg (1997)
18. Gilbert, H.: An untwisted representation of AES. Early Symmetric Crypto Seminar, Mondorf-les-Bains, Luxemburg (January 2013)
19. Cho, J.Y., Hermelin, M., Nyberg, K.: A new technique for multidimensional linear cryptanalysis with applications on reduced round serpent. In: Lee, P.J., Cheon, J.H. (eds.) ICISC 2008. LNCS, vol. 5461, pp. 383–398. Springer, Heidelberg (2009)
20. Hermelin, M., Cho, J.Y., Nyberg, K.: Multidimensional extension of Matsui's algorithm 2. In: Dunkelman, O. (ed.) FSE 2009. LNCS, vol. 5665, pp. 209–227. Springer, Heidelberg (2009)
21. Knudsen, L.R.: Truncated and Higher Order Differentials. In: Preneel, B. (ed.) FSE 1994. LNCS, vol. 1008, pp. 196–211. Springer, Heidelberg (1995)
22. Knudsen, L.R., Rijmen, V.: Known-key distinguishers for some block ciphers. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 315–324. Springer, Heidelberg (2007)
23. Leander, G.: On Linear Hulls, Statistical Saturation Attacks, PRESENT and a Cryptanalysis of PUFFIN. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 303–322. Springer, Heidelberg (2011)
24. Linial, N., Mansour, Y., Nisan, N.: Constant Depth Circuits, Fourier Transform, and Learnability. Journal of the Association for Computing Machinery 40(3), 607–620 (1993)

25. Matsui, M.: Linear cryptanalysis method for DES cipher. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 386–397. Springer, Heidelberg (1994)
26. Matsui, M.: On Correlation Between the Order of S-boxes and the Strength of DES. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 366–375. Springer, Heidelberg (1995)
27. Nyberg, K.: Linear approximation of block ciphers. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 439–444. Springer, Heidelberg (1995)
28. Nyberg, K.: S-boxes and round functions with controllable linearity and differential uniformity. In: Preneel, B. (ed.) FSE 1994. LNCS, vol. 1008, pp. 111–130. Springer, Heidelberg (1995)
29. Nakahara Jr., J., Sepehrdad, P., Zhang, B., Wang, M.: Linear (Hull) and algebraic cryptanalysis of the block cipher PRESENT. In: Garay, J.A., Miyaji, A., Otsuka, A. (eds.) CANS 2009. LNCS, vol. 5888, pp. 58–75. Springer, Heidelberg (2009)
30. Ohkuma, K.: Weak keys of reduced-round PRESENT for linear cryptanalysis. In: Jacobson Jr., M.J., Rijmen, V., Safavi-Naini, R. (eds.) SAC 2009. LNCS, vol. 5867, pp. 249–265. Springer, Heidelberg (2009)
31. Selçuk, A.A.: On Probability of Success in Linear and Differential Cryptanalysis. Journal of Cryptology 21(1), 131–147 (2008)
32. Wang, C., Heys, H.M.: An ultra compact block cipher for serialized architecture implementations. In: Fanucci, L. (ed.) CCECE 2009, pp. 1085–1090. IEEE (2009)
33. Wang, M.: Differential cryptanalysis of reduced-round PRESENT. In: Vaudenay, S. (ed.) AFRICACRYPT 2008. LNCS, vol. 5023, pp. 40–49. Springer, Heidelberg (2008)
34. Wang, M., Sun, Y., Tischhauser, E., Preneel, B.: A Model for Structure Attacks, with Applications to PRESENT and Serpent. In: Canteaut, A. (ed.) FSE 2012. LNCS, vol. 7549, pp. 49–68. Springer, Heidelberg (2012)

A Appendices

Figure 2 presents results of some experiments on a reduced-round version of PUFFIN. Distribution over r rounds was computed using Formula (6) to simulate multiple differential attack on $r + 1$ rounds. Different experiments have been conducted. In this figure we illustrate the results for the input difference $\delta = \texttt{0x2}$ and output projected space V concentrated on $S3$ and $S5$ (for $|V| = 2^8$) and $S3$, $S5$, $S9$ (for $|V| = 2^{12}$).

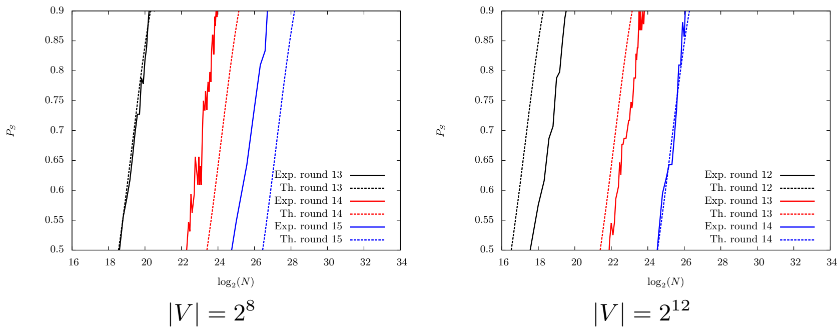


Fig. 2. Data complexity of attacks on 12, 13, 14 rounds of PUFFIN

Towards Key-Length Extension with Optimal Security: Cascade Encryption and Xor-cascade Encryption

Jooyoung Lee[⋆]

Faculty of Mathematics and Statistics
Sejong University, Seoul, Korea 143-747
jlee05@sejong.ac.kr

Abstract. This paper discusses provable security of two types of cascade encryptions. The first construction CE^l, called l-cascade encryption, is obtained by sequentially composing l blockcipher calls with independent keys. The security of CE^l has been a longstanding open problem until Gaži and Maurer [9] proved its security up to $2^{\kappa+\min\{\frac{n}{2},\kappa\}}$ query complexity for large cascading length, where κ and n denote the key size and the block size of the underlying blockcipher, respectively. We improve this limit by proving the security of CE^l up to $2^{\kappa+\min\{\kappa,n\}-\frac{16}{l}\left(\frac{n}{2}+2\right)}$ query complexity: this bound approaches $2^{\kappa+\min\{\kappa,n\}}$ with increasing cascade length l.

The second construction XCE^l is a natural cascade version of the DESX scheme with intermediate keys xored between blockcipher calls. This can also be viewed as an extension of double XOR-cascade proposed by Gaži and Tessaro [10]. We prove that XCE^l is secure up to $2^{\kappa+n-\frac{8}{l}\left(\frac{n}{2}+2\right)}$ query complexity. As cascade length l increases, this bound approaches $2^{\kappa+n}$.

In the ideal cipher model, one can obtain all the evaluations of the underlying blockcipher by making $2^{\kappa+n}$ queries, so the $(\kappa+n)$-bit security becomes the maximum that key-length extension based on a single κ-bit key n-bit blockcipher is able to achieve. Cascade encryptions CE^l (with $n \leq \kappa$) and XCE^l provide almost optimal security with large cascade length.

1 Introduction

The key length of a blockcipher, say κ, is a crucial factor that limits its achievable security level: no matter how carefully designed, one can recover its secret key simply by trying all possible 2^κ keys. For example, the Data Encryption Standard (DES) [1] using 56-bit keys was one of the most predominant algorithms for encryption of data. No feasible attacks faster than a brute-force attack have

[⋆] The work of J. Lee was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology(2012-0003157).

been proposed (as most of them require a huge amount of data), while advances in computational power made a brute-force attack itself practical. As a result, DES was replaced by a new standard algorithm AES [4]. On the other hand, in order to protect legacy applications based on DES, there have been considerable research on constructing DES-based encryption schemes which employ longer keys. This approach is called *key-length extension*, for which Triple-DES [2,3,5] and DESX (due to Rivest) are the most popular constructions.

The Triple-DES approach transforms a κ-bit key n-bit blockcipher E into an encryption scheme that accepts three κ-bit keys $k_1, k_2, k_3 \in \{0,1\}^\kappa$ and encrypts an n-bit message block u as $v = E_{k_3}(E_{k_2}(E_{k_1}(u)))$. Bellare and Rogaway [6] proved its security up to $2^{\kappa+\frac{\min\{n,\kappa\}}{2}}$ query complexity assuming E is an ideal blockcipher. The triple encryption can be naturally extended to sequentially composing more than three blockcipher calls with independent keys. It has been a longstanding open problem if the security of cascade encryption improves with increasing cascade length. Recently, Gaži and Maurer [9] partially answered this question by showing the security bound (in terms of the threshold number of queries) approaches roughly the value $2^{\kappa+\min\{\frac{n}{2},\kappa\}}$. In this paper, we will revisit this question.

The DESX approach transforms a κ-bit key n-bit blockcipher E into an encryption scheme that accepts a κ-bit key $k \in \{0,1\}^\kappa$ and additional n-bit whitening keys $k_i, k_o \in \{0,1\}^n$ and encrypts an n-bit message block u as $v = k_o \oplus E_k(k_i \oplus u)$. Killan and Rogaway [13] proved its security up to $2^{\frac{\kappa+n}{2}}$ query complexity. As an efficient key-length extension, Gaži and Tessaro [10] proposed a cascade of two DESX schemes with some refinement, and proved its security up to $2^{\kappa+\frac{n}{2}}$ query complexity.

Our Contribution. Since one can obtain all the evaluations of a κ-bit key n-bit blockcipher by making $2^{\kappa+n}$ queries, the $(\kappa+n)$-bit security becomes the maximum that key-length extension based on a single κ-bit key n-bit blockcipher is able to achieve: a standard brute-force attack of $2^{\kappa+n}$ query complexity is given in Appendix A.

Therefore it is natural to ask if there is key-length extension with the optimal $(\kappa+n)$-bit security. In order to answer this question, we consider two types of cascade encryptions. The first construction is a regular cascade encryption. Formally, *l-cascade encryption* CE^l accepts an $l\kappa$-bit key $\mathbf{k} = (k_1, \ldots, k_l) \in (\{0,1\}^\kappa)^l$ and encrypts a plaintext $u \in \{0,1\}^n$ by computing

$$v = \mathsf{CE}^l_{\mathbf{k}}[E](u) = E_{k_l} \circ E_{k_{l-1}} \circ \cdots \circ E_{k_2} \circ E_{k_1}(u).$$

In this paper, we prove that CE^l is pseudorandom up to $2^{\kappa+\min\{\kappa,n\}-\frac{16}{l}\left(\frac{n}{2}+2\right)}$ query complexity (ignoring log factor). As cascade length l increases, this bound approaches $2^{\kappa+\min\{\kappa,n\}}$, improving the limit $2^{\kappa+\min\{\frac{n}{2},\kappa\}}$ given by Gaži and Maurer when $\frac{n}{2} < \kappa$.

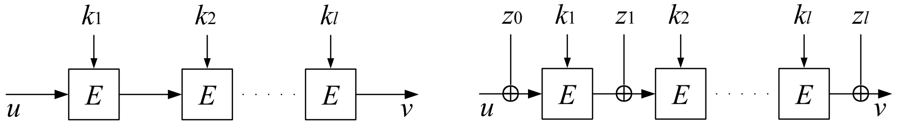
The second construction can be viewed as a cascade of DESX: *l-xor-cascade encryption* XCE^l accepts an $(l\kappa+(l+1)n)$-bit key $(\mathbf{k}, \mathbf{z}) \in (\{0,1\}^\kappa)^l \times (\{0,1\}^n)^{l+1}$ and for

$$\mathbf{k} = (k_1, \ldots, k_l) \in (\{0,1\}^\kappa)^l \text{ and } \mathbf{z} = (z_0, \ldots, z_l) \in (\{0,1\}^n)^{l+1},$$

encrypts a plaintext $u \in \{0,1\}^n$ by computing

$$v = \mathsf{XCE}^l_{\mathbf{k},\mathbf{z}}[E](u) = \oplus_{z_l} \circ E_{k_l} \circ \oplus_{z_{l-1}} \circ \cdots \circ \oplus_{z_1} \circ E_{k_1} \circ \oplus_{z_0}(u),$$

where for $z \in \{0,1\}^n$, $\oplus_z$ denotes the mapping $x \mapsto x \oplus z$ from $\{0,1\}^n$ to itself. We prove the security of XCE^l up to $2^{\kappa+n-\frac{8}{l}(\frac{n}{2}+2)}$ query complexity. With increasing cascade length, this bound approaches $2^{\kappa+n}$. So XCE^l asymptotically provides optimal security with large cascade length, and this observation also applies to cascade encryption CE^l if $n \leq \kappa$ (as in the case of DES and AES). See Figure 1 for pictorial representation of CE^l and XCE^l.



(a) l-cascade encryption CE^l (b) l-xor-cascade encryption XCE^l

Fig. 1. Two types of cascade encryptions

Proof Techniques. We will use a combinatorial framework that lifts the NCPA-security of $l/2$-cascade construction to the CPA security of l-cascade construction. Maurer, Pietrzak, and Renner [15] proved that if two independent encryption schemes F and G are NCPA-secure, then $F \circ G^{-1}$ is CPA-secure. Combinatorial interpretation of this property, based on Lemma 2, was first introduced in [14], where the key-alternating cipher of t rounds is viewed as a composition of two independent key-alternating cipher of $t/2$ rounds, and the NCPA-security of each component is analyzed. A similar approach can be applied to our constructions, while a difficulty comes from the fact that the two components are commonly based on a single blockcipher. We address this problem by using *random key space separation*: randomly partition a key space into two subspaces of the same size and make the first $l/2$ blockcipher calls use keys from one key subspace and the last $l/2$ calls from the other. The modified key sampling process is shown to be indistinguishable from the original one, while by having the two components use their keys from separate key subspaces, we can view a cascade encryption as a composition of two independent ones.

The NCPA-security of each component is proved by coupling technique. Since first introduced by Mironov [16] in a cryptographic context and recently revisited by Morris, Rogaway and Stegers [17] to analyze the security of maximally unbalanced Feistel networks, it became a powerful tool used for the security proof of various types of iterated constructions including generalized Feistel networks, shuffling-based encryption schemes and key-alternating ciphers [14,11,12]. Careful definition and analysis of a coupling, given in the proof of Lemma 5, is the core of our security proof.

Other Related Work. Recently, Gaži [8] presented a distinguishing attack on cascade encryption of odd (resp. even) length l using roughly $2^{\kappa+\frac{l-1}{l+1}n}$ (resp. $2^{\kappa+\frac{l-2}{l}n}$) queries. For xor-cascade encryption of length l (and its generalization), a distinguishing attack of $2^{\kappa+\frac{l-1}{l}n}$ query complexity is presented. In the random system framework, the security of xor-cascade encryption of odd (resp. even) length l is proved up to $2^{\kappa+\frac{l-1}{l+1}n}$ (resp. $2^{\kappa+\frac{l-2}{l}n}$) query complexity, and especially up to $2^{\kappa+\frac{l-1}{l}n}$ query complexity for $l \in \{3,4\}$. These lower bounds are tighter than ours.

2 Preliminaries

2.1 General Notation

For an integer $n \geq 1$, let $I_n = \{0,1\}^n$ be the set of binary strings of length n. The set of all permutations on I_n will be denoted $\mathcal{P}_n$. We will usually write $N = 2^n$.

For a set T and an integer $s \geq 1$, T^{*s} denotes the set of all sequences that consists of s pairwise distinct elements of T. For integers $1 \leq s \leq t$, we will write $(t)_s = t(t-1)\cdots(t-s+1)$. If $|T| = t$, then $(t)_s$ becomes the size of T^{*s}.

2.2 The Ideal Cipher Model

A blockcipher is a function family $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ such that for all $k \in \mathcal{K}$ the mapping $E(k,\cdot)$ is a permutation on I_n. We write $BC(\mathcal{K},n)$ to mean the set of all such blockciphers, shortening to $BC(\kappa,n)$ when $\mathcal{K} = \{0,1\}^\kappa$. In the ideal cipher model, a blockcipher E is chosen from $BC(\mathcal{K},n)$ uniformly at random. It allows for two types of oracle queries $E(k,x)$ and $E^{-1}(k,y)$ for $x,y \in \{0,1\}^n$ and $k \in \mathcal{K}$.[1] The response to an inverse query $E^{-1}(k,y)$ is $x \in \{0,1\}^n$ such that $E(k,x) = y$.

2.3 Indistinguishability

Let $\mathsf{C} \in \{\mathsf{CE}^l, \mathsf{XCE}^l\}$ be an n-bit encryption scheme that employs λ-bit keys and makes oracle queries to a blockcipher $E \in BC(\kappa,n)$. So each key $\mathbf{k} \in \{0,1\}^\lambda$ and a blockcipher $E \in BC(\kappa,n)$ define a permutation $\mathsf{C}_\mathbf{k}[E]$ on I_n. In the *indistinguishability* framework (in the ideal cipher model), $\mathsf{C}_\mathbf{k}[E]$ uses a random secret key $\mathbf{k}$ and makes oracle queries to an ideal blockcipher E, while a permutation P is chosen uniformly at random from $\mathcal{P}_n$. A distinguisher $\mathcal{A}$ would like to tell apart two worlds $(\mathsf{C}_\mathbf{k}[E], E)$ and (P, E) by adaptively making forward and backward queries to the permutation and the blockcipher. Formally, $\mathcal{A}$'s distinguishing advantage is defined by

$$\begin{aligned}\mathbf{Adv}_{\mathsf{C}}^{\mathsf{PRP}}(\mathcal{A}) = \mathbf{Pr}&\left[P \xleftarrow{\$} \mathcal{P}_n, E \xleftarrow{\$} BC(\kappa,n) : \mathcal{A}[P,E] = 1\right] \\ &- \mathbf{Pr}\left[\mathbf{k} \xleftarrow{\$} \{0,1\}^\lambda, E \xleftarrow{\$} BC(\kappa,n) : \mathcal{A}[\mathsf{C}_\mathbf{k}[E],E] = 1\right].\end{aligned}$$

[1] We interchangeably use both representations $E(k,x)$ and $E_k(x)$, and similarly $E^{-1}(k,y)$ and $E_k^{-1}(y)$.

For q_1, $q_2 > 0$, we define

$$\mathbf{Adv}_{\mathsf{C}}^{\mathsf{PRP}}(q_1, q_2) = \max_{\mathcal{A}} \mathbf{Adv}_{\mathsf{C}}^{\mathsf{PRP}}(\mathcal{A}),$$

where the maximum is taken over all adversaries $\mathcal{A}$ making at most q_1 queries to the outer permutation and at most q_2 queries to the underlying blockcipher.

Combinatorial Framework. We assume that a distinguisher $\mathcal{A}$ making q_1 forward and/or backward queries to the permutation oracle records a query history

$$\mathcal{Q}_1 = (u^i, v^i)_{1 \le i \le q_1},$$

where (u^i, v^i) represents the evaluation obtained by the i-th query to the permutation oracle. So according to the instantiation, it implies either $\mathsf{C}_{\mathbf{k}}[E](u^i) = v^i$ or $P(u^i) = v^i$. By making q_2 queries to the underlying blockcipher E, $\mathcal{A}$ also records the second query history

$$\mathcal{Q}_2 = (x^i, k^i, y^i)_{1 \le i \le q_2},$$

where (x^i, k^i, y^i) represents the evaluation $E(k^i, x^i) = y^i$ obtained by the i-th query to the blockcipher. The pair of the query histories

$$\mathcal{T} = (\mathcal{Q}_1, \mathcal{Q}_2)$$

is called the *transcript* of the attack; it contains all the information that $\mathcal{A}$ has obtained at the end of the attack. In this work, we will only consider information theoretic distinguishers. Therefore we can assume that a distinguisher is deterministic without making any redundant queries, and hence the output of $\mathcal{A}$ can be regarded as a function of $\mathcal{T}$, denoted $\mathcal{A}(\mathcal{T})$ or $\mathcal{A}(\mathcal{Q}_1, \mathcal{Q}_2)$.

If a permutation $\mathsf{C}_{\mathbf{k}}[E]$(resp. P) is consistent with $\mathcal{Q}_1$, i.e., $\mathsf{C}_{\mathbf{k}}[E](u^i) = v^i$(resp. $P(u^i) = v^i$) for every $i = 1, \ldots, q_1$, then we will write $\mathsf{C}_{\mathbf{k}}[E] \vdash \mathcal{Q}_1$(resp. $P \vdash \mathcal{Q}_1$). Similarly, if a blockcipher $E \in BC(\kappa, n)$ is consistent with $\mathcal{Q}_2$ (i.e., $E(k^i, x^i) = y^i$ for $i = 1, \ldots, q_2$), then we will write $E \vdash \mathcal{Q}_2$. Using these notations, we have

$$\mathbf{Adv}_{\mathsf{C}}^{\mathsf{PRP}}(\mathcal{A}) = \sum_{\mathcal{A}(\mathcal{Q}_1,\mathcal{Q}_2)=1} \mathbf{Pr}\left[P \xleftarrow{\$} \mathcal{P}_n, E \xleftarrow{\$} BC(\kappa, n) : P \vdash \mathcal{Q}_1 \wedge E \vdash \mathcal{Q}_2\right]$$
$$- \sum_{\mathcal{A}(\mathcal{Q}_1,\mathcal{Q}_2)=1} \mathbf{Pr}\left[\mathbf{k} \xleftarrow{\$} \{0,1\}^{\lambda}, E \xleftarrow{\$} BC(\kappa, n) : \mathsf{C}_{\mathbf{k}}[E] \vdash \mathcal{Q}_1 \wedge E \vdash \mathcal{Q}_2\right], \quad (1)$$

where the sum is taken over all the possible transcripts $\mathcal{T} = (\mathcal{Q}_1, \mathcal{Q}_2)$ such that $\mathcal{A}(\mathcal{Q}_1, \mathcal{Q}_2) = 1$.[2]

[2] Here we only consider "valid" transcripts that $\mathcal{A}$ might produce by communicating with a permutation $P \in \mathcal{P}_n$ and a blockcpher $E \in BC(\kappa, n)$. For example, in a valid transcript $\mathcal{T} = (\mathcal{Q}_1, \mathcal{Q}_2)$, (x, y) and (x', y) with $x \neq x'$ could not be both contained in $\mathcal{Q}_1$.

2.4 Coupling Technique

Given a finite event space Ω and two probability distributions μ and ν defined on Ω, the *total variation distance* between μ and ν, denoted $\|\mu - \nu\|$, is defined as

$$\|\mu - \nu\| = \frac{1}{2} \sum_{x \in \Omega} |\mu(x) - \nu(x)|.$$

The following definitions are also all equivalent.

$$\|\mu - \nu\| = \max_{S \subset \Omega}\{\mu(S) - \nu(S)\} = \max_{S \subset \Omega}\{\nu(S) - \mu(S)\} = \max_{S \subset \Omega}\{|\mu(S) - \nu(S)|\}.$$

A *coupling* of μ and ν is a distribution τ on $\Omega \times \Omega$ such that for all $x \in \Omega$, $\sum_{y \in \Omega} \tau(x, y) = \mu(x)$ and for all $y \in \Omega$, $\sum_{x \in \Omega} \tau(x, y) = \nu(x)$. In other words, τ is a joint distribution whose marginal distributions are respectively μ and ν. We will use the following two lemmas in subsequent security proofs.

Lemma 1. *Let μ and ν be probability distributions on a finite event space Ω, let τ be a coupling of μ and ν, and let (X, Y) be a random variable sampled according to distribution τ. Then $\|\mu - \nu\| \leq \mathbf{Pr}[X \neq Y]$.*

Lemma 2. *Let Ω be some finite event space and ν be the uniform probability distribution on Ω. Let μ be a probability distribution on Ω such that $\|\mu - \nu\| \leq \epsilon$. Then there is a set $S \subset \Omega$ such that*

1. $|S| \geq (1 - \sqrt{\epsilon})|\Omega|$,
2. $\mu(x) \geq (1 - \sqrt{\epsilon})\nu(x)$ *for every* $x \in S$.

The proof of the above lemmas is given in [14]. For completeness, we include the same proof in Appendix B.

3 Security Proofs

In the security proof of cascade encryption CE^l, we will assume that for any x, $y \in I_n$, there are at most β keys k such that $(x, k, y) \in \mathcal{Q}_2$. Define the weight of $\mathcal{Q}_2$ by

$$\omega(\mathcal{Q}_2) = \max_{x,y \in I_n} |\{k : (x, k, y) \in \mathcal{Q}_2\}|.$$

Then we have

$$\mathbf{Pr}\left[E \xleftarrow{\$} BC(\kappa, n) : \omega(\mathcal{Q}_2) > \beta\right] \leq 2^{2n-\beta} \tag{2}$$

for any $\beta \geq e2^{\kappa-n+1}$. Note that a distinguisher $\mathcal{A}$ is deterministic, so once E is chosen then $\mathcal{Q}_2$, and hence $\omega(\mathcal{Q}_2)$ is uniquely determined. This bound has already been used in [6,9], while for completeness we give a proof in Appendix C. With this probabilistic restriction, the security proof of cascade encryption CE^l will use the following lemma.

Lemma 3. *Let $\delta > 0$ and $\beta \geq e2^{\kappa-n+1}$. Assume that for any transcript $\mathcal{T} = (\mathcal{Q}_1, \mathcal{Q}_2)$ such that $|\mathcal{Q}_1| = q_1$, $|\mathcal{Q}_2| = q_2$ and $\omega(\mathcal{Q}_2) \leq \beta$, we have*

$$\mathsf{p}_1(\mathcal{Q}_1|\mathcal{Q}_2) \geq (1-\delta)\mathsf{p}_2(\mathcal{Q}_1|\mathcal{Q}_2),$$

where

$$\mathsf{p}_1(\mathcal{Q}_1|\mathcal{Q}_2) = \mathbf{Pr}\left[\mathbf{k} \xleftarrow{\$} I_\kappa^l, E \xleftarrow{\$} BC(\kappa,n) : \mathsf{CE}_\mathbf{k}^l[E] \vdash \mathcal{Q}_1 \;\middle|\; E \vdash \mathcal{Q}_2\right],$$

$$\mathsf{p}_2(\mathcal{Q}_1|\mathcal{Q}_2) = \mathbf{Pr}\left[P \xleftarrow{\$} \mathcal{P}_n, E \xleftarrow{\$} BC(\kappa,n) : P \vdash \mathcal{Q}_1 \;\middle|\; E \vdash \mathcal{Q}_2\right] = 1/(N)_{q_1}.$$

Then we have

$$\mathbf{Adv}_{\mathsf{CE}^l}^{\mathsf{PRP}}(\mathcal{A}) \leq \delta + 2^{2n-\beta}.$$

Proof. For a transcript $\mathcal{T} = (\mathcal{Q}_1, \mathcal{Q}_2)$, define

$$\begin{aligned}
\mathsf{p}(\mathcal{Q}_2) &= \mathbf{Pr}\left[E \xleftarrow{\$} BC(\kappa,n) : E \vdash \mathcal{Q}_2\right],\\
\mathsf{p}_1(\mathcal{Q}_1,\mathcal{Q}_2) &= \mathbf{Pr}\left[\mathbf{k} \xleftarrow{\$} I_\kappa^l, E \xleftarrow{\$} BC(\kappa,n) : \mathsf{CE}_\mathbf{k}^l[E] \vdash \mathcal{Q}_1 \wedge E \vdash \mathcal{Q}_2\right]\\
&= \mathsf{p}_1(\mathcal{Q}_1|\mathcal{Q}_2)\mathsf{p}(\mathcal{Q}_2),\\
\mathsf{p}_2(\mathcal{Q}_1,\mathcal{Q}_2) &= \mathbf{Pr}\left[P \xleftarrow{\$} \mathcal{P}_n, E \xleftarrow{\$} BC(\kappa,n) : P \vdash \mathcal{Q}_1 \wedge E \vdash \mathcal{Q}_2\right]\\
&= \mathsf{p}_2(\mathcal{Q}_1|\mathcal{Q}_2)\mathsf{p}(\mathcal{Q}_2).
\end{aligned}$$

Then by (1) and (2), we have

$$\begin{aligned}
\mathbf{Adv}_{\mathsf{CE}^l}^{\mathsf{PRP}}(\mathcal{A}) &= \sum_{\mathcal{A}(\mathcal{Q}_1,\mathcal{Q}_2)=1} \mathsf{p}_2(\mathcal{Q}_1,\mathcal{Q}_2) - \sum_{\mathcal{A}(\mathcal{Q}_1,\mathcal{Q}_2)=1} \mathsf{p}_1(\mathcal{Q}_1,\mathcal{Q}_2)\\
&= \sum_{\substack{\mathcal{A}(\mathcal{Q}_1,\mathcal{Q}_2)=1\\ \omega(\mathcal{Q}_2)\leq\beta}} \mathsf{p}_2(\mathcal{Q}_1|\mathcal{Q}_2)\mathsf{p}(\mathcal{Q}_2) - \sum_{\substack{\mathcal{A}(\mathcal{Q}_1,\mathcal{Q}_2)=1\\ \omega(\mathcal{Q}_2)\leq\beta}} \mathsf{p}_1(\mathcal{Q}_1|\mathcal{Q}_2)\mathsf{p}(\mathcal{Q}_2)\\
&\quad + \sum_{\substack{\mathcal{A}(\mathcal{Q}_1,\mathcal{Q}_2)=1\\ \omega(\mathcal{Q}_2)>\beta}} \mathsf{p}_2(\mathcal{Q}_1,\mathcal{Q}_2) - \sum_{\substack{\mathcal{A}(\mathcal{Q}_1,\mathcal{Q}_2)=1\\ \omega(\mathcal{Q}_2)>\beta}} \mathsf{p}_1(\mathcal{Q}_1,\mathcal{Q}_2)\\
&\leq \sum_{\substack{\mathcal{A}(\mathcal{Q}_1,\mathcal{Q}_2)=1\\ \omega(\mathcal{Q}_2)\leq\beta}} \mathsf{p}_2(\mathcal{Q}_1|\mathcal{Q}_2)\mathsf{p}(\mathcal{Q}_2) - \sum_{\substack{\mathcal{A}(\mathcal{Q}_1,\mathcal{Q}_2)=1\\ \omega(\mathcal{Q}_2)\leq\beta}} \mathsf{p}_1(\mathcal{Q}_1|\mathcal{Q}_2)\mathsf{p}(\mathcal{Q}_2) + \sum_{\omega(\mathcal{Q}_2)>\beta} \mathsf{p}_2(\mathcal{Q}_2)\\
&\leq \sum_{\substack{\mathcal{A}(\mathcal{Q}_1,\mathcal{Q}_2)=1\\ \omega(\mathcal{Q}_2)\leq\beta}} \mathsf{p}_2(\mathcal{Q}_1|\mathcal{Q}_2)\mathsf{p}(\mathcal{Q}_2) - (1-\delta) \sum_{\substack{\mathcal{A}(\mathcal{Q}_1,\mathcal{Q}_2)=1\\ \omega(\mathcal{Q}_2)\leq\beta}} \mathsf{p}_2(\mathcal{Q}_1|\mathcal{Q}_2)\mathsf{p}(\mathcal{Q}_2) + 2^{2n-\beta}\\
&\leq \delta \sum_{\substack{\mathcal{A}(\mathcal{Q}_1,\mathcal{Q}_2)=1\\ \omega(\mathcal{Q}_2)\leq\beta}} \mathsf{p}_2(\mathcal{Q}_1,\mathcal{Q}_2) + 2^{2n-\beta} \leq \delta + 2^{2n-\beta}. \qquad \square
\end{aligned}$$

In the security proof of xor-cascade encryption XCE^l, we put no restriction on the weight of $\mathcal{Q}_2$. In this case, we can use the following lemma whose proof is similar as Lemma 3. (We might simply apply $\beta = \infty$ to Lemma 3.)

Lemma 4. *Let $\delta > 0$. Assume that for any transcript $\mathcal{T} = (\mathcal{Q}_1, \mathcal{Q}_2)$ such that $|\mathcal{Q}_1| = q_1$ and $|\mathcal{Q}_2| = q_2$, we have*

$$\mathsf{p}_1(\mathcal{Q}_1|\mathcal{Q}_2) \geq (1-\delta)\mathsf{p}_2(\mathcal{Q}_1|\mathcal{Q}_2),$$

where

$$\mathsf{p}_1(\mathcal{Q}_1|\mathcal{Q}_2) = \mathbf{Pr}\left[\mathbf{k} \xleftarrow{\$} I_\kappa^l, \mathbf{z} \xleftarrow{\$} I_n^{l+1}, E \xleftarrow{\$} BC(\kappa, n) : \mathsf{XCE}_{\mathbf{k},\mathbf{z}}^l[E] \vdash \mathcal{Q}_1 \;\middle|\; E \vdash \mathcal{Q}_2\right],$$

$$\mathsf{p}_2(\mathcal{Q}_1|\mathcal{Q}_2) = \mathbf{Pr}\left[P \xleftarrow{\$} \mathcal{P}_n, E \xleftarrow{\$} BC(\kappa, n) : P \vdash \mathcal{Q}_1 \;\middle|\; E \vdash \mathcal{Q}_2\right] = 1/(N)_{q_1}.$$

Then we have

$$\mathbf{Adv}_{\mathsf{XCE}^l}^{\mathsf{PRP}}(q_1, q_2) \leq \delta.$$

3.1 Security of Cascade Encryption

In this section, we analyze the security of cascade encryption CE^l for even length $l = 2d$. We begin with slightly modifying the key sampling process of CE^l. Consider the following three key sampling processes.

A: Choose $\mathbf{k} \in I_\kappa^l$ uniformly at random.
B: Choose $\mathbf{k} \in (I_\kappa)^{*l}$ uniformly at random.
C: Randomly partition $T_1 \cup T_2 = I_\kappa$ so that $|T_1| = |T_2|$, choose $\mathbf{k}' \in (T_1)^{*d}$ and $\mathbf{k}'' \in (T_2)^{*d}$ uniformly at random, and then define $\mathbf{k} = (\mathbf{k}', \mathbf{k}'')$.

One can distinguish sampling processes **A** and **B** with advantage at most

$$\binom{l}{2}\frac{1}{2^\kappa} \leq \frac{l^2}{2^{\kappa+1}}. \tag{3}$$

On the other hand, sampling processes **B** and **C** have exactly the same probability distribution. (See Appendix D for the proof.) Taking into account (3), we will analyze the security of

$$\mathsf{CE}_{\mathbf{k}}^l[E] = \mathsf{CE}_{\mathbf{k}''}^d[E] \circ \mathsf{CE}_{\mathbf{k}'}^d[E],$$

where $\mathbf{k}$, $\mathbf{k}'$ and $\mathbf{k}''$ are defined by key sampling process **C** instead of the original process **A**.

If $\mathsf{CE}_{\mathbf{k}}^l[E] \vdash \mathcal{Q}_1$ for a query history $\mathcal{Q}_1 = (u^i, v^i)_{1 \leq i \leq q_1}$, then it follows that

$$\mathsf{CE}_{\mathbf{k}'}^d[E] \vdash (u^i, w^i)_{1 \leq i \leq q_1} \text{ and } \mathsf{CE}_{\mathbf{k}''}^d[E] \vdash (w^i, v^i)_{1 \leq i \leq q_1},$$

for some $\mathbf{w} = (w^i)_{1 \leq i \leq q_1} \in (I_n)^{*q_1}$. Therefore for a transcript $\mathcal{T} = (\mathcal{Q}_1, \mathcal{Q}_2)$, we have

$$\mathsf{p}_1(\mathcal{Q}_1|\mathcal{Q}_2) = \sum_{\mathbf{w} \in \Omega} \mathbf{Pr}\Big[T_1 \xleftarrow{\$} \mathcal{P}_{2^{\kappa-1}}(I_\kappa), \mathbf{k}' \xleftarrow{\$} (T_1)^{*d}, \mathbf{k}'' \xleftarrow{\$} (T_2)^{*d},$$

$$E \xleftarrow{\$} BC(\kappa, n) : \mathsf{CE}_{\mathbf{k}'}^d[E] \vdash (u^i, w^i) \wedge \mathsf{CE}_{\mathbf{k}''}^d[E] \vdash (w^i, v^i) \;\Big|\; E \vdash \mathcal{Q}_2\Big],$$

where $\Omega = (I_n)^{*q_1}$, $\mathcal{P}_{2^{\kappa-1}}(I_\kappa)$ is the set of all subsets of I_κ of size $2^{\kappa-1}$, and $T_2 = I_\kappa \backslash T_1$.

Given a partition (T_1, T_2) of I_κ, a blockcipher $E \in BC(\kappa, n)$ is naturally partitioned into two blockciphers $E' \in BC(T_1, n)$ and $E'' \in BC(T_2, n)$, and vice versa. Given a query history $\mathcal{Q}_2$ for E, then this partition also induces two query histories $\mathcal{Q}_2'$ for E' and $\mathcal{Q}_2''$ for E''. Namely, for $\mathcal{Q}_2 = (x^i, k^i, y^i)_{1\le i\le q_2}$, $\mathcal{Q}_2' = (x^i, k^i, y^i)_{1\le i\le q_2, k^i\in T_1}$ and $\mathcal{Q}_2'' = (x^i, k^i, y^i)_{1\le i\le q_2, k^i\in T_2}$. With these notations, we have

$$\mathbf{Pr}\Big[T_1 \xleftarrow{\$} \mathcal{P}_{2^{\kappa-1}}(I_\kappa), \mathbf{k}' \xleftarrow{\$} (T_1)^{*d}, \mathbf{k}'' \xleftarrow{\$} (T_2)^{*d}, E \xleftarrow{\$} BC(\kappa, n) :$$

$$\mathsf{CE}^d_{\mathbf{k}'}[E] \vdash (u^i, w^i) \wedge \mathsf{CE}^d_{\mathbf{k}''}[E] \vdash (w^i, v^i) \;\Big|\, E \vdash \mathcal{Q}_2\Big]$$

$$= \frac{1}{\binom{2^\kappa}{2^{\kappa-1}}} \sum_{\substack{T_1\cup T_2 = I_\kappa \\ |T_1|=|T_2|=2^{\kappa-1}}} \mathbf{Pr}\Big[\mathbf{k}' \xleftarrow{\$} (T_1)^{*d}, \mathbf{k}'' \xleftarrow{\$} (T_2)^{*d}, E' \xleftarrow{\$} BC(T_1, n),$$

$$E'' \xleftarrow{\$} BC(T_2, n) : \mathsf{CE}^d_{\mathbf{k}'}[E'] \vdash (u^i, w^i) \wedge \mathsf{CE}^d_{\mathbf{k}''}[E''] \vdash (w^i, v^i) \;\Big|\, E' \vdash \mathcal{Q}_2' \wedge E'' \vdash \mathcal{Q}_2''\Big],$$

and hence

$$\mathsf{p}_1(\mathcal{Q}_1|\mathcal{Q}_2) = \frac{1}{\binom{2^\kappa}{2^{\kappa-1}}} \sum_{\substack{T_1\cup T_2 = I_\kappa \\ |T_1|=|T_2|=2^{\kappa-1}}} \sum_{\mathbf{w}\in\Omega} \mathbf{Pr}\Big[\mathbf{k}' \xleftarrow{\$} (T_1)^{*d}, \mathbf{k}'' \xleftarrow{\$} (T_2)^{*d}, E' \xleftarrow{\$} BC(T_1, n),$$

$$E'' \xleftarrow{\$} BC(T_2, n) : \mathsf{CE}^d_{\mathbf{k}'}[E'] \vdash (u^i, w^i) \wedge \mathsf{CE}^d_{\mathbf{k}''}[E''] \vdash (w^i, v^i) \;\Big|\, E' \vdash \mathcal{Q}_2' \wedge E'' \vdash \mathcal{Q}_2''\Big],$$

where

$$\sum_{\mathbf{w}\in\Omega} \mathbf{Pr}\Big[\mathbf{k}' \xleftarrow{\$} (T_1)^{*d}, \mathbf{k}'' \xleftarrow{\$} (T_2)^{*d}, E' \xleftarrow{\$} BC(T_1, n), E'' \xleftarrow{\$} BC(T_2, n) :$$

$$\mathsf{CE}^d_{\mathbf{k}'}[E'] \vdash (u^i, w^i) \wedge \mathsf{CE}^d_{\mathbf{k}''}[E''] \vdash (w^i, v^i) \;\Big|\, E' \vdash \mathcal{Q}_2' \wedge E'' \vdash \mathcal{Q}_2''\Big]$$

$$= \sum_{\mathbf{w}\in\Omega} \mathbf{Pr}\Big[\mathbf{k} \xleftarrow{\$} (T_1)^{*d}, E \xleftarrow{\$} BC(T_1, n) : \mathsf{CE}^d_{\mathbf{k}}[E] \vdash (u^i, w^i) \;\Big|\, E \vdash \mathcal{Q}_2'\Big]$$

$$\times \mathbf{Pr}\Big[\mathbf{k} \xleftarrow{\$} (T_2)^{*d}, E \xleftarrow{\$} BC(T_2, n) : \mathsf{CE}^d_{\mathbf{k}}[E] \vdash (w^i, v^i) \;\Big|\, E \vdash \mathcal{Q}_2''\Big]. \quad (4)$$

In order to upper bound each factor of the products appearing in (4), we fix a query history $\mathcal{Q}_2 = (x^i, k^i, y^i)_{1\le i\le q'}$ such that $q' \le q_2$ and $\omega(\mathcal{Q}_2) \le \beta$, and define a probability distribution $\mu_{\mathbf{s}}$ for each $\mathbf{s} = (s^i)_{1\le i\le q_1} \in \Omega$, where for each

$\mathbf{w} = (w^i)_{1 \le i \le q_1} \in \Omega$,

$$\mu_{\mathbf{s}}(\mathbf{w}) = \mathbf{Pr}\left[\mathbf{k} \xleftarrow{\$} (I_\kappa)^{*d}, E \xleftarrow{\$} BC(\kappa, n) : \mathsf{CE}^d_{\mathbf{k}}[E] \vdash (s^i, w^i)_{1 \le i \le q_1} \;\middle|\; E \vdash \mathcal{Q}_2\right].$$

Using the coupling technique, we can upper bound the statistical distance between $\mu_{\mathbf{s}}$ and the uniform probability distribution. The proof will be given at the end of this section.

Lemma 5. *Let d be even, let $\mu_{\mathbf{s}}$ be the probability distribution defined as above, and let ν be the uniform probability distribution on Ω. Then for $M > 0$, we have $\|\mu_{\mathbf{s}} - \nu\| \le \epsilon$, where*

$$\epsilon = q_1 \left(\frac{2q_2}{M(2^\kappa - d)} + \frac{2M\beta}{2^\kappa - d} + \frac{2M}{N - M}\right)^{\frac{d}{2}}.$$

Applying Lemma 5 with $\mathbf{s} = \mathbf{u} = (u^i)_{1 \le i \le q_1}$, $\mathcal{Q}_2 = \mathcal{Q}_2'$ and

$$\mu_{\mathbf{u}}(\mathbf{w}) = \mathbf{Pr}\left[\mathbf{k} \xleftarrow{\$} (T_1)^{*d}, E \xleftarrow{\$} BC(T_1, n) : \mathsf{CE}^d_{\mathbf{k}}[E] \vdash (u^i, w^i) \;\middle|\; E \vdash \mathcal{Q}_2'\right],$$

and using Lemma 2, we have a subset $S_1 \subset \Omega$ such that $|S_1| \ge (1 - \sqrt{\epsilon})|\Omega|$ and

$$\mathbf{Pr}\left[\mathbf{k} \xleftarrow{\$} (T_1)^{*d}, E \xleftarrow{\$} BC(T_1, n) : \mathsf{CE}^d_{\mathbf{k}}[E] \vdash (u^i, w^i) \;\middle|\; E \vdash \mathcal{Q}_2'\right]$$
$$\ge (1 - \sqrt{\epsilon})\nu(\mathbf{w}) = \frac{1 - \sqrt{\epsilon}}{(N)_{q_1}}$$

for every $\mathbf{w} \in S_1$, where

$$\epsilon = q_1 \left(\frac{2q_2}{M(2^{\kappa-1} - d)} + \frac{2M\beta}{2^{\kappa-1} - d} + \frac{2M}{N - M}\right)^{\frac{d}{2}}.$$

Here $BC(T_1, n)$ is viewed as equivalent to $BC(\kappa - 1, n)$.

For $\mathcal{Q}_2''$, define $\mathcal{Q}_2'''$ where $(x, k, y) \in \mathcal{Q}_2''$ if and only if $(y, k, x) \in \mathcal{Q}_2'''$. Again, applying Lemma 5 with $\mathbf{s} = \mathbf{v} = (v^i)_{1 \le i \le q_1}$, $\mathcal{Q}_2 = \mathcal{Q}_2'''$ and

$$\mu_{\mathbf{v}}(\mathbf{w}) = \mathbf{Pr}\left[\mathbf{k} \xleftarrow{\$} (T_2)^{*d}, E \xleftarrow{\$} BC(T_2, n) : \mathsf{CE}^d_{\mathbf{k}}[E] \vdash (v^i, w^i) \;\middle|\; E \vdash \mathcal{Q}_2'''\right]$$
$$= \mathbf{Pr}\left[\mathbf{k} \xleftarrow{\$} (T_2)^{*d}, E \xleftarrow{\$} BC(T_2, n) : \mathsf{CE}^d_{\mathbf{k}}[E] \vdash (w^i, v^i) \;\middle|\; E \vdash \mathcal{Q}_2''\right],$$

we have a subset $S_2 \subset \Omega$ such that $|S_2| \ge (1 - \sqrt{\epsilon})|\Omega|$ and

$$\mathbf{Pr}\left[\mathbf{k} \xleftarrow{\$} (T_2)^{*d}, E \xleftarrow{\$} BC(T_2, n) : \mathsf{CE}^d_{\mathbf{k}}[E] \vdash (w^i, v^i) \;\middle|\; E \vdash \mathcal{Q}_2''\right]$$
$$\ge (1 - \sqrt{\epsilon})\nu(\mathbf{w}) = \frac{1 - \sqrt{\epsilon}}{(N)_{q_1}}$$

for every $w \in S_2$. Let $S = S_1 \cap S_2$. Since $|S| \geq (1-2\sqrt{\epsilon})|\Omega|$, it follows that

$$\sum_{\mathbf{w}\in\Omega} \mathbf{Pr}\Big[\mathbf{k} \xleftarrow{\$} (T_1)^{*d}, E \xleftarrow{\$} BC(T_1,n) : \mathsf{CE}^d_{\mathbf{k}}[E] \vdash (u^i, w^i) \Big| E \vdash \mathcal{Q}'_2\Big]$$
$$\times \mathbf{Pr}\Big[\mathbf{k} \xleftarrow{\$} (T_2)^{*d}, E \xleftarrow{\$} BC(T_2,n) : \mathsf{CE}^d_{\mathbf{k}}[E] \vdash (w^i, v^i) \Big| E \vdash \mathcal{Q}''_2\Big]$$
$$\geq (1-2\sqrt{\epsilon})|\Omega| \cdot \left(\frac{1-\sqrt{\epsilon}}{(N)_{q_1}}\right)^2 \geq (1-4\sqrt{\epsilon})\mathsf{p}_2(\mathcal{Q}_1|\mathcal{Q}_2).$$

Therefore we have

$$\begin{aligned}\mathsf{p}_1(\mathcal{Q}_1|\mathcal{Q}_2) &\geq \frac{1}{\binom{2^\kappa}{2^{\kappa-1}}} \sum_{\substack{T_1\cup T_2 = I_\kappa \\ |T_1|=|T_2|=2^{\kappa-1}}} (1-4\sqrt{\epsilon})\mathsf{p}_2(\mathcal{Q}_1|\mathcal{Q}_2) \\ &= (1-4\sqrt{\epsilon})\mathsf{p}_2(\mathcal{Q}_1|\mathcal{Q}_2). \qquad (5)\end{aligned}$$

By (3), (5) and Lemma 3, we have the following theorem.

Theorem 1. *Let CE^l be an l-cascade encryption scheme using a κ-bit key n-bit blockcipher. If $l = 2d$ and d is even, then for $M > 0$ and $\beta \geq e2^{\kappa-n}$,*

$$\mathbf{Adv}^{\mathsf{PRP}}_{\mathsf{CE}^l}(q_1,q_2) \leq \frac{l^2}{2^{\kappa+1}} + 4q_1^{\frac{1}{2}}\left(\frac{2q_2}{M(2^{\kappa-1}-d)} + \frac{2M\beta}{2^{\kappa-1}-d} + \frac{2M}{N-M}\right)^{\frac{d}{4}} + 2^{2n-\beta}.$$

Optimizing parameters. Let $\beta \geq \max\{3n, e2^{\kappa-n}\}$. Then $3n \leq \beta$, and hence $2^{2n-\beta} \leq 1/2^n$. Let $M = \sqrt{\frac{q_2}{\beta}}$ by solving $\frac{2q_2}{M(2^{\kappa-1}-d)} = \frac{2M\beta}{2^{\kappa-1}-d}$. Then for $q_2 \leq 2^{\kappa+n}$, $M \leq \sqrt{\frac{2^{\kappa+n}}{e2^{\kappa-n}}} \leq \frac{N}{\sqrt{e}}$ and hence

$$\frac{1}{N-M} \leq \left(1 - \frac{1}{\sqrt{e}}\right)^{-1}\frac{1}{N} \leq \frac{e}{N}.$$

This implies

$$\frac{2M}{N-M} \leq \frac{2M\cdot e2^{\kappa-n}}{2^\kappa} \leq \frac{2M\beta}{2^{\kappa-1}-d}\left(= \frac{4M\beta}{2^\kappa - l}\right).$$

Using this inequality, the upper bound of Theorem 1 is simplified as follows.

Corollary 1. *Let CE^l be an l-cascade encryption scheme using a κ-bit key n-bit blockcipher. If l is a multiple of 4, then for $\beta \geq \max\{3n, e2^{\kappa-n}\}$,*

$$\mathbf{Adv}^{\mathsf{PRP}}_{\mathsf{CE}^l}(q_1,q_2) \leq \frac{l^2}{2^{\kappa+1}} + 4q_1^{\frac{1}{2}}\left(\frac{12\sqrt{\beta q_2}}{2^\kappa - l}\right)^{\frac{l}{8}} + \frac{1}{2^n}.$$

Interpretation. Assuming that $l^2/2^{\kappa+1}$ and $1/2^n$ are negligible, focus on the second term of the above upper bound. If we set $q_1 = 2^n$ to the maximum number of queries to the outer permutation and approximate $2^\kappa - l \approx 2^\kappa$, then the distinguishing advantage becomes negligible when

$$q_2 \ll \frac{2^{2\kappa - \frac{16}{l}(\frac{n}{2}+2)}}{144\beta} \le \min\left\{ \frac{2^{2\kappa - \frac{16}{l}(\frac{n}{2}+2)}}{432n}, \frac{2^{\kappa+n-\frac{16}{l}(\frac{n}{2}+2)}}{144e} \right\}.$$

Alternatively, let $q_2 = \min\left\{ \frac{2^{2\kappa}}{432n}, \frac{2^{\kappa+n}}{144e} \right\}$. Then the second term is upper bounded by $2^{\frac{n}{2}+2-\frac{l}{8}}$, approaching zero as the length l increases.

Proof of Lemma 5. Fix $\mathbf{s} = (s^i)_{1\le i \le q_1}$ and for $m = 0, \ldots, q_1$, define probability distributions π_m where for each $\mathbf{w} = (w^1, \ldots, w^{q_1}) \in \Omega$,

$$\pi_m(\mathbf{w}) = \mathbf{Pr}\Big[(u^{m+1}, \ldots, u^{q_1}) \xleftarrow{\$} (I_n \backslash \{s^1, \ldots, s^m\})^{*(q_1-m)}, \mathbf{k} \xleftarrow{\$} (I_\kappa)^{*d},$$

$$E \xleftarrow{\$} BC(\kappa, n) : \mathsf{CE}^d_{\mathbf{k}}[E] \vdash (s^i, w^i)_{1\le i\le m} \wedge \mathsf{CE}^d_{\mathbf{k}}[E] \vdash (u^i, w^i)_{m+1\le i \le q_1} \Big| E \vdash \mathcal{Q}_2 \Big].$$

Then we can check that $\pi_0 = \nu$ and $\pi_{q_1} = \mu_{\mathbf{s}}$. Since

$$\|\mu_{\mathbf{s}} - \nu\| \le \sum_{m=0}^{q_1-1} \|\pi_{m+1} - \pi_m\|, \tag{6}$$

we will focus on upper bounding $\|\pi_{m+1} - \pi_m\|$ for each $m = 0, \ldots, q_1 - 1$. In order to couple π_{m+1} and π_m, we will define a random variable (T, V) on $\Omega \times \Omega$ by the sampling process described in Figure 2. In this description,

$$\mathsf{D}(k) = \{x \in I_n : (x, k, y) \in \mathcal{Q}_2 \text{ for some } y\},$$
$$\mathsf{R}(k) = \{y \in I_n : (x, k, y) \in \mathcal{Q}_2 \text{ for some } x\},$$

for each key $k \in I_\kappa$. So they denote the domain points and the range points of the evaluations of $E(k, \cdot)$ determined by $\mathcal{Q}_2$, respectively.

In lines 1 to 4, the first $m + 1$ elements are initialized. They are updated in lines 5 to 23 along cascade encryption. Specifically, the first m elements are faithfully updated in lines 7 to 11, while the $(m + 1)$-th element is updated in lines 12 to 23 according to four conditions. The last $q_1 - m - 1$ elements of the output are determined in lines 24 to 29 without any update process.
As for this random variable, we point out some noteworthy properties.

1. In any case, the first m elements of T and V are equal.
2. If $t[d] = v[d]$, then $T = V$ at the end of the experiment.
3. By ignoring the steps used to sample V, we obtain the process for sampling T as described in Figure 3(a). Similarly, we obtain the process for sampling V as described in Figure 3(b). We can check that T and V follow probability distributions π_{m+1} and π_m, respectively.

```
1: for $i \leftarrow 1$ to $m$ do
2:     $w^i[0] \leftarrow s^i$
3: $t[0] \leftarrow s^{m+1}$
4: $v[0] \xleftarrow{\$} I_n \backslash \{s^1, \ldots, s^m\}$
5: for $j \leftarrow 1$ to $d$ do
6:     $k[j] \xleftarrow{\$} I_\kappa \backslash \{k[1], \ldots, k[j-1]\}$
7:     for $i \leftarrow 1$ to $m$ do
8:         if $w^i[j-1] \in \mathsf{D}(k[j])$ then
9:             $w^i[j] \leftarrow E(k[j], w^i[j-1])$
10:        else if $w^i[j-1] \notin \mathsf{D}(k[j])$ then
11:            $w^i[j] \xleftarrow{\$} I_n \backslash (\{w^1[j], \ldots, w^{i-1}[j]\} \cup \mathsf{R}(k[j]))$
12:    if $t[j-1] \in \mathsf{D}(k[j])$ and $v[j-1] \in \mathsf{D}(k[j])$ then
13:        $t[j] \leftarrow E(k[j], t[j-1])$
14:        $v[j] \leftarrow E(k[j], v[j-1])$
15:    else if $t[j-1] \in \mathsf{D}(k[j])$ and $v[j-1] \notin \mathsf{D}(k[j])$ then
16:        $t[j] \leftarrow E(k[j], t[j-1])$
17:        $v[j] \xleftarrow{\$} I_n \backslash (\{w^1[j], \ldots, w^m[j]\} \cup \mathsf{R}(k[j]))$
18:    else if $t[j-1] \notin \mathsf{D}(k[j])$ and $v[j-1] \in \mathsf{D}(k[j])$ then
19:        $t[j] \xleftarrow{\$} I_n \backslash (\{w^1[j], \ldots, w^m[j]\} \cup \mathsf{R}(k[j]))$
20:        $v[j] \leftarrow E(k[j], v[j-1])$
21:    else if $t[j-1] \notin \mathsf{D}(k[j])$ and $v[j-1] \notin \mathsf{D}(k[j])$ then
22:        $t[j] \xleftarrow{\$} I_n \backslash (\{w^1[j], \ldots, w^m[j]\} \cup \mathsf{R}(k[j]))$
23:        $v[j] \leftarrow t[j]$
24: if $t[d] = v[d]$ then
25:    $(v^{m+2}, \ldots, v^{q_1}) \xleftarrow{\$} (I_n \backslash \{w^1[d], \ldots, w^m[d], v[d]\})^{*(q_1-m-1)}$
26:    $(t^{m+2}, \ldots, t^{q_1}) \leftarrow (v^{m+2}, \ldots, v^{q_1})$
27: else
28:    $(v^{m+2}, \ldots, v^{q_1}) \xleftarrow{\$} (I_n \backslash \{w^1[d], \ldots, w^m[d], v[d]\})^{*(q_1-m-1)}$
29:    $(t^{m+2}, \ldots, t^{q_1}) \xleftarrow{\$} (I_n \backslash \{w^1[d], \ldots, w^m[d], t[d]\})^{*(q_1-m-1)}$
30: $T \leftarrow (w^1[d], \ldots, w^m[d], t[d], t^{m+2}, \ldots, t^{q_1})$
31: $V \leftarrow (w^1[d], \ldots, w^m[d], v[d], v^{m+2}, \ldots, v^{q_1})$
32: return $(T, V)$
```

Fig. 2. Sampling process for random variable (T, V)

```
1: for $i \leftarrow 1$ to $m$ do
2:     $w^i[0] \leftarrow s^i$
3: $t[0] \leftarrow s^{m+1}$
4: for $j \leftarrow 1$ to $d$ do
5:     $k[j] \xleftarrow{\$} I_\kappa \backslash \{k[1], \ldots, k[j-1]\}$
6:     for $i \leftarrow 1$ to $m$ do
7:         if $w^i[j-1] \in \mathsf{D}(k[j])$ then
8:             $w^i[j] \leftarrow E(k[j], w^i[j-1])$
9:         else if $w^i[j-1] \notin \mathsf{D}(k[j])$ then
10:            $w^i[j] \xleftarrow{\$} I_n \backslash (\{w^1[j], \ldots, w^{i-1}[j]\} \cup \mathsf{R}(k[j]))$
11:     if $t[j-1] \in \mathsf{D}(k[j])$ then
12:         $t[j] \leftarrow E(k[j], t[j-1])$
13:     else if $t[j-1] \notin \mathsf{D}(k[j])$ then
14:         $t[j] \xleftarrow{\$} I_n \backslash (\{w^1[j], \ldots, w^m[j]\} \cup \mathsf{R}(k[j]))$
15: $(t^{m+2}, \ldots, t^{q_1}) \xleftarrow{\$} (I_n \backslash \{w^1[d], \ldots, w^m[d], t[d]\})^{*(q_1-m-1)}$
16: return $T = (w^1[d], \ldots, w^m[d], t[d], t^{m+2}, \ldots, t^{q_1})$
```

(a) Sampling T

```
1: for $i \leftarrow 1$ to $m$ do
2:     $w^i[0] \leftarrow s^i$
3: $v[0] \xleftarrow{\$} I_n \backslash \{s^1, \ldots, s^m\}$
4: for $j \leftarrow 1$ to $d$ do
5:     $k[j] \xleftarrow{\$} I_\kappa \backslash \{k[1], \ldots, k[j-1]\}$
6:     for $i \leftarrow 1$ to $m$ do
7:         if $w^i[j-1] \in \mathsf{D}(k[j])$ then
8:             $w^i[j] \leftarrow E(k[j], w^i[j-1])$
9:         else if $w^i[j-1] \notin \mathsf{D}(k[j])$ then
10:            $w^i[j] \xleftarrow{\$} I_n \backslash (\{w^1[j], \ldots, w^{i-1}[j]\} \cup \mathsf{R}(k[j]))$
11:     if $v[j-1] \in \mathsf{D}(k[j])$ then
12:         $v[j] \leftarrow E(k[j], v[j-1])$
13:     else if $v[j-1] \notin \mathsf{D}(k[j])$ then
14:         $v[j] \xleftarrow{\$} I_n \backslash (\{w^1[j], \ldots, w^m[j]\} \cup \mathsf{R}(k[j]))$
15: $(v^{m+2}, \ldots, v^{q_1}) \xleftarrow{\$} (I_n \backslash \{w^1[d], \ldots, w^m[d], v[d]\})^{*(q_1-m-1)}$
16: return $V = (w^1[d], \ldots, w^m[d], v[d], v^{m+2}, \ldots, v^{q_1})$
```

(b) Sampling V

Fig. 3. Sampling T and V separately

Therefore by Lemma 1, we have

$$\|\pi_{m+1} - \pi_m\| \leq \mathbf{Pr}\left[T \neq V\right] = \mathbf{Pr}\left[t[d] \neq v[d]\right]. \tag{7}$$

Since $t[j] = v[j]$ implies $t[j+2] = v[j+2]$ for $j = 0, \ldots, d-2$ (actually, $t[j'] = v[j']$ for every $j' > j$), we have

$$\mathbf{Pr}\left[t[d] \neq v[d]\right] \leq \prod_{h=1}^{\frac{d}{2}} \mathbf{Pr}\left[t[2h] \neq v[2h] \,\middle|\, t[2h-2] \neq v[2h-2]\right]. \tag{8}$$

For a fixed $h = 1, \ldots, \frac{d}{2}$, assume that $t[2h-2] \neq v[2h-2]$, and on this condition, consider the probability that $t[2h]$ and $v[2h]$ are different. In order for this event to happen, either $t[2h-2]$ or $v[2h-2]$ should map to a point within $\mathsf{D}(k[2h])$ since otherwise $t[2h-1]$ and $v[2h-1]$ both outside $\mathsf{D}(k[2h])$ would map to an identical point $t[2h] = v[2h]$. We divide this event into three subcases. In the following description, we fix a parameter $M > 0$, and call a key k *heavy* if $|\mathsf{R}(k)| = |\mathsf{D}(k)| > M$.

Case 1: Either $k[2h-1]$ or $k[2h]$ is heavy. Since there are at most q_2/M heavy keys and $k[2h-1]$ and $k[2h]$ are chosen from the set of size at least $2^\kappa - d$, the probability of this case is at most

$$\frac{2q_2}{M(2^\kappa - d)}. \tag{9}$$

Case 2: $k[2h]$ is not heavy, and either $(t[2h-2], k[2h-1], y) \in \mathcal{Q}_2$ or $(v[2h-2], k[2h-1], y) \in \mathcal{Q}_2$ for some $y \in \mathsf{D}(k[2h])$. First, assume that $k[2h]$ is not heavy. Since $|\mathsf{D}(k[2h])| \leq M$ and $\omega(\mathcal{Q}_2) \leq \beta$, the number of keys k such that either $(t[2h-2], k, y) \in \mathcal{Q}_2$ or $(v[2h-2], k, y) \in \mathcal{Q}_2$ for some $y \in \mathsf{D}(k[2h])$ is at most $2M\beta$. Therefore the probability that one of such keys is chosen as $k[2h-1]$ is at most

$$\frac{2M\beta}{2^\kappa - d}. \tag{10}$$

Case 3: The remaining case. Here we assume that any of $k[2h-1]$ and $k[2h]$ is not heavy. Furthermore, $k[2h-1]$ and $\mathcal{Q}_2$ do not determine a mapping from one of $t[2h-2]$ and $v[2h-2]$ to any point within $\mathsf{D}(k[2h])$. However either $t[2h-2]$ or $v[2h-2]$ might still go into $\mathsf{D}(k[2h])$ by probabilistic sampling. Since $|\mathsf{D}(k[2h])| \leq M$ and $|\mathsf{R}(k[2h-1])| \leq M$, this case occurs with probability at most

$$\frac{2M}{N - M}. \tag{11}$$

We notice that the update of $w^i[2h-2]$, $i = 1, \ldots, m$, does not affect this upper bounding. By (6), (7), (8), (9), (10) and (11), we obtain

$$\|\mu_{\mathbf{s}} - \nu\| \leq q_1 \left(\frac{2q_2}{M(2^\kappa - d)} + \frac{2M\beta}{2^\kappa - d} + \frac{2M}{N - M} \right)^{\frac{d}{2}}.$$

3.2 Security of Xor-cascade Encryption

In this section, we analyze the security of xor-cascade encryption XCE^l for even length $l = 2d$. The argument is very similar to the security proof of the original cascade encryption except modifying key sampling process and applying Lemma 6. First, the following original key sampling process **A** is modified into **B**:

A: Choose $\mathbf{k} \in I_\kappa^l$ and $\mathbf{z} \in I_n^{l+1}$ uniformly at random.
B: Randomly partition $T_1 \cup T_2 = I_\kappa$ so that $|T_1| = |T_2|$, choose $\mathbf{k}' \in (T_1)^{*d}$ and $\mathbf{k}'' \in (T_2)^{*d}$ uniformly at random, and then define $\mathbf{k} = (\mathbf{k}', \mathbf{k}'')$. Next, choose $\mathbf{z}' = (z'_0, \dots, z'_d) \in I_n^{d+1}$ and $\mathbf{z}'' = (z''_0, \dots, z''_d) \in I_n^{d+1}$ uniformly at random, and then define

$$\mathbf{z} = (z'_0, \dots, z'_d \oplus z''_0, \dots, z''_d) \in I_n^{l+1}.$$

One can distinguish sampling processes **A** and **B** with advantage at most

$$\binom{l}{2}\frac{1}{2^\kappa} \le \frac{l^2}{2^{\kappa+1}}. \tag{12}$$

Taking into account (12), we analyze the security of

$$\mathsf{XCE}^l_{\mathbf{k},\mathbf{z}}[E] = \mathsf{XCE}^d_{\mathbf{k}'',\mathbf{z}''}[E] \circ \mathsf{XCE}^d_{\mathbf{k}',\mathbf{z}'}[E],$$

where $(\mathbf{k}, \mathbf{z})$, $(\mathbf{k}', \mathbf{z}')$ and $(\mathbf{k}'', \mathbf{z}'')$ are defined by key sampling process **B**.

For $\mathcal{Q}_1 = (u^i, v^i)_{1\le i\le q_1}$ and $\mathcal{Q}_2 = (x^i, k^i, y^i)_{1\le i\le q_2}$, we can prove

$$\mathsf{p}_1(\mathcal{Q}_1|\mathcal{Q}_2) = \frac{1}{\binom{2^\kappa}{2^{\kappa-1}}} \times$$
$$\sum_{\substack{T_1\cup T_2 = I_\kappa \\ |T_1|=|T_2|=2^{\kappa-1}}} \sum_{\mathbf{w}\in\Omega} \Bigg(\mathbf{Pr}\Big[\mathbf{k} \xleftarrow{\$} (T_1)^{*d}, \mathbf{z} \xleftarrow{\$} I_n^{d+1}, E \xleftarrow{\$} BC(T_1, n) : \mathsf{XCE}^d_{\mathbf{k}}[E] \vdash (u^i, w^i) \Big| E \vdash \mathcal{Q}'_2\Big]$$
$$\times \mathbf{Pr}\Big[\mathbf{k} \xleftarrow{\$} (T_2)^{*d}, \mathbf{z} \xleftarrow{\$} I_n^{d+1}, E \xleftarrow{\$} BC(T_2, n) : \mathsf{XCE}^d_{\mathbf{k}}[E] \vdash (w^i, v^i) \Big| E \vdash \mathcal{Q}''_2\Big]\Bigg),$$

with the same notations as the previous section. In order to estimate the probabilities appearing as the summands, we fix a query history $\mathcal{Q}_2 = (x^i, k^i, y^i)_{1\le i\le q'}$ such that $q' \le q_2$, and for each $\mathbf{s} \in \Omega$ define a probability distribution $\mu_\mathbf{s}$ such that for each $\mathbf{w} = (w^i)_{1\le i\le q_1} \in \Omega$,

$$\mu_\mathbf{s}(\mathbf{w}) = \mathbf{Pr}\Big[\mathbf{k} \xleftarrow{\$} (I_\kappa)^{*d}, \mathbf{z} \xleftarrow{\$} I_n^{d+1}, E \xleftarrow{\$} BC(\kappa, n) :$$
$$\mathsf{XCE}^d_{\mathbf{k}}[E] \vdash (s^i, w^i)_{1\le i\le q_1} \Big| E \vdash \mathcal{Q}_2\Big].$$

Then we have the following lemma.

Lemma 6. *Let $d > 0$, let $\mu_{\mathbf{s}}$ be the probability distribution defined as above, and let ν be the uniform probability distribution on Ω. Then for $M > 0$, we have $\|\mu_{\mathbf{s}} - \nu\| \leq \epsilon$, where*

$$\epsilon = q_1 \left(\frac{q_2}{M(2^\kappa - d)} + \frac{2M}{N} \right)^d .$$

The proof will be given at Appendix E. Using this lemma and exactly the same argument for the original cascade encryption, we can also prove the following theorem.

Theorem 2. *Let XCE^l be an l-xor-cascade encryption scheme using a κ-bit key n-bit blockcipher. If $l = 2d$, then for $M > 0$,*

$$\mathbf{Adv}^{\mathsf{PRP}}_{\mathsf{XCE}^l}(q_1, q_2) \leq \frac{l^2}{2^{\kappa+1}} + 4q_1^{\frac{1}{2}} \left(\frac{q_2}{M(2^{\kappa-1} - d)} + \frac{2M}{N} \right)^{\frac{d}{2}} .$$

Optimizing Parameters. By solving $\frac{q_2}{M(2^{\kappa-1}-d)} = \frac{2M}{N}$, we set $M = \sqrt{\frac{Nq_2}{2^\kappa - l}}$, obtaining the following corollary.

Corollary 2. *Let XCE^l be an l-cascade encryption scheme using a κ-bit key n-bit blockcipher. If l is even, then*

$$\mathbf{Adv}^{\mathsf{PRP}}_{\mathsf{XCE}^l}(q_1, q_2) \leq \frac{l^2}{2^{\kappa+1}} + 4q_1^{\frac{1}{2}} \left(\frac{16q_2}{N(2^\kappa - l)} \right)^{\frac{l}{8}} .$$

Interpretation. Assuming that $l^2/2^{\kappa+1}$ is negligible, set $q_1 = 2^n$ and approximate $2^\kappa - l \approx 2^\kappa$. Then the distinguishing advantage becomes negligible when

$$q_2 \ll 2^{\kappa+n-4-\frac{8}{l}(\frac{n}{2}+2)}.$$

Alternatively, let $q_2 = 2^{\kappa+n-5}$. Then we can check that $\mathbf{Adv}^{\mathsf{PRP}}_{\mathsf{CE}^l}(2^n, 2^{\kappa+n-5})$ approaches zero as the length l increases (up to the condition that $l^2/2^{\kappa+1}$ is negligible).

References

1. FIPS PUB 46: Data Encryption Standard (DES). National Institute of Standards and Technology (1977)
2. ANSI X9.52: Triple Data Encryption Algorithm Modes of Operation (1998)
3. FIPS PUB 46-3: Data Encryption Standard (DES). National Institute of Standards and Technology (1999)
4. FIPS PUB 197: Advanced Encryption Standard (AES). National Institute of Standards and Technology (2001)
5. NIST ST 800-67: Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher. National Institute of Standards and Technology (2004)
6. Bellare, M., Rogaway, P.: The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 409–426. Springer, Heidelberg (2006)

7. Bogdanov, A., Knudsen, L.R., Leander, G., Standaert, F.-X., Steinberger, J., Tischhauser, E.: Key-Alternating Ciphers in a Provable Setting: Encryption Using a Small Number of Public Permutations. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 45–62. Springer, Heidelberg (2012)
8. Gaži, P.: Plain versus Randomized Cascading-Based Key-Length Extension for Block Ciphers. Cryptology ePrint Archive, Report 2013/019 (2013), `http://eprint.iacr.org/2013/019`
9. Gaži, P., Maurer, U.: Cascade Encryption Revisited. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 37–51. Springer, Heidelberg (2009)
10. Gaži, P., Tessaro, S.: Efficient and Optimally Secure Key-Length Extension for Block Ciphers via Randomized Cascading. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 63–80. Springer, Heidelberg (2012)
11. Hoang, V.T., Morris, B., Rogaway, P.: An Enciphering Scheme Based on a Card Shuffle. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 1–13. Springer, Heidelberg (2012)
12. Hoang, V.T., Rogaway, P.: On Generalized Feistel Networks. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 613–630. Springer, Heidelberg (2010)
13. Kilian, J., Rogaway, P.: How to Protect DES Against Exhaustive Key Search (an Analysis of DESX). Journal of Cryptology 14, 17–35 (2001)
14. Lampe, R., Patarin, J., Seurin, Y.: An Asymptotically Tight Security Analysis of the Iterated Even-Mansour Cipher. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 278–295. Springer, Heidelberg (2012)
15. Maurer, U., Pietrzak, K., Renner, R.: Indistinguishability amplification. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 130–149. Springer, Heidelberg (2007)
16. Mironov, I.: (Not so) random shuffles of RC4. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 304–319. Springer, Heidelberg (2002)
17. Morris, B., Rogaway, P., Stegers, T.: How to Encipher Messages on a Small Domain: Deterministic Encryption and the Thorp Shuffle. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 286–302. Springer, Heidelberg (2009)

A A Brute-force Attack of $2^{\kappa+n}$ Query Complexity

In this section, we describe a standard information theoretic brute-force attack against a λ-bit key m-bit encryption scheme C that makes a certain number of calls to a κ-bit key n-bit blockcipher E.[3] An adversary $\mathcal{A}$ executes the following steps.

1. $\mathcal{A}$ makes all possible $2^{\kappa+n}$ queries to the underlying blockcipher E.
2. $\mathcal{A}$ makes t nonadaptive forward queries to the outer permutation, recording query history $\mathcal{Q} = (u^i, v^i)_{1\le i\le t}$.
3. If there is a λ-bit key $\mathbf{k}$ such that $\mathsf{C}_{\mathbf{k}}[E](u^i) = v^i$ for every $i = 1, \ldots, t$, then $\mathcal{A}$ outputs 0. Otherwise, $\mathcal{A}$ outputs 1.

Since we have

$$\mathbf{Pr}\left[\mathbf{k} \xleftarrow{\$} \{0,1\}^\lambda, E \xleftarrow{\$} BC(\kappa, n) : \mathcal{A}[\mathsf{C}_{\mathbf{k}}[E], E] = 0\right] = 1,$$

$$\mathbf{Pr}\left[P \xleftarrow{\$} \mathcal{P}_n, E \xleftarrow{\$} BC(\kappa, n) : \mathcal{A}[P, E] = 0\right] \le \frac{2^\lambda}{(2^m)_t},$$

$\mathbf{Adv}_{\mathsf{C}}^{\mathsf{PRP}}(\mathcal{A})$ gets close to 1 as $t \gg \frac{\lambda}{m}$.

[3] The output size m of C might be different from the block size n of E.

B Proof of Lemma 1 and Lemma 2

Proof of Lemma 1. Let λ be a coupling of μ and ν and let $(X,Y) \sim \lambda$. By definition, for any $z \in \Omega$, $\lambda(z,z) \le \min\{\mu(z),\nu(z)\}$. Since $\mathbf{Pr}[X=Y] = \sum_{z\in\Omega}\lambda(z,z)$, we have

$$\mathbf{Pr}[X=Y] \le \sum_{z\in\Omega} \min\{\mu(z),\nu(z)\}.$$

Therefore we have

$$\begin{aligned}\mathbf{Pr}[X\neq Y] &\ge 1-\sum_{z\in\Omega}\min\{\mu(z),\nu(z)\} = \sum_{z\in\Omega}(\mu(z)-\min\{\mu(z),\nu(z)\})\\ &= \sum_{\substack{z\in\Omega\\ \mu(z)\ge\nu(z)}}(\mu(z)-\nu(z)) = \max_{S\subset\Omega}\{\mu(S)-\nu(S)\} = \|\mu-\nu\|.\end{aligned}$$

Proof of Lemma 2. Let $S=\{x\in\Omega : \mu(x)\ge(1-\sqrt{\epsilon})\nu(x)\}$. By definition, any element of S satisfies the second condition. Contary to the first condition, suppose that $|S|<(1-\sqrt{\epsilon})|\Omega|$. This implies $\nu(\Omega\backslash S) = 1-|S|/|\Omega| > \sqrt{\epsilon}$, and hence

$$\nu(\Omega\backslash S)-\mu(\Omega\backslash S) \ge \nu(\Omega\backslash S) - \left(1-\sqrt{\epsilon}\right)\nu(\Omega\backslash S) = \sqrt{\epsilon}\nu(\Omega\backslash S) > \left(\sqrt{\epsilon}\right)^2 = \epsilon.$$

This is a contradiction to $\|\mu-\nu\|\le\epsilon$.

C Proof of Inequality (2)

Fix x, $y\in I_n$. For any $\beta>0$,

$$\mathbf{Pr}\left[E \xleftarrow{\$} BC(\kappa,n) : |\{k : E(k,x)=y\}|\ge\beta\right] \le \binom{2^\kappa}{\beta}\left(\frac{1}{2^n}\right)^\beta \le \left(\frac{e2^\kappa}{\beta 2^n}\right)^\beta.$$

Therefore for any $\mathcal{Q}_2$ (which might be the maximum query history of size $2^{\kappa+n}$ including all the evaluations of E), $\omega(\mathcal{Q}_2)$ is smaller than β except with probability

$$\mathsf{P} = 2^{2n}\left(\frac{e2^\kappa}{\beta 2^n}\right)^\beta,$$

where $\mathsf{P}\le 2^{2n-\beta}$ if $\beta\ge e2^{\kappa-n+1}$.

D Equivalence of Key Sampling Processes B and C for CE^l

Fix a key $\mathbf{k}=(\mathbf{k}',\mathbf{k}'')\in(I_\kappa)^{*l}$, where $\mathbf{k}'=(k'_1,\ldots,k'_d)$ and $\mathbf{k}''=(k''_1,\ldots,k''_d)$. Then the number of partitions (T_1,T_2) such that $\{k'_1,\ldots,k'_d\}\subset T_1$ and $\{k''_1,\ldots,k''_d\}\subset T_2$ is $\binom{2K-2d}{K-d}$, where $K=2^{\kappa-1}$. For each (T_1,T_2), key sampling process $\mathbf{C}$ chooses $\mathbf{k}'$ and $\mathbf{k}''$ from T_1 and T_2, respectively, with probability $(1/(K)_d)^2$. So the probability that $\mathbf{C}$ chooses $\mathbf{k}'$ and $\mathbf{k}''$ is

$$\frac{\binom{2K-2d}{K-d}}{\binom{2K}{K}}\left(\frac{1}{(K)_d}\right)^2 = \frac{(2K-2d)!(K!)^2}{(2K)!((K-d)!)^2}\cdot\left(\frac{(K-d)!}{K!}\right)^2 = \frac{(2K-2d)!}{(2K)!} = \frac{1}{(2^\kappa)_{2d}},$$

which is the same as the probability that key sampling process $\mathbf{B}$ chooses $\mathbf{k} = (\mathbf{k}', \mathbf{k}'')$.

E Proof of Lemma 6

Fix $\mathbf{s} = (s^i)_{1\le i\le q_1}$ and for $m = 0, \dots, q_1$, define probability distributions π_m where for each $\mathbf{w} = (w^1, \dots, w^{q_1}) \in \Omega$,

$$\pi_m(\mathbf{w}) = \mathbf{Pr}\Big[(u^{m+1}, \dots, u^{q_1}) \xleftarrow{\$} (I_n\backslash\{s^1, \dots, s^m\})^{*(q_1-m)}, \mathbf{k} \xleftarrow{\$} (I_\kappa)^{*d}, \mathbf{z} \xleftarrow{\$} I_n^{d+1},$$
$$E \xleftarrow{\$} BC(\kappa, n) : \mathsf{XCE}^d_{\mathbf{k}}[E] \vdash (s^i, w^i)_{1\le i\le m} \wedge \mathsf{XCE}^d_{\mathbf{k}}[E] \vdash (u^i, w^i)_{m+1\le i\le q_1} \;\Big|\; E \vdash \mathcal{Q}_2\Big].$$

Then we can check that $\pi_0 = \nu$ and $\pi_{q_1} = \mu_{\mathbf{s}}$. Since

$$\|\mu_{\mathbf{s}} - \nu\| \le \sum_{m=0}^{q_1-1} \|\pi_{m+1} - \pi_m\|, \tag{13}$$

we focus on upper bounding $\|\pi_{m+1} - \pi_m\|$ for each $m = 0, \dots, q_1 - 1$. In order to couple π_{m+1} and π_m, we will define a random variable (T, V) on $\Omega \times \Omega$ by the sampling process described in Figure 4. Then we can check that their marginal distributions are π_{m+1} and π_m, and

$$\|\pi_{m+1} - \pi_m\| \le \mathbf{Pr}\left[T \ne V\right] = \mathbf{Pr}\left[t[d] \ne v[d]\right]. \tag{14}$$

Since $t[j] = v[j]$ implies $t[j+1] = v[j+1]$ for $j = 0, \dots, l-1$, we have

$$\mathbf{Pr}\left[t[d] \ne v[d]\right] \le \prod_{j=1}^{d} \mathbf{Pr}\left[t[j] \ne v[j] \;\middle|\; t[j-1] \ne v[j-1]\right]. \tag{15}$$

In order to upper bound $\mathbf{Pr}\left[t[j] \ne v[j] \;\middle|\; t[j-1] \ne v[j-1]\right]$ for each j, we first choose $k[j]$ from the set of size at least $2^\kappa - d$. For a parameter $M > 0$, there are at most q_2/M heavy keys k such that

$$|\mathsf{R}(k)| = |\mathsf{D}(k)| > M.$$

Therefore the probability that $k[j]$ is heavy is at most

$$\frac{q_2}{M(2^\kappa - d)}. \tag{16}$$

Conditioned on the case that $k[j]$ is not heavy, either $t[j-1] \oplus z[j-1]$ or $v[j-1] \oplus z[j-1]$ should map to a point within $\mathsf{D}(k[j])$ since otherwise $t[j] \oplus z[j-1]$ and $v[j] \oplus z[j-1]$ both outside $\mathsf{D}(k[j])$ would map to an identical point $t[j+1] = v[j+1]$. The probability of this event over the random choice of $z[j-1]$ is at most

$$\frac{2M}{N}. \tag{17}$$

Then by (13), (14), (15), (16) and (17), we obtain

$$\|\mu_{\mathbf{s}} - \nu\| \leq q_1 \left(\frac{q_2}{M(2^\kappa - d)} + \frac{2M}{N} \right)^d .$$

```
1: for $i \leftarrow 1$ to $m$ do
2:     $w^i[0] \leftarrow s^i$
3: $t[0] \leftarrow s^{m+1}$
4: $v[0] \xleftarrow{\$} I_n \backslash \{s^1, \ldots, s^m\}$
5: for $j \leftarrow 1$ to $d$ do
6:     $k[j] \xleftarrow{\$} I_\kappa \backslash \{k[1], \ldots, k[j-1]\}$
7:     $z[j-1] \xleftarrow{\$} I_n$
8:     for $i \leftarrow 1$ to $m$ do
9:         $w^i[j-1] \leftarrow w^i[j-1] \oplus z[j-1]$
10:        if $w^i[j-1] \in \mathsf{D}(k[j])$ then
11:            $w^i[j] \leftarrow E(k[j], w^i[j-1])$
12:        else if $w^i[j-1] \notin \mathsf{D}(k[j])$ then
13:            $w^i[j] \xleftarrow{\$} I_n \backslash (\{w^1[j], \ldots, w^{i-1}[j]\} \cup \mathsf{R}(k[j]))$
14:    if $t[j-1] \oplus z[j-1] \in \mathsf{D}(k[j])$ and $v[j-1] \oplus z[j-1] \in \mathsf{D}(k[j])$ then
15:        $t[j] \leftarrow E(k[j], t[j-1] \oplus z[j-1])$
16:        $v[j] \leftarrow E(k[j], v[j-1] \oplus z[j-1])$
17:    else if $t[j-1] \oplus z[j-1] \in \mathsf{D}(k[j])$ and $v[j-1] \oplus z[j-1] \notin \mathsf{D}(k[j])$ then
18:        $t[j] \leftarrow E(k[j], t[j-1] \oplus z[j-1])$
19:        $v[j] \xleftarrow{\$} I_n \backslash (\{w^1[j], \ldots, w^m[j]\} \cup \mathsf{R}(k[j]))$
20:    else if $t[j-1] \oplus z[j-1] \notin \mathsf{D}(k[j])$ and $v[j-1] \oplus z[j-1] \in \mathsf{D}(k[j])$ then
21:        $t[j] \xleftarrow{\$} I_n \backslash (\{w^1[j], \ldots, w^m[j]\} \cup \mathsf{R}(k[j]))$
22:        $v[j] \leftarrow E(k[j], v[j-1] \oplus z[j-1])$
23:    else if $t[j-1] \oplus z[j-1] \notin \mathsf{D}(k[j])$ and $v[j-1] \oplus z[j-1] \notin \mathsf{D}(k[j])$ then
24:        $t[j] \xleftarrow{\$} I_n \backslash (\{w^1[j], \ldots, w^m[j]\} \cup \mathsf{R}(k[j]))$
25:        $v[j] \leftarrow t[j]$
26: $z[d] \xleftarrow{\$} I_n$
27: if $t[d] = v[d]$ then
28:     $(v^{m+2}, \ldots, v^{q_1}) \xleftarrow{\$} (I_n \backslash \{w^1[d], \ldots, w^m[d], v[d]\})^{*(q_1-m-1)}$
29:     $(t^{m+2}, \ldots, t^{q_1}) \leftarrow (v^{m+2}, \ldots, v^{q_1})$
30: else
31:     $(v^{m+2}, \ldots, v^{q_1}) \xleftarrow{\$} (I_n \backslash \{w^1[d], \ldots, w^m[d], v[d]\})^{*(q_1-m-1)}$
32:     $(t^{m+2}, \ldots, t^{q_1}) \xleftarrow{\$} (I_n \backslash \{w^1[d], \ldots, w^m[d], t[d]\})^{*(q_1-m-1)}$
33: $T \leftarrow (w^1[d] \oplus z[d], \ldots, w^m[d] \oplus z[d], t[d] \oplus z[d], t^{m+2}, \ldots, t^{q_1})$
34: $V \leftarrow (w^1[d] \oplus z[d], \ldots, w^m[d] \oplus z[d], v[d] \oplus z[d], v^{m+2}, \ldots, v^{q_1})$
35: return $(T, V)$
```

Fig. 4. Sampling process for random variable (T, V)

Ideal-Cipher (Ir)reducibility for Blockcipher-Based Hash Functions

Paul Baecher[1], Pooya Farshim[1], Marc Fischlin[1], and Martijn Stam[2]

[1] Department of Computer Science, Darmstadt University of Technology, Germany
www.cryptoplexity.de
[2] Department of Computer Science, University of Bristol, UK

Abstract. Preneel et al. (Crypto 1993) assessed 64 possible ways to construct a compression functions out of a blockcipher. They conjectured that 12 out of these 64 so-called PGV constructions achieve optimal security bounds for collision resistance and preimage resistance. This was proven by Black et al. (Journal of Cryptology, 2010), if one assumes that the blockcipher is ideal. This result, however, does not apply to "non-ideal" blockciphers such as AES. To alleviate this problem, we revisit the PGV constructions in light of the recently proposed idea of random-oracle reducibility (Baecher and Fischlin, Crypto 2011). We say that the blockcipher in one of the 12 secure PGV constructions reduces to the one in another construction, if *any* secure instantiation of the cipher, ideal or not, for one construction also makes the other secure. This notion allows us to relate the underlying assumptions on blockciphers in different constructions, and show that the requirements on the blockcipher for one case are not more demanding than those for the other. It turns out that this approach divides the 12 secure constructions into two groups of equal size, where within each group a blockcipher making one construction secure also makes all others secure. Across the groups this is provably not the case, showing that the sets of "good" blockciphers for each group are qualitatively distinct. We also relate the ideal ciphers in the PGV constructions with those in double-block-length hash functions such as Tandem-DM, Abreast-DM, and Hirose-DM. Here, our results show that, besides achieving better bounds, the double-block-length hash functions rely on weaker assumptions on the blockciphers to achieve collision and everywhere preimage resistance.

1 Introduction

The design of hash functions (or compression functions) from blockciphers has been considered very early in modern cryptography. Preneel, Govaerts, and Vandewalle [27] initiated a systematic study of designing a compression function $\mathsf{F} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ out of a blockcipher $\mathsf{E} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ by analyzing all 64 possible ways to combine the relevant inputs and outputs using xors only. Preneel et al. conjectured only 12 out of these 64 PGV constructions to be secure, including the well-known constructions of Matyas–Meyer–Oseas (MMO) and Davies–Meyer (DM). The idea continues to influence hash-function design till today. Indeed, one of the former five final candidates in the

SHA-3 competition, Skein [13], explicitly refers to this design methodology, and other former candidates like Grøstl [15] are based on similar principles.

The conjecture about the 12 secure PGV variants was later shown to be true in the *ideal-cipher model* (ICM) by Black et al. [9,10]. Roughly speaking, Black et al. show that assuming E implements a random blockcipher, the 12 secure PGV compression functions achieve optimal security of $\Theta(q^2 \cdot 2^{-n})$ for collision resistance and $\Theta(q \cdot 2^{-n})$ for preimage resistance, where q is the number of queries to the ideal cipher (and its inverse). Black et al. also discuss 8 further variants which, if used in an iteration mode, attain optimal collision resistance and sub-optimal preimage resistance of $\Theta(q^2 \cdot 2^{-n})$. The remaining 44 PGV versions are insecure.

Idealized models. As pointed out by Black et al. [10], security proofs for the PGV schemes in the ICM should be treated with care. Such results indicate that in order to break the security of the PGV scheme one would need to take advantage of structural properties of the blockcipher. Yet blockciphers such as AES, or the Threefish blockcipher used in Skein, clearly display a structure which is far from an ideal object. For instance, IDEA seems quite unsuitable to base a compression function on [33], while for AES recent related-key attacks [7,8] cast some shadow on its suitability for this purpose. Indeed, Khovratovich [20, Corollary 2] states unambiguously that "AES-256 in the Davies–Meyer hashing mode leads to an insecure hash function," but remarks that it is not known how to attack, for instance, double-block-length constructions. Moreover, it is currently still unknown how to exploit these weaknesses in AES-256 to break the standard collision or preimage security of any AES-instantiated PGV compression function. Consequently it may well be that AES makes some of the 12 PGV constructions secure, whereas others turn out to be insecure, despite a proof in the ICM. Unfortunately, it is very hard to make any security claims about specific PGV constructions with respect to a "real" blockcipher, or to even determine exactly the necessary requirements on the blockcipher for different PGV constructions to be secure.

Recently, a similar issue for the random-oracle model, where a monolithic idealized hash function is used, has been addressed by Baecher and Fischlin [4] via the so-called random-oracle reducibility. The idea is to relate the idealized hash functions in different (primarily public-key) schemes, allowing to conclude that the requirements on the hash function in one scheme are weaker than those in the other scheme. That is, Baecher and Fischlin consider two cryptographic schemes A and B with related security games in the random-oracle model. They define that the random oracle in scheme B reduces to the one in scheme A, if *any* instantiation $\mathcal{H}$ of the random oracle, possibly through an efficient hash function or again by an oracle-based solution, which makes scheme A secure, also makes scheme B secure. As such, the requirements on the hash function for scheme B are weaker than those for the one in scheme A. To be precise, Baecher and Fischlin allow an efficient but deterministic and stateless transformation $\mathcal{T}^{\mathcal{H}}$ for instantiating the random oracle in scheme B, to account for, say, different input or output sizes of the hash functions in the schemes. Using such transformations they are able to relate the random oracles in some public-key encryption schemes, including some ElGamal-type schemes.

Our results for the PGV constructions. We apply the idea of oracle reducibility to the ideal-cipher model and the PGV constructions. Take any two of the 12 PGV constructions, PGV_i and PGV_j, which are secure in the ICM. The goal is to show that any blockcipher (ideal or not) which makes PGV_i secure, also makes PGV_j secure. Here, security may refer to different games such as standard notion for collision resistance, preimage resistance, or everywhere preimage resistance [30]. Although we can ask the same question for indifferentiability from random functions [25], the PGV constructions, as pointed out in [11,21], do not achieve this level of security.[1]

Our first result divides the 12 secure PGV constructions into two groups $\mathcal{G}_1$ and $\mathcal{G}_2$ of size 6, where within each group the ideal cipher in each construction reduces to the ideal cipher in any other construction (with respect to collision resistance, [everywhere] preimage resistance, and preimage awareness). We sometimes call these the PGV_1-group and the PGV_2-group respectively: these two schemes are representatives of their respective groups. Across different groups, however, and for any of the security games, starting with the ideal cipher we can derive a blockcipher which makes all schemes in one group secure, whereas any scheme in the other group becomes insecure under this blockcipher. This separates the PGV_1-group and the PGV_2-group in terms of *direct* ideal-cipher reducibility. In direct reducibility we use the blockcipher in question without any modifications in another construction. This was one of the reasons to investigate different PGV constructions in the first place. For *free* reductions allowing arbitrary transformations $\mathcal{T}$ of the blockcipher, we show that the PGV constructions can be seen as transformations of each other, and under suitable $\mathcal{T}$ all 12 PGV constructions reduce to each other.

Preneel et al. [27] already discussed equivalence classes from an attack perspective. Our work reaffirms these classes and puts them on a solid theoretical foundation. Dividing the 12 constructions into two groups allows us to say that, within each group, one can use a blockcipher in a construction under the same *qualitative* assumptions on the blockcipher as for schemes; only across the groups this becomes invalid. In other words, the sets (or more formally, distributions) of "good" blockciphers for the groups are not equal, albeit they clearly share the ideal cipher as a common member making both groups simultaneously secure. We note that our results are also *quantitatively* tight in the sense that the blockciphers within a group are proven to be tightly reducible to each other in terms of the number of queries, running times, and success probabilities.

PGV and double-block-length hashing. Double-block-length (DBL) hash or compression functions aim at surpassing the $2^{n/2}$ upper bound for collision resistance of the PGV constructions by using two "PGV-like" constructions in parallel, doubling the output length. There are three major such compression

[1] This mainly motivates why we chose the oracle reducibility notion of [4] rather than the indifferentiability reducibility notion in [25].

functions, namely, Tandem-DM (TDM, [22]), Abreast-DM (ADM, [22]), and Hirose's construction (HDM, [19]). Several results underline the optimality of collision-resistance [19,23,24] and preimage-resistance bounds [2] for these functions in the ICM.

We next establish a connection between the basic PGV constructions and the double-block-length compression functions. Since all the DBL constructions have a "PGV_1-part" (with twice the key size) built in, it follows that any collision for any of the DBL functions immediately yields a collision for PGV_1 built from a blockcipher with $2n$-bit key. In other words, the ideal cipher in the DBL constructions directly reduces to the one in double-key PGV_1. We also prove that there is a free reduction to single-key PGV_1 from this double-key variant, thereby relating DBL functions to PGV_1 for free transformations. It follows, via a free reduction to PGV_1 and a free reduction from PGV_1 to PGV_2, that DBL functions reduce to PGV_2 for free transformations. An analogous result also applies to the everywhere preimage-resistance game, but, somewhat curiously, we show such a result cannot hold for the (standard) preimage-resistance game.

When it comes to free reducibility from PGV to DBL functions, we present irreducibility results for the collision-resistance and [everywhere] preimage-resistance games. We achieve this by making use of an interesting relationship to (lower bounds for) hash combiners [17,16,26]. Namely, if one can turn a collision (or preimage) for, say, PGV_1 into one for a DBL compression function, then we can think of PGV_1, which has n-bit digests, as a sort of robust hash combiner for the DBL function (which has $2n$-bit outputs). However, known lower bounds for hash combiners [26] tell us that such a combiner (with tight bounds and being black-box) cannot exist, and this transfers to ideal-cipher reducibility. More in detail, by combining Pietrzak's techniques [26] with a lower bound on generic collision finders by Bellare and Kohno [5] on compression functions, we confirm the irreducibility result formally for the simple case of black-box reductions making only a single call to the PGV collision-finder oracle (as also discussed in [26]). We leave the analysis of the full case to the final version. In summary, not only do the DBL functions provide stronger guarantees in terms of quantitative security (as well as efficiency and output length), but they also provably rely on qualitatively weaker assumptions on the blockcipher for the collision-resistance and everywhere preimage-resistance games.

Finally, we demonstrate that for none of the aforementioned DBL constructions the ideal cipher directly reduces to the one in either of the other schemes. That is, starting with the ideal cipher, for each target DBL function we construct a blockcipher which renders it insecure but preserves collision resistance for the other two functions. We are not aware of an analogous result for free reductions, but can exclude transformations which are involutions.

2 Preliminaries

NOTATION. We write $x \leftarrow y$ for assigning value y to variable x. We write $x \leftarrow_{\$} \mathsf{X}$ for sampling x from (finite) set X uniformly at random. If $\mathcal{A}$ is a probabilistic algorithm we write $y \leftarrow_{\$} \mathcal{A}(x_1, \ldots, x_n)$ for the action of running $\mathcal{A}$ on inputs $x_1, \ldots, x_n$ with coins chosen uniformly at random, and assigning the result to y. We use "|" for string concatenation, denote the bit complement of $x \in \{0,1\}^\star$ by $\overline{x}$. We set $[n] := \{1, \ldots, n\}$. We say $\epsilon(\lambda)$ is negligible if $|\epsilon(\lambda)| \in \lambda^{-\omega(1)}$.

Blockciphers. A blockcipher with key length k and block length n is a set of permutations and their inverses on $\{0,1\}^n$ indexed by a key in $\{0,1\}^k$. This set can therefore be thought of as a pair of functions

$$\mathsf{E} : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n \quad \text{and} \quad \mathsf{E}^{-1} : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n .$$

We denote the set of all such blockciphers by $\mathsf{Block}(k,n)$. A blockcipher is efficient if the above functions can be implemented by an efficient Turing machine.

Ideal and idealized (block)ciphers. An idealized (block)cipher with key length k and block length n is a distribution $\mathcal{E}$ on $\mathsf{Block}(k,n)$. We often consider an $\mathcal{E}$-idealized model of computation where all parties are given oracle access to a blockcipher chosen according to $\mathcal{E}$. *The* ideal-cipher model is the $\mathcal{E}$-idealized model where $\mathcal{E}$ is the uniform distribution on $\mathsf{Block}(k,n)$. We denote the set of all idealized ciphers with key length k and block length n (i.e., the set of all distributions on $\mathsf{Block}(k,n)$) by $\mathsf{Ideal}(k,n)$. Below, when saying that one has oracle access to an idealized cipher $\mathcal{E}$ it is understood that a blockcipher is sampled according to $\mathcal{E}$ and that one gets oracle access to this blockcipher.

Compression functions. A compression function is a function mapping $\{0,1\}^l$ to $\{0,1\}^m$ where $m < l$. We are primarily interested in compression functions which are built from a blockcipher. In this case we write $\mathsf{F}^{\mathsf{E},\mathsf{E}^{-1}} : \{0,1\}^l \to \{0,1\}^m$. A compression function is often considered in an idealized model where its oracles are sampled according to an idealized cipher $\mathcal{E}$.

2.1 Security Notions for Compression Functions

We now recall a number of fundamental security properties associated with blockcipher-based hashing.

Definition 1 (Everywhere preimage and collision resistance [30]). *Let* $\mathsf{F}^{\mathsf{E},\mathsf{E}^{-1}} : \{0,1\}^l \to \{0,1\}^m$ *be a compression function with oracle access to a blockcipher in* $\mathsf{Block}(k,n)$. *Let* $\mathcal{E}$ *denote an idealized cipher on* $\mathsf{Block}(k,n)$. *The preimage- (resp., everywhere preimage-, resp., collision-) resistance advantage of an adversary* $\mathcal{A}$ *in the* $\mathcal{E}$*-idealized model against* $\mathsf{F}^{\mathsf{E},\mathsf{E}^{-1}}$ *are defined by*

$$\mathbf{Adv}^{\mathsf{pre}}_{\mathsf{F},\mathcal{E}}(\mathcal{A}) := \Pr\left[\mathsf{F}^{\mathsf{E},\mathsf{E}^{-1}}(X') = Y \;:\; \begin{array}{c}(\mathsf{E},\mathsf{E}^{-1}) \leftarrow_{\$} \mathcal{E}; X \leftarrow_{\$} \{0,1\}^l; \\ Y \leftarrow \mathsf{F}^{\mathsf{E},\mathsf{E}^{-1}}(X); X' \leftarrow_{\$} \mathcal{A}^{\mathsf{E},\mathsf{E}^{-1}}(Y)\end{array}\right],$$

$$\mathbf{Adv}^{\mathsf{epre}}_{\mathsf{F},\mathcal{E}}(\mathcal{A}) := \Pr\left[\mathsf{F}^{\mathsf{E},\mathsf{E}^{-1}}(X) = Y \;:\; (\mathsf{E},\mathsf{E}^{-1}) \leftarrow_{\$} \mathcal{E}; (Y, \mathrm{st}) \leftarrow_{\$} \mathcal{A}_1; X \leftarrow_{\$} \mathcal{A}_2^{\mathsf{E},\mathsf{E}^{-1}}(\mathrm{st})\right],$$

$$\mathbf{Adv}^{\mathsf{coll}}_{\mathsf{F},\mathcal{E}}(\mathcal{A}) := \Pr\left[X_0 \neq X_1 \wedge \mathsf{F}^{\mathsf{E},\mathsf{E}^{-1}}(X_0) = \mathsf{F}^{\mathsf{E},\mathsf{E}^{-1}}(X_1) \;:\; \begin{array}{c}(\mathsf{E},\mathsf{E}^{-1}) \leftarrow_{\$} \mathcal{E}; \\ (X_0, X_1) \leftarrow_{\$} \mathcal{A}^{\mathsf{E},\mathsf{E}^{-1}}\end{array}\right].$$

For the set S_q of all adversaries which place at most q queries to their E or E^{-1} oracles in total we define

$$\mathbf{Adv}^{\mathsf{pre}}_{\mathsf{F},\mathcal{E}}(q) := \max_{\mathcal{A}\in S_q}\left\{\mathbf{Adv}^{\mathsf{pre}}_{\mathsf{F},\mathcal{E}}(\mathcal{A})\right\} ,$$

and similarly for the everywhere preimage-resistance and collision-resistance games. We note that although a compression function cannot be collision-resistant nor everywhere preimage-resistance with respect to a *fixed* blockcipher, reducibility arguments still apply [29].

Some of our results also hold for "more advanced" properties of hash or compression functions like preimage awareness [12]. (The definition can be found in the full version of the paper.) If so, we mention this briefly.

2.2 Reducibility

In order to define what it means for an idealized cipher to reduce to another, we begin with a semantics for security games similar to that in [6]. We capture the three security properties above by our notion, but can also extend the framework to cover a larger class of security games, such as complex multi-stage games and simulation-based notions. In the simpler case, we will consider a game between a challenger or a game Game and a sequence $\mathcal{A}_1, \mathcal{A}_2, \ldots$ of admissible adversaries (e.g., those which run in polynomial time). When the game terminates by outputting 1, this is deemed a success for the adversary (in that instance of the game). To determine the overall success of the adversaries, we then measure the success probability with respect to threshold t (e.g., 0 for computational games, or $\frac{1}{2}$ for decisional games). We present our formalism in the concrete setting. However, our definitions can be easily extended to the asymptotic setting by letting the game, its parameters, and adversaries to depend on a security parameter.

Definition 2 (Secure $\mathcal{E}$-idealized games). *An $\mathcal{E}$-idealized game consists of an oracle Turing machine* Game *(also called the challenger) with access to an idealized cipher $\mathcal{E}$ and n adversary oracles, a threshold $t \in [0,1]$, and a set S of n-tuples of admissible adversaries. The game terminates by outputting a bit. The advantage of adversaries $\mathcal{A}_1, \ldots, \mathcal{A}_n$ against* Game *is defined as*

$$\mathbf{Adv}^{\mathsf{Game}}_{\mathcal{E}}(\mathcal{A}_1, \ldots, \mathcal{A}_n) := \left|\Pr\left[\mathsf{Game}^{\mathsf{E},\mathsf{E}^{-1},\mathcal{A}_1^{\mathsf{E},\mathsf{E}^{-1}},\ldots,\mathcal{A}_n^{\mathsf{E},\mathsf{E}^{-1}}} = 1\right] - t\right| ,$$

where the probability is taken over Game*, $\mathcal{A}_1, \ldots, \mathcal{A}_n$, and $(\mathsf{E}, \mathsf{E}^{-1}) \leftarrow_{\$} \mathcal{E}$. For bounds $\epsilon \in [0,1]$ and $T, Q \in \mathbb{N}$ we say* Game *is (Q, T, ϵ)*-secure *if*

$$\forall(\mathcal{A}_1, \ldots, \mathcal{A}_n) \in S : \mathbf{Adv}^{\mathsf{Game}}_{\mathcal{E}}(\mathcal{A}_1, \ldots, \mathcal{A}_n) \le \epsilon$$

and Game *together with any set of admissible adversaries runs in time at most T and makes at most Q queries to the sample of the idealized cipher, including those of the adversaries.*

For example, the above notion captures everywhere preimage resistance by having $\mathcal{A}_1$ terminate by outputting (Y, st) with no access to the blockcipher, and $\mathcal{A}_2^{\mathsf{E},\mathsf{E}^{-1}}(\mathrm{st})$ return some X; the challenger then outputs 1 if and only if $\mathsf{F}^{\mathsf{E},\mathsf{E}^{-1}}(X) = Y$. Note that in particular, the construction F is usurped, together with the everywhere preimage experiment, in the general notation Game. We also note that with the above syntax we can combine multiple games into one by having a "master" adversary $\mathcal{A}$ first send a label to the challenger deciding which sub-game to play and then invoking the corresponding parties and game. Note also that as in [4] we assume that an idealized cipher can be given as an entirely ideal object, as a non-ideal object through a full description of an efficient Turing machine given as input to the parties, or a mixture thereof.

Ideal-cipher transformations. A transformation of ideal ciphers is a function $\mathcal{T}$ which maps a blockcipher from $\mathsf{Block}(k, n)$ to another blockcipher in $\mathsf{Block}(k', n')$. Typically, we will only be interested in *efficient* transformations i.e., those which can be implemented by efficient oracle Turing machines in the $\mathcal{E}$-idealized model, written $\mathcal{T}^{\mathsf{E}}$. Note that the requirement of $\mathcal{T}$ being a function implies that, algorithmically, the oracle Turing machine is deterministic and stateless. Below we envision the (single) transformation $\mathcal{T}$ to work in different modes $\mathsf{Enc}, \mathsf{Dec}$ to provide the corresponding interfaces for a blockcipher $(\mathsf{E}', \mathsf{E}'^{-1})$. Slightly abusing notation, we simply write $\mathcal{T}$ and $\mathcal{T}^{-1}$ for the corresponding interfaces E' and E'^{-1} (instead of $\mathcal{T}_{\mathsf{Enc}}^{\mathsf{E},\mathsf{E}^{-1}}$ for E' and $\mathcal{T}_{\mathsf{Dec}}^{\mathsf{E},\mathsf{E}^{-1}}$ for E'^{-1}). The transformation is written as

$$\mathsf{E}'(K, M) := \mathcal{T}^{\mathsf{E},\mathsf{E}^{-1}}(K, M) \quad \text{and} \quad \mathsf{E}'^{-1}(K, M) := \mathcal{T}^{-1^{\mathsf{E},\mathsf{E}^{-1}}}(K, M) .$$

Any transformation $\mathcal{T}$ also induces a mapping from $\mathsf{Ideal}(k, n)$ to $\mathsf{Ideal}(k', n')$. When E is sampled according to $\mathcal{E}$, then $\mathcal{T}$ induces an idealized cipher $\mathcal{E}' \in \mathsf{Ideal}(k', n')$ which we occasionally denote by $\mathcal{T}^{\mathcal{E}}$.

Definition 3 (Ideal-cipher reducibility). *Let* Game_1 *and* Game_2 *be two idealized games relying on blockciphers in* $\mathsf{Block}(k, n)$ *and* $\mathsf{Block}(k', n')$ *respectively. We say* the idealized cipher in Game_2 reduces to the idealized cipher in Game_1, *if for any* $\mathcal{E}_1 \in \mathsf{Ideal}(k, n)$ *there is a deterministic, stateless, and efficient transformation* $\mathcal{T} : \mathsf{Block}(k, n) \to \mathsf{Block}(k', n')$ *such that if*

$$\forall (\mathcal{A}_{1,1}, \ldots, \mathcal{A}_{1,n_1}) \in S_1 \ : \ \mathbf{Adv}_{\mathcal{E}_1}^{\mathsf{Game}_1}(\mathcal{A}_{1,1}, \ldots, \mathcal{A}_{1,n_1}) \leq \epsilon_1 ,$$

whenever Game_1 *runs in time at most* t_1 *and makes at most* Q_1 *queries to the block cipher sampled according to* $\mathcal{E}_1$*, then setting* $\mathcal{E}_2 := \mathcal{T}^{\mathcal{E}_1}$*, we have that*

$$\forall (\mathcal{A}_{2,1}, \ldots, \mathcal{A}_{2,n_2}) \in S_2 \ : \ \mathbf{Adv}_{\mathcal{E}_2}^{\mathsf{Game}_2}(\mathcal{A}_{2,1}, \ldots, \mathcal{A}_{2,n_2}) \leq \epsilon_2 ,$$

where Game *runs in time at most* t_2 *and makes at most* Q_2 *queries to the blockcipher sampled according to* $\mathcal{E}_2$*. In this case we say the reduction is* $(Q_1/Q_2, T, t_1/t_2, \epsilon_1/\epsilon_2)$*-tight, where* T *is an upper bound on the number of queries that* $\mathcal{T}$ *places to its oracle per invocation. When* $k = k'$*,* $n = n'$*, and* $\mathcal{T}$ *is the identity transformation, we say the reduction is* direct*; else it is called* free*.*

Definitional choices. In this work, our focus is on reducibility among blockcipher-based hash functions. In this setting, there are often no assumptions beyond the idealized cipher being chosen from a certain distribution. In this case, the strict, strong, and weak reducibility notions as discussed in [4] all collapse to the one given above. Of particular interest to us are two types of transformations. First, *free* transformations which can be arbitrary, and second the identity/dummy transformation which does not change the cipher. This latter type of direct reducibility asks if any idealized cipher making one construction secure makes the other secure too. The former type, however, apart from appropriately modifying the syntactical aspects of the blockcipher (such as the key or the block size), asks if the *model* for which one primitive is secure can be reduced to the model for which the other is secure.

3 Reducibility among the PGV Functions

We start by recalling the blockcipher-based constructions of hash functions by Preneel et al. [27,10]. The PGV compression functions rely on a blockcipher $\mathsf{E} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$, and map $\{0,1\}^{2n}$ to $\{0,1\}^n$:

$$\mathsf{PGV}_i^{\mathsf{E}} : \{0,1\}^{2n} \to \{0,1\}^n \quad \text{for} \quad \mathsf{E} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n .$$

There are 64 basic combinations to build such a compression function, of which 12 were first believed [27] (under category "✓" or "FP") and later actually proven to be secure [10] (under category "group-1"). We denote these secure compression functions by $\mathsf{PGV}_1, \dots, \mathsf{PGV}_{12}$ and adopt the s-index of [10] (as defined in Figure 2 there); they are depicted in Figure 1. The PGV_1 and PGV_5 functions can be instantiated with a blockcipher whose key length and message length are not equal. The remaining function, however, do not natively support this feature but they can be generalized such that they do [32].

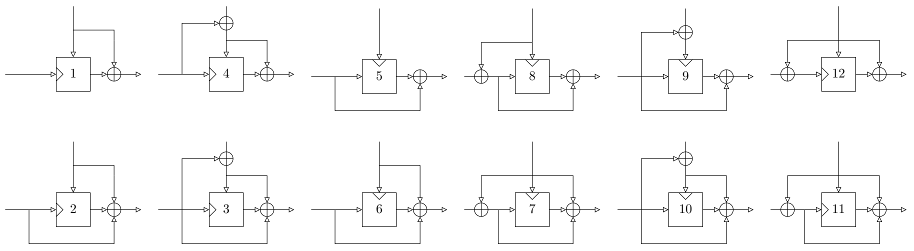


Fig. 1. The 12 optimally secure PGV constructions $\mathsf{PGV}_i^{\mathsf{E}}$ for $i \in [12]$. A triangle denotes the location of the key input. When used in an iteration mode, the top input is a message block and the left input is the chaining value. The first (resp. second) row corresponds to the PGV_1-group (resp. PGV_2-group).

For $i \in [12]$ and $q \geq 0$, the security bounds for uniform $\mathcal{E}$ according to [9,32,10] are

$$\mathbf{Adv}^{\mathsf{coll}}_{\mathsf{PGV}_i,\mathcal{E}}(q) \leq \frac{q^2}{2^n}, \qquad \mathbf{Adv}^{\mathsf{pre}}_{\mathsf{PGV}_i,\mathcal{E}}(q) \leq \frac{2q}{2^n}, \quad \text{and} \quad \mathbf{Adv}^{\mathsf{epre}}_{\mathsf{PGV}_i,\mathcal{E}}(q) \leq \frac{2q}{2^n}.$$

These bounds also hold when the key length and block length are not equal. Furthermore, for uniform $\mathcal{E}$, there exist adversaries $\mathcal{A}$ and $\mathcal{B}$ making q queries to their E and E^{-1} oracles in total such that [10][2]

$$\mathbf{Adv}^{\mathsf{coll}}_{\mathsf{PGV}_i,\mathcal{E}}(\mathcal{A}) \geq \frac{1}{8e}\frac{q^2+1}{2^n}, \qquad \mathbf{Adv}^{\mathsf{pre}}_{\mathsf{PGV}_i,\mathcal{E}}(\mathcal{B}) \geq \frac{q+1}{2^{n+1}}, \quad \text{and}$$

$$\mathbf{Adv}^{\mathsf{epre}}_{\mathsf{PGV}_i,\mathcal{E}}(\mathcal{B}) \geq \frac{q+1}{2^{n+1}}.$$

As we will show in the two following theorems, when it comes to ideal-cipher reducibility, the 12 secure PGV constructions can be further partitioned into two subgroups as follows, which we call the PGV_1-group and PGV_2-group, respectively.

$$\mathcal{G}_1 := \{\mathsf{PGV}_1, \mathsf{PGV}_4, \mathsf{PGV}_5, \mathsf{PGV}_8, \mathsf{PGV}_9, \mathsf{PGV}_{12}\}$$
$$\mathcal{G}_2 := \{\mathsf{PGV}_2, \mathsf{PGV}_3, \mathsf{PGV}_6, \mathsf{PGV}_7, \mathsf{PGV}_{10}, \mathsf{PGV}_{11}\}$$

The PGV_1 and PGV_2 functions will be representative of their respective groups.

The next proposition shows that, within a group, the compression functions are ideal-cipher reducible to each other in a direct and tight way (i.e., with the identity transformation and preserving the security bounds). It is worth pointing out that Preneel et al. [27] already discussed equivalence classes from an attack perspective. Present work reaffirms these classes and puts them on a solid theoretical foundation. As noted before, we cannot hope that any PGV compression function construction is indifferentiable from random (given access to E and E^{-1}), so we do not cover this property here; we can, however, include the notion of preimage awareness [12] to the games which are preserved.

Proposition 1. *Any two PGV constructions in $\mathcal{G}_1$ (resp., in $\mathcal{G}_2$) directly and $(1,1,1,1)$-tightly reduce the idealized cipher to each other for the [everywhere] preimage-resistance, collision-resistance, and preimage-awareness games.*

The proof of Proposition 1 appears in the full version of this paper.

Note that since we can combine the individual games into one, we can conclude that any blockcipher making a scheme from one group secure for all games simultaneously, would also make any other scheme in the group simultaneously secure. Also, the above equivalence still holds for PGV_1 and PGV_5 in case they work with a blockcipher with different key and message length.

The next theorem separates the two groups with respect to the collision-resistance and [everywhere] preimage-resistance games.

[2] The "plus one" terms are introduced in order to compactly capture the zero-query lower bounds.

Theorem 1. *No PGV construction in $\mathcal{G}_1$ (resp., in $\mathcal{G}_2$) directly reduces to any PGV construction in $\mathcal{G}_2$ (resp., in $\mathcal{G}_1$) for any of the collision-resistance and [everywhere] preimage-resistance games.*

For collision resistance and preimage resistance we assume the ideal cipher, whereas for everywhere preimage resistance we only need the minimal property that there exists *some* blockcipher making the schemes in one group secure, in order to achieve the separation. Due to space constraints we present the proof in the full version of this paper.

Proposition 2. *Any two PGV constructions* PGV_i *and* PGV_j *for* $i, j \in [12]$ $(1,1,1,1)$*-tightly reduce the idealized cipher to each other for the [everywhere] preimage-resistance and collision-resistance games (under free transformations).*

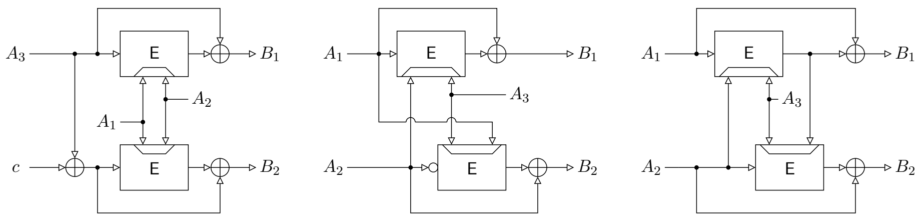
To prove this (which we do in the full version of this paper), we first show that there is a transformation such that there is an inter-group reduction, i.e., $\mathsf{PGV}_2 \in \mathcal{G}_2$ reduces to $\mathsf{PGV}_1 \in \mathcal{G}_1$ and vice versa—indeed we will use the same transformation for either direction. By transitivity we then obtain a reduction for any two constructions through Proposition 1, where we view the identity transformation as a special case of an arbitrary one.

4 Double-Block-Length Hashing and PGV

4.1 Reducibility from DBL to PGV

In this section we study the relation between three prominent double-block-length hash function constructions in the literature, namely, Hirose-DM [18,19], Abreast-DM [22,23], and Tandem-DM [22,24,14], and the PGV constructions. All the DBL compression functions under consideration here map $3n$-bit inputs to $2n$-bit outputs, and rely on a blockcipher with $2n$-bit keys and n-bit block. More precisely, these constructions are of the form

$$\mathsf{F}^{\mathsf{E}} : \{0,1\}^{3n} \to \{0,1\}^{2n} \quad \text{where} \quad \mathsf{E} : \{0,1\}^{2n} \times \{0,1\}^n \to \{0,1\}^n .$$



(2.a) Hirose-DM (2.b) Abreast-DM (2.c) Tandem-DM

Fig. 2. The three double-block-length compression functions. The hollow circle in Abreast-DM denotes bitwise complement.

We denote the Hirose-DM for a constant $c \in \{0,1\}^n \setminus \{0^n\}$, the Abreast-DM, and the Tandem-DM compression functions by HDM_c, ADM, and TDM, respectively. These functions are defined as follows (see Figure 2 for pictorial representations).

$$\begin{aligned}\mathsf{HDM}_c^{\mathsf{E}}(A_1, A_2, A_3) &:= (\mathsf{E}(A_1|A_2, A_3) \oplus A_3, \mathsf{E}(A_1|A_2, A_3 \oplus c) \oplus A_3 \oplus c)\\ \mathsf{ADM}^{\mathsf{E}}(A_1, A_2, A_3) &:= \big(\mathsf{E}(A_2|A_3, A_1) \oplus A_1, \mathsf{E}(A_3|A_1, \overline{A_2}) \oplus A_2\big)\\ \mathsf{TDM}^{\mathsf{E}}(A_1, A_2, A_3) &:= (\mathsf{E}(A_2|A_3, A_1) \oplus A_1, \mathsf{E}(A_3|\mathsf{E}(A_2|A_3, A_1), A_2) \oplus A_2)\end{aligned}$$

The next proposition shows that collisions (resp., somewhere preimages) in HDM_c directly lead to collisions (resp., somewhere preimages) for the double-key versions of PGV_1 and PGV_5 functions.

Proposition 3. *The idealized ciphers in* HDM_c*, for any* $c \in \{0,1\}^n \setminus \{0^n\}$, ADM*, and* TDM *compression functions directly and* $(1,1,1,1)$*-tightly reduce to those in the (double-key versions of the)* PGV_1 *and* PGV_5 *functions for the everywhere preimage-resistance and collision-resistance games.*

The proof of the proposition appears in the full version of this paper. Note that despite the tightness of the reduction, a blockcipher that makes the schemes PGV_1 and PGV_5 ideally secure is not guaranteed to make the double-block-length compression functions secure beyond the implied single-length security bound.

Curiously, the above argument fails for the preimage-resistance game as we cannot extend a challenge value for PGV_1 to a full challenge value for a DBL construction. The proof of the following proposition appears in the full version.

Proposition 4. *The idealized cipher in none of the DBL constructions directly reduces to the idealized cipher in* PGV_1 *(and hence neither to the one in* PGV_5*) for the (standard) preimage-resistance game.*

Direct ideal-cipher reducibility to the other PGV constructions is not syntactically possible as only the PGV_1 and PGV_5 constructions can be natively instantiated with a double-block-length blockcipher.[3] Note that the above proposition leaves open the (im)possibility of free reductions from DBL to PGV, which we leave to future work.

We next show that under *free* transformations a double-block-length instantiation of PGV_1 reduces to a single-block-length instantiation of PGV_1. By the transitivity of reductions we obtain reducibility of the idealized cipher in the DBL constructions to that in any of the PGV constructions.

Proposition 5. *The idealized cipher in* PGV_1 *instantiated with an idealized cipher in* $\mathsf{Ideal}(2n, n)$ $(2,2,1,1)$*-tightly reduces to the one in* PGV_1 *when instantiated with an idealized cipher in* $\mathsf{Ideal}(n, n)$ *for the everywhere preimage-resistance and collision-resistance games.*

[3] There exist modifications of the PGV constructions which can be instantiated with DBL blockciphers [32]. We leave their treatment to future work.

For space reasons, we defer the proof to the full version of this paper.

Remark. Although Merkle–Damgård chaining does *not* in general preserve the preimage resistance of the underlying compression function, there exist more sophisticated chaining rules, such as ROX [1], which do so. If such chaining rules are used to compress the keys in the proposition above, we also obtain reducibility for the preimage-resistance game.

4.2 Separations among the DBL Compression Functions

We now investigate direct reducibility among the DBL compression functions, as well as PGV_1 and DBL functions. We focus on collision resistance, but similar techniques (for separations) may be applicable to the other security games. For this game, there are twelve relations to be considered, three of which have already been settled by Proposition 3. We study the remaining relations by providing separations among all the possible pairs. In doing so, we give blockciphers E such that one of the DBL constructions (and hence by Proposition 3 the PGV_1 function, too) admits a trivial collision, whereas the other two constructions are *simultaneously* secure.

We start with the HDM_c compression function where $c \neq 0^n$. Let E be a blockcipher. Define a modified blockcipher $\tilde{\mathsf{E}}$ as follows.

$$M_c := \mathsf{E}^{-1}(0^n|0^n, \mathsf{E}(0^n|0^n, 0^n) \oplus c)\,, \quad C_0 := \mathsf{E}(0^n|0^n, 0^n)\,, \quad C_c := \mathsf{E}(0^n|0^n, c)\,.$$

$$\tilde{\mathsf{E}}(K_1|K_2, M) := \begin{cases} C_0 \oplus c & \text{if } (K_1|K_2, M) = (0^n|0^n, c)\,; \\ C_c & \text{if } (K_1|K_2, M) = (0^n|0^n, M_c)\,; \\ \mathsf{E}(K_1|K_2, M) & \text{otherwise.} \end{cases}$$

$$\tilde{\mathsf{E}}^{-1}(K_1|K_2, C) := \begin{cases} c & \text{if } (K_1|K_2, C) = (0^n|0^n, C_0 \oplus c)\,; \\ M_c & \text{if } (K_1|K_2, C) = (0^n|0^n, C_c)\,; \\ \mathsf{E}^{-1}(K_1|K_2, C) & \text{otherwise.} \end{cases}$$

Note that $\tilde{\mathsf{E}}$ and $\tilde{\mathsf{E}}^{-1}$ above define a blockcipher and we have $c \neq 0^n$. Hence,

$$\mathsf{HDM}_c^{\tilde{\mathsf{E}}}(0^n, 0^n, 0^n) = (\tilde{\mathsf{E}}(0^n|0^n, 0^n) \oplus 0^n, \tilde{\mathsf{E}}(0^n|0^n, c) \oplus c) = (C_0, C_0)\,,$$
$$\mathsf{HDM}_c^{\tilde{\mathsf{E}}}(0^n, 0^n, c) = (\tilde{\mathsf{E}}(0^n|0^n, c) \oplus c, \tilde{\mathsf{E}}(0^n|0^n, 0^n) \oplus 0^n) = (C_0, C_0)\,.$$

and the pair $((0^n, 0^n, 0^n), (0^n, 0^n, c))$ thus constitutes a non-trivial collision for $\mathsf{HDM}_c^{\tilde{\mathsf{E}}}$. However, the next lemma shows that ADM and TDM remain collision-resistant for this cipher. The proof appears in the full version of this paper.

Lemma 1. *Let $\tilde{\mathsf{E}}$ be a blockcipher as above with a distribution according to $(\mathsf{E}, \mathsf{E}^{-1}) \leftarrow_{\$} \mathsf{Block}(2n, n)$. Then $\mathsf{ADM}^{\tilde{\mathsf{E}}}$ and $\mathsf{TDM}^{\tilde{\mathsf{E}}}$ are both collision-resistant.*

Due to space constraints we also provide the remaining separating examples in the full version of this paper.

Theorem 2. *Let $c \in \{0,1\}^n \setminus \{0^n\}$. Then among the compression functions* HDM_c*,* ADM*, and* TDM *neither one directly reduces the idealized cipher in either one of the other two functions for the collision-resistance game.*

As a corollary of the above results we get that there is no direct reduction from PGV to any of the DBL compression functions: otherwise we also obtain direct reducibility to any other DBL compression function via Theorem 3, which we have shown to be impossible in the above theorem. In the next section we will extend this irreducibility result to free reductions.

4.3 Irreducibility of PGV to DBL

We now turn our attention to the converse of Propositions 3 and 5: can one convert any idealized cipher which makes a DBL construction secure to one which makes a PGV construction secure? We show strong evidence towards the impossibility of such a reduction. To this end, we restrict the class of reductions under the construction to *black-box* ones [28]. Such a reduction is a pair of oracle Turing machines $(\mathcal{T}, \mathcal{R})$. Both machines have access to a blockcipher, $\mathcal{T}$ is a transformation which implements an idealized cipher, and $\mathcal{R}$ is a reduction which given oracle access to an algorithm $\mathcal{B}$ breaking the security of a PGV construction when instantiated with $\mathcal{T}^{\mathsf{E}}$, breaks the security of a DBL construction with respect to E (for random E). As it will become apparent from the proof of the theorem, the type of reductions that we actually rule out allow both the transformation and the reduction to depend on the blockcipher and hence, in the terminology of [28], the class of reductions that we rule out lies somewhere in between fully black-box and $\forall\exists$semi-black-box reductions. More concisely, this class is captured as an NBN reduction in the CAP taxonomy of [3], meaning that the *C*onstruction may make non-black-box use of primitive, and that the reduction makes black-box use of the *A*dversary resp. non-black-box use of the *P*rimitive.

We make two further simplifications on the structure of the reduction. First we assume that $\mathcal{R}$ queries its break oracle $\mathcal{B}$ once. We call this a single-query reduction. Second, we require the reduction to succeed with a constant probability whenever $\mathcal{B}$ is successful. Now, the intuition behind the impossibility of the existence of such a reduction follows that for lower bounds on the output size of hash combiners [26]. The underlying idea is that the collision-resistance security of any of the DBL constructions is *beyond* that of the PGV constructions. More precisely, around $\Theta(2^n)$ queries are needed to break the collision resistance of any of the DBL constructions with noticeable probability, whereas this bound is only $\Theta(2^{n/2})$ for the PGV constructions. To derive a contradiction, we may simulate the break algorithm $\mathcal{B}$ for the reduction with only $\Theta(2^{n/2})$ queries, and the reduction will translate this collision efficiently to a DBL construction collision, which contradicts the $\Theta(2^n)$ collision-resistance bound.

We are now ready to state our irreducibility theorem. Since we are dealing with an impossibility result, for the sake of clarity of the presentation we present the theorem in asymptotic language. The proof appears in the full version.

Theorem 3. *There is no single-query fully black-box ideal-cipher reduction from any of the PGV constructions to any of the DBL constructions for the collision-resistance and [everywhere] preimage-resistance games as long as the reduction is tight: when the number of queries, run times, and success probabilities are parameterized by a security parameter, the reduction is* $(\mathcal{O}(1), \mathcal{O}(1), \mathcal{O}(1), \mathcal{O}(1))$*-tight.*

It is conceivable that the techniques of [26] can be leveraged to derive a more general theorem which rules out reductions that call the break oracle multiple times. Furthermore, one might also be able to extended the result to arbitrary games for two given constructions, as long as a lower bound on the success probability of an attack on the security of the first construction is noticeably higher than an upper bound on the security of the second.

5 Summary and Future Work

We summarize our reducibility results in Figure 3 and refer to the caption for details. One important observation from these results is that we do not have one single "Y" column, i.e., a compression function which reduces to all of the other ones—or, equivalently, a compression function which is secure if any of the others is secure. This would be a clear winner in the sense that it is the safest choice for practice.

For the "n" entries of Table 3.b we can show that there is a separation for a large class of potential transformation functions. More specifically, we show that there is no surjective transformation $\mathcal{T}$ to reduce, say, ADM to HDM_{1^n}, as long as the transformation also preserves HDM-security "backwards." Here, surjectivity means that $\mathcal{T}^{\mathsf{E}}$ varies over all possible blockciphers if E runs through all blockciphers, and backward security preservation means that $\mathcal{E}$ is secure for HDM if $\mathcal{T}^{\mathcal{E}}$ is. Transformations which are covered by this include, for example, those of the form $\mathcal{T}^{\mathsf{E}}_{\pi_1,\pi_2}(K_1|K_2, M) = \pi_2(\mathsf{E}(K_1|K_2, \pi_1(M)))$ for fixed involutions π_1, π_2 over $\{0,1\}^n$, or more generally, any transformation which is an involution (over $\mathsf{Block}(2n, n)$).[4] The argument is as follows. Assume that there exists such a $\mathcal{T}$. Then for any blockcipher E which makes HDM secure, the blockcipher $\mathcal{T}^{\mathsf{E}}$ makes ADM secure. However, we also know that there is a blockcipher $\mathsf{E}^\star$ such that $\mathsf{E}^\star$ gives rise to a collision-resistant $\mathsf{HDM}^{\mathsf{E}^\star}_{1^n}$ but renders $\mathsf{ADM}^{\mathsf{E}^\star}$ collision-tractable (see the full version of this paper). Now define E to be any blockcipher in the preimage of $\mathsf{E}^\star$ under $\mathcal{T}$ (such an E exists as $\mathcal{T}$ is surjective). The transformation now maps E to $\mathsf{E}^\star$, which means that it fails to provide security for ADM. Furthermore, E makes $\mathsf{HDM}^{\mathsf{E}}_{1^n}$ collision-resistant by assumption about backward security. This, however, contradicts the requirement of reducibility from ADM to HDM, because E makes HDM secure but $\mathcal{T}^{\mathsf{E}}$ is insecure for ADM.

[4] An example of a surjective transformation which is not backward-secure for PGV_1 is $\mathcal{T}^{\mathsf{E}}(K, M) = \mathsf{E}(K, M) \oplus K$, because it maps PGV_1 for $\mathcal{T}^{\mathsf{E}}$ to PGV_2 for E, and we know that there are idealized ciphers making PGV_2 secure but PGV_1 insecure.

	$\mathcal{G}_1$	$\mathcal{G}_2$	TDM	HDM_c	ADM
$\mathcal{G}_1$	Y 1	N 1	Y 3	Y 3	Y 3
$\mathcal{G}_2$	N 1	Y 1	–	–	–
TDM	N 2	–	Y	N 2	N 2
HDM_c	N 2	–	N 2	Y	N 2
ADM	N 2	–	N 2	N 2	Y

(3.a) Results for the identity transformation.

	$\mathcal{G}_1$	$\mathcal{G}_2$	TDM	HDM_c	ADM
$\mathcal{G}_1$	Y $\rightarrow$	Y 2	Y $\rightarrow$	Y $\rightarrow$	Y $\rightarrow$
$\mathcal{G}_2$	Y 2	Y $\rightarrow$	Y $\star$	Y $\star$	Y $\star$
TDM	N 3	N 3	Y	n 2	n 2
HDM_c	N 3	N 3	n 2	Y	n 2
ADM	N 3	N 3	n 2	n 2	Y

(3.b) Results for arbitrary transformations.

Fig. 3. Summary of our reducibility results for collision resistance. A "Y" or "N" in a cell means that any cipher which makes the compression function corresponding to the row collision-resistant also makes the compression function corresponding to the column collision-resistant. A "–" in direct reductions indicates a syntax mismatch. The number below an entry indicates the theorem/proposition supporting the claim. An arrow "$\rightarrow$" means that the result is implied by the left table. Reductions on the diagonal of TDM, HDM_c, and ADM trivially follow by self-reductions. Note that for arbitrary transformations each cell might be using different transformations. The star symbol "$\star$" denotes reducibility by transitivity. An "n" is a separation for a restricted class of transformations; see Section 5.

Open problems. Recall that we showed that one can transform a good blockcipher E (or rather distribution $\mathcal{E}$) for the PGV_1-group into a good one $\mathcal{T}^{\mathsf{E}}$ for the PGV_2-group. We also presented a transformation in the opposite direction. Ideally, though, one would be interested in a *single* transformation $\mathcal{T}$ which, given $\mathcal{E}$ making a PGV construction secure, turns it into $\mathcal{T}^{\mathcal{E}}$ which *simultaneously* makes both the PGV_1-group and the PGV_2-group secure. Such a transformation would be of interest because incorporating it into the compression function would result in a construction that relies on a weaker assumption than either just PGV_1 or PGV_2. Consequently, it would provide a handle to *strengthen* existing schemes (in a provable way). Note that such a result would not contradict the separation of direct reducibility between the PGV_1-group and the PGV_2-group, because simultaneous security looks for a (transformed) cipher in the intersection of good (distributions over) blockciphers for both groups. This intersection is clearly nonempty because it contains the ideal cipher; the question to address here is how hard it is to hit a distribution when starting with the minimal security assumption that (a potentially non-ideal) $\mathcal{E}$ is good for at least one PGV construction. We remark our technique of separating the DBL constructions from PGV_1 does not seem to apply here, as the simultaneous security bound for PGV_1 and PGV_2 is $\Theta(q^2/2^n)$. However, surjective, backward-secure transformations are still ruled out according to the same argument as in the HDM vs. ADM case.

Another direction of research left open here is the existence of reductions among two compression functions for *different* games. For example, one might ask whether the collision resistance of one construction for a blockcipher gives preimage resistance in another (or perhaps the same) construction with the same cipher. In particular, using Simon's result [31] one might be able to demonstrate the impossibility of reducing collision resistance to preimage resistance for any of the PGV constructions.

Finally, let us emphasize that all results in this work apply directly to compression functions. Needless to say, in practice compression functions are iterated in order to hash arbitrary lengths of data. This could extend the set of $\mathcal{E}$ that provide security, potentially changing the scope for transformations between constructions. We leave the question of the existence of reductions among iterated hash functions as an interesting open problem.

Acknowledgments. We thank the anonymous reviewers for their valuable comments. The work described in this paper has been supported in part by the European Commission through the ICT programme under contract ICT-2007-216676 ECRYPT II. Paul Baecher and Pooya Farshim are supported by grant Fi 940/4-1 of the German Research Foundation (DFG). Marc Fischlin is supported by grant Fi 940/3-1 of the German Research Foundation (DFG).

References

1. Andreeva, E., Neven, G., Preneel, B., Shrimpton, T.: Seven-property-preserving iterated hashing: ROX. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 130–146. Springer, Heidelberg (2007)
2. Armknecht, F., Fleischmann, E., Krause, M., Lee, J., Stam, M., Steinberger, J.: The preimage security of double-block-length compression functions. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 233–251. Springer, Heidelberg (2011)
3. Baecher, P., Brzuska, C., Fischlin, M.: Notions of black-box reductions, revisited. Cryptology ePrint Archive, Report 2013/101 (2013), `http://eprint.iacr.org/`
4. Baecher, P., Fischlin, M.: Random oracle reducibility. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 21–38. Springer, Heidelberg (2011)
5. Bellare, M., Kohno, T.: Hash function balance and its impact on birthday attacks. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 401–418. Springer, Heidelberg (2004)
6. Bellare, M., Rogaway, P.: The security of triple encryption and a framework for code-based game-playing proofs. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 409–426. Springer, Heidelberg (2006)
7. Biryukov, A., Khovratovich, D.: Related-key cryptanalysis of the full AES-192 and AES-256. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 1–18. Springer, Heidelberg (2009)
8. Biryukov, A., Khovratovich, D., Nikolić, I.: Distinguisher and related-key attack on the full AES-256. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 231–249. Springer, Heidelberg (2009)
9. Black, J.A., Rogaway, P., Shrimpton, T.: Black-box analysis of the block-cipher-based hash-function constructions from PGV. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 320–335. Springer, Heidelberg (2002)

10. Black, J., Rogaway, P., Shrimpton, T., Stam, M.: An analysis of the blockcipher-based hash functions from PGV. Journal of Cryptology 23(4), 519–545 (2010)
11. Coron, J.-S., Dodis, Y., Malinaud, C., Puniya, P.: Merkle-damgård revisited: How to construct a hash function. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 430–448. Springer, Heidelberg (2005)
12. Dodis, Y., Ristenpart, T., Shrimpton, T.: Salvaging merkle-damgård for practical applications. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 371–388. Springer, Heidelberg (2009)
13. Ferguson, N., Lucks, S., Schneier, B., Whiting, D., Bellare, M., Kohno, T., Callas, J., Walker, J.: The skein hash function family (2008)
14. Fleischmann, E., Gorski, M., Lucks, S.: On the security of TANDEM-DM. In: Dunkelman, O. (ed.) FSE 2009. LNCS, vol. 5665, pp. 84–103. Springer, Heidelberg (2009)
15. Gauravaram, P., Knudsen, L.R., Matusiewicz, K., Mendel, F., Rechberger, C., Schläfffer, M., Thomsen, S.S.: Grøstl — a SHA-3 candidate (2011)
16. Harnik, D., Kilian, J., Naor, M., Reingold, O., Rosen, A.: On robust combiners for oblivious transfer and other primitives. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 96–113. Springer, Heidelberg (2005)
17. Herzberg, A.: On tolerant cryptographic constructions. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 172–190. Springer, Heidelberg (2005)
18. Hirose, S.: Provably secure double-block-length hash functions in a black-box model. In: Park, C.-S., Chee, S. (eds.) ICISC 2004. LNCS, vol. 3506, pp. 330–342. Springer, Heidelberg (2005)
19. Hirose, S.: Some plausible constructions of double-block-length hash functions. In: Robshaw, M. (ed.) FSE 2006. LNCS, vol. 4047, pp. 210–225. Springer, Heidelberg (2006)
20. Khovratovich, D.: New Approaches to the Cryptanalysis of Symmetric Primitives. Ph.D. thesis, University of Luxembourg (2010)
21. Kuwakado, H., Morii, M.: Indifferentiability of single-block-length and rate-1 compression functions. IEICE Transactions 90-A(10), 2301–2308 (2007)
22. Lai, X., Massey, J.L.: Hash functions based on block ciphers. In: Rueppel, R.A. (ed.) EUROCRYPT 1992. LNCS, vol. 658, pp. 55–70. Springer, Heidelberg (1993)
23. Lee, J., Kwon, D.: The security of abreast-dm in the ideal cipher model. IEICE Transactions 94-A(1), 104–109 (2011)
24. Lee, J., Stam, M., Steinberger, J.: The collision security of tandem-DM in the ideal cipher model. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 561–577. Springer, Heidelberg (2011)
25. Maurer, U.M., Renner, R.S., Holenstein, C.: Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 21–39. Springer, Heidelberg (2004)
26. Pietrzak, K.: Compression from collisions, or why CRHF combiners have a long output. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 413–432. Springer, Heidelberg (2008)
27. Preneel, B., Govaerts, R., Vandewalle, J.: Hash functions based on block ciphers: A synthetic approach. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 368–378. Springer, Heidelberg (1994)
28. Reingold, O., Trevisan, L., Vadhan, S.P.: Notions of reducibility between cryptographic primitives. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 1–20. Springer, Heidelberg (2004)
29. Rogaway, P.: Formalizing human ignorance. In: Nguyen, P.Q. (ed.) VIETCRYPT 2006. LNCS, vol. 4341, pp. 211–228. Springer, Heidelberg (2006)

30. Rogaway, P., Shrimpton, T.: Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In: Roy, B., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 371–388. Springer, Heidelberg (2004)
31. Simon, D.R.: Findings collisions on a one-way street: Can secure hash functions be based on general assumptions? In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 334–345. Springer, Heidelberg (1998)
32. Stam, M.: Blockcipher-based hashing revisited. In: Dunkelman, O. (ed.) FSE 2009. LNCS, vol. 5665, pp. 67–83. Springer, Heidelberg (2009)
33. Wei, L., Peyrin, T., Sokołowski, P., Ling, S., Pieprzyk, J., Wang, H.: On the (In)Security of IDEA in various hashing modes. In: Canteaut, A. (ed.) FSE 2012. LNCS, vol. 7549, pp. 163–179. Springer, Heidelberg (2012)

Limitations of the Meta-reduction Technique: The Case of Schnorr Signatures

Marc Fischlin[1] and Nils Fleischhacker[2]

[1] Technische Universität Darmstadt
www.cryptoplexity.de
[2] Saarland University
ca.cs.uni-saarland.de

Abstract. We revisit the security of Fiat-Shamir signatures in the non-programmable random oracle model. The well-known proof by Pointcheval and Stern for such signature schemes (Journal of Cryptology, 2000) relies on the ability to re-program the random oracle, and it has been unknown if this property is inherent. Pailler and Vergnaud (Asiacrypt 2005) gave some first evidence of the hardness by showing via meta-reduction techniques that algebraic reductions cannot succeed in reducing key-only attacks against unforgeability to the discrete-log assumptions. We also use meta-reductions to show that the security of Schnorr signatures cannot be proven equivalent to the discrete logarithm problem without programming the random oracle. Our result also holds under the one-more discrete logarithm assumption but applies to a large class of reductions, we call *single-instance* reductions, subsuming those used in previous proofs of security in the (programmable) random oracle model. In contrast to algebraic reductions, our class allows arbitrary operations, but can only invoke a single resettable adversary instance, making our class incomparable to algebraic reductions.

Our main result, however, is about meta-reductions and the question if this technique can be used to further strengthen the separations above. Our answer is negative. We present, to the best of our knowledge for the first time, limitations of the meta-reduction technique in the sense that finding a meta-reduction for general reductions is most likely infeasible. In fact, we prove that finding a meta-reduction against a potential reduction is equivalent to finding a "meta-meta-reduction" against the strong existential unforgeability of the signature scheme. This means that the existence of a meta-reduction implies that the scheme must be insecure (against a slightly stronger attack) in the first place.

1 Introduction

On a technical level, we investigate the security of Fiat-Shamir (FS) signatures [10] in the non-programmable random oracle model (NPROM), i.e., where programming the hash function is prohibited. Such programming has been exploited in the security proof for common FS signatures by Pointcheval and Stern [23], bringing forward the question if the security result remains valid in the more stringent model of non-programmable random oracles. Conceptually, though, the more interesting result in the paper refers to limitations of so-called meta-reductions. Such meta-reductions are also called "reductions against the reductions" as they basically treat the reduction as an adversary itself and reduce the existence of such a reduction to a presumably hard problem, ruling out reductions and therefore security proofs for the underlying scheme. This proof technique recently gained quite some attention as it rules out certain reductions, especially those which only treat the adversary but not the underlying primitive as a black box (e.g., [9,20,15,14,21,16,2,27]). We show, via a "meta-meta-reduction", that one cannot use the meta-reduction technique to show impossibility results for FS signatures in the NPROM.

1.1 Fiat-Shamir Signatures in the NPROM

The class of FS signatures comprises all transformed three-move identification schemes in which the challenge $\textsc{ch}$, sent by the verifier in return of the prover's initial commitment $\textsc{com}$, is replaced by the hash value $H(\textsc{com}, m)$ for message m. The prover's response $\textsc{resp}$, together with $\textsc{com}$, then yields the signature $(\textsc{com}, \textsc{resp})$ for m. For some cases, like the Schnorr signature scheme [26], the signature can be shortened by using $(\textsc{ch}, \textsc{resp})$ instead.

The common security proof for FS signature schemes in the random oracle model [6], given in [23], basically works as follows. The reduction to the underlying problem, such as the discrete logarithm problem for the Schnorr scheme, runs the adversary twice. In the first runs the reduction gets a signature forgery $(\textsc{com}, \textsc{resp})$ for message m and challenge $\textsc{ch} = H(\textsc{com}, m)$. In the second run it re-programs H to yield a distinct challenge $\textsc{ch}' = H(\textsc{com}, m)$ and response $\textsc{resp}'$. From both signatures the reduction can then compute a solution to the underlying problem. Clearly, this technique relies on the programmability of the hash function.[1]

Fischlin et al. [13] later defined reductions in the non-programmable random oracle model (NPROM) by externalizing the hash function to both the adversary *and* the reduction.[2] In the NPROM the reduction may still observe the adversary's queries to the hash function, but cannot change the reply. Obviously, this non-programming property matches much closer our understanding of "real" hash functions and instantiations through, say, SHA-3. Interestingly, though, Fischlin et al. [13] do not investigate this arguably most prominent application of the random oracle methodology. Instead, they separate programming and non-programming reductions (and an intermediate notion called weakly programming reductions) through the case of OAEP encryption, FDH signatures,

[1] Note that this proof reduces the security of the signature scheme to the underlying number-theoretic problem via special soundness. Abdalla et al. [1] more generally consider FS schemes with reductions to the identification schemes. We do not cover the latter type of reductions and schemes here.

[2] The role of programmability was first investigated by Nielsen [18], even though not for reductions as in the proofs of Fiat-Shamir schemes.

and trapdoor permutation based KEMs. Weakly programming reductions are allowed to reset the random oracle and redirect the value to some (external) random answer. Our first result is to formally confirm the intuition that FS signatures should still be secure in the weakly programmable random oracle model.

1.2 Limitations Through Meta-Reductions

The more interesting question is if FS signatures can be shown to be secure in the NPROM. Our first result in this regard is negative and applies to discrete log schemes like the Schnorr signature scheme [26] or the RSA-based Guillou-Quisquater scheme [17]. Namely, we first consider any reduction $\mathcal{R}$ which initiates only a single (black-box) instance of the adversary $\mathcal{A}$ for some public key pk, but such that it can reset $\mathcal{A}$ arbitrarily to the point after having handed over the public key (from the fixed group). Note that the reduction in the *programmable* ROM in [23] is of this kind, only that it can also change the behavior of the random oracle, unlike our reductions here in the NPROM. We show that this type of *single-instance* reduction to the discrete log problem cannot succeed in the NPROM under the one-more discrete log assumption [5].

Our impossibility result follows from presenting a meta-reduction $\mathcal{M}$ against $\mathcal{R}$. That is, we show that if one can find a reduction $\mathcal{R}$ which successfully solves the DL problem given black-box access to any successful adversary $\mathcal{A}$ against the signature scheme, then there is a meta-reduction $\mathcal{M}$ breaking the one-more DL problem directly. Since we also present a successful (unbounded) adversary $\mathcal{A}$ which $\mathcal{M}$ can simulate towards $\mathcal{R}$ efficiently, we conclude that the existence of reduction $\mathcal{R}$ would already contradict the one-more DL problem.

We observe that our meta-reduction, too, works in the NPROM and thus cannot program the random oracle for $\mathcal{R}$; else the meta-reduction would violate the idea of modeling hash functions as non-programmable. It is also easy to show that, if the meta-reduction, unlike the reduction, was allowed to program the random oracle, this "unfair" situation would straightforwardly dismiss the possibility of such reductions. However, such an approach seems to violate the idea behind non-programmable oracles as a mean to capture real-world hash functions over which no party, not even the meta-reduction, has control.

The noteworthy property of our meta-reduction $\mathcal{M}$ is that, unlike most of the previous proposals (cf. [11]), it does not work by resetting the reduction $\mathcal{R}$. The reset strategy is usually used to rewind the reduction and, in case of signature schemes, get an additional signature through a signing query in an execution branch, and display this signature back to $\mathcal{R}$ as a forgery in the main branch. However, this means that one needs to take care of correlations between the additional signature and the reduction's state. Instead of using such resets, our meta-reduction will essentially run two independent copies of the reduction and use the signatures of one execution in the other one. The independence of the executions thus "decorrelates" the additional signature from the reduction's state, avoiding many complications from the resetting strategy.

1.3 Limitations of Meta-reductions

Does our meta-reduction impossibility result for non-programming reductions extend to other cases like the discrete logarithm problem? We show that this is unlikely, thus showing limitations of the meta-reduction technique. The idea is to consider the meta-reduction itself as a reduction, and to use the meta-reduction technique against this reduction. Hence, we obtain a "meta-meta-reduction" $\mathcal{N}$ which now simulates the reduction $\mathcal{R}$ for $\mathcal{M}$, just as the meta-reduction simulates the adversary for $\mathcal{R}$.

More concretely, assume that we consider reductions $\mathcal{R}$ transforming an adversary $\mathcal{A}$ against the signature scheme in a black-box way into a solver for some cryptographic problem $\Pi_{\mathcal{R}}$. Then, a meta-reduction $\mathcal{M}$ should turn $\mathcal{R}$ (to which it has black-box access) into a successful solver for some problem $\Pi_{\mathcal{M}}$. For technical reasons, in our case this problem $\Pi_{\mathcal{M}}$ has to be non-interactive, e.g., correspond to the discrete logarithm problem; this also circumvents the case of our previous meta-reduction for the interactive one-more DL problem. Then we show that such a meta-reduction can be used to build a meta-meta-reduction $\mathcal{N}$ against the strong unforgeabilty of the signature scheme.

In other words, the meta-reduction technique cannot help to rule out black-box reductions to arbitrary problems, unless the signature is insecure in the first place. Here, insecurity refers to the notion of strong unforgeability where the adversary also succeeds by outputting a new signature to a previously signed message. In fact, in the programmable ROM the security proof in [23] actually shows that the FS schemes achieve this stronger notion.

1.4 Related Work

As mentioned before, meta-reductions have been used in several recent results to rule out black-box reductions for Fiat-Shamir schemes, and especially for Schnorr signatures. Paillier and Vergnaud [20] analyzed the security of Schnorr Signatures in the standard model. They showed with the help of meta-reductions that, if the one-more discrete logarithm assumption holds, the security of Schnorr signatures cannot be reduced to the (one-more) discrete logarithm problem, at least using algebraic reductions. While algebraic reductions where first defined by Boneh and Venkatesan [7], Paillier and Vergnaud [20], however, use a slightly more liberal definition of algebraicity. Their notion basically states that, given the discrete logarithm of all of the reduction's inputs and access to the reduction, it is possible to compute the discrete logarithm of any group element output by the reduction. We note that the ability to trace the discrete logarithms of the group elements produced by the reduction is important to their result and allows them to prove impossibility even for key-only attacks.

Paillier and Vergnaud [20] also extended their result to other signature schemes, including the Guillou-Quisquater scheme [17] and the one-more RSA assumption [5]. They also considered the tightness loss in the Pointcheval-Stern proof for the Schnorr signature scheme in the programmable random oracle model. They showed, again for algebraic reductions, that the security loss of a factor $\sqrt{q_H}$ is inevitable, where q_H is the maximum number of random oracle queries by the adversary. This bound was later raised to $q_H^{2/3}$ by Garg et al. [15] in the same setting. Seurin [27] recently improved this bound further to $\mathcal{O}(q_H)$. Using meta-reduction techniques and considering algebraic reductions, too, he proved it is unlikely that a tighter reduction exists.

In a recent work, Baldimtsi and Lysyanskaya [4] showed, via meta-reductions, that one cannot prove blind Schnorr signatures secure via black-box reductions. Their meta-reduction, like ours here, has the interesting feature of being non-resetting. Remarkably, though, they seem to rule out the more liberal *programming* reductions, whereas our result is against the "more confined" *non-programming* reductions. However, their result considers a special type of programming reduction, called naive. This roughly means that one can predict the reduction's programmed random oracle answers by reading the reduction's random tape. This property is inherently tied to the programmability and is clearly not fulfilled by non-programmable, external random oracles; for such oracles even the reduction does not know the answers in advance. This, unfortunately, also means that their meta-reduction technique may not apply to non-programmable hash functions. In other words, one may be able to bypass their impossibility result and may still be able to find a cryptographic security proof for such schemes, by switching to the non-programmable random oracle model, or even to standard-model hash functions.

1.5 Organization

In Section 2 we first recall some basic facts about signatures and (general and discrete-log specific) cryptographic problems. Then we show that FS signatures are secure in the weakly programmable random oracle model, and prove our meta-reduction impossiblity result for single-instance reductions in the NPROM in Section 3. Our main result about meta-meta-reductions appears in Section 4.

2 Preliminaries

We use standard notions for digital signature schemes $\mathcal{S} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vrfy})$ such as existential unforgeability and strong existential unforgeability. We usually assume (non-trivially) randomized signature schemes, where the signature algorithm has super-logarithmic min-entropy for the security parameter κ, i.e., $H_\infty(\mathsf{Sign}(\mathsf{sk}, m)) \in \omega(\log(\kappa))$ for all keys sk, all messages m, and given the random oracle. For formal definitions refer to the full version of this paper.

2.1 Cryptographic Problems

We define a cryptographic problem as a game between a challenger and an adversary. The challenger uses an instance generator to generate a fresh instance of the problem. The adversary is then supposed to find a solution for said instance. The challenger may assist the adversary by providing access to some oracle, like a decryption oracle in a chosen-ciphertext attack against indistinguishability. Eventually the adversary outputs a solution for the problem instance and the challenger uses a verification algorithm to check whether the solution is correct.

For many problems there exist trivial adversaries, e.g., succeeding in an indistinguishability game by pure guessing. One is usually interested in the advantage of adversaries beyond such trivial strategies. We therefore introduce a so-called *threshold algorithm* to cover such trivial attacks and measure any adversary against this threshold adversary.

Definition 1 (Cryptographic Problem). *A cryptographic problem* $\Pi = (\mathsf{IGen}, \mathsf{Orcl}, \mathsf{Vrfy}, \mathsf{Thresh})$ *consists of four algorithms:*

- *The instance generator* IGen *takes as input the security parameter* 1^κ *and outputs a problem instance* z*. The set of all possible instances output by* IGen *is called* $\mathbf{Inst}$*.*
- *The computationally unbounded and stateful oracle algorithm* Orcl *takes as input a query* $q \in \{0,1\}^*$ *and outputs a response* $r \in \{0,1\}^*$ *or a special symbol* $\perp$ *indicating that* q *was not a valid query.*
- *The deterministic verification algorithm* Vrfy *takes as input a problem instance* $z \in \mathbf{Inst}$ *and a candidate solution* $x \in \mathbf{Sol}$*. The algorithm outputs* $b \in \{0,1\}$*. We say* x *is a valid solution to instance* z *if and only if* $b \overset{?}{=} 1$*.*
- *The efficient threshold algorithm* Thresh *takes as input a problem instance* z *and outputs some* x*. The threshold algorithm is a special adversary and as such also has access to* Orcl*.*

We note that the algorithms IGen*,* Orcl*,* Vrfy *potentially have access to shared state that persists for the duration of an experiment.*

Definition 2 (Hard Cryptographic Problem). *For a cryptographic problem* $\Pi = (\mathsf{IGen}, \mathsf{Orcl}, \mathsf{Vrfy}, \mathsf{Thresh})$ *and an adversary* $\mathcal{A}$ *we define the following experiment:*

$$\mathsf{Exp}_\Pi^{\mathcal{A}}(\kappa) : [z \leftarrow \mathsf{IGen}(1^\kappa); x \leftarrow \mathcal{A}^{\mathsf{Orcl}}(z); b \leftarrow \mathsf{Vrfy}(z,x); \text{output } b].$$

The problem Π *is said to be* hard *if and only if for all probabilistic polynomial-time algorithms* $\mathcal{A}$ *the following advantage function is negligible in the security parameter* κ*:*

$$\mathsf{Adv}_\Pi^{\mathcal{A}}(\kappa) = \Pr\left[\mathsf{Exp}_\Pi^{\mathcal{A}}(\kappa) \overset{?}{=} 1\right] - \Pr\left[\mathsf{Exp}_\Pi^{\mathsf{Thresh}}(\kappa) \overset{?}{=} 1\right],$$

where the probability is taken over the random tapes of IGen *and* $\mathcal{A}$*.*

We sometimes require some additional properties of cryptographic problems, summarized in the following definition:

Definition 3 (Specific Cryptographic Problems). *Let* $\Pi = (\mathsf{IGen}, \mathsf{Orcl}, \mathsf{Vrfy}, \mathsf{Thresh})$ *be a cryptographic problem as defined in Definition 1.*

- *The problem* Π *is said to be* non-interactive *if and only if* $\Pi.\mathsf{Orcl}$ *is the algorithm that always outputs* $\perp$ *and never changes the shared state.*

– *The problem Π is said to be* efficiently generatable *if and only if $\Pi.\mathsf{IGen}$ is a polynomial-time algorithm.*
– *The problem Π is said to be* solvable *if and only if $\Pi.\mathbf{Sol}$ is recursively enumerable, and the following holds:*

$$\forall z \leftarrow \Pi.\mathsf{IGen}(1^\kappa) : (\exists x \in \Pi.\mathbf{Sol} : \Pi.\mathsf{Vrfy}(z, x) \stackrel{?}{=} 1).$$

– *The problem Π is said to be* monotone *if and only if for all instances $z \leftarrow \Pi.\mathsf{IGen}(1^\kappa)$, all solutions $x \in \Pi.\mathbf{Sol}$, all $n \in \mathbb{N}$, and all sequences of queries $(q_1, \ldots, q_n)$ the following holds: If $\Pi.\mathsf{Vrfy}(z, x) \stackrel{?}{=} 1$ holds after executing the queries $\Pi.\mathsf{Orcl}(q_1); \ldots; \Pi.\mathsf{Orcl}(q_n)$, then this already held before $\Pi.\mathsf{Orcl}(q_n)$ was executed.*

Intuitively, an algorithm solving a monotone problem is not punished for issuing fewer queries. In particular, if a solution is valid after some sequence of queries, it is also valid if no queries were executed at all.

2.2 Discrete Logarithm Assumptions

The discrete logarithm problem with its corresponding hardness assumption is a specific instance of a non-interactive, efficiently generatable, and solvable problem. The assumption about the computational infeasibility of computing logarithms in certain groups is formally defined in Definition 4.

Definition 4 (Discrete Logarithm Assumption). *Let $\mathbb{G} = \langle g \rangle$ be a group of prime order q with $|q| = \kappa$. The* discrete logarithm (DL) *problem over $\mathbb{G}$ —written $\mathsf{DL}_\mathbb{G}$— is defined as follows:*

Instance and Solution space: *The instance space* **Inst** *is $\mathbb{G}$ and the solution space* **Sol** *is $\mathbb{Z}_q$.*

Instance Generation: *The instance generator $\mathsf{IGen}(1^\kappa)$ chooses $z \xleftarrow{\$} \mathbb{G}$ and outputs z. Note that this sampling of z may require to pick a random $w \xleftarrow{\$} \mathbb{Z}_q$ and compute $z = g^w$.*

Verification: *The verification algorithm $\mathsf{Vrfy}(z, x)$ computes $z' = g^x$. If $z' \stackrel{?}{=} z$, then it outputs 1, otherwise it outputs 0.*

Threshold: *The threshold algorithm $\mathsf{Thresh}(z)$ chooses $x \xleftarrow{\$} \mathbb{Z}_q$ and outputs x.*

The discrete logarithm *assumption is said to hold over $\mathbb{G}$ if $\mathsf{DL}_\mathbb{G}$ is hard.*

A natural extension of the discrete logarithm problem are the interactive, efficiently generatable, monotone, and solvable *one-more discrete logarithm* problems first introduced by Bellare et al. [5]. They are interactive, as the adversary is given access to an oracle capable of solving the $\mathsf{DL}_\mathbb{G}$ problem. However, an adversary computing $n + 1$ discrete logarithms can only request at most n discrete logarithms from the DL oracle, hence, the name *one-more* discrete-log problem. The problems with their corresponding hardness assumptions are formally described in Definition 5. The assumptions are believed to be stronger than the regular DL assumption [8].

Definition 5 (n-One-More Discrete Logarithm Assumption [5]). *Let $\mathbb{G} = \langle g \rangle$ be a group of prime order q with $|q| = \kappa$. The* n-one-more discrete logarithm (n-DL) *problem over $\mathbb{G}$ –written n-$\mathsf{DL}_{\mathbb{G}}$– is defined as follows:*

Instance and Solution space: *The instance space* **Inst** *is $\mathbb{G}^{n+1}$ and the solution space* **Sol** *is $\mathbb{Z}_q^{n+1}$.*

Shared State: *The shared state consists only of a single counter variable i.*

Instance Generation: *The instance generator $\mathsf{IGen}(1^\kappa)$ initializes $i := 0$ in the shared state, chooses $z_0, \ldots, z_n \xleftarrow{\$} \mathbb{G}$, and outputs $(z_0, \ldots, z_n)$.*

Oracles: *The oracle algorithm $\mathsf{Orcl}(z)$, on input $z \in \mathbb{G}$, increments $i := i + 1$. It then exhaustively searches $\mathbb{Z}_q$ for an x such that $g^x \overset{?}{=} z$ and outputs x. On input some $z \notin \mathbb{G}$, Orcl outputs $\perp$.*

Verification: *The verification algorithm $\mathsf{Vrfy}((z_0, \ldots, z_n), (x_0, \ldots, x_n))$ computes $z'_j = g^{x_j}$. If $z'_j \overset{?}{=} z_j$ for all j and if $i \leq n$, then it outputs 1, otherwise it outputs 0.*

Threshold: *The threshold algorithm $\mathsf{Thresh}(z)$ chooses $x_0, \ldots, x_n \xleftarrow{\$} \mathbb{Z}_q$ and outputs $(x_0, \ldots, x_n)$.*

The n-one-more discrete logarithm (n-DL) *assumption is said to hold over $\mathbb{G}$, if and only if the n-$\mathsf{DL}_{\mathbb{G}}$ problem is hard.*

3 Security of Schnorr Signatures

We first recall the definition of the Schnorr signature scheme (SSS) [25,26] as derived from the Schnorr identification scheme via the Fiat-Shamir transform [10]. Afterwards, we analyze the security of the resulting signature scheme in two variants of the random oracle model, in which reductions are limited in the way they can program the random oracle.

Definition 6 (Schnorr Signature Scheme). *Let $\mathbb{G}$ be a cyclic group of prime order q with generator g and let $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_q$ be a hash function modeled as a random oracle. The Schnorr signature scheme, working over $\mathbb{G}$, is defined as follows:*

Key Generation: *The key generation algorithm $\mathsf{KGen}(1^\kappa)$ proceeds as follows: Pick $\mathsf{sk} \xleftarrow{\$} \mathbb{Z}_q$, compute $\mathsf{pk} := g^{\mathsf{sk}}$, and output $(\mathsf{sk}, \mathsf{pk})$.*

Signature Generation: *The signing algorithm $\mathsf{Sign}(\mathsf{sk}, m; r)$ proceeds as follows: Use $r \in \mathbb{Z}_q$ and compute $R := g^r$. Compute $c := \mathcal{H}(R, m)$ and $y := r + \mathsf{sk} \cdot c \mod q$. Output $\sigma := (c, y)$.*

Signature Verification *The verification algorithm $\mathsf{Vrfy}(\mathsf{pk}, m, \sigma)$ proceeds as follows: Parse σ as (c, y). If $c \overset{?}{=} \mathcal{H}(\mathsf{pk}^{-c} g^y, m)$, then output 1, otherwise output 0.*

3.1 Unforgeability of Schnorr Signatures under Randomly Programming Reductions

We begin by showing that the original proof by Pointcheval and Stern [22,23] still holds for randomly programming reductions. Randomly programming reductions as defined in [13] do not simulate the random oracle themselves. Instead, they can re-set the random oracle to another hash value. As shown in [13] such randomly programming reductions are equivalent to the weakly-programmable random oracle model (WPROM) which is in between the programmable and non-programmable ROM. Whereas a conventional random oracle has only a single interface implementing a random mapping from domain **Dom** to range **Rng**, a weakly programmable random oracle has three interfaces, which allow for programming but only in a weak sense: one cannot freely re-program the hash values but only re-set them to another random value:

Definition 7 (Weakly Programmable Random Oracle). *A* weakly programmable random oracle *(WPRO) exposes three interfaces to the caller:*

Evaluation: *The evaluation interface* RO_{eval} *behaves as a conventional random oracle, mapping* $\mathbf{Dom} \rightarrow \mathbf{Rng}$.

Random: *The random interface* RO_{rand} *takes as input bit strings of arbitrary length and implements a random mapping* $\{0,1\}^* \rightarrow \mathbf{Rng}$.

Programming: *The programming interface* RO_{prog} *takes as input a pair* $(a, b) \in \mathbf{Dom} \times \{0,1\}^*$ *and programs* $\mathsf{RO}_{eval}(a)$ *to evaluate to* $\mathsf{RO}_{rand}(b)$.

The randomly programming reduction gets oracle access to all three interfaces, whereas the adversary only gets access to the RO_{eval} interface. We now show that randomly programming reductions are sufficient to prove SSS secure in the ROM.

Theorem 1 (EUF-CMA Security of SSS Under Randomly Programming Reductions). *The* EUF-CMA *security of* SSS *is reducible to the* discrete logarithm *problem over* $\mathbb{G}$ *using a randomly programming reduction* $\mathcal{R}$.

The proof is close to the one in the programmable case and omitted here; the reader may refer to the full version for a sketch. We thus show that the limited programmability of a randomly programming random oracle is sufficient to obtain a (loose) proof of security for Schnorr signatures. In particular, choosing range points of the random oracle at will is not required for the proof. We note that the above result transfers to other FS schemes such as [19,17,12].

3.2 Schnorr Signatures are not Provably Secure under Non-programming Single-Instance Reductions

We now show that the Schnorr Signature Scheme cannot be proven existentially unforgeable under chosen message attacks without programming the random oracle —at least with respect to a slightly restricted type of reduction. We actually prove that, if such a reduction exists, the 1-one-more discrete logarithm assumption does not hold over $\mathbb{G}$.

We term the restricted class of reductions as *single-instance reductions.* Such single-instance reductions only invoke a single instance of the adversary and, while they may rewind the adversary, they may not rewind it to a point before it received the public key for the first time. This class of reductions is especially relevant, because both the original security reduction by Pointcheval and Stern [23] as well as the one in Theorem 1 are of this type.

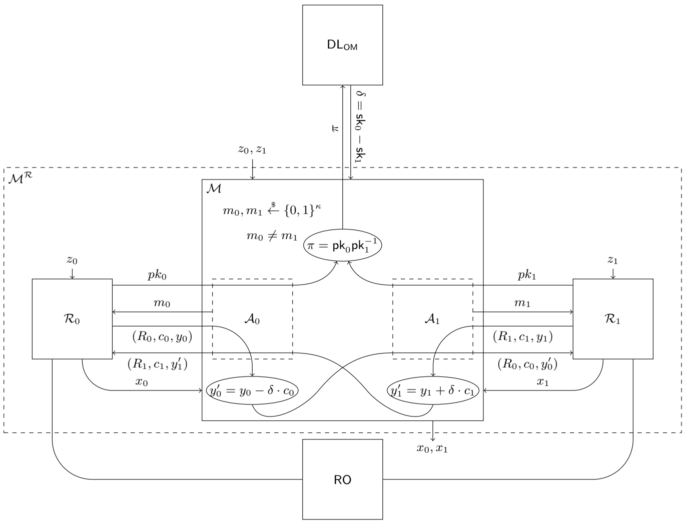
Instead of simulating the random oracle itself, a non-programming reduction works relative to an external fixed random function and it is required to honestly answer all random oracle queries. That is, the black-box reduction can observe the adversary's queries to the random oracle, but cannot change the answers. We omit a formal approach (see [13]) because the definition reflects the intuition straightforwardly. We remark that the approach assumes fully-black-box reductions [24] (or, in terms of the CAP taxonomy of [3], the BBB-type of reduction) which need to work for any (unbounded) adversary oracle. In particular, and we will in fact exploit this below, the adversary can thus depend on the reduction. We may therefore think of the adversary as a family $\mathbb{A}$ of adversaries $\mathcal{A}_{\mathcal{R},a}$, depending on the reduction $\mathcal{R}$ and using some randomness a. We believe it is conceptually easier in this case here to make the randomness a explicit, as opposed to having a single adversary that internally chooses a at the beginning of the execution. It is nonetheless sometimes convenient to omit these subindices and to simply write $\mathcal{A}$.

Theorem 2 (Non-Programming Irreducibility for SSS). *Assume that the* 1-one-more discrete logarithm *assumption holds over* $\mathbb{G}$. *There exists no non-programming single-instance fully-black-box reduction that reduces the* EUF-CMA *security of* SSS *over* $\mathbb{G}$ *to the discrete logarithm problem over* $\mathbb{G}$.

More precisely, assume there exists a non-programming single-instance fully-black-box reduction $\mathcal{R}$ *that converts any adversary* $\mathcal{A}$ *against the* EUF-CMA *security of* SSS *working in group* $\mathbb{G}$ *into an adversary against the* DL *problem over* $\mathbb{G}$. *Assume further that the reduction has success probability* $\mathsf{Succ}_{\mathsf{DL},\mathbb{G}}^{\mathcal{R}^{\mathcal{A}}}(\kappa)$ *for given* $\mathcal{A}$ *and runtime* $\mathsf{Time}_{\mathcal{R}}(\kappa)$. *Then, there exists a family* $\mathbb{A}$ *of successful (but possibly inefficient) adversaries* $\mathcal{A}_{\mathcal{R},a}$ *against the* EUF-CMA *security of* SSS *and a meta-reduction* $\mathcal{M}$ *that breaks the* 1-DL *assumption over* $\mathbb{G}$ *with non-negligible success probability* $\mathsf{Succ}_{\text{1-DL},\mathbb{G}}^{\mathcal{M}}(\kappa) \geq (\mathsf{Succ}_{\mathsf{DL},\mathbb{G}}^{\mathcal{R}^{\mathcal{A}_{\mathcal{R},a}}}(\kappa))^2$ *for a random* $\mathcal{A}_{\mathcal{R},a} \in \mathbb{A}$ *and runtime* $\mathsf{Time}_{\mathcal{M}}(\kappa) = 2 \cdot \mathsf{Time}_{\mathcal{R}}(\kappa) + \mathsf{poly}(\kappa)$.

Note that the fact that $\mathcal{A}$ breaks SSS working over $\mathbb{G}$ implies that for any public key pk output by $\mathcal{R}$ it holds that $\mathsf{pk} \in \mathbb{G}$.

Proof. (Sketch) Roughly, the meta-reduction $\mathcal{M}$ – depicted in Figure 1 – with inputs z_0, z_1 works as follows: It invokes two instances $\mathcal{R}_0$ and $\mathcal{R}_1$ of the reduction in a black-box way, on inputs z_0 and z_1, respectively, and independent random tapes. When the instances of $\mathcal{R}$ invoke the forger with public key pk_0 and pk_1 respectively, $\mathcal{M}$ simulates a specific (inefficient) forger. To do so, the meta-reduction queries random messages to the sign oracles and obtains signatures on them. It then queries the quotient of the two public keys, i.e., $\mathsf{pk}_0\mathsf{pk}_1^{-1}$, to the



$\mathsf{DL}_{\mathsf{OM}}$
π
$\delta = \mathsf{sk}_0 - \mathsf{sk}_1$
z_0, z_1
$\mathcal{M}^{\mathcal{R}}$
$\mathcal{M}$
$m_0, m_1 \xleftarrow{\$} \{0,1\}^\kappa$
$m_0 \neq m_1$
$\pi = \mathsf{pk}_0\mathsf{pk}_1^{-1}$
z_0
z_1
pk_0
pk_1
m_0
m_1
$\mathcal{R}_0$
$\mathcal{A}_0$
$\mathcal{A}_1$
$\mathcal{R}_1$
(R_0, c_0, y_0)
(R_1, c_1, y_1)
(R_1, c_1, y_1')
(R_0, c_0, y_0')
x_0
x_1
$y_0' = y_0 - \delta \cdot c_0$
$y_1' = y_1 + \delta \cdot c_1$
x_0, x_1
RO

Fig. 1. The meta-reduction uses two instances of $\mathcal{R}$ and simulates the adversary $\mathcal{A}$ by obtaining the difference between the secret keys and adapting the signatures output by $\mathcal{R}$ to the other key, respectively.

$\mathsf{DL}_{\mathsf{OM}}$ oracle, thus obtaining the difference between the secret keys. The difference between the secret keys can then be used to adapt the obtained signatures to the other public key, respectively. These adapted signatures are then returned to the reductions as forgeries. As we are working in the (non-programmable) random oracle model, the instances of $\mathcal{R}$ expect to see all the random oracle queries, the (simulated) adversary would issue. The meta-reduction $\mathcal{M}$ therefore makes sure to issue exactly those queries. Then the meta-reduction mimics the behavior of the adversary closely, and succeeds in solving the 1-one-more DL problem. □

Remark 1. Note that the restriction to single-instance reductions is crucial at this point. Consider a reduction that would output a second public key, either by invoking another instance of $\mathcal{A}$ or by rewinding the adversary to a point before it received the public key. The meta-reduction would then need to issue another query to the $\mathsf{DL}_{\mathsf{OM}}$ oracle to simulate the signing oracle. Obviously, $\mathcal{M}$ would then have made 2 queries to the $\mathsf{DL}_{\mathsf{OM}}$ oracle and could, thus, no longer win in the 1-DL experiment.

Remark 2. It should be noted, that the meta-reduction employed in the proof of Theorem 2 only works because SSS is defined relative to a single fixed random oracle. If one uses a common variant of the Fiat-Shamir transform, in which the random oracle is "personalized" by including the public key in the hash query, $c = H(\mathsf{pk}, R, m)$, the meta-reduction no longer works. This is due to the fact that in this case signatures can no longer be simply adapted to another public key, using only the secret keys' difference.

Remark 3. The idea immediately applies to other FS signature schemes with unique keys, where there is a related one-more problem, such as the RSA-GQ scheme [17].

4 Limitations of the Meta-reduction Technique

Paillier and Vergnaud [20], as well as we here, have used meta-reductions to provide evidence that, once we drop programmability, the security of Schnorr signatures might not be equivalent to the discrete log problem after all. However, it is interesting to note that in both cases the meta-reduction-based proofs rely on the one-more discrete log assumption. As the discrete log assumption does not seem to imply its one-more variants [8] the results are, thus, conditional and not as strong as they could be. The obvious question is therefore: "Can we do better?" Unfortunately, the answer turns out to be "Not without finding an actual adversary."

Our results actually holds for *any* randomized signature scheme $\mathcal{S}$ (where, as explained in Section 2, the signing algorithm has super-logarithmic min entropy) for which the signing algorithm's hash queries in any signature generation can always be reconstructed from the signature alone, in the right order. We call them randomized signature schemes *with reconstructible hash queries*, for a formal definition refer to the full version. These schemes include Fiat-Shamir transformed schemes such as Schnorr but also cover (randomized versions of) FDH-RSA signatures. We show that finding a meta-reduction to a non-interactive problem such as the discrete log problem is at least as hard as finding an adversary against the strong existential unforgeability of $\mathcal{S}$. For this, we first describe an inefficient reduction $\mathcal{R}$ that is capable of detecting when the forgery it receives is actually one of the signatures it produced itself as an answer to a signing query. For example, a meta-reduction may make several signing requests to a reduction and then reset these requests in order to use one additional message-signature pairs as the forgery. Our reduction will be able to spot such attempts.

The meta-reduction result of the previous section does not apply here, even though our reduction here will be of the single-instance type. The reason is that the meta-reduction there assumed an (interactive) one-more DL problem –and made use of the DL oracle– whereas the meta-reduction here should work for non-interactive problems such as the discrete log problem.

4.1 An Inefficient Reduction for Randomized Signature Schemes with Reconstructible Hash Queries

Let $\mathcal{S}$ be a randomized signature scheme and let $\Pi_{\mathcal{R}}$ be a monotone solvable problem. Let $\mathbb{Q}$ be the set of message-signature pairs (m_i, σ_i) resulting from queries to $\mathcal{R}$'s signing oracle. Furthermore, let p be the maximum number of signature queries issued by a forger $\mathcal{A}$ and assume that $\mathcal{R}$ knows the polynomially bounded p. We note that for the adversary in our single-instance reductions in the previous section, the reduction could have been given $p = 1$, too.

For the moment, the reader may think of the meta-reduction as running a single instance of the reduction; we will later reduce the multi-instance case to the single-instance case via standard "guess-and-insert" techniques.

The reduction $\mathcal{R}$ on input an instance z of $\Pi_{\mathcal{R}}$ first generates a key pair $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathcal{S}.\mathsf{KGen}(1^\kappa)$, then initializes the counter variable $i := 0$, and chooses a random function $\mathcal{O} : \{0,1\}^{2^\kappa} \times \mathbb{Z}_p \to \mathit{Coins}_{\mathsf{pk}}$. The public key pk is then output as the key under which the forger is supposed to forge a signature. When the forger queries a message m to the signing oracle, $\mathcal{R}$ determines random coins $\omega \leftarrow \mathcal{O}(m, i)$, computes the signature as $\sigma \leftarrow \mathcal{S}.\mathsf{Sign}(\mathsf{sk}, m; \omega)$, and returns σ to the forger. The counter i is then incremented by one. If the counter is ever incremented to $p+1$, then $\mathcal{R}$ aborts, as it is obviously not interacting with the real adversary.

Eventually, the forger outputs a forgery (m^*, σ^*). If the signature does not verify, i.e., $\mathcal{S}.\mathsf{Vrfy}(\mathsf{pk}, m^*, \sigma^*) \overset{?}{=} 0$, then $\mathcal{R}$ immediately aborts. Otherwise, the reduction computes $\sigma_j \leftarrow \mathcal{S}.\mathsf{Sign}(\mathsf{sk}, m^*; \omega_j)$ with $\omega_j \leftarrow \mathcal{O}(m^*, j)$ for all $j \in \mathbb{Z}_p$ and checks whether $\sigma_j \overset{?}{=} \sigma^*$. If the check holds for any σ_j, then $\mathcal{R}$ also immediately aborts. Otherwise, $\mathcal{R}$ enumerates all possible solutions $x \in \Pi_{\mathcal{R}}.\mathbf{Sol}$ and checks whether $\Pi_{\mathcal{R}}.\mathsf{Vrfy}(x, z) \overset{?}{=} 1$. Once such an x is found, it is output by $\mathcal{R}$ as the solution. Because $\Pi_{\mathcal{R}}$ is monotone and solvable, it is guaranteed that there exists a valid solution even though $\mathcal{R}$ never issues a single oracle query and that the enumeration of possible solutions will terminate in finite time.

Observe that the adversary $\mathcal{A}$ used by $\mathcal{R}$ is an $\mathsf{EUF\text{-}CMA}$ adversary, therefore, whenever $\mathcal{A}$ forges successfully, it forges a signature for a message m^* that has not been queried before. The probability that $\mathcal{R}$ will reject such a forgery is the probability that at least one of the σ_i collides with σ^*. As $\mathcal{O}$ is a random function, all values to which $\mathcal{O}$ evaluates on input m^* and some number i are uniformly and independently distributed. For each σ_i, the probability that it matches σ^* is thus bounded through the min-entropy of the random variable describing $\mathcal{S}.\mathsf{Sign}(\mathsf{sk}, m^*)$, i.e., $\forall i \in \mathbb{Z}_p : (\Pr[\sigma_i \overset{?}{=} \sigma^*] \leq 2^{-H_\infty(\mathcal{S}.\mathsf{Sign}(\mathsf{sk}, m^*))})$.

Therefore, the probability that $\mathcal{R}$ will accept a forgery is at least $1 - p \cdot 2^{-H_\infty(\mathcal{S}.\mathsf{Sign}(\mathsf{sk}, m^*))}$. As $\mathcal{S}$ is randomized, the probability for each σ_i to match is negligible and thus

$$\mathsf{Succ}_{\Pi_{\mathcal{R}}}^{\mathcal{R}^{\mathcal{A}}}(\kappa) \geq (1 - p \cdot \epsilon(\kappa)) \cdot \mathsf{Succ}_{\mathsf{EUF\text{-}CMA}}^{\mathcal{S},\mathcal{A}}(\kappa) = \mathsf{Succ}_{\mathsf{EUF\text{-}CMA}}^{\mathcal{S},\mathcal{A}}(\kappa) - \epsilon'(\kappa)$$

for negligible functions ϵ, ϵ'. Therefore, we conclude that $\mathsf{Succ}_{\Pi}^{\mathcal{R}^{\mathcal{A}}}(\kappa)$ is non-negligible for any successful adversary $\mathcal{A}$ and that $\mathcal{R}$ is, thus, a successful –albeit inefficient– reduction from problem $\Pi_{\mathcal{R}}$ to the $\mathsf{EUF\text{-}CMA}$ security of $\mathcal{S}$.

We next show that the checks of our reduction prevent the meta-reduction to replay signatures to the reduction. This step relies on the fact that the meta-reduction can only use the reduction in a black-box way, $\mathcal{M}^{\mathcal{R}}$, and has for example no control over the coin tosses of $\mathcal{R}$. First, we show that we can restrict ourselves to meta-reductions which actually take advantage of the reduction, at least if the meta-reduction's problem $\Pi_{\mathcal{M}}$ is hard:

Lemma 1 (Meta-Reductions Rely on the Reduction). *Let $\mathcal{M}$ be a non-programming meta-reduction that converts any $(\mathsf{EUF\text{-}CMA}_{\mathcal{S}} \rightsquigarrow \Pi_{\mathcal{R}})$ reduction in a black-box way into an adversary against some hard problem $\Pi_{\mathcal{M}}$. Further, let the reduction used by $\mathcal{M}$ be $\mathcal{R}$ as described above. Then it holds that $\mathcal{M}$ provides $\mathcal{R}$ with a forgery (m^*, σ^*) with non-negligible advantage.*

Proof. Assume that this was not the case. Then one could easily simulate $\mathcal{R}$ and the meta-reduction interacting with this reduction would solve $\Pi_{\mathcal{M}}$ efficiently with non-negligible advantage. This contradicts the hardness of the problem. □

Hence, from now on we condition on the meta-reduction to always provide the reduction with a forgery, without losing more than a negligible advantage. In this case we have:

Lemma 2 (Meta-Reductions Cannot Replay Signatures). *Let $\mathcal{M}$ be a non-programming meta-reduction that converts any $(\mathsf{EUF\text{-}CMA}_{\mathcal{S}} \rightsquigarrow \Pi_{\mathcal{R}})$ reduction in a black-box way into an adversary against some hard problem $\Pi_{\mathcal{M}}$. Let $\mathbb{Q}$ be the set of message-signature pairs (m_i, σ_i) resulting from $\mathcal{M}$'s queries to the reduction's signing oracle, and let (m^*, σ^*) be the message-signature pair output by $\mathcal{M}$ as a forgery on behalf of the adversary. Further, let the reduction used by $\mathcal{M}$ be $\mathcal{R}$ as described above. Then it holds that $(m^*, \sigma^*) \notin \mathbb{Q}$.*

Proof. The proof is rather straightforward. Observe that by construction of $\mathcal{R}$ the following holds: $\forall (m, \sigma) \in \mathbb{Q} : \exists i \in \mathbb{Z}_p : \omega \leftarrow \mathcal{O}(m, i) \wedge \sigma \stackrel{?}{=} \mathcal{S}.\mathsf{Sign}(m, i; \omega)$. Therefore, it follows directly that, for $(m^*, \sigma^*) \in \mathbb{Q}$, the reduction $\mathcal{R}$ will abort and $\mathsf{Succ}_{\Pi_{\mathcal{R}}}^{\mathcal{R}^{\mathcal{M}}}(\kappa) = 0$ for $\mathcal{M}$ if it replays an element of $\mathbb{Q}$ as a forgery. Note that here we rely on the previous Lemma which assumes that $\mathcal{M}$ always provides such a forgery. As it, thus, would not be a successful meta-reduction it must hold that $(m^*, \sigma^*) \notin \mathbb{Q}$. □

4.2 A Reduction against the Meta-reduction

Using the reduction described in the previous section, we now prove that finding an efficient meta-reduction for a randomized signature scheme is at least as hard as finding a strong existential forger.

Theorem 3 (Meta-Reductions to Non-Interactive Problems Are Hard). *Let $\mathcal{S}$ be a randomized signature scheme with reconstructible hash queries, $\Pi_{\mathcal{R}}$ be a monotone solvable problem, and $\Pi_{\mathcal{M}}$ be a non-interactive, efficiently generatable problem. If $\Pi_{\mathcal{M}}$ is hard, then finding an efficient meta-reduction $\mathcal{M}$ that converts any successful $(\mathsf{EUF\text{-}CMA}_{\mathcal{S}} \rightsquigarrow \Pi_{\mathcal{R}})$-reduction in a black-box way into an efficient successful adversary against $\Pi_{\mathcal{M}}$ is at least as hard as finding an* sEUF-CMA *adversary against $\mathcal{S}$.*

More precisely, assume there exists an efficient non-programming black-box meta-reduction $\mathcal{M}$ that converts any $(\mathsf{EUF\text{-}CMA}_{\mathcal{S}} \rightsquigarrow \Pi_{\mathcal{R}})$-reduction into an adversary against $\Pi_{\mathcal{M}}$. Then, there exists a meta-meta-reduction $\mathcal{N}$ that converts $\mathcal{M}$ into an adversary against the sEUF-CMA *security of $\mathcal{S}$ with non-negligible* *success probability* $\mathsf{Succ}^{\mathcal{S},\mathcal{N}^{\mathcal{M}}}_{\mathsf{sEUF\text{-}CMA}}(\kappa) \geq \frac{1}{r} \cdot \mathsf{Succ}^{\mathcal{M}^{\mathcal{R}}}_{\Pi_{\mathcal{M}}}(\kappa)$ *and runtime* $\mathsf{Time}_{\mathcal{N}^{\mathcal{M}}}(\kappa) = \mathsf{Time}_{\mathcal{M}^{\mathcal{R}}}(\kappa) + \mathsf{poly}(\kappa)$*, where* $\mathcal{R}$ *is the reduction described above and* r *is the maximal number of reduction instances invoked by* $\mathcal{M}$*.*

Note that, since $\mathcal{M}$ needs to work for any (black-box) $\mathcal{R}$, we may assume that $\mathcal{R}$ knows r. Indeed, we take advantage of this fact in the proof.

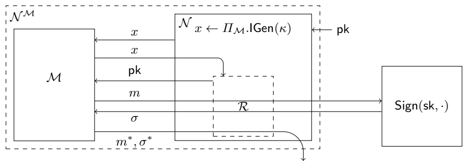


Fig. 2. The meta-meta-reduction relies on the fact that $\mathcal{M}$ cannot replay an old signature. It simply outputs the forgery provided by $\mathcal{M}$. The meta-meta-reduction can be generalized for multiple instances of $\mathcal{R}$ using a standard guess-and-insert approach.

The proof is omitted here, but can be found in the full version. The idea is outlined in Figure 2. The meta-meta-reduction picks one of the r reduction instances run by $\mathcal{M}$ at random and substitutes this instance with the help of its external signature oracle. The reconstruction property guarantees that $\mathcal{N}$ can still pretend towards $\mathcal{M}$ to have made the hash queries of externally provided signatures locally. All other reduction instances are simulated by $\mathcal{N}$ itself. In order to be successful, $\mathcal{M}$ needs to provide a forgery to some of the $\mathcal{R}$-instances, and with probability $1/r$ it will be the one which is “externalized” by $\mathcal{N}$. In this case, Lemma 2 ensures that the forgery is a strong forgery against the external signature oracle.

Acknowledgments. We thank the anonymous reviewers for valuable comments. Marc Fischlin was supported by a Heisenberg grant Fi 940/3-1 of the German Research Foundation (DFG). Part of this work was also supported by the German Federal Ministry of Education and Research (BMBF) within EC SPRIDE. Nils Fleischhacker was supported by the German Federal Ministry of Education and Research (BMBF) through funding for the Center for IT-Security, Privacy and Accountability (CISPA – `www.cispa-security.org`).

References

1. Abdalla, M., An, J.H., Bellare, M., Namprempre, C.: From identification to signatures via the Fiat-Shamir transform: Minimizing assumptions for security and forward-security. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 418–433. Springer, Heidelberg (2002)
2. Abe, M., Groth, J., Ohkubo, M.: Separating short structure-preserving signatures from non-interactive assumptions. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 628–646. Springer, Heidelberg (2011)

3. Baecher, P., Brzuska, C., Fischlin, M.: Notions of black-box reductions, revisited. IACR Cryptology ePrint Archive (2013)
4. Baldimtsi, F., Lysyanskaya, A.: On the security of one-witness blind signature schemes. IACR Cryptology ePrint Archive (2012)
5. Bellare, M., Namprempre, C., Pointcheval, D., Semanko, M.: The one-more-RSA-inversion problems and the security of Chaum's blind signature scheme. Journal of Cryptology 16(3), 185–215 (2003)
6. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Ashby, V. (ed.) ACM CCS 1993: 1st Conference on Computer and Communications Security, Fairfax, Virginia, USA, November 3-5, pp. 62–73. ACM Press (1993)
7. Boneh, D., Venkatesan, R.: Breaking RSA may not be equivalent to factoring. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 59–71. Springer, Heidelberg (1998)
8. Bresson, E., Monnerat, J., Vergnaud, D.: Separation results on the "one-more" computational problems. In: Malkin, T. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 71–87. Springer, Heidelberg (2008)
9. Coron, J.-S.: Optimal security proofs for PSS and other signature schemes. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 272–287. Springer, Heidelberg (2002)
10. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987)
11. Fischlin, M.: Black-box reductions and separations in cryptography. In: Mitrokotsa, A., Vaudenay, S. (eds.) AFRICACRYPT 2012. LNCS, vol. 7374, pp. 413–422. Springer, Heidelberg (2012)
12. Fischlin, M., Fischlin, R.: The representation problem based on factoring. In: Preneel, B. (ed.) CT-RSA 2002. LNCS, vol. 2271, pp. 96–113. Springer, Heidelberg (2002)
13. Fischlin, M., Lehmann, A., Ristenpart, T., Shrimpton, T., Stam, M., Tessaro, S.: Random oracles with(out) programmability. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 303–320. Springer, Heidelberg (2010)
14. Fischlin, M., Schröder, D.: On the impossibility of three-move blind signature schemes. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 197–215. Springer, Heidelberg (2010)
15. Garg, S., Bhaskar, R., Lokam, S.V.: Improved bounds on security reductions for discrete log based signatures. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 93–107. Springer, Heidelberg (2008)
16. Gentry, C., Wichs, D.: Separating succinct non-interactive arguments from all falsifiable assumptions. In: Fortnow, L., Vadhan, S.P. (eds.) 43rd Annual ACM Symposium on Theory of Computing, San Jose, California, USA, June 6-8, pp. 99–108. ACM Press (2011)
17. Guillou, L.C., Quisquater, J.-J.: A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory. In: Günther, C.G. (ed.) EUROCRYPT 1988. LNCS, vol. 330, pp. 123–128. Springer, Heidelberg (1988)
18. Nielsen, J.B.: Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 111–126. Springer, Heidelberg (2002)

19. Okamoto, T.: Provably secure and practical identification schemes and corresponding signature schemes. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 31–53. Springer, Heidelberg (1993)
20. Paillier, P., Vergnaud, D.: Discrete-log-based signatures may not be equivalent to discrete log. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 1–20. Springer, Heidelberg (2005)
21. Pass, R.: Limits of provable security from standard assumptions. In: Fortnow, L., Vadhan, S.P. (eds.) 43rd Annual ACM Symposium on Theory of Computing, San Jose, California, USA, June 6-8, pp. 109–118. ACM Press (2011)
22. Pointcheval, D., Stern, J.: Security proofs for signature schemes. In: Maurer, U. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 387–398. Springer, Heidelberg (1996)
23. Pointcheval, D., Stern, J.: Security arguments for digital signatures and blind signatures. Journal of Cryptology 13(3), 361–396 (2000)
24. Reingold, O., Trevisan, L., Vadhan, S.P.: Notions of reducibility between cryptographic primitives. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 1–20. Springer, Heidelberg (2004)
25. Schnorr, C.P.: Efficient identification and signatures for smart cards. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 239–252. Springer, Heidelberg (1990)
26. Schnorr, C.P.: Efficient signature generation by smart cards. Journal of Cryptology 4(3), 161–174 (1991)
27. Seurin, Y.: On the exact security of schnorr-type signatures in the random oracle model. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 554–571. Springer, Heidelberg (2012)

Practical Signatures from Standard Assumptions

Florian Böhl[1], Dennis Hofheinz[1], Tibor Jager[2], Jessica Koch[1], Jae Hong Seo[3], and Christoph Striecks[1]

[1] Karlsruhe Institute of Technology, Karlsruhe, Germany
[2] Ruhr-Universität Bochum, Bochum, Germany
[3] Myongji University, Yongin, Republic of Korea

Abstract. We put forward new techniques for designing signature schemes. As a result, we present practical signature schemes based on the CDH, the RSA, and the SIS assumptions. Our schemes compare favorably with existing schemes based on these assumptions.

Our core idea is the use of *tag-based* signatures. Concretely, each signatures contains a tag which is uniformly chosen from a suitable tag set. Intuitively, the tag provides a way to embed instances of computational problems. Indeed, carefully choosing these tag spaces provides new ways to partition the set of possible message-tag pairs into "signable" and "unsignable" pairs. In our security proof, we will thus be able to sign all adversarially requested messages, and at the same time use an adversarially generated forgery with suitably large probability.

Keywords: digital signatures, CDH assumption, pairing-friendly groups, RSA assumption, SIS assumption.

1 Introduction

On the Difficulty of Constructing Digital Signature Schemes. From a purely theoretical point of view, digital signatures turn out to be a weaker primitive than public-key encryption (PKE): digital signature schemes are equivalent to one-way functions [29], while PKE appears to be a stronger primitive [24]. However, somewhat surprisingly, it seems much harder to construct *practical* signature schemes than PKE schemes. For instance, there exist practical and even chosen-ciphertext secure PKE schemes from a variety of assumptions (e.g., DDH [12], CDH [9], DCR [14], factoring [18], or LPN [28]), while it seems much harder to construct practical signature schemes from any of the above assumptions.[1] Indeed, the most efficient known schemes (e.g., [13, 15, 25, 5, 19]) are based on what [20] call "strong" assumptions.[2] Intuitively, to contradict a strong assumption, it suffices to solve one out of many possible problem instances given by the challenge. For instance, the strong RSA assumption demands that finding *any* $e \geq 2$ and $C^{1/e} \bmod N$ when given N and $C \in \mathbb{Z}_N$ is hard.

[1] We ignore here schemes based on random oracles (e.g., full-domain hash [3]), since these come only with heuristic proofs.

[2] There are also practical schemes based on standard, non-strong assumptions, e.g., [6, 33, 22, 20]; these however suffer from large keys or signatures, or from a comparatively inefficient signing process. A notable exception are dual systems and dual form signatures [32, 17], which are however based on decisional assumptions like DLIN.

We believe that this reliance on strong assumptions is very natural for a signature scheme. Namely, in the standard security experiment for digital signatures, an adversary A wins if it generates a signature for a (fresh) message of his own choice. This gives A much more freedom (by choosing signature *and* message freely) than, e.g., an adversary in an encryption security experiment. Thus, if we use A in a straightforward way as a problem-solver in a security reduction to a computational assumption, A itself may select which instance of the particular problem it is going to solve (by choosing the forgery message). Note that we cannot simply guess which instance A is going to solve, since there usually will be superpolynomially many possible messages (and thus problem instances).[3]

Our Approach: Tag-Based Signatures. In this work, we explore *tag-based signature schemes* as a means to enable security reductions to standard computational assumptions. In a tag-based signature scheme, each signature carries a tag t that can be chosen freely during signature time. Intuitively, the tag further parameterizes the problem instance we want to let A solve during a security reduction. The benefit of this additional parameterization becomes apparent when one considers tags from a small domain: if there are only few (i.e., polynomially many) tags, we could try to guess the tag t^* used in A's forgery in advance. Our security reduction could then set up things such that precisely signatures with tags $t \neq t^*$ can be generated, and any signature with tag t^* can be used to solve an underlying computational problem. (For now, let us assume that A never reuses a tag from a previously signed message in his forgery.)

Showcase: Compact CDH-Based Signatures. To showcase our ideas, we first consider a tag-based signature scheme in pairing-friendly groups. The scheme itself can be seen as a variant of the stateful signature scheme of Hohenberger and Waters [23], with tags (chosen from a suitably-sized tag space) in place of states. At this point, our work forks up into two directions: first, we show that this scheme achieves a *bounded* form of security in which an upper bound on the number of adversarial signature queries is known prior to setting the scheme's parameters. This yields an extremely compact and efficient scheme for reasonable security parameters. Next, we show how to achieve full security by employing a different, somewhat more generic proof strategy. This yields an *asymptotically* more efficient scheme, at the cost of a qualitatively worse security reduction. (With "qualitatively worse", we mean that the reduction *loss* depends – in a polynomial way – on the adversary's runtime and success.) In both cases, we prove security under the computational Diffie-Hellman (CDH) assumption.

More on the Bounded Security of our Scheme. First, consider an adversary A that makes only a fixed, a-priori bounded number q of signing queries. Our tag space will then consist of *vectors* $\overrightarrow{t}$ over a polynomial domain whose size depends on q. In the security reduction, we will guess a suitable vector prefix $\overrightarrow{t}^{(i)}$ of the challenge tag that is different from all tag prefixes that arise during the signature generation for A. This allows us to embed a CDH challenge into the prefix $\overrightarrow{t}^{(i)}$ during the security proof.

[3] There are more clever ways of embedding a computational problem into a signature scheme (e.g., partitioning [11, 33]). These techniques however usually require special algebraic features such as homomorphic properties or pairing-friendly groups. For instance, partitioning is not known to apply in the (standard-model) RSA setting.

On a technical level, this strategy opens the door to an interesting tradeoff between public key and signature size. Namely, using a partitioning argument, we can allow a very limited number of tag-prefix-collisions (in the sense that tags with the prefix $\overrightarrow{t}^{(i)}$ used in the forgery occur in the signatures generated for A). This yields smaller public keys, at the cost of additional group elements in the signatures. For reasonable security parameters (and assuming $q \leq 2^{30}$ signature queries), we thus obtain a scheme with 4 and 15 group elements per signature, resp. public key.

Confined Guessing. To prove our scheme fully secure, we rely on a new technique we call "confined guessing". Concretely, we view our scheme as the combination of several instances of a mildly secure tag-based scheme, each instance with a different tag size. Here, mild security means that an adversary has to commit in advance on the tag t^* of his forgery. The basic version of our CDH-based scheme can be proven mildly secure, since the CDH challenge can be embedded precisely in signatures with tag t^*. (In particular for small tag sets, it may be necessary to generate a *constant* number of signatures with tag t^* for A. We can solve this problem using a partitioning technique, which – since the number of required t^*-signatures is small – can be very efficient.)

A signature in our fully secure scheme consists of $\log(\lambda)$ signatures $(\sigma_i)_{i=1}^{\log(\lambda)}$ of a mildly secure scheme. (In the CDH case, these signatures can be aggregated.) The i-th signature component σ_i is created with tag chosen as uniform 2^i-bit string. Hence, tag-collisions (i.e., multiply used tags) are likely to occur after a few signatures in instances with small i, while instances with larger i will almost never have tag-collisions.

We will reduce the (full) security of the new scheme *generically* to the mild security of the underlying scheme. When reducing a concrete (full) adversary B to a mild adversary A, we will first single out an instance i^* such that (a) the set of all tags is polynomially small (so we can guess the i^*-th challenge tag $t^*_{i^*}$ in advance), and (b) tag-collisions occur only with sufficiently small (but possibly non-negligible) probability in an attack with A (so only a constant number of $t^*_{i^*}$-signatures will have to be generated for A). This instance i^* is the challenge instance, and all other instances are simulated by A for B. Any valid forgery of B *must* contain a valid signature under instance i^* with 2^{i^*}-bit tag. Hence any B-forgery implies an A-forgery. This leads to a very compact scheme (e.g., in the CDH case, with $O(1)$ and $O(\log(\lambda))$ group elements per signature, resp. public key). However, the *loss* in the security reduction depends (polynomially) on an adversary's success and runtime.

Other Applications. We also show how to generalize our confined guessing paradigm to other computational settings. In particular, we construct mildly secure schemes from the RSA and SIS assumptions. Combining this with our generic transformation, this gives compact and very efficient new fully secure signature schemes.

Efficiency Comparison. The most efficient previous CDH-based signature scheme [33] has signatures and public keys of size $O(\lambda)$, resp. $O(1)$ group elements. Our CDH-based scheme also has constant-sized signatures, and more compact public keys. Concretely, we can get public keys of $O(\sqrt{\lambda/\log(\lambda)})$ group elements when only aiming at bounded security (see Table 1 for exact figures). Besides, we can get public keys of $O(\log(\lambda))$ group elements at the price of a worse security reduction. Our RSA-based scheme has similar key and signature sizes as existing RSA-based schemes [22, 20], but requires significantly fewer (i.e., only $O(\log(\lambda))$ instead of $O(\lambda)$, resp. $O(\lambda/\log(\lambda))$

many) generations of large primes. Again, this improvement is bought with a worse security reduction. Our SIS-based scheme offers an alternative to the existing scheme of [7]. Concretely, our scheme has larger (by a factor of $\log(\lambda)$) signatures and a worse security reduction, but significantly smaller (by a factor of $\lambda/\log(\lambda)$) public keys.

Note on the History of This Paper. This paper is the result of a merge of two papers submitted to Eurocrypt. Both submissions contained essentially the same CDH-based scheme. One submission, by Seo, contained its bounded security analysis. The other, by Böhl, Hofheinz, Jager, Koch, and Striecks, contained the confined guessing strategy. In this merged paper, the results of Seo, resp. BHJKS, are contained in Section 4, resp. 5. During the merge, Seo acted as corresponding author.

2 Preliminaries

Notation. For $n \in \mathbb{R}$, let $[n] := \{1, \ldots, \lfloor n \rfloor\}$. We write $[a, b]$ to denote the set of integers $\{a, \ldots, b\}$. Throughout the paper, $\lambda \in \mathbb{N}$ denotes the security parameter. For a finite set $\mathcal{S}$, we denote by $s \leftarrow \mathcal{S}$ the process of sampling s uniformly from $\mathcal{S}$. For a probabilistic algorithm A, we write $y \leftarrow A(x)$ for the process of running A on input x with uniformly chosen random coins, and assigning y the result. If A's running time is polynomial in λ, then A is called probabilistic polynomial-time (PPT). A function $f : \mathbb{N} \rightarrow \mathbb{R}$ is negligible if it vanishes faster than the inverse of any polynomial (i.e., if $\forall c \exists \lambda_0 \forall \lambda \geq \lambda_0 : |f(\lambda)| \leq 1/\lambda^c$). On the other hand, f is significant if it dominates the inverse of some polynomial (i.e., if $\exists c, \lambda_0 \forall \lambda \geq \lambda_0 : f(\lambda) \geq 1/\lambda^c$).

Signature Schemes. A signature scheme SIG consists of three PPT algorithms (Gen, Sig, Ver). The key generation algorithm $\mathsf{Gen}(1^\lambda)$ outputs a public key pk and a secret key sk. The signing algorithm $\mathsf{Sig}(sk, M)$, given the secret key sk and a message M, outputs a signature σ. Given the public key pk, a message M, and a signature σ, $\mathsf{Ver}(pk, M, \sigma)$ outputs a verdict $b \in \{0, 1\}$. For correctness, we require for any $\lambda \in \mathbb{N}$, all $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$, all M, and all $\sigma \leftarrow \mathsf{Sig}(sk, M)$ that $\mathsf{Ver}(pk, M, \sigma) = 1$.

EUF-(na)CMA Security. A signature scheme SIG is existential unforgeable under adaptive chosen-message attacks (EUF-CMA) iff for any PPT forger F in the following experiment the probability to win is negligible. F receives a public key pk generated as $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$, and has access to a signing oracle $\mathsf{Sig}(sk, \cdot)$. F wins if it outputs a valid signature for a message M such that it has never queried $\mathsf{Sig}(sk, M)$. In the non-adaptive (EUF-naCMA) case the adversary is forced to output messages $M_1, \ldots, M_q$ it wants to see signed before obtaining the public key pk. F wins if it outputs a valid signature for a message $M \neq M_j \ \forall j \in [q]$. We define $\mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\mathsf{SIG},F}(\lambda)$ and $\mathsf{Adv}^{\mathsf{euf\text{-}nacma}}_{\mathsf{SIG},F}(\lambda)$ to be F's winning probability in the adaptive, resp. non-adaptive case.

EUF-q-(na)CMA Security. In addition to the previous notions, we define two weaker security notions for signatures. The notions existential unforgeability with respect to q-bounded adaptive chosen-message-attacks (EUF-q-CMA), resp. non-adaptive chosen-message-attacks (EUF-q-naCMA) are exacly the same as the EUF-CMA, resp. EUF-naCMA security notions above, except that the adversary is restricted to at most q

signature queries. We define $\mathsf{Adv}^{\text{euf-q-cma}}_{\mathsf{SIG},F}(\lambda)$ and $\mathsf{Adv}^{\text{euf-q-nacma}}_{\mathsf{SIG},F}(\lambda)$ to be F's winning probability in the adaptive, resp. non-adaptive case. Concretely, if for any PPT algorithm F, which issues at most (polynomial) q signing queries and runs in time T, $\mathsf{Adv}^{\text{euf-q-cma}}_{\mathsf{SIG},F}(\lambda) < \varepsilon$ and $\mathsf{Adv}^{\text{euf-q-nacma}}_{\mathsf{SIG},F}(\lambda) < \varepsilon$, then we say that SIG is (q, ε, T)- EUF-CMA secure, resp. (q, ε, T)-EUF-naCMA secure.

Pseudorandom Functions. For any set $\mathcal{S}$ a *pseudorandom function (PRF) with range* $\mathcal{S}$ is an efficiently computable function $\mathsf{PRF}^{\mathcal{S}} : \{0,1\}^{\lambda} \times \{0,1\}^{*} \to \mathcal{S}$. We may also write $\mathsf{PRF}^{\mathcal{S}}_{\kappa}(x)$ for $\mathsf{PRF}^{\mathcal{S}}(\kappa, x)$ with key $\kappa \in \{0,1\}^{\lambda}$. Additionally we require that

$$\mathsf{Adv}^{\mathsf{prf}}_{\mathsf{PRF}^{\mathcal{S}},A}(\lambda) := \left|\Pr\left[A^{\mathsf{PRF}^{\mathcal{S}}_{\kappa}(\cdot)} = 1 \text{ for } \kappa \leftarrow \{0,1\}^{\lambda}\right] - \Pr\left[A^{\mathcal{U}_{\mathcal{S}}(\cdot)} = 1\right]\right|$$

is negligible in k where $\mathcal{U}_{\mathcal{S}}$ is a truly uniform function to $\mathcal{S}$. Note that for any efficiently samplable set $\mathcal{S}$ with uniform sampling algorithm Samp we can generically construct a PRF with range $\mathcal{S}$ from a PRF $\mathsf{PRF}^{\{0,1\}^{\lambda}}$ by using the output of $\mathsf{PRF}^{\{0,1\}^{\lambda}}_{\kappa}$ as random coins for Samp. (We can assume without loss of generality that Samp requires only λ bits of randomness.) Following this principle we can construct $(\mathsf{PRF}^{\mathcal{S}_i})_{i\in[n]}$ for a family of sets $(\mathcal{S}_i)_{i\in[n]}$ from a single PRF $\mathsf{PRF}^{\{0,1\}^{\lambda}}$.

Chameleon Hashing. A chameleon hash scheme CHS consists of two PPT algorithms $(\mathsf{CHGen}, \mathsf{CHTrapColl})$. $\mathsf{CHGen}(1^{\lambda})$ outputs a tuple (CH, τ) where CH is the description of an efficiently computable function $\mathsf{CH} : \mathcal{M} \times \mathcal{R} \to \mathcal{N}$ which maps a message M and randomness r to a hash value $\mathsf{CH}(M, r)$. The trapdoor τ allows to produce collisions in the following sense: given arbitrary M, r, M', $\mathsf{CHTrapColl}(\tau, M, r, M')$ finds r' with $\mathsf{CH}(M, r) = \mathsf{CH}(M', r')$. We require that the distribution of r' is uniform given only CH and M'. We say that CH is collision-resistant iff for $(\mathsf{CH}, \tau) \leftarrow \mathsf{CHGen}(1^{\lambda})$ and any PPT adversary C, which receives as input CH, the probability $\mathsf{Adv}^{\mathsf{cr}}_{\mathsf{CH},C}(\lambda)$ to find $(M, r) \neq (M', r')$ with $\mathsf{CH}(M, r) = \mathsf{CH}(M', r')$ is negligible in λ.

Generic Transformation from Non-Adaptive to Adaptive Secure Signatures. There is a well known generic transformation from EUF-naCMA secure signatures to EUF-CMA secure signatures which was used in many previously proposed signature schemes (e.g., [26, 31, 4, 23, 22]). Analogously, we can construct a generic transformation from EUF-q-naCMA secure signatures to EUF-q-CMA secure signatures.

Lemma 1. *If* SIG *is* (q, ε, T)*-EUF-naCMA secure signature scheme and* CHS *is a chameleon hash scheme, then there is a generic transformation taking* SIG *and* CHS *as input and outputting* $(q, 2(\varepsilon + \varepsilon_{ch}), T')$*-EUF-CMA secure signature scheme, where the chameleon hash function satisfies* $(\varepsilon_{ch}, T_{ch})$*-collision resistance and* $T' \approx T \approx T_{ch}$.

We provide the details of the generic transformation in [30].

3 Our CDH-Based Signature Scheme

Overview. Our starting point is an interpretation of the stateful signature scheme of Hohenberger and Waters [23] as a tag-based scheme. In this section, we will only describe the scheme (and a natural generalization); the next two sections will present two surprisingly different analyses of this scheme. But first, we start with a few preparatory definitions.

Bilinear Groups. We say that $\mathcal{G}$ is a bilinear group generator if on inputting the security parameter λ, it outputs a tuple $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, e)$, where p is a $(2\lambda + 1)$-bit prime, $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_t$ are finite abelian groups of order p, and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_t$ is a non-degenerate bilinear map, that is, (bilinearity) for all $a, b \in \mathbb{Z}_p$ and $g \in \mathbb{G}_1$, $g' \in \mathbb{G}_2$, $e(g^a, g'^b) = e(g, g')^{ab}$ and (non-degeneracy) for generators $g \in \mathbb{G}_1$ and $g' \in \mathbb{G}_2$, $e(g, g') \neq 1$. If $\mathbb{G}_1 = \mathbb{G}_2$, then we use a notation $\mathbb{G}$ to denote $\mathbb{G}_1 = \mathbb{G}_2$ and we say that e is a type-1 pairing. If $\mathbb{G}_1 \neq \mathbb{G}_2$ but there is an efficiently computable homomorphism $\phi : \mathbb{G}_2 \to \mathbb{G}_1$, then we say that e is a type-2 pairing. Otherwise (that is, $\mathbb{G}_1 \neq \mathbb{G}_2$ and there are no efficiently computable homomorphisms between $\mathbb{G}_1$ and $\mathbb{G}_2$), we say that e is a type-3 pairing.

CDH Assumption. Let $\mathcal{G}$ be a bilinear group generator. We say that $\mathcal{G}$ satisfies the (ϵ_{dh}, T_{dh})-DH assumption if for any T_{dh}- time probabilistic algorithm B the following advantage $\mathsf{Adv}^{dh}_{\mathcal{G},B}$ is less than ϵ_{dh}:

$$\mathsf{Adv}^{dh}_{\mathcal{G},B} = \Pr\left[B(p, \mathbb{G}, \mathbb{G}_t, e, g, g^a, g^b) \to g^{ab} : \mathcal{G}(\lambda) \to (p, \mathbb{G}, \mathbb{G}_t, e), a, b \leftarrow \mathbb{Z}_p, g \leftarrow \mathbb{G}\right].$$

Tag-Based Signature Schemes. Our basic scheme will be tag-based; that means that signature and verification take an additional *tag* as input. More formally, a tag-based signature scheme $\mathsf{SIG_t} = (\mathsf{Gen_t}, \mathsf{Sig_t}, \mathsf{Ver_t})$ with message space $\mathcal{M}_\lambda$ and tag space $\mathcal{T} = \mathcal{T}_\lambda$ consists of three PPT algorithms. Key generation $(pk, sk) \leftarrow \mathsf{Gen_t}(1^\lambda)$ takes as input a security parameter and outputs a key pair (pk, sk). Signing $\sigma \leftarrow \mathsf{Sig_t}(sk, M, t)$ computes a signature σ on input a secret key sk, message M, and tag t. Verification $\mathsf{Ver_t}(pk, M, \sigma, t) \in \{0, 1\}$ takes a public key pk, message M, signature σ, and a tag t, and outputs a bit. For correctness, we require for any $\lambda \in \mathbb{N}$, all $(pk, sk) \leftarrow \mathsf{Gen_t}(1^\lambda)$, all $M \in \mathcal{M}_\lambda$, all $t \in \mathcal{T}$, and all $\sigma \leftarrow \mathsf{Sig_t}(sk, M, t)$ that $\mathsf{Ver_t}(pk, M, \sigma, t) = 1$.

The Basic (Tag-Based) Scheme. The signature scheme $\mathsf{SIG^{CDH}}$ from Figure 1 is derived from the stateful CDH-based scheme of [23], but with states interpreted as tags, and with two additional modifications. First, we substitute the implicit chameleon hash function $u^M v^r$ used in [23] with a product $\mathbf{u}^M = \prod_{i=0}^{m} u_i^{M^i}$. (The parameter m will be fixed later.) Second, we omit the $w^{\lceil \log(t) \rceil}$-factor in the "Boneh-Boyen hash function", which simplifies this part to $(z^t h)^s$.

The Generalized (Non-Tag-Based) Scheme. We also provide a natural generalization of $\mathsf{SIG^{CDH}}$, which we call $\mathsf{SIG^{CDH}_{gen}}$ (see Figure 2). Compared to $\mathsf{SIG^{CDH}}$, $\mathsf{SIG^{CDH}_{gen}}$ first hashes the message to be signed (using a chameleon hash with images in $\mathbb{Z}_p$); besides, $\mathsf{SIG^{CDH}_{gen}}$ uses a pseudorandom function to derive l tags t_i that are incorporated into the mentioned "Boneh-Boyen hash function". The number l of tags and the respective sets $\mathcal{T}_i$ from which the t_i are chosen which will be defined in our respective analyses.

```
$\mathsf{Gen_t}(1^\lambda)$
  $(p, \mathbb{G}, \mathbb{G}_t, e) \leftarrow \mathcal{G}(\lambda)$
  $\alpha \leftarrow \mathbb{Z}_p$
  $g, u_0, \ldots, u_m, z, h \leftarrow \mathbb{G}$
  $sk := \alpha$
  $pk :=$
      $(g, g^\alpha, u_0, \ldots, u_m, z, h)$
  return $(pk, sk)$
```

```
$\mathsf{Sig_t}(sk, M, t)$
  $s \leftarrow \mathbb{Z}_p$
  $\mathbf{u}^M := \prod_{i=0}^{m} u_i^{M^i}$
  $\tilde{\sigma}_1 := (\mathbf{u}^M)^\alpha (z^t h)^s$
  $\tilde{\sigma}_2 := g^s$
  return $(\tilde{\sigma}_1, \tilde{\sigma}_2)$
```

```
$\mathsf{Ver_t}(pk, M, \sigma = (\tilde{\sigma}_1, \tilde{\sigma}_2), t)$
  if $t \notin \mathcal{T}_\lambda$
    return 0
  if $e(\tilde{\sigma}_1, g) \neq$
        $e(\mathbf{u}^M, g^\alpha) e(\tilde{\sigma}_2, z^t h)$
    return 0
  else
    return 1
```

Fig. 1. The modified Hohenberger-Waters CDH-based signature scheme $\mathsf{SIG}^{\mathsf{CDH}}$ [23]

```
$\mathsf{Gen}(1^\lambda)$
  $(p, \mathbb{G}, \mathbb{G}_t, e) \leftarrow \mathcal{G}(\lambda)$
  $(\mathsf{CH}, \tau) \leftarrow \mathsf{CHGen}(1^\lambda)$
  $\alpha \leftarrow \mathbb{Z}_p$
  $g, h, u_0, \ldots, u_m,$
      $z_1, \ldots, z_l \leftarrow \mathbb{G}$
  $\kappa \leftarrow \{0,1\}^\lambda$
  $sk := (g, \alpha, \mathsf{CH})$
  $pk := (g, g^\alpha, (u_j)_{j=0}^m,$
        $(z_i)_{i=1}^l, h, \mathsf{CH}, \kappa)$
  return $(pk, sk)$
```

```
$\mathsf{Sig}(sk, M)$
  $s, r \leftarrow \mathbb{Z}_p$
  $x := \mathsf{CH}(M, r)$
  $\mathbf{u}^x := \prod_{i=0}^m u_i^{x^i}$
  for $i := 1$ to $l$ do
    $t_i := \mathsf{PRF}_\kappa^{\mathcal{T}_i}(x)$
  $z := \prod_{i=1}^l z_i^{t_i}$
  $\tilde{\sigma}_1 := (\mathbf{u}^x)^\alpha (z \cdot h)^s$
  $\tilde{\sigma}_2 := g^s$
  return $(\tilde{\sigma}_1, \tilde{\sigma}_2, r)$
```

```
$\mathsf{Ver}(pk, M, \sigma = (\tilde{\sigma}_1, \tilde{\sigma}_2, r))$
  $x := \mathsf{CH}(M, r)$
  for $i := 1$ to $l$ do
    $t_i := \mathsf{PRF}_\kappa^{\mathcal{T}_i}(x)$
  if $e(\tilde{\sigma}_1, g) \neq$
    $e(\mathbf{u}^x, g^\alpha) e(\tilde{\sigma}_2, h \prod_{i=1}^{l} z_i^{t_i})$
    return 0
  else
    return 1
```

Fig. 2. The generalized CDH-based signature scheme $\mathsf{SIG}_{\mathsf{gen}}^{\mathsf{CDH}}$

4 Bounded CMA Security

In this section, we first analyze the security of the basic tag-based signature scheme so that it is a EUF-q-naCMA secure signature scheme with somewhat short public key. Then, we prove that the generalized tag-based signature scheme is a EUF-q-naCMA secure signature scheme with short public key. The proposed signature scheme is for fixed length messages, but we note that we can easily modify it for arbitrary length messages by using collision resistant hash functions; first, compute a hash value of a long message, and then use it as a message for the signature scheme.

4.1 Combining Two Techniques: 'Somewhat' Short Public Key

We begin with exploring two techniques for obtaining short signatures in the standard model. In the simulation of the EUF-q-naCMA model, the simulator should give a set of signatures on messages queried by the adversary, but the simulator should not be able to create signatures on all messages other than those queried by the adversary. If the simulator can create signatures on all messages, then the simulator does not need help from the adversary to obtain the forgery since the simulator can sign on all messages himself; hence, we cannot extract the solution of the DH problem from the output of the adversary. We can use programmable hash functions [19] to allow the simulator to produce only signatures on messages queried by the adversary. In particular, we use weak programmable hash functions [20] to construct EUF-q-naCMA secure short signatures. For a 2λ-bit message M, we consider M as an element of $\mathbb{Z}_p$. $(\prod_{i=0}^{m} u_i^{M^i})$ is a weak programmable hash function on input M that, in the EUF-q-naCMA model, allows the simulator to sign on at most m messages, which are given by the adversary before generating the public key.[4] Furthermore, we can construct a simulator that extracts the solution of g^{ab} from the forgery by imbedding g^a in u_i and setting g and g^α by g and g^b, respectively.

There is the other technique that obtains short signatures with short public key by maintaining the index counter in the signer side [23]. The idea of this technique is first to restrict the adversary to attack one of the polynomially many indexes and then uses the technique for selectively-secure signatures such as that used in the Boneh-Boyen signature scheme [5]. We can combine this technique with programmable hash functions. Since our aim is a stateless signature scheme, we should modify this technique so that the signer does not maintain the current index but randomly chooses it from some fixed set for each signature. Then, we obtain a short signature with somewhat short public key, which is our basic tag-based scheme in Figure 1, where t is uniformly chosen from $t^* = [1, Q]$ for another parameter $Q \geq q$. For the basic scheme, we set Q to be polynomial in λ. The strategy of the simulation in the EUF-q-naCMA model is as follows: The simulator guesses t^*, the tag of the forgery, (with non-negligible $\frac{1}{Q}$ probability) and uses the technique for the selectively-secure signature scheme of the Boneh-Boyen signatures. For each signature, the tag is randomly chosen so that there may exist several signatures containing the same tag as t^* among the resulting signatures of singing queries. Under normal circumstances, the simulator cannot produce signatures with tag t^* (since we use technique for selectively-secure scheme). We can resolve this by using the weak programmable hash functions. If we uniformly choose a tag from $[1, Q]$ at most q times for polynomial $Q \geq q$, there are at most $\Theta(\frac{\lambda}{\log \lambda})$ same tags as the tag of the forgery with overwhelming probability. Therefore, we can set $m = \Theta(\frac{\lambda}{\log \lambda})$ and the simulator can create m signatures, which has the same tag as that of the forgery. Since our main scheme in the next subsection is a generalization of the scheme in Figure 1, we omit the detailed security analysis of the scheme in Figure 1.

Remark. We used the combination of the two techniques in this section for signature schemes based on the DH assumption. There is similar approach for signature schemes based on the RSA assumption and q-DH assumption [20]. Note that our original contribution is explained in Section 4.2.

4.2 Asymmetric Trade: Realizing Short Public Key

Let Q, m, and l be functions in λ. For readers who want to see the specific parameters a little early, we give an example parameter below. We will explain about selecting parameters in the last part of this subsection.

Example Parameter 1. $Q = 2^3 q$, $m = \left\lceil \sqrt{\frac{\lambda}{\log \lambda}} \right\rceil$, and $l = \left\lceil \sqrt{\frac{\lambda}{\log \lambda}} \right\rceil$.

[4] M^i is not the i-th bit of M, but the i times product of M.

We first describe a signature scheme which is EUF-q-naCMA secure under the DH assumption in Figure 3. By applying standard techniques using the chameleon hashes and the pseudo-random functions, we can obtain the generalized (non-tag-based) scheme in Figure 2.

```
$\mathsf{Gen}(1^\lambda)$
  $(p, \mathbb{G}, \mathbb{G}_t, e) \leftarrow \mathcal{G}(\lambda)$
  $\alpha \leftarrow \mathbb{Z}_p$
  $g, h, u_0, \ldots, u_m,$
    $z_1, \ldots, z_l \leftarrow \mathbb{G}$
  $sk := (g, \alpha)$
  $pk := (g, g^\alpha, (u_j)_{j=0}^m,$
      $(z_i)_{i=1}^l, h)$
  return $(pk, sk)$
```

```
$\mathsf{Sig}(sk, M)$
  $s \leftarrow \mathbb{Z}_p$
  $\mathbf{u}^M := \prod_{i=0}^m u_i^{M^i}$
  for $i := 1$ to $l$ do
    $t_i \leftarrow [1, Q]$
  $z := \prod_{i=1}^l z_i^{t_i}$
  $\tilde{\sigma}_1 := (\mathbf{u}^M)^\alpha (z \cdot h)^s$
  $\tilde{\sigma}_2 := g^s$
  return $(\tilde{\sigma}_1, \tilde{\sigma}_2, \overrightarrow{t})$
```

```
$\mathsf{Ver}(pk, M, \sigma = (\tilde{\sigma}_1, \tilde{\sigma}_2, \overrightarrow{t}))$
  for $i := 1$ to $l$ do
    if $t_i \notin [1, Q]$
      return 0
  if $e(\tilde{\sigma}_1, g) \neq$
      $e(\mathbf{u}^M, g^\alpha) e(\tilde{\sigma}_2, h \prod_{i=1}^l z_i^{t_i})$
    return 0
  else
    return 1
```

Fig. 3. EUF-q-naCMA secure signature with short public key

For each signature $\sigma = (\tilde{\sigma}_1, \tilde{\sigma}_2, \overrightarrow{t})$, we call $\overrightarrow{t}$ tag vector. In contrast to the basic tag-based scheme, we use a vector $\overrightarrow{t}$ instead of an integer t_1 in signatures. Roughly speaking, our analysis shows that the signature scheme in Figure 3 with $\mathcal{T}_i := [1, Q]$ for $i \in \{1, \ldots, l\}$ satisfies non-adaptive unforgeability (against bounded CMA) when $ml = \Omega(\frac{\lambda}{\log \lambda})$ (this result contains the signatures with somewhat short public key in Figure 1). In addition (roughly speaking again), since the public key size is $\Theta(m + l)$ group elements, we can attain the minimal public key size when m and l are nearly equal. On the other hand, the size of signatures will increase when the parameter l increases. However, each t_i is a $\log Q$-bit integer, and so $\overrightarrow{t}$ is asymptotically much shorter than $\Theta(\lambda)$-bit (if we set Q as a polynomial in λ). This is an *asymmetric trade* between the public key and tag vectors. When we apply the example parameter 1, the signature size will be bounded by two group and a field element, that is, the signature size is $\Theta(\lambda)$ bits. We give precise analysis of the efficiency of the proposed signature scheme in Section 4.2.

Our construction of the short signatures with short public key in Figure 3 is a simple generalization of the basic tag-based scheme (short signatures with somewhat short public key) in Figure 1. However, the analysis of the security in the EUF-q-naCMA model is more challenging than the construct itself. The basic strategy of the simulator in the EUF-q-naCMA model of the signature scheme in Figure 1 is guessing the tag t^* of the forgery and then using the programmability of the weak programmable hash function $(\prod_{i=0}^m u_i^{M^i})$ to sign for the signature with the same tag. We cannot naively apply this proof strategy to the generalized construction. To obtain short public key, we should set l sufficiently large (but not too much). However, if l is large, then the simulator cannot guess the tag vector of the forgery, $t^* \in [1, Q]^l$, with non-negligible probability. That is, we would fail to construct a polynomial-time reduction. We developed a proof technique to resolve this problem.

Our Proof Strategy. We now explain our proof strategy for polynomial-time reduction from solving the DH problem to the breaking the non-adaptive unforgeability of the proposed signature scheme. In particular, we explain the method to guess the tag vector t^* of the forgery with non-negligible probability. In fact, we cannot guess all the bits of t^*, but only part of t^* with non-negligible probability. This is sufficient for our proof strategy.

We begin with defining notations for efficient explanation. Let T and T^i be sets $[1, Q]$ and $[1, Q]^i$ (i times canonical product set), respectively. For $j \in [1, q]$, let $t_j \in T^l$ be the tag vector (randomly chosen by the simulator) of the signature on the jth message (queried by the adversary). Let $\overrightarrow{t}^* = (t_1^*, \ldots, t_l^*) \in T^l$ be the tag vector of the forgery output by the adversary. For $\overrightarrow{t} \in T^l$ and $i \leq l$, let $\overrightarrow{t}^{(i)} \in T^i$ be the first i entries of $\overrightarrow{t}$ (e.g., $\overrightarrow{t} = (t_1, \ldots, t_l)$ and $\overrightarrow{t}^{(i)} = (t_1, \ldots, t_i)$). We separate the adversaries into several types according to the relations between $\overrightarrow{t}^*$ and $\{\overrightarrow{t}_i\}_{i \in [1,q]}$. To this end, for fixed $\{\overrightarrow{t}_i\}_{i \in [1,q]}$, we first define the set T_i as

$$\{\hat{t} \in T^i \mid \exists \text{ at least } (m+1) \text{ distinct } j_1, \ldots, j_{m+1} \in [1, q] \text{ such that } \hat{t} = \overrightarrow{t}^{(i)}_{j_1} = \ldots = \overrightarrow{t}^{(i)}_{j_{m+1}}\}.$$

Let us consider an example to help the readers understand the definition of T_i.

Example. Suppose that

$$\begin{aligned} \overrightarrow{t}^{(i)}_1 = \ldots = \overrightarrow{t}^{(i)}_{m+2} \neq \overrightarrow{t}^{(i)}_j \quad & \text{for } j \in [m+3, q], \\ \overrightarrow{t}^{(i+1)}_1 = \ldots = \overrightarrow{t}^{(i+1)}_{m+1} \neq \overrightarrow{t}^{(i+1)}_j \quad & \text{for } j \in [m+2, q], \end{aligned}$$

and $\overrightarrow{t}^{(i)}_{m+3}, \ldots, \overrightarrow{t}^{(i)}_q$ are distinct. Then,

$$\begin{cases} \overrightarrow{t}^{(i)}_j \in \mathcal{T}_i \text{ for } j \in [1, m+2] \\ \overrightarrow{t}^{(i)}_j \notin \mathcal{T}_i \text{ for } j \in [m+3, q], \end{cases}, \quad \begin{cases} \overrightarrow{t}^{(i+1)}_j \in \mathcal{T}_{i+1} \text{ for } j \in [1, m+1] \\ \overrightarrow{t}^{(i+1)}_j \notin \mathcal{T}_{i+1} \text{ for } j \in [m+2, q], \end{cases}$$

and $|T_i| = |T_{i+1}| = 1$. □

We can easily see that $|T_{i+1}| \leq |T_i|$. Let n be the largest integer in $[1, l]$ such that $T_n \neq \emptyset$. If we choose m, l, and Q appropriately, we then obtain the following two properties with overwhelming probability, where the probability is taken over the choice of $\{\overrightarrow{t}_i\}_{i \in [1,q]}$.

1. $|\mathcal{T}_1| < \lambda$
2. $n < l$ (equivalently $T_l = \emptyset$, that is, $|T_l| < 1$)

When $Q \geq q$, the following lemma implies the above two properties. (e.g., we obtain the above properties when we apply the example parameter 1 to Lemma 2.)

Lemma 2. $\Pr_{\overrightarrow{t}_1, \ldots, \overrightarrow{t}_q \leftarrow T_l}[|T_i| \geq j] < (\frac{q^{m+1}}{(m+1)! Q^{im}})^j$.

Proof. Let F be the set of all functions from $[1, q]$ to S^i. For $\overrightarrow{y} \in S^i$ and $f \in F$, let $|f^{-1}(\overrightarrow{y})|$ be the number of the distinct pre-images of $\overrightarrow{y}$. Let T_f be the set of all

$\overrightarrow{y} \in Im(f)$ such that $|f^{-1}(\overrightarrow{y})| \geq m+1$, where $Im(f)$ means the set of all images of f. Then, we can consider $\Pr_{\overrightarrow{s}_1,\ldots,\overrightarrow{s}_q \leftarrow S^k}[|S_i| \geq j]$ as

$$\Pr_{f \leftarrow F}[|T_f| \geq j].$$

To compute $\Pr_{f \leftarrow F}[|T_f| \geq j]$, we count all functions f such that $|T_f| \geq j$, then divide the result by $|S^i|^q$ (the number of all elements in F). In fact, we count the number of f such that $|T_f| \geq j$, allowing duplications, so that we compute the upper bound of $\Pr_{f \leftarrow F}[|T_f| \geq j]$. To define an f, we choose j distinct subsets $A_1,\ldots,A_j$ of size $m+1$ from $[1,q]$ and j distinct vectors $\overrightarrow{y}_1,\ldots,\overrightarrow{y}_j$ from S^i, and then set $f(a) = \overrightarrow{y}_t$ for all $a \in A_t$ and $t \in [1,j]$. For other integers $a \in [1,q] \backslash (A_1 \cup \ldots \cup A_j)$, we arbitrarily define $f(a)$. This way of defining a function covers all f such that $|T_f| \geq j$. We count all f that are defined as above. Then, the number of such f is bounded by

$$\Big(\prod_{t=0}^{j-1} \binom{q-t(m+1)}{m+1} \cdot (|S^i| - t)\Big) \cdot (|S^i|)^{(q-j(m+1))},$$

where the notation $\binom{\cdot}{\cdot}$ denotes the binomial coefficient.

Therefore, we can obtain the desired result as follows:

$$\begin{aligned}
\Pr_{\overrightarrow{s}_1,\ldots,\overrightarrow{s}_q \leftarrow S^k}[|S_i| \geq j] &= \Pr_{f \leftarrow F}[|T_f| \geq j] \\
&< \frac{\Big(\prod_{t=0}^{j-1} \binom{q-t(m+1)}{m+1} \cdot (Q^i - t)\Big) \cdot (Q^i)^{(q-j(m+1))}}{|S^i|^q} \\
&< \frac{\Big(\frac{q^{m+1}}{(m+1)!}\Big)^j Q^{ij+i(q-j(m+1))}}{Q^{iq}} \\
&= \big(\tfrac{q^{m+1}}{(m+1)!Q^{im}}\big)^j.
\end{aligned}$$

□

For now, let us assume that we have m, l, and Q such that the above two properties hold. We separate the types of adversaries according to $\overrightarrow{t}^*$ as follows.

$$\begin{aligned}
\text{Type-1}: &\ \overrightarrow{t}^{*(1)} \notin T_1. \\
\text{Type-2}: &\ \overrightarrow{t}^{*(1)} \in T_1, \text{ and } \overrightarrow{t}^{*(2)} \notin T_2. \\
&\ \vdots \\
\text{Type-}i: &\ \overrightarrow{t}^{*(i-1)} \in T_{i-1}, \text{ and } \overrightarrow{t}^{*(i)} \notin T_i. \\
&\ \vdots \\
\text{Type-}n: &\ \overrightarrow{t}^{*(n-1)} \in T_{n-1}, \text{ and } \overrightarrow{t}^{*(n)} \notin T_n. \\
\text{Type-}(n+1): &\ \overrightarrow{t}^{*(n)} \in T_n.
\end{aligned}$$

Here, $\overrightarrow{t}^{*(i-1)} \in T_{i-1}$ implies that $\overrightarrow{t}^{*(j)} \in T_j$ for all $j \in [1, i-1]$. Therefore, we can see that the above $n+1$ types of adversaries are pairwise disjoint and cover all possible adversaries. For the type-i adversary, the simulator can guess $\overrightarrow{t}^{*(i)}$ with probability $\frac{1}{|T_{i-1}|\cdot|T|}$; it guesses $\overrightarrow{t}^{*(i-1)}$ with $\frac{1}{|T_{i-1}|}$ and t_i^* with $\frac{1}{|T|}$ (we use the second property for the case $i = n+1$). Since the simulator can guess the type of the adversary with probability $\frac{1}{l}$, it can guess the tag vector of the forgery with at least probability $\frac{1}{l\lambda Q}$ (we use the first property for the inequality $|T_{i-1}| \leq |T_1| < \lambda$).

The other parts of the proof strategy are similar to the strategy for the short signatures with somewhat short public key in Figure 1 as we mentioned in Section 4.1; (1) guess $\overrightarrow{t}^{*(i)}$, (2) use the proof technique for the reduction from solving the DH problem to breaking the selectively secure signatures, and (3) generate for the signature with the same tag vector as $t^{*(i)}$, using the programmability of the weak programmable hash functions. Since $\overrightarrow{t}^{*(i)} \notin T_i$ (for $i = n+1$, $\overrightarrow{t}^{*(i)} \notin T_i = \emptyset$) implies that there are at most m tag vectors same as $\overrightarrow{t}^{*(i)}$, the simulator can response m signatures with tag vector $\overrightarrow{t}^{*(i)}$ using the programmability of $(\prod_{i=0}^{m} u_i^{M^i})$. If $l\lambda Q$ is bounded by a polynomial in λ, then we obtain the polynomial-time reduction.

By applying the above strategy, we give the following theorem.

Theorem 1. *The signature scheme in Figure 3 is (q, ε, T)-EUF-naCMA secure assuming the $(\varepsilon_{dh}, T_{dh})$-DH assumption holds such that*

$$\varepsilon_{dh} = \frac{1}{l\lambda Q}(\varepsilon - \frac{q^{m+1}}{(m+1)!Q^{lm}} - \frac{q}{p} - (\frac{q^{m+1}}{(m+1)!Q^m})^\lambda) \quad and \quad T \approx T_{dh}.$$

Because of space constraints, we relegate the proof Theorem 1 in [30].

To derive a meaningful result about the asymptotic security from Theorem 1, we need the following three conditions.

Condition 1. $l\lambda Q$ is polynomially bounded in λ.
Condition 2. $\frac{q^{m+1}}{(m+1)!Q^{lm}}$ is a negligible function in λ.
Condition 3. $(\frac{q^{m+1}}{(m+1)!Q^m})^\lambda$ is a negligible function in λ.

We give asymptotic values of m, l, and Q for satisfying the above conditions and short public key in Section 4.2. For such parameters (e.g., example parameter 1), we obtain the following corollary by applying the generic transformation using the chameleon hashes.

Corollary 1. *Let* SIG *be the signature scheme resulting from the generic transformation on the signature scheme in Figure 3 and* CHSetup*. Then,* SIG *is $(q, 2(\varepsilon + \varepsilon_{\mathsf{CH}}), T)$-EUF-CMA secure assuming $(\frac{\varepsilon}{l\lambda Q} - neg(\lambda), T_{dh})$-DH assumption holds, where $\varepsilon_{\mathsf{CH}}$ is the advantage for breaking the collision resistance of* CHSetup*, $T \approx T_{dh}$.*

Tag-Free Scheme by Pseudorandom Functions. We apply a trick for tag-free scheme using (non-adaptive) pseudorandom functions (PRF). Note that similar techniques are used in the RSA-based signatures [23, 22, 20, 34] to generate random prime numbers used in each signature. If we use this trick (and the chameleon hashes), we can obtain the generalized (tag-based) signature scheme in Figure 2, where $\forall \mathcal{T}_i = [1, Q]$; each signature has a tag vector that is uniformly chosen from its domain. Thus, a signer can use pseudorandom functions (PRF) mapping from messages to tag vectors, and publishes the PRF the signer used along with its key. Even though the signer publishes the PRF key, (in the non-adaptive security model) we can use the fact that the distribution of tag vectors is indistinguishable from the uniform distribution. The resulting signature size is reduced by eliminating tag vectors from signatures but augmenting signing/verification costs and adding constant factor in public key size (that is, public key size is still $\Theta(\sqrt{\frac{\lambda}{\log\lambda}})$ group elements);

Corollary 2. *Let* SIG *be the signature scheme in Figure 2 with* $\forall \mathcal{T}_i = [1, Q]$. *Then,* SIG *is* $(q, 2(\varepsilon + \varepsilon_{\mathsf{CH}} + \varepsilon_{\mathsf{PRF}}), T)$*-EUF-CMA secure assuming* $(\frac{\varepsilon}{l\lambda Q} - neg(\lambda), T_{dh})$*-DH assumption holds, where* $\varepsilon_{\mathsf{CH}}$ *and* $\varepsilon_{\mathsf{PRF}}$ *are advantages for breaking the collision resistance of the chameleon hash functions and the pseudo-randomness of PRF, respectively,* $T \approx T_{dh}$, *and* $neg(\lambda)$ *is a negligible function in* λ.

Parameter Selection for Short Public Key. The description of our construction did not explain how to choose m, l, and Q. We show how to minimize public key size. From Theorem 1, we obtained three conditions for polynomial-time reduction to the DH problem. First, $l\lambda Q$ should be polynomially bounded in λ. Second, $\frac{q^{m+1}}{(m+1)!Q^{lm}}$ and $(\frac{q^{m+1}}{(m+1)!Q^{m}})^{\lambda}$ should be a negligible function in λ. For simple analysis, we assume that $Q = Cq$ for small constant $C > 1$ and compute conditions for m and l when $\frac{q^{m+1}}{(m+1)!Q^{lm}}$ and $(\frac{q^{m+1}}{(m+1)!Q^{m}})^{\lambda}$ are smaller than $\frac{1}{2^{\lambda}}$. We compute asymptotically minimal values of m and l for short public key size, and then provide practical parameters with reasonable reduction loss, which is comparable to that of Waters signature in [33].

Condition 1. $l\lambda q$ is polynomially bounded in λ.
Condition 2. $\frac{q^{m+1}}{(m+1)!Q^{lm}} < \frac{1}{2^{\lambda}}$.
Condition 3. $(\frac{q^{m+1}}{(m+1)!Q^{m}})^{\lambda} < \frac{1}{2^{\lambda}}$.

From the condition 2, at least the denominator should be larger than 2^{λ}. Since $Q = Cq$ and $(m+1)! \approx \sqrt{2\pi(m+1)}(\frac{m+1}{e})^{m+1}$ (by Stirling's approximation), where e is the Euler's number, $lm = \Omega(\frac{\lambda}{\log\lambda})$ or $m = \Omega(\frac{\lambda}{\log\lambda})$. For minimizing public key size, we should minimize $m + l$ since the size of public key is $\Theta(m + l)$. Therefore, $m = \Theta(\sqrt{\frac{\lambda}{\log\lambda}})$ and $l = \Theta(\sqrt{\frac{\lambda}{\log\lambda}})$ are (asymptotically) minimal parameters for minimal public key size. In fact, if we set $m = \Theta(\sqrt{\frac{\lambda}{\log\lambda}})$ and $l = \Theta(\sqrt{\frac{\lambda}{\log\lambda}})$, then the condition 1 and 3 also hold.

Next, we provide practical parameters for $\lambda \in \{80, 256\}$ and $q \in \{2^{30}, 2^{40}\}$, where λ is the security parameter and q is the bound for adversarial signing queries. If the above condition 2 and condition 3 hold, our security proof loses $4l\lambda Q$ factor in the simulation (when we ignore negligible factors), which is asymptotically larger than Waters signature scheme's reduction loss. However, for practical choices of λ and q, l is a small constant (at most 3 in our example parameters) and Q is Cq with small constant ($C = 2^3$ in our example parameters); and thus, the reduction loss in our example parameters is at most $96\lambda q$, which is comparable to that given in [33][5]. The example

[5] Hofheinz et al. proposed a variant of Waters signatures using a special encoding for optimal security reduction $\Theta(\frac{1}{q})$ [21]. In [21], however, they do not provide a concrete constant factor of Θ notation for practical security parameters, but only asymptotic analysis for optimal security reduction.

Table 1. Practical parameters for EUF-q-CMA secure signature scheme: '$\tau_{\mathbb{G}_1}$' and '$\tau_{\mathbb{G}_2}$' are the bit-lengths to represent elements in $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. For type-1 pairings, $\tau_{\mathbb{G}_1} = \tau_{\mathbb{G}_2}$. '$\tau_{\mathbb{Z}_p}$' is the size of prime p that is order of cyclic groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_t$. '$\tau_{\mathbb{G}_{ch}}$' is the bit length to represent an element in the group $\mathbb{G}_{ch}$ (of order $p' \leq p$), over which the chameleon hashes defined. '$|K|$' is a size of a PRF key.

Security Parameter λ		q	m	l	PK size	Sig. size
80	tag-based scheme	2^{30}	7	2	$12\tau_{\mathbb{G}_2} + \tau_{\mathbb{G}_1} + 2\tau_{\mathbb{G}_{ch}}$	$2\tau_{\mathbb{G}_1} + 2\tau_{\mathbb{Z}_p}$
		2^{40}	8	2	$13\tau_{\mathbb{G}_2} + \tau_{\mathbb{G}_1} + 2\tau_{\mathbb{G}_{ch}}$	$2\tau_{\mathbb{G}_1} + 2\tau_{\mathbb{Z}_p}$
	tag-free scheme	2^{30}	7	2	$12\tau_{\mathbb{G}_2} + \tau_{\mathbb{G}_1} + 2\tau_{\mathbb{G}_{ch}} + \|K\|$	$2\tau_{\mathbb{G}_1} + \tau_{\mathbb{Z}_p}$
		2^{40}	8	2	$13\tau_{\mathbb{G}_2} + \tau_{\mathbb{G}_1} + 2\tau_{\mathbb{G}_{ch}} + \|K\|$	$2\tau_{\mathbb{G}_1} + \tau_{\mathbb{Z}_p}$
256	tag-based scheme	2^{30}	7	3	$13\tau_{\mathbb{G}_2} + \tau_{\mathbb{G}_1} + 2\tau_{\mathbb{G}_{ch}}$	$2\tau_{\mathbb{G}_1} + 2\tau_{\mathbb{Z}_p}$
		2^{40}	8	2	$13\tau_{\mathbb{G}_2} + \tau_{\mathbb{G}_1} + 2\tau_{\mathbb{G}_{ch}}$	$2\tau_{\mathbb{G}_1} + 2\tau_{\mathbb{Z}_p}$
	tag-free scheme	2^{30}	7	3	$13\tau_{\mathbb{G}_2} + \tau_{\mathbb{G}_1} + 2\tau_{\mathbb{G}_{ch}} + \|K\|$	$2\tau_{\mathbb{G}_1} + \tau_{\mathbb{Z}_p}$
		2^{40}	8	2	$13\tau_{\mathbb{G}_2} + \tau_{\mathbb{G}_1} + 2\tau_{\mathbb{G}_{ch}} + \|K\|$	$2\tau_{\mathbb{G}_1} + \tau_{\mathbb{Z}_p}$

practical parameters are given in the table 1. To get the table 1, we firstly set $Q = 2^3 q$ and $q \in \{2^{30}, 2^{40}\}$, and then find small m and l satisfying the above three conditions. The size of a tag vector is a $l\lceil \log Q \rceil$-bit string, which is asymptotically smaller than 2λ if $l = \Theta(\sqrt{\frac{\lambda}{\log \lambda}})$ and Q is a polynomial in λ; and thus, we can assume that a tag vector is a field element of $\mathbb{Z}_p$. In particular, when we apply practical parameters in the table 1, the size of a tag vector is still smaller than 2λ (e.g., a tag vector is 129-bit string when $l = 3$ and $Q = 2^{43}$).

Instantiation Using Asymmetric Pairings. Although we described our construction using type-1 pairings, we can easily modify our construction to be instantiated using type-2 pairings or type-3 parings. The scheme using type-1 pairings and its security proof does not use the symmetry property; Our main idea to achieve sub-linear public key is to divide adversarial types according to tag vectors in the security proof, and this technique is independent of pairing's type. We provide the details of the scheme using type-2 or type-3 parings in [30].

5 Confined Guessing

Overview. In this section, we will explain the schemes from Section 3 as arising from a more general transformation. The final schemes (and in particular $\mathsf{SIG}_{\mathsf{gen}}^{\mathsf{CDH}}$) are fully EUF-CMA secure (in contrast to EUF-q-CMA security), and have asymptotically very short public keys. The downside of this analysis is that the security reduction is qualitatively worse than the one from Section 4. Namely, the reduction *loss* depends on the adversary's success. Nonetheless, the reduction loss is always polynomial for polynomially bounded adversaries.

5.1 From Mild to Full Security

We start with a completely generic transformation from mildly secure tag-based signature schemes to fully secure schemes. We first define a mild security notion for tag-based schemes, dubbed EUF-naCMA$^*_m$ security, which requires an adversary F to initially specify all messages M_i it wants signed, along with corresponding tags t_i. Only then, F gets to see a public key, and is subsequently expected to produce a forgery for an arbitrary fresh message M^*, but with respect to an already used tag $t^* \in \{t_i\}_i$. As a slightly technical (but crucial) requirement, we only allow F to initially specify at most m messages M_i with tag $t_i = t^*$. We call m the *tag-collision parameter*; it influences key and signature sizes, and the security reduction.

Definition 1 (EUF-naCMA$^*_m$ security). *Let $m \in \mathbb{N}$. A tag-based signature scheme* $\mathsf{SIG_t}$ *is existentially unforgeable under non-adaptive chosen-message attacks with m-fold tag-collisions (short: EUF-naCMA$^*_m$ secure) iff the function* $\mathsf{Adv}^{\mathsf{euf\text{-}nacma}^*_\mathsf{m}}_{\mathsf{SIG_t},F}(\lambda) := \Pr\left[\mathsf{Exp}^{\mathsf{euf\text{-}nacma}^*_\mathsf{m}}_{\mathsf{SIG_t},F}(\lambda) = 1\right]$ *is negligible for any PPT adversary F. Here, experiment* $\mathsf{Exp}^{\mathsf{euf\text{-}nacma}^*_\mathsf{m}}_{\mathsf{SIG_t},F}(\lambda)$ *is defined in Figure 4.*

```
Experiment $\mathsf{Exp}^{\mathsf{euf\text{-}nacma}^*_\mathsf{m}}_{\mathsf{SIG_t},F}(\lambda)$
  $(M_j, t_j)_{j=1}^{q(\lambda)} \leftarrow F(1^\lambda)$
  $(pk, sk) \leftarrow \mathsf{Gen_t}(1^\lambda)$
  $\sigma_j \leftarrow \mathsf{Sig_t}(sk, M_j, t_j)$ for $j \in [q(\lambda)]$
  $(M^*, \sigma^*, t^*) \leftarrow F(pk, (\sigma_j)_{j=1}^{q(\lambda)})$
  if $\mathsf{Ver_t}(pk, M^*, \sigma^*, t^*) = 0$
     or $M \in \{M_j\}_{j=1}^{q(\lambda)}$
     or $|\{j \in [q(\lambda)] : t_j = t^*\}| > m$
     or $t \notin \{t_j\}_{j=1}^{q(\lambda)}$
  then return 0, else return 1
```

Fig. 4. The EUF-naCMA$^*_m$ experiment for tag-based signature schemes

In this subsection, we will show how to use a EUF-naCMA$^*_m$ secure scheme $\mathsf{SIG_t}$ to build an EUF-naCMA secure scheme SIG. (Full EUF-CMA security can then be achieved using chameleon hashing [26].)

To this end, we separate the tag space $\mathcal{T}_\lambda$ into $l := \lfloor \log_c(\lambda) \rfloor$ *pairwise disjoint* sets $\mathcal{T}'_i$, such that $|\mathcal{T}'_i| = 2^{\lceil c^i \rceil}$. Here $c > 1$ is a *granularity parameter* that will affect key and signature sizes, and the security reduction. For instance, if $c = 2$ and $\mathcal{T}_\lambda = \{0,1\}^\lambda$, then we may set $\mathcal{T}'_i := \{0,1\}^i$. The constructed signature scheme SIG assigns to each message M a vector of tags $(t_1, \ldots, t_l)$, where each tag is derived from the message M by applying a pseudorandom function as $t_i := \mathsf{PRF}^{\mathcal{T}'_i}_\kappa(M)$. The PRF seed κ is part of SIG's public key.[6]

A SIG-signature is of the form $\sigma = (\sigma_i)_{i=1}^l$, where each $\sigma_i \leftarrow \mathsf{Sig_t}(sk, M, t_i)$ is a signature according to $\mathsf{SIG_t}$ with message M and tag t_i. This signature is considered valid if all σ_i are valid w.r.t. $\mathsf{SIG_t}$.

The crucial idea is to define the sets $\mathcal{T}'_i$ of allowed tags as sets quickly growing in i. This means that $(m+1)$-tag-collisions (i.e., the same tag t_i being chosen for $m+1$ different signed messages) are very likely for small i, but become quickly less likely for larger i. Concretely, let $\mathsf{SIG_t} = (\mathsf{Gen_t}, \mathsf{Sig_t}, \mathsf{Ver_t})$ be a tag-based signature scheme with

[6] It will become clear in the security proof that actually a function with weaker security properties than a fully-secure PRF is sufficient for our application. However, we stick to standard PRF security for simplicity. Thanks to an anonymous reviewer for pointing this out.

```
$\mathsf{Gen}(1^\lambda)$
    $(pk', sk) \leftarrow \mathsf{Gen}_\mathsf{t}(1^\lambda)$
    $\kappa \leftarrow \{0,1\}^\lambda$
    $pk := (pk', \kappa)$
    return $(pk, sk)$
```

```
$\mathsf{Sig}(sk, M)$
    $t_i := \mathsf{PRF}^{\mathcal{T}_i}_\kappa(M)$ for $i \in [l]$
    $\sigma_i \leftarrow \mathsf{Sig}_\mathsf{t}(sk, M, t_i)$
    return $\sigma := (\sigma_i)_{i=1}^l$
```

```
$\mathsf{Ver}((pk', \kappa), M, \sigma = (\sigma_i)_{i=1}^l)$
    $t_i := \mathsf{PRF}^{\mathcal{T}_i}_\kappa(M)$ for $i \in [l]$
    return $\bigwedge_{i=1}^{l} \mathsf{Ver}_\mathsf{t}(pk', M, \sigma_i, t_i)$
```

Fig. 5. Our EUF-naCMA secure signature scheme

tag space $\mathcal{T}_\lambda = \bigcup_{i=1}^l \mathcal{T}_i'$, let $m \in \mathbb{N}$ and $c > 1$, and let PRF be a PRF. SIG is described in Figure 5.

It is straightforward to verify SIG's correctness. Before turning to the formal proof, we first give an intuition why SIG is EUF-naCMA secure. We will map an adversary F on SIG's EUF-naCMA security to an adversary F' on SIG_t's EUF-naCMA$^*_m$ security. Intuitively, F' will internally simulate the EUF-naCMA security experiment for F and embed its own SIG_t-instance (with public key pk') in the SIG-instance of F by setting $pk := (pk', \kappa)$. Additionally, the seed κ for PRF is chosen internally by F'.

Say that F makes $q = q(\lambda)$ (non-adaptive) signing requests for messages M_j, for all $j \in [q]$. To answer these q requests, F' can obtain signatures under pk' from its own EUF-naCMA$^*_m$ experiment. The corresponding tags are chosen as in SIG, as $t_i^{(j)} = \mathsf{PRF}^{\mathcal{T}_i}_\kappa(M_j)$. Once F produces a forgery $\sigma^* = (\sigma_i^*)_{i=1}^l$, F' will try to use $\sigma^*_{i^*}$ (with tag $t^*_{i^*} = \mathsf{PRF}^{\mathcal{T}_{i^*}}_\kappa(M^*)$ for some appropiate $i^* \in [l]$) as its own forgery.

Indeed, $\sigma^*_{i^*}$ will be a valid SIG_t-forgery (in the EUF-naCMA$^*_m$ experiment) if (a) F' did not initially request signatures for more than m messages for the forgery tag $t^*_{i^*}$, and (b) $t^*_{i^*}$ already appears in one of F''s initial signature requests. Our technical handle to make this event likely will be a suitable choice of i^*. First, recall that the i-th SIG_t-instance in SIG uses $\lceil c^i \rceil$-bit tags. We will hence choose i^* such that

(i) the probability of an $(m+1)$-tag-collision among the $t_{i^*}^{(j)}$ is significantly lower than F's success probability (so F will sometimes have to forge signatures when no $(m+1)$-tag collision occurs), and

(ii) $|\mathcal{T}'_{i^*}| = 2^{\lceil c^{i^*} \rceil}$ is polynomially small (so all tags in $\mathcal{T}'_{i^*}$ can be queried by F').

We turn to a formal proof:

Theorem 2. *If* PRF *is a PRF and* SIG_t *is an EUF-naCMA$^*_m$ secure tag-based signature scheme, then* SIG *is EUF-naCMA secure. Concretely, let F be an EUF-naCMA forger on* SIG *with non-negligible advantage* $\varepsilon := \mathsf{Adv}^{\mathsf{euf\text{-}nacma}}_{\mathsf{SIG},F}(\lambda)$ *and making $q = q(\lambda)$ signature queries. Then $\varepsilon(\lambda) > \frac{1}{p(\lambda)}$ for some polynomial p and $\lambda \in K$ for an infinite set $K \subseteq \mathbb{N}$. For $\lambda \in K$ there exists a EUF-naCMA$^*_m$ forger F' on* SIG_t *with advantage* $\varepsilon' := \mathsf{Adv}^{\mathsf{euf\text{-}nacma}^*_\mathsf{m}}_{\mathsf{SIG}_\mathsf{t},F'}(\lambda)$ *and making* $q'(\lambda) \leq \left(\frac{q^{m+1}}{\varepsilon(\lambda)}\right)^{c/m}$ *signature queries, such that* $\varepsilon' \geq \varepsilon/2 - \varepsilon_{\mathsf{PRF}} - \frac{1}{|\mathcal{M}_\lambda|}$*, where $\varepsilon_{\mathsf{PRF}}$ is the advantage of a suitable PRF distinguisher on* PRF *and $\mathcal{M}_\lambda$ the message space.*

Proof. First, F' receives messages $M_1, \ldots, M_q$ from F. Let $\varepsilon(\lambda)$ be F's advantage in the EUF-naCMA experiment. F' chooses the challenge instance i^* such that the probability of an $(m+1)$-tag collision is at most $\varepsilon(\lambda)/2$, i.e.,

$$\Pr\left[\exists\{j_0,\ldots,j_m\}\subseteq[q]: t_{i^*}^{(j_0)}=\ldots=t_{i^*}^{(j_m)} \mid \forall j\in[q]: t_{i^*}^{(j)}\leftarrow\mathcal{T}_{i^*}\right]\leq\frac{\varepsilon(\lambda)}{2}, \quad (1)$$

and such that $|\mathcal{T}'_{i^*}|$ is polynomial in λ. Concretely, $i^* := \lceil\log_c(\log_2((\frac{q^{m+1}}{\varepsilon(\lambda)})^{1/m}))\rceil$ is an index that fulfills these conditions. (See [8, Lemma 3.5] for a complete analysis.) F' then chooses a PRF-key $\kappa \leftarrow \{0,1\}^\lambda$.

Recall that a signature $\sigma = (\sigma_1,\ldots,\sigma_l)$ of SIG consists of l signatures of $\mathsf{SIG_t}$. In the sequel we write $\sigma^{(j)} = (\sigma_1^{(j)},\ldots,\sigma_l^{(j)})$ to denote the SIG-signature for message M_j, for all $j \in \{1,\ldots,q\}$. Adversary F' uses its signing oracle provided from the $\mathsf{SIG_t}$-security experiment to simulate these SIG-signatures. To this end, it proceeds as follows.

Simulation of Signatures. In order to simulate all signatures $\sigma_i^{(j)}$ with $i \neq i^*$, F' computes $t_i^{(j)} := \mathsf{PRF}_\kappa^{\mathcal{T}'_i}(M_j)$ and defines message-tag pair $(M_j, t_i^{(j)})$. F' will later request signatures for these message-tag pairs from its EUF-naCMA$^*_m$-challenger. Note that $t_i^{(j)} \notin \mathcal{T}'_{i^*}$ for all $i \neq i^*$, since the sets $\mathcal{T}'_1,\ldots,\mathcal{T}'_l$ are pairwise disjoint.

To compute the i^*-th $\mathsf{SIG_t}$-signature $\sigma_{i^*}^{(j)}$ contained in $\sigma^{(j)}$, F' proceeds as follows. First it computes $t_{i^*}^{(j)} := \mathsf{PRF}_\kappa^{\mathcal{T}'_{i^*}}(M_j)$ for all $j \in \{1,\ldots,q\}$. If a $(m+1)$-fold tag-collision occurs, then F' aborts. This defines q message-tag-pairs (M_j, t_j) for $j \in \{1,\ldots,q\}$. Note that the list $(t_{i^*}^{(1)},\ldots,t_{i^*}^{(q)})$ need not contain all elements of $\mathcal{T}'_{i^*}$, that is, it might hold that $\mathcal{T}'_{i^*} \setminus \{t_{i^*}^{(1)},\ldots,t_{i^*}^{(q)}\} \neq \emptyset$. If this happens, then F' chooses a dummy message $M \leftarrow \mathcal{M}_\lambda$ uniformly at random and associates it with all tags $t \in \mathcal{T}'_{i^*} \setminus \{t_{i^*}^{(1)},\ldots,t_{i^*}^{(q)}\}$ that are not contained in $\{t_{i^*}^{(1)},\ldots,t_{i^*}^{(q)}\}$. This defines further message-tag- pairs (M,t) for each $t \in \mathcal{T}'_{i^*} \setminus \{t_{i^*}^{(1)},\ldots,t_{i^*}^{(q)}\}$. We do this since F' has to re-use an already queried tag for a valid forgery later and F' does not know at this point which tag F is going to use in his forgery later.

Finally F' requests signatures for all message-tag-pairs from its challenger, and receives in return signatures $\sigma_{i^*}^{(j)}$ for all j, as well as a public key pk'.

F' defines $pk := (pk', \kappa)$ and hands $(pk, \sigma^{(1)},\ldots,\sigma^{(q)})$ to F. Note that each $\sigma^{(j)}$ is a valid SIG- signature for message M_j.

Extraction. Suppose F eventually generates a forged signature $\sigma^* = (\sigma_i^*)_{i=1}^l$ for a fresh message $M^* \notin \{M_1,\ldots,M_q\}$. If $M^* = M$, then F' aborts. Otherwise it forwards $((\sigma_{i^*}^*, \mathsf{PRF}_\kappa^{\mathcal{T}'_{i^*}}(M^*)), M^*)$ to the EUF-CMA$^*_m$ challenger.

This concludes the description of F'.

Analysis. Let $\mathsf{bad_{abort}}$ be the event that F' aborts. It is clear that F' successfully forges a signature whenever F does so, and $\mathsf{bad_{abort}}$ does not occur. Note that message M is independent of the view of F, thus we have $\Pr[M = M^*] \leq 1/|\mathcal{M}_\lambda|$. Hence, to prove our theorem, it suffices to show that $\Pr[\mathsf{bad_{abort}}] \leq \varepsilon/2 + \varepsilon_{\mathsf{PRF}} + 1/|\mathcal{M}_\lambda|$ since this leaves a non-negligible advantage for F'.

First note that the probability of an $(m+1)$-tag collision would be at most $\varepsilon/2$ by (1) if the tags $t_{i^*}^{(j)}$ were chosen truly uniformly from $\mathcal{T}'_{i^*}$. Now recall that the actual choice of the $t_{i^*}^{(j)} = \mathsf{PRF}_\kappa^{\mathcal{T}'_{i^*}}(M_j)$ was performed in a way that uses PRF only in a black-box way. Hence, if $(m+1)$-tag collisions (and thus $\mathsf{bad}_{\mathsf{abort}}$) occurred significantly more often than with truly uniform tags, we had a contradiction to PRF's pseudorandomness. Concretely, a PRF distinguisher that simulates F' until the decision to abort is made shows $\Pr[\mathsf{bad}_{\mathsf{abort}}] \leq \varepsilon/2 + \varepsilon_{\mathsf{PRF}} + 1/|\mathcal{M}_\lambda|$, and thus the theorem. □

In order to obtain a fully EUF-CMA secure signature scheme, one may combine our EUF-naCMA-secure scheme with a suitable chameleon hash function or a one-time signature scheme. This is a very efficient standard construction, see for instance [22, Lemma 2.3] for details.

5.2 The CDH-Based Scheme

In this subsection we explain the schemes from Section 3 in the our confined guessing framework. We start with with the tag-based scheme $\mathsf{SIG}^{\mathsf{CDH}}$ (Figure 1) and prove it EUF-naCMA$^*_m$-secure. Then we can apply our generic transformation from Section 5.1 to achieve full EUF-CMA security. Finally, we illustrate some optimizations that allow us to reduce the size of public keys and signatures, for instance by aggregation. The result is essentially the generic scheme $\mathsf{SIG}^{\mathsf{CDH}}_{\mathsf{gen}}$.

Theorem 3. *If CDH holds in $\mathbb{G}$, then $\mathsf{SIG}^{\mathsf{CDH}}$ from Figure 1 is EUF-naCMA$^*_m$-secure. Let F be a PPT adversary with advantage $\varepsilon := \varepsilon(\lambda) := \mathsf{Adv}^{\mathsf{euf\text{-}nacma}^*_{\mathsf{m}}}_{\mathsf{SIG}^{\mathsf{CDH}},F}(\lambda)$ and runtime T asking for $q = q(\lambda)$ signatures, then it can be used to solve a CDH challenge with probability $\epsilon_{dh} = (\varepsilon \cdot m)/q$ and $T_{dh} \approx T$.*

Proof sketch. Because of space constraints we will only sketch the proof here. The complete proof can be found in [8].

Public Key Setup. The simulation receives a CDH challenge (g, g^a, g^b) and signature queries $(M_i, t_i)_{i \in [q]}$.We guess an index $i^* \leftarrow [q]$ for which we expect F to forge a signature on a new message $M^* \neq M_{i^*}$, but with $t^* = t_{i^*}$. Let M^*_j (for $j \in [m]$) denote the corresponding messages to t_{i^*}. We set up a polynomial $f(X) := \prod_{i=1}^m (X - M^*_i) = \sum_{i=0}^m d_i X^i \in \mathbb{Z}_p[X]$ and choose random exponents $r_0, \ldots, r_m, x_z, x_h \in \mathbb{Z}_p$. Write $r(X) := \sum_{i=0}^m r_i X^i$, so $\mathbf{u}^M = g^{bf(M)+r(M)}$. We embed our challenge, the coefficients d_i, t_{i^*} and the random exponents in the public key as follows: we set $pk := (g, g^a, u_0, \ldots, u_m, z, h)$ for $u_i := (g^b)^{d_i} g^{r_i}$, $z := g^{b+x_z}$, and $h := g^{-bt_{i^*}} g^{x_h}$. (Observe that this implicitly sets $sk := a$.)

Signing. We have to consider two cases. If $t_i = t_{i^*}$ and thus $M_i = M^*_j$ for some j, we choose a random $s_i \leftarrow \mathbb{Z}_p$ and set $\tilde{\sigma}_{1,i} = (g^a)^{r(M^*_j)} \cdot (z^{t_{i^*}} h)^{s_i}$, $\tilde{\sigma}_{2,i} = g^{s_i}$. Since $f(M^*_j) = 0$, this signature is valid:

$$\tilde{\sigma}_{1,i} = (g^a)^{r(M^*_j)} \cdot (z^{t_{i^*}} h)^{s_i} = (g^{bf(M^*_j)} g^{r(M^*_j)})^a \cdot (z^{t_{i^*}} h)^{s_i} = (\mathbf{u}^{M^*_j})^a \cdot (z^{t_{i^*}} h)^{s_i}.$$

If $t_i \neq t_{i^*}$, let $s'_i \leftarrow \mathbb{Z}_p$ and $S_i := g^{s'_i}/(g^a)^{f(M_i)/(t_i - t_{i^*})} = g^{s'_i - af(M_i)/(t_i - t_{i^*})}$. A valid signature $\sigma_i := (\tilde{\sigma}_{1,i}, \tilde{\sigma}_{2,i})$ can then be computed as follows: $\tilde{\sigma}_{1,i} = (g^a)^{r(M_i)} \cdot S_i^{x_z t_i + x_h} \cdot (g^b)^{s'_i(t_i - t_{i^*})}$ and $\tilde{\sigma}_{2,i} = S_i$.

Extract from Forgery. If F generates a valid signature (M^*, t^*, σ^*) for $t^* = t_{i^*}$, then $\tilde{\sigma}_1^* = ((g^b)^{f(M^*)}(g^{r(M^*)}))^a((g^{b+x_z})^{t^*}(g^{x_h - bt_{i^*}}))^{s^*} = g^{abf(M^*)}g^{ar(M^*)}g^{s^*(x_z t^* + x_h)}$ As $M^* \neq M_j^*$, we have $f(M^*) \neq 0$, so $(\tilde{\sigma}_1^* / (g^{ar(M^*)}\tilde{\sigma}_2^{*(x_z t^* + x_h)}))^{1/f(M^*)} = g^{ab}$.

Analysis. We denote by ε the advantage of the adversary F in the experiment and by success the event that the simulation outputs a solution g^{ab}. Since the exponents are randomly chosen this yields the correct distribution for F. The simulator is successful if F is successful and it guesses t^* correctly. So we have $\Pr[\text{success}] = \frac{\varepsilon \cdot m}{q}$. $\square$

Optimizations. Now, with this result and our generic transformation from Section 5.1 we can construct a stateless signature scheme, which is proven EUF-naCMA secure by Theorem 2. By applying a chameleon hash function CH, we obtain a fully EUF-CMA-secure signature scheme. This signature scheme does have a constant size public key but signatures consist of $O(\log(\lambda))$ group elements.

Now, we concentrate on how we can improve this and achieve constant size signatures. This will be done by aggregation, essentially by multiplying the signatures of each instance similar to [27]. We re-use $u_0, \ldots, u_m$, one $sk := \alpha$ and one randomness s for all instances i (see Figure 2). Unfortunately, we need additional elements in the public key for the aggregation to work. In this sense our optimization is rather a tradeoff: We prefer constant-size signatures with public keys of logarithmic length over logarithmic-length signatures with constant-size public keys. The result is scheme $\mathsf{SIG}_{\mathsf{gen}}^{\mathsf{CDH}}$ (Figure 2). The proof of the following theorem can be found in [8].

Theorem 4. *If the CDH assumption holds in $\mathbb{G}$ and CH is a chameleon hash function, then $\mathsf{SIG}_{\mathsf{gen}}^{\mathsf{CDH}}$ (Figure 2) is EUF-CMA secure. Let F be a PPT adversary with advantage $\varepsilon := \varepsilon(\lambda) := \mathsf{Adv}_{\mathsf{SIG}_{\mathsf{gen}}^{\mathsf{CDH}}, F}^{\mathsf{euf\text{-}cma}}(\lambda)$ and runtime T asking for $q := q(\lambda)$ signatures, then it can be used to solve a CDH challenge with probability at least $\epsilon_{dh} = \frac{\varepsilon^{c/m+1}}{2^{c/m+1} \cdot q^{c(m+1)/m}} - \varepsilon_{\mathsf{PRF}} - \varepsilon_{\mathsf{CH}}$, where $\varepsilon_{\mathsf{PRF}}$ and $\varepsilon_{\mathsf{CH}}$ correspond to the advantages for breaking the PRF and the chameleon hash respectively. $T_{dh} \approx T$.*

5.3 Our RSA-Based Scheme

In this subsection we construct a stateless signature scheme $\mathsf{SIG}_{\mathsf{opt}}^{\mathsf{RSA}}$ that is EUF-CMA-secure under the RSA assumption. The result is the most efficient RSA-based scheme currently known.

The prototype for our construction is the stateful RSA-based scheme of Hohenberger and Waters [23], to which we refer as $\mathsf{SIG}_{\mathsf{HW09}}^{\mathsf{RSA}}$ from now on. We first show that a stripped-to-the-basics variation of their scheme (which is tag-based but stateless), denoted $\mathsf{SIG}^{\mathsf{RSA}}$, is mildly secure, i.e., EUF-naCMA$_m^*$ secure. Subsequently, we apply our generic transformation from Section 5.1 and add a chameleon hash to construct a fully secure stateless scheme. Finally we apply common aggregation techniques which yields the optimized scheme $\mathsf{SIG}_{\mathsf{opt}}^{\mathsf{RSA}}$.

Definition 2 (RSA assumption). *Let $N \in \mathbb{N}$ be the product of two distinct safe primes P and Q with $2^{\frac{\lambda}{2}} \leq P, Q \leq 2^{\frac{\lambda}{2}+1} - 1$. Let e be a randomly chosen positive integer less than and relatively prime to $\varphi(N) = (P-1)(Q-1)$. For $y \leftarrow \mathbb{Z}_N^\times$ we call the triple*

(N, e, y) RSA challenge. *The* RSA assumption *holds if for every PPT algorithm A the probability*

$$\Pr\left[A(N,e,y) = x \wedge x^e \equiv y \bmod N\right]$$

is negligible for a uniformly chosen RSA challenge (N, e, y).

EUF-naCMA$^*_m$-Secure Signature Scheme. We start with the basic scheme $\mathsf{SIG}^{\mathsf{RSA}}$. Let $N = PQ$ be an RSA modulus consistent with the RSA assumption (Definition 2). Basically, a $\mathsf{SIG}^{\mathsf{RSA}}$ signature for a message-tag-pair (M, t) is a tuple $((\mathbf{u}^M)^{\frac{1}{p}} \bmod N, t)$ where p is a prime derived from the tag t. Analogously to our CDH scheme (Section 3), we define $\mathbf{u}^M := \prod_{i=0}^{m} u_i^{M^i}$ using quadratic residues $(u_i)_{i=0}^m$ to allow for the signing of up to m messages with the same tag. The message space is $\{0,1\}^\ell$ where we pick $\ell = \lambda/2$ for our realization – we will need later that $\frac{1}{2^{\lambda-\ell}}$ is negligible. To construct a mapping from tags to primes we use a technique from [22] and [20]: For a PRF $\mathsf{PRF}^{\{0,1\}^\lambda}$, a corresponding key $\kappa \leftarrow \{0,1\}^\lambda$, and a random bitstring $b \leftarrow \{0,1\}^\lambda$, we define

$$\mathsf{P}_{(\kappa,b)}(t) := \mathsf{PRF}_\kappa^{\{0,1\}^\lambda}(t||\mu_t) \oplus b$$

where $\mu_t := \min\{\mu \in \mathbb{N} : \mathsf{PRF}_\kappa^{\{0,1\}^\lambda}(t||\mu) \oplus b \text{ is prime}\}$ and $||$ denotes the concatenation of bitstrings.[7] We call μ_t the *resolving index* of t. The complete scheme $\mathsf{SIG}^{\mathsf{RSA}}$ is depicted in Figure 6.

```
$\mathsf{Gen_t}(1^\lambda)$
  Pick modulus $N = PQ$,
  $u_i \leftarrow QR_N$
    $(i \in \{0,\dots,m\})$
  $\kappa \leftarrow \{0,1\}^\lambda$
  $b \leftarrow \{0,1\}^\lambda$
  $pk := (N, (u_i)_{i=0}^m, \kappa, b)$
  $sk := (P, Q)$
  return $(pk, sk)$
```

```
$\mathsf{Sig_t}(sk, M, t)$
  $p := \mathsf{P}_{(\kappa,b)}(t)$
  $\hat{\sigma} :=$
    $(\prod_{i=0}^m u_i^{M^i})^{\frac{1}{p}} \bmod N$
  return $(\hat{\sigma}, t)$
```

```
$\mathsf{Ver_t}(pk, M, \sigma = (\hat{\sigma}, t))$
  if $t \notin \mathcal{T}$
    return 0
  $p := \mathsf{P}_{(\kappa,b)}(t)$
  if $\hat{\sigma}^p \not\equiv \prod_{i=0}^m u_i^{M^i} \bmod N$
    return 0
  else
    return 1
```

Fig. 6. The tag-based RSA scheme $\mathsf{SIG}^{\mathsf{RSA}}$

Differences to $\mathsf{SIG}^{\mathsf{RSA}}_{\mathsf{HW09}}$. We give a brief overview how $\mathsf{SIG}^{\mathsf{RSA}}$ relates to $\mathsf{SIG}^{\mathsf{RSA}}_{\mathsf{HW09}}$. To reduce overhead, we first removed all components from $\mathsf{SIG}^{\mathsf{RSA}}_{\mathsf{HW09}}$ that are not required to prove EUF-naCMA$^*_m$-security. This includes the chameleon hash (we are in a non-adaptive setting) and the logarithm-of-tag-construction (we guess from a small set of tags only). Our setup of $\mathsf{P}_{(\kappa,b)}$ slightly differs from the one in $\mathsf{SIG}^{\mathsf{RSA}}_{\mathsf{HW09}}$ since we do need that *every* tag is mapped to a prime.

[7] $\mathsf{P}_{(\kappa,b)}(t)$ can be computed in expected polynomial time but not in strict polynomial time. However, one can simply pick an upper bound $\overline{\mu}$ and set $\mathsf{P}_{(\kappa,b)}(t) = p$ for some arbitrary but fix prime p if $\mu_t > \overline{\mu}$ for the resolving index of t μ_t. For a proper $\overline{\mu}$ the event $\mu_t > \overline{\mu}$ will only occur with negligible probability (see [8], Theorem 5.6).

Theorem 5. *If F is a PPT EUF-naCMA$^*_m$-adversary for* $\mathsf{SIG}^{\mathsf{RSA}}$ *with advantage* $\varepsilon := \mathsf{Adv}^{\text{euf-nacma}^*_m}_{\mathsf{SIG}^{\mathsf{RSA}},F}(\lambda)$ *asking for* $q := q(\lambda)$ *signatures, then it can be used to efficiently solve an RSA challenge according to Definition 2 with probability at least*

$$\frac{\varepsilon}{q'\lambda^2} - \lambda^2 \varepsilon_{\mathsf{PRF}} - O(\frac{1}{2^{\lambda/2}})$$

where q' denotes the number of distinct tags queried by F and $\varepsilon_{\mathsf{PRF}}$ is the advantage of a suitable distinguisher for the PRF.

The proof for Theorem 5 can be found in [8]. The main idea is exactly that of the proof for Theorem 3. One additional challenges is that we have to map every tag to a prime (realized by $\mathsf{P}_{(\kappa,b)}$). Furthermore, in the extraction phase, we need to use Shamir's trick here to perform a division in the exponent – this requires coprime exponents. Now, by Theorem 5, our generic transformation from Section 5.1 applied to $\mathsf{SIG}^{\mathsf{RSA}}$ yields an EUF-naCMA-secure signature scheme. Finally, we use chameleon hashing [26] to generically construct the fully secure scheme $\mathsf{SIG}^{\mathsf{RSA}}_{\mathsf{gen}}$, like for instance the RSA-based chameleon hash from [22, Appendix C].

Optimizations. The resulting signature scheme of the previous section $\mathsf{SIG}^{\mathsf{RSA}}_{\mathsf{gen}}$ may be EUF-CMA-secure but is not very compact yet. In addition to parameters for the chameleon hash, a signature of $\mathsf{SIG}^{\mathsf{RSA}}_{\mathsf{gen}}$ consists of $l = \lfloor \log_c(\lambda) \rfloor$ $\mathsf{SIG}^{\mathsf{RSA}}$ signatures. This can be improved considerably to constant size signatures by generic aggregation.

Figure 7 depicts the resulting scheme $\mathsf{SIG}^{\mathsf{RSA}}_{\mathsf{opt}}$ for the two parameters l (which implicitly contains the granularity parameter c) and m. We still use l tags (intuitively representing the l instances of the original scheme) for signing and verification. However, the public key's size depends only on m (which is a fixed parameter) and the signature size is constant: We need one group element and randomness for the chameleon hash (which is typically also about the size of a group element). As mentioned, the functions $(\mathsf{PRF}^{\mathcal{T}_i})_{i\in[l]}$ necessary to generate the tags for a signature can be generically constructed from $\mathsf{PRF}^{\{0,1\}^\lambda}$.

```
$\mathsf{Gen}(1^\lambda)$
  Pick modulus $N = PQ$
  $u_i \leftarrow QR_N$,
    $(i \in \{0,\dots,m\})$
  $\kappa \leftarrow \{0,1\}^\lambda$
  $b \leftarrow \{0,1\}^\lambda$
  $(\mathsf{CH}, \tau) \leftarrow \mathsf{CHGen}(1^\lambda)$
  $pk :=$
    $(N, (u_i)_{i=0}^m, \kappa, b, \mathsf{CH})$
  $sk := (P, Q)$
  return $(pk, sk)$
```

```
$\mathsf{Sig}(sk, M)$
  Pick uniform $r$ for $\mathsf{CH}$
  $x := \mathsf{CH}(M, r)$
  for $i := 1$ to $l$ do
    $t_i := \mathsf{PRF}^{\mathcal{T}_i}_\kappa(x)$
    $p_i := \mathsf{P}_{(\kappa,b)}(t_i)$
  $p := \prod_{i\in[l]} p_i$
  $\hat{\sigma} :=$
    $(\prod_{i=0}^m u_i^{x^i})^{\frac{1}{p}} \bmod N$
  return $(\hat{\sigma}, r)$
```

```
$\mathsf{Ver}(pk, M, (\hat{\sigma}, r))$
  $x := \mathsf{CH}(M, r)$
  for $i := 1$ to $l$ do
    $t_i := \mathsf{PRF}^{\mathcal{T}_i}_\kappa(x))$
    $p_i := \mathsf{P}_{(\kappa,b)}(t_i)$
  $p := \prod_{i\in[l]} p_i$
  if $\hat{\sigma}^p \not\equiv \prod_{i=0}^m u_i^{x^i} \bmod N$
    return 0
  else
    return 1
```

Fig. 7. The optimized RSA-based signature scheme $\mathsf{SIG}^{\mathsf{RSA}}_{\mathsf{opt}}$

Theorem 6. *Let F be a PPT EUF-CMA adversary against $\mathsf{SIG}^{\mathsf{RSA}}_{\mathsf{opt}}$ with advantage $\varepsilon := \mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\mathsf{SIG}^{\mathsf{RSA}},F}(\lambda)$ asking for $q := q(\lambda)$ signatures (at most). Then it can be used to efficiently solve an RSA challenge according to Definition 2 with probability at least*

$$\frac{\varepsilon(\lambda)^{c/m}}{2^{c/m} \cdot q^{c(m+1)/m}} - \frac{\varepsilon(\lambda)}{2} - \varepsilon_{\mathsf{PRF}} - \varepsilon_{\mathsf{CH}} - \mathcal{O}\left(\frac{1}{2^{\lambda/2}}\right)$$

where $\varepsilon_{\mathsf{PRF}}$ and $\varepsilon_{\mathsf{CH}}$ are the success probabilities for breaking PRF*, resp.* CH.

The proof for Theorem 6 can be found in [8] and is analogous to the proof for Theorem 4. Again, we have the additional challenges of mapping tags to primes and meeting the prerequisite for Shamir's trick in the extraction phase.

5.4 Our SIS-Based Scheme

Let us now sketch the SIS-based signature scheme. Due to space limitations, this sketch is extremely brief. We refer to [8] for details.

In previous chapters we have used the character m to denote the number of repeating tags in the EUF-naCMA$^*_m$ security experiment. Unfortunately, the same character is commonly used in lattice-based cryptography to denote the dimension of a matrix $\mathbb{Z}_p^{n \times m}$. In order to be consistent with the literature, and since we consider only EUF-naCMA$^*_1$-security in the sequel, we will from now on use m to denote the dimension of matrices.

Again we construct a tag-based signature scheme first and prove EUF-naCMA$^*_1$-security. The scheme is described in Figure 8. It is based on techniques from [16, 2, 10, 7], in particular it exhibits many similarities to the identity-key generation algorithm of the IBE scheme from [2], where our tags correspond to identities of [2].

Figure 8 uses two algorithms TrapGen and SampleLeft, which we unfortunately can not describe in detail here. Essentially, TrapGen computes a matrix $A \in \mathbb{Z}_q^{n \times m}$ together with a short basis T_A of $\Lambda_p^{\perp}(A)$, and SampleLeft samples a short vector $e \in \mathbb{Z}_p^{2m}$ statistically close to $\mathcal{D}_{\Lambda_p^u(A|G_t),\gamma}$. A difference to [2] is that we must be able to issue one signature for message-tag-pair (M_i, t_i) with $t_i = t^*$, but without knowing any trapdoor. This is resolved by a suitable set-up of the vector v contained in the public key in the proof. See [8] for details.

```
$\mathsf{Gen_t}(1^\lambda)$
  $(A, T_A) \leftarrow$
    $\mathsf{TrapGen}(p, n)$
  $Z, Y \leftarrow \mathbb{Z}_q^{n \times m}$
  $U \leftarrow \mathbb{Z}_q^{n \times \ell}$
  $v \leftarrow \mathbb{Z}_p^n$
  $sk := T_A$
  $pk := (U, A, Z, Y, v)$
  return $(sk, pk)$
```

```
$\mathsf{Sig_t}(sk, M, t)$
  $G_t := Z + H(t)Y \bmod p$
  $u := UM + v$
  $e \leftarrow \mathsf{SmpL}(A, T_A, G_t, u, \gamma)$
  return $(e, t) \in \mathbb{Z}_p^{2m} \times \mathcal{T}$
```

```
$\mathsf{Ver_t}(pk, M, \sigma = (e, t))$
  if $t \notin \mathcal{T}$ or $M \notin \{0,1\}^\ell$
    return 0
  if $e \leq 0$ or $\|e\| > \sqrt{2m} \cdot \gamma$
    return 0
  $G_t := Z + H(t)Y \in \mathbb{Z}_p^{n \times 2m}$
  if $(A|G_t)e = UM + v \bmod p$
    return 1
  else return 0
```

Fig. 8. The EUF-naCMA$^*_1$-secure SIS scheme

EUF-CMA-Secure Scheme. By applying the generic transformation from Section 5.1 to our lattice-based EUF-naCMA$_1^*$-secure signature scheme, we obtain EUF-naCMA-secure signatures. Concretely, suppose we use message space $\{0,1\}^\ell$ with $\ell = m$. Then the resulting EUF-naCMA-secure signature scheme has public keys consisting of $4nm + n$ elements of $\mathbb{Z}_p$ plus a key κ for the PRF. Signatures consist of l low-norm vectors in $\mathbb{Z}_p^n$, where $l = \lfloor \log_c(\lambda) \rfloor = O(\log \lambda)$ is defined as in Section 5.1. Unfortunately we are not able to aggregate signatures, like we did for the optimized CDH- and RSA-based constructions, due to the lack of signature aggregation techniques for lattice-based signatures. We leave this as an interesting open problem.

To obtain a fully EUF-CMA-secure signature scheme, one can combine this EUF-naCMA-secure scheme with a suitable chameleon hash function, like for instance the SIS-based construction from [10, Section 4.1]. This adds another $2mn$ elements of $\mathbb{Z}_p$ to the public key, plus one additional low-norm vector $e \in \mathbb{Z}_p^m$ to each signature.

Acknowledgement. Jae Hong Seo thanks members of Shin-Akarui-Angou-Benkyou-Kai for their helpful comments. Part of his work was done while he was in NICT, Japan and it was made public through [1]. Böhl was supported by MWK grant "MoSeS". Koch was supported by BMBF project "KASTEL". Hofheinz and Jager were supported by DFG grant GZ HO 4534/2-1. Part of this work was done while Jager was with KIT.

References

[1] Memoirs of the 6th Cryptology Paper Contest, arranged by Korea Communications Commission (2012)

[2] Agrawal, S., Boneh, D., Boyen, X.: Efficient lattice (H)IBE in the standard model. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 553–572. Springer, Heidelberg (2010)

[3] Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Ashby, V. (ed.) ACM CCS 1993, Fairfax, Virginia, USA, November 3-5, pp. 62–73. ACM Press (1993)

[4] Boneh, D., Boyen, X.: Short signatures without random oracles. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004)

[5] Boneh, D., Boyen, X.: Short signatures without random oracles and the SDH assumption in bilinear groups. Journal of Cryptology 21(2), 149–177 (2008)

[6] Boneh, D., Mironov, I., Shoup, V.: A secure signature scheme from bilinear maps. In: Joye, M. (ed.) CT-RSA 2003. LNCS, vol. 2612, pp. 98–110. Springer, Heidelberg (2003)

[7] Boyen, X.: Lattice mixing and vanishing trapdoors: A framework for fully secure short signatures and more. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 499–517. Springer, Heidelberg (2010)

[8] Böhl, F., Hofheinz, D., Jager, T., Koch, J., Striecks, C.: Confined guessing: New signatures from standard assumptions. Cryptology ePrint Archive, Report 2013/171 (2013), `http://eprint.iacr.org/`

[9] Cash, D., Kiltz, E., Shoup, V.: The twin diffie-hellman problem and applications. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 127–145. Springer, Heidelberg (2008)

[10] Cash, D., Hofheinz, D., Kiltz, E., Peikert, C.: Bonsai trees, or how to delegate a lattice basis. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 523–552. Springer, Heidelberg (2010)

[11] Coron, J.-S.: On the exact security of full domain hash. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 229–235. Springer, Heidelberg (2000)

[12] Cramer, R., Shoup, V.: A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 13–25. Springer, Heidelberg (1998)
[13] Cramer, R., Shoup, V.: Signature schemes based on the strong RSA assumption. In: ACM CCS 1999, Kent Ridge Digital Labs., Singapore, November 1-4, pp. 46–51. ACM Press (1999)
[14] Cramer, R., Shoup, V.: Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 45–64. Springer, Heidelberg (2002)
[15] Fischlin, M.: The Cramer-Shoup strong-RSA signature scheme revisited. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 116–129. Springer, Heidelberg (2002)
[16] Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Ladner, R.E., Dwork, C. (eds.) 40th ACM STOC, Victoria, British Columbia, Canada, May 17-20, pp. 197–206. ACM Press (2008)
[17] Gerbush, M., Lewko, A., O'Neill, A., Waters, B.: Dual form signatures: An approach for proving security from static assumptions. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 25–42. Springer, Heidelberg (2012)
[18] Hofheinz, D., Kiltz, E.: Practical chosen ciphertext secure encryption from factoring. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 313–332. Springer, Heidelberg (2009)
[19] Hofheinz, D., Kiltz, E.: Programmable hash functions and their applications. Journal of Cryptology 25(3), 484–527 (2012)
[20] Hofheinz, D., Jager, T., Kiltz, E.: Short signatures from weaker assumptions. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 647–666. Springer, Heidelberg (2011)
[21] Hofheinz, D., Jager, T., Knapp, E.: Waters signatures with optimal security reduction. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 66–83. Springer, Heidelberg (2012)
[22] Hohenberger, S., Waters, B.: Short and stateless signatures from the RSA assumption. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 654–670. Springer, Heidelberg (2009)
[23] Hohenberger, S., Waters, B.: Realizing hash-and-sign signatures under standard assumptions. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 333–350. Springer, Heidelberg (2009)
[24] Impagliazzo, R., Rudich, S.: Limits on the provable consequences of one-way permutations. In: 21st ACM STOC, Seattle, Washington, USA, May 15-17, pp. 44–61. ACM Press (1989)
[25] Joye, M.: An efficient on-line/Off-line signature scheme without random oracles. In: Franklin, M.K., Hui, L.C.K., Wong, D.S. (eds.) CANS 2008. LNCS, vol. 5339, pp. 98–107. Springer, Heidelberg (2008)
[26] Krawczyk, H., Rabin, T.: Chameleon signatures. In: NDSS 2000, San Diego, California, USA, February 2-4. The Internet Society (2000)
[27] Lu, S., Ostrovsky, R., Sahai, A., Shacham, H., Waters, B.: Sequential aggregate signatures and multisignatures without random oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 465–485. Springer, Heidelberg (2006)
[28] Döttling, N., Müller-Quade, J., Nascimento, A.C.A.: IND-CCA Secure Cryptography Based on a Variant of the LPN Problem. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 485–503. Springer, Heidelberg (2012)
[29] Rompel, J.: One-way functions are necessary and sufficient for secure signatures. In: 22nd ACM STOC, Baltimore, Maryland, USA, May 14-16, pp. 387–394. ACM Press (1990)
[30] Seo, J.H.: Short signature from Diffie-Hellman: Realizing short public key, Cryptology ePrint Archive, Report 2012/480 (2012), `http://eprint.iacr.org/2012/480`

[31] Shamir, A., Tauman, Y.: Improved online/offline signature schemes. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 355–367. Springer, Heidelberg (2001)
[32] Waters, B.: Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 619–636. Springer, Heidelberg (2009)
[33] Waters, B.: Efficient identity-based encryption without random oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)
[34] Yamada, S., Hanaoka, G., Kunihiro, N.: Space efficient signature schemes from the RSA assumption. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 102–119. Springer, Heidelberg (2012)

Locally Computable UOWHF with Linear Shrinkage

Benny Applebaum[⋆] and Yoni Moses

School of Electrical Engineering, Tel-Aviv University
`bennyap@post.tau.ac.il, ymoses@gmail.com`

Abstract. We study the problem of constructing locally computable Universal One-Way Hash Functions (UOWHFs) $\mathcal{H} : \{0,1\}^n \to \{0,1\}^m$. A construction with constant *output locality*, where every bit of the output depends only on a constant number of bits of the input, was established by [Applebaum, Ishai, and Kushilevitz, SICOMP 2006]. However, this construction suffers from two limitations: (1) It can only achieve a sub-linear shrinkage of $n - m = n^{1-\epsilon}$; and (2) It has a super-constant *input locality*, i.e., some inputs influence a large super-constant number of outputs. This leaves open the question of realizing UOWHFs with constant output locality and linear shrinkage of $n - m = \epsilon n$, or UOWHFs with constant input locality and minimal shrinkage of $n - m = 1$.

We settle both questions simultaneously by providing the first construction of UOWHFs with linear shrinkage, constant input locality, and constant output locality. Our construction is based on the one-wayness of "random" local functions – a variant of an assumption made by Goldreich (ECCC 2000). Using a transformation of [Ishai, Kushilevitz, Ostrovsky and Sahai, STOC 2008], our UOWHFs give rise to a digital signature scheme with a minimal *additive* complexity overhead: signing n-bit messages with security parameter κ takes only $O(n + \kappa)$ time instead of $O(n\kappa)$ as in typical constructions. Previously, such signatures were only known to exist under an *exponential* hardness assumption. As an additional contribution, we obtain new locally-computable hardness amplification procedures for UOWHFs that preserve linear shrinkage.

1 Introduction

The question of minimizing the parallel time complexity of cryptographic primitives has been the subject of an extensive body of research. At the extreme, one would aim for an ultimate level of efficiency at the form of *constant*-parallel time implementation. Namely, the goal is to have "local" cryptographic constructions in which each bit of the output depends only on a small constant number of input bits, and each bit of the input influences only a constant number of outputs. Achieving both constant *input locality* and constant *output locality* allows an implementation by constant-depth circuit of bounded fan-in and bounded

⋆ Supported by Alon Fellowship, ISF grant 1155/11, Israel Ministry of Science and Technology (grant 3-9094), and GIF grant 1152/2011.

fan-out [7]. Furthermore, such local constructions have turned to be surprisingly helpful in speeding-up the *sequential complexity* of cryptography [17]. At a more abstract level, the study of locally computable cryptography allows to understand whether extremely simple functions can generate cryptographic hardness.

Intuitively, one may suspect that functions with local input-output dependencies may be vulnerable to algorithmic attacks. Still, during the last decade it was shown that, under standard intractability assumptions, many cryptographic tasks can be implemented by local functions [6,5,7]. This includes basic primitives such as one-way functions and pseudorandom generators, as well as, more complicated primitives such as public-key encryption schemes. One notable exception, for which such a result is unknown, is hash functions with *linear* shrinkage.

A collection of hash functions $\mathcal{H} = \{h : \{0,1\}^n \to \{0,1\}^m\}$ shrinks a long n-bit string into a shorter string of length $m < n$ such that, given a random function $h \overset{R}{\leftarrow} \mathcal{H}$ and a target string x, it is hard to find a sibling $y \neq x$ that collide with x under h. The exact specification of the above game corresponds to different notions of hashing. We will mainly consider *universal one-way hash functions* (UOWHFs) [21], in which the adversary specifies the target string x without seeing the function h. (This property is also known as *target collision resistance* [8], TCR in short.) A central parameter of a hash function is the amount of shrinkage it provides. We measure this as the difference between the output length m and the input length n, namely the *additive shrinkage* $n - m$. We say that the shrinkage is linear if $n - m = \Omega(n)$, i.e., $m < (1 - \varepsilon)n$ for some constant ε. In this paper we ask:

> Are there UOWHFs with *linear shrinkage* and *constant* output and/or input locality ?

Previous Results. The results of [6] show that any log-space computable UOWHF can be converted into a UOWHF with constant output locality and sub-linear shrinkage of $n - m = n^{\varepsilon}$, for a constant $\varepsilon < 1$. (A similar result holds for collision-resistance hash functions.) This gives rise to UOWHFs with constant output locality based on standard cryptographic assumptions (e.g., factoring), or, more generally, on any log-space computable one-way function [21,24,15]. Although there are several ways to amplify the shrinkage of a UOWHF (cf. [21,8]), none of these transformations preserve low locality, and so the question of obtaining UOWHFs with linear shrinkage and constant output locality has remained wide open.

The situation is even worse for constant input locality. In [7] it was shown that tasks which involve secrecy (e.g., one-wayness, pseudorandomness, symmetric or public-key encryption) can be implemented with constant input locality (under plausible assumptions), while tasks which require some form of non-malleability (e.g., MACs, signatures, non-malleable encryption) cannot be implemented with constant input locality. Interestingly, hash functions escaped this characterization. Although it is easy to find near-collisions in a function with constant input locality (simply flip the first bit of the target x), it is unknown how to extend this to a full collision. Overall, the question of computing UOWHFs with constant input locality has remained open, even for the case of a single-bit shrinkage $n - m = 1$.[1]

1.1 Main Result

We construct the first locally computable UOWHF with linear shrinkage. Our construction has both constant input locality and constant output locality, and is based on the one-wayness of random local functions (also known as Goldreich's one-way function [14]). The latter assumption asserts that a random local function $f : \{0,1\}^n \to \{0,1\}^m$ is one-way where f is chosen uniformly at random as follows. View the n inputs and m outputs as vertices in a bipartite graph G and connect each output node y_i to a random set of d distinct input nodes. To compute the i-th output apply some fixed d-local predicate $P : \{0,1\}^d \to \{0,1\}$ to the d inputs that are connected to y_i. This experiment defines a distribution $\mathcal{F}_{P,n,m}$ over functions with output locality of d. (See Section 3 for a formal definition.) We prove the following theorem.

Theorem 1. *There exists a constant* d *and a predicate* $P : \{0,1\}^d \to \{0,1\}$ *for which the following holds. If the collection* $\mathcal{F}_{P,n,m=O(n^3)}$ *is one-way then there exists a collection* $\mathcal{H}$ *of UOWHF with linear shrinkage, constant input locality, and constant output locality.*

The theorem is constructive, and can be applied to every predicate which satisfies a simple criteria. In particular, we show that the predicate $\mathsf{MST}_{d_1,d_2}(x,y) = (x_1 \wedge \ldots \wedge x_{d_1}) \oplus (y_1 \oplus \ldots \oplus y_{d_2})$ defined by [20] satisfies the theorem for every $d_1 \geq 2$ and every sufficiently large constant d_2. The hypothesis of the theorem (one-wayness of random local functions) was extensively studied in the last few years and it is supported both experimentally [22,13] and theoretically [14,2,13,10]. In fact, recent evidence suggest that, for a proper predicate, this collection may even be pseudorandom [4,3]. Interestingly, Theorem 1 can be proved under the (possibly weaker) assumption that $\mathcal{F}_{P,n,m=O(n)}$ is a weak pseudorandom generator (i.e., its output cannot be distinguished from truly random string with advantage better than, say, 0.1).

There are several interesting corollaries that follow from Theorem 1. First, it is possible to reduce the output locality to 4 (which is almost optimal) while preserving (tiny) linear shrinkage (i.e., $m = (1 - \varepsilon)n$ for some small ε) via the compiler of [6].[2] Second, by self-composing $\mathcal{H}$ a constant number times, one can get arbitrary linear shrinkage (i.e., $m = \varepsilon n$ for arbitrary constant $\varepsilon > 0$) at the expense of increasing the locality to a larger constant. Furthermore, by iterating $\mathcal{H}$ a logarithmic number of times we get a linear-time computable hash function $\mathcal{H}'$ with polynomial shrinkage factor of $m = n^{\varepsilon}$ (the i-th level of the

[1] We note that standard transformations from one-way functions to UOWHFs [21,24,15] are inherently non-local as they employ primitives such as k-wise independent hash functions which cannot be computed locally.

[2] When applied to *local functions*, the AIK compiler preserves linear shrinkage.

circuit contains $O(n/2^i)$ gates). As observed by [17], one can then employ the Naor-Yung transform [21] and sign n-bit messages with linear time complexity and only *additive cryptographic overhead*, i.e., $O(n + \kappa)$. (See [17] for details.) This is contrasted with standard signature schemes whose complexity grows multiplicatively with the security parameter, i.e., $O(n\kappa)$. Previously, such linear-time computable UOWHFs and signatures were only known to exist assuming that Goldreich's collection is *exponentially*-hard to invert [17].[3]

1.2 Techniques

Hashing via Random Local Functions? As a starting point, we ask whether the collection $\mathcal{F}_{P,n,m=n(1-\varepsilon)}$ itself can be used, even heuristically, as a UOWHF. To make the question non-trivial, let us assume that the distribution of the input-output dependency graph is slightly modified such that the graph is (c, d)-regular, i.e., each input affects c outputs and each output depends on d inputs. (Otherwise, we are likely to have some inputs of degree 0, with no influence at all.) For concreteness let us think of P as the majority predicate. A moment of reflection suggests that collisions are easy to find even with respect to a random target string x. Indeed, suppose that there exists an input variable x_i that all of its neighboring inputs (i.e., the inputs that share an output with x_i) turn to be zero. In this case, we can flip the *insensitive* input x_i without affecting the output of the function, and this way obtain a trivial collision. Observe that each input variable has a constant probability of being insensitive as it has at most $cd = O(1)$ neighbors. Overall, one is likely to find $\Omega(n)$ insensitive inputs. Furthermore, by collecting an independent set I of insensitive inputs (that do not share any common output) one can simultaneously flip any subset of the inputs in I without changing the output. Hence, we find exponentially many collisions x' which form a "ball" around x of diameter $\Omega(n)$. It is not hard to show that a similar attack can be applied to $\mathcal{F}_{P,n,m}$ for every predicate P except for XOR or its negation. (Unfortunately, in the latter case collisions can be found via Gaussian elimination.)

Despite this failure, let us keep asking: Can $\mathcal{F}_{P,n,m}$ achieve some, possibly weak, form of collision resistance ? Specifically, one may hope to show that it is hard to find collisions which are β-far from the target x, for some (non-trivial) constant β. This assumption is intuitively supported by study of the *geometry of the solutions* of random Constraint Satisfaction Problems (e.g., Random SAT) [1]. Thinking of each output as inducing a local constraint on the inputs, it can be essentially showed that, for under-constraint problems where $m < n$, the space of solutions (siblings of x) is shattered into far-apart clusters of Hamming-close solutions. It is believed that efficient algorithms cannot move from one cluster to another as such a transition requires to pass through solutions x' which violate many constraints (i.e., $f(x')$ is far, in Hamming distance, from $f(x)$). Therefore, it seems plausible to conjecture that the collection $\mathcal{F}_{P,n,m}$ is secure with respect to β-far collisions.

[3] Exponential hardness assumptions do not seem to help in the context of *locally computable* UOWHFs.

As our main technical contribution, we prove that a weak form of this conjecture holds assuming the one-wayness of $\mathcal{F}_{P,n,m'}$ (where $m' > n > m$). Specifically, we prove that, for some constants $\varepsilon, \beta, \delta \in (0,1)$, it is hard to find β-far target collisions in $\mathcal{F}_{P,n,(1-\varepsilon)n}$ with probability better than δ. To prove Theorem 1, we show that (δ, β)-target collision resistance (TCR) can be *locally* amplified into standard TCR while preserving *linear* shrinkage. Let us sketch the main ideas behind each of these steps.

One-wayness ⇒ (δ, β)-TCR. Assume that we have an algorithm $\mathcal{A}$ that, given a random function $h \overset{R}{\leftarrow} \mathcal{F}_{P,n,m=(1-\varepsilon)n}$ and a random target w, finds a β-far sibling with probability δ. We show how to use $\mathcal{A}$ to invert the collection $\mathcal{F}_{P,n,m'}$ with output length of $m' \approx 2m$. Given a random function $f_G \overset{R}{\leftarrow} \mathcal{F}_{P,n,m'}$ specified by a random input-output dependencies graph G, and an image $y = f_G(x)$ of a random point $x \overset{R}{\leftarrow} \{0,1\}^n$, we will recover the preimage x as follows. First, we choose a target w uniformly at random and partition the graph G into two subgraphs: G_0 which contains only the output nodes for which $f_G(w)$ agrees with y (and all input nodes), and G_1 which contains the remaining subgraph. Assuming that P is balanced, each subgraph contains roughly m' outputs. Next, we define $h = f_{G_0}$ to be the restriction of f_G to the output nodes for which $f_G(w)$ agrees with y, and ask $\mathcal{A}$ for a β-far sibling w' of w under h. Let us (optimistically) assume that w' is statistically independent of the sub-graph G_1 that was not used by h. That is, imagine that this part of the dependencies graph is chosen uniformly at random after w' is obtained. Since w is far from w', this pair is expected to disagree on a constant fraction γ of the remaining coordinate of f_{G_1}. Remembering that the pair (w, x) did not agree on any of these coordinates, we conclude that x and w agree on a fraction of $\frac{1}{2} + \gamma/2$ of the outputs of f_G (i.e., γ-fraction of the coordinates of f_{G_1} and all the coordinates of $h = f_{G_0}$). Assuming that P is *sensitive* enough, it follows that w' and x must be *correlated* – their Hamming distance is bounded by a constant which is strictly smaller than $\frac{1}{2}$. At this point we employ a result of [9] that allows to fully recover x given such a correlated string w' (and additional $O(n)$ outputs).

The above argument is over-optimistic, as there is no reason to assume that w' is statistically independent of the subgraph G_1. Fortunately, we can show that a failure of the above approach allows to distinguish the string $y = f(x)$ from a truly random string. At this point, we employ the result of [3] which shows that this string is somewhat pseudorandom assuming the one-wayness of $\mathcal{F}_{P,n,m''}$ for larger m''. Hence, we are in a win-win situation: we invert $\mathcal{F}$ either by finding a correlated string, or by distinguishing its output from a random string. (See Section 4 for details.)

(δ, β)-TCR ⇒ δ-TCR. The above reduction leaves us with a δ-secure β-TCR $\mathcal{H}$ of linear shrinkage $n - m = \varepsilon n$, where $\delta, \beta, \varepsilon$ are constants. Our first goal is to get rid of β (i.e., obtain security with respect to standard, possibly close, collisions). A tempting approach would be to compose $\mathcal{H}$ with an error correcting code C, i.e., map an input x to a codeword $C(x)$ and hash the result via $h \in \mathcal{H}$.

A code of constant relative distance larger than β and constant rate smaller than ε will fully eliminate β-close collisions (in an information theoretic sense), while preserving linear shrinkage. Unfortunately, this transformation is inherently non-local, as local functions cannot compute codes with constant relative distance and constant rate.[4] We solve the problem via a dual approach: Instead of computing a codeword $C(x)$ and composing the result with h, we concatenate $h(x)$ with the syndrome Mx where M is a sparse parity-check matrix M whose dual relative distance is β. It is not hard to show that a pair of β-close strings x and x' will always be mapped by M to different outputs $y \neq y'$, and so the mapping $x \mapsto (h(x), Mx)$ is immunized against β-close collisions. Unlike the case of sparse generating matrices, whose distance is deemed to be non-constant, the dual distance of sparse parity-check matrices can be constant (aka LDPC) and so the transformation is locally computable. (See Section 5.2.)

δ-hard TCR $\Rightarrow$ TCR. We move on to amplify the error parameter δ from constant to negligible. Typically this is done via t-wise direct-product, i.e., $x \mapsto (h_1(x), \ldots, h_t(x))$ where the h_i's are chosen independently from $\mathcal{H}$. The error δ decreases exponentially fast and so any super-logarithmic t leads to a negligible error [11]. Unfortunately, in our case even a super-constant t will completely ruin the shrinkage and the input locality. An alternative, more economic, approach is to first stretch the input x into a longer string $C(x) = (c_1, \ldots, c_t) \in (\{0,1\}^n)^t$ via an error-correcting code C, and then apply t-wise direct product [18,11]. If the code has a constant relative distance, any collision (x', x) is translated into a pair $C(x), C(x')$ which collide under $\Omega(t)$ of the coordinates of $(h_1, \ldots, h_t) \overset{R}{\leftarrow} \mathcal{H}^t$. Hence, the error parameter decreases exponentially with t while keeping the shrinkage linear (for properly chosen parameters). Unfortunately, this optimization is inherently non-local as it requires a code with good distance. Nevertheless, we observe that even if C is replaced with a sparse generating matrix G, the resulting transformation is not completely useless. Although the distance of G is bad, it can be shown any pair of β-far inputs x, x' will be mapped by G to a pair (y, y') which is $\Omega(t)$ far apart. As a result, the modified construction amplifies hardness with respect to β-far collisions, but does not amplify hardness with respect to close collisions. Fortunately, such collisions can be again eliminated via LDPCs.[5] (See Section 5.3.)

We note that the above transformations can also be used to locally amplify *collision resistance.*

[4] In fact, such codes are as bad as possible as their relative distance is $O(1/n)$.

[5] One can change the order of the transformations, namely, transform (δ, β)-TCR to β-TCR and then to TCR. This allows to use LDPCs only once. Still we prefer the current order as once β is eliminated (in the first step), it is easy to amplify the shrinkage factor to a small constant via a constant number of self-compositions. Overall, this results in a more flexible reduction that works for a wider range of parameters.

2 Preliminaries

General. By default, logarithms are taken to base 2. For reals $p, q \in (0,1)$ we let $D_2(p\|q) := p\log(\frac{p}{q}) + (1-p)\log(\frac{1-p}{1-q})$ denote the relative entropy function. Observe that $1 - D_2(p\|\frac{1}{2})$ equals to the binary entropy function $H_2(p) := -p\log(p) - (1-p)\log(1-p)$. We will use the following additive form of Chernoff-Hoeffding bound. Let $X_1, \ldots, X_n$ be i.i.d. random variables where $X_i \in [0,1]$ and $\mathsf{E}[X_i] = p$. Then, for every $\varepsilon > 0$, the average $\bar{X} = n^{-1}\sum_i X_i$ satisfies

$$\Pr\left[\bar{X} \geq p + \varepsilon\right] < 2^{-D_2(p+\varepsilon\|p)n} \quad \text{and} \quad \Pr\left[\bar{X} \leq p - \varepsilon\right] < 2^{-D_2(p-\varepsilon\|p)n}.$$

A simpler form follows by noting that $D_2(p+\varepsilon\|p) > 2\varepsilon^2$.

Collection of Functions. We model cryptographic primitives as collections of functions $\mathcal{F} = \left\{f_k : \{0,1\}^n \to \{0,1\}^{m(n)}\right\}_{k\in\{0,1\}^{s(n)}}$ equipped with a pair of efficient algorithms: (1) an evaluation algorithm which given $(k \in \{0,1\}^s, x \in \{0,1\}^n)$ outputs $f_k(x)$; and (2) a key-sampling algorithm $\mathcal{K}$ which given 1^n samples a index $k \in \{0,1\}^{s(n)}$. We will sometimes keep the key-sampler implicit and write $f \xleftarrow{R} \mathcal{F}$ to denote the experiment where $k \xleftarrow{R} \mathcal{K}(1^n)$ and $f = f_k$. A collection of functions has constant *output locality* (resp., constant *input locality*) if there exists a constant d (which does not grow with n) such that for every fixed k each output of the function f_k depends on at most d inputs (resp., each input of f_k affects at most d outputs). The collection is locally computable if it has both constant input locality and constant output locality.

One-wayness and Pseudorandomness. A collection of functions $\mathcal{F}$ is δ-secure β *approximation-resilient one-way* (in short, (δ,β) one-way) if for every efficient adversary $\mathcal{A}$ the following event happens with probability at most δ: Given $f \xleftarrow{R} \mathcal{F}$ and $y = f(x)$ for random $x \xleftarrow{R} \{0,1\}^n$, the adversary $\mathcal{A}$ outputs a list of candidates X' which contains some string x' which is β-close to some preimage of y. The special case of $\beta = 0$ corresponds to the standard notion of δ-secure one-wayness, or simply one-wayness when $\delta = \mathrm{neg}(n)$. A collection of functions $\mathcal{F}$ is δ-pseudorandom if the distribution ensemble $(f, f(x))$ is δ computationally-indistinguishable from the distribution ensemble (f, y), where $f \xleftarrow{R} \mathcal{F}, x \xleftarrow{R} \{0,1\}^n$ and $y \xleftarrow{R} \{0,1\}^m$.

Hash Functions. Let $m = m(n) < n$ be an integer-valued function. A collection of functions $\mathcal{H} = \{h : \{0,1\}^n \to \{0,1\}^m\}$ is δ-secure β *target-collision resistance* ((δ,β)-TCR) if for every pair of efficient adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ it holds that

$$\Pr_{\substack{(x,r)\xleftarrow{R}\mathcal{A}_1(1^n)\\ h\xleftarrow{R}\mathcal{H}}}\left[\mathcal{A}_2(h,x,r) = x' \text{ s.t. } \Delta(x',x) > \beta \text{ and } h(x) = h(x')\right] \leq \delta,$$

where $\Delta(\cdot,\cdot)$ denotes relative Hamming distance. That is, first the adversary $\mathcal{A}_1$ specifies a target string x and a state information r, then a random hash function h is selected, and then $\mathcal{A}_2$ tries to form a β-far collision x' with x under h. The collection is δ-secure *β random target-collision resistance* ((δ, β) RTCR) if the above holds in the special case where $\mathcal{A}_1$ outputs a uniformly chosen target string $x \xleftarrow{R} \{0,1\}^n$ and empty state information. (As we will see, there are standard local transformations from RTCR to TCR). The standard notions of δ-RTCR and δ-TCR correspond to the case where $\beta = 0$ (or just $\beta < 1/n$). If, in addition, δ is negligible we obtain standard RTCR and TCR. The *shrinking factor* of $\mathcal{H}$ is the ratio m/n. When $m/n < 1/(1+H_2(\beta))$ and $\delta = o(1)$ TCR and RTCR become non-trivial in the sense that their existence implies the existence of one-way functions. For an extensive study of hash functions see [8,23].

3 Random Local Functions and Sensitivity

Let $P : \{0,1\}^d \to \{0,1\}$ be a predicate, and let $G = (S_1, \dots, S_m)$ where each $S_i \subseteq [n]$ is a subset of $[n]$ that contains d distinct ordered elements $S_{i,1}, \dots, S_{i,d} \in [n]$. We will think of G as a bipartite graph with n input vertices and m output vertices where each output i is connected to the d inputs in S_i. We define the function $f_{G,P} : \{0,1\}^n \to \{0,1\}^m$ as follows: Given an n-bit input x, the i-th output bit y_i is computed by applying P to the restriction of x to the i-th set S_i, i.e., $y_i = P(x_{S_i})$. For $m = m(n)$ and some fixed predicate $P : \{0,1\}^d \to \{0,1\}$, we let $\mathcal{F}_{P,n,m}$ denote the collection $\left\{f_{G,P} : \{0,1\}^n \to \{0,1\}^{m(n)}\right\}$ where the key G is sampled by selecting $m(n)$ sets uniformly and independently at random from all the possible $n \cdot (n-1) \cdot \ldots \cdot (n-d+1)$ ordered sets. We refer to the latter distribution as the uniform distribution over (n, m, d) graphs and denote it by $\mathcal{G}_{n,m,d}$. When the predicate P is clear from the context, we omit it from the subscript and write f_G and $\mathcal{F}_{n,m}$.

By definition, the ensemble $\mathcal{F}_{P,n,m}$ has a constant output locality of d. However, some inputs will have large (super-constant) locality. Still, one can show, via simple probabilistic argument, that the locality of most inputs will be close to the expectation md/n which is constant when $m = O(n)$. We will later use this fact to reduce the input locality to constant.

3.1 Sensitivity

Let $P : \{0,1\}^d \to \{0,1\}$ be a d-local predicate. For a pair of strings $x, x' \in \{0,1\}^n$ let $s_P(x, x')$ be the expected relative Hamming distance between the images $f(x)$ and $f(x')$ where f is randomly chosen from $\mathcal{F}_{P,n,m}$. Equivalently, we may write $s_P(x, x')$ as

$$\Pr_S[P(x_S) \neq P(x'_S)], \tag{1}$$

where S is a random set of d distinct indices $i_1, \dots, i_d$ which are chosen from $[n]$ uniformly at random without replacement. Imagine the following experiment: first x is chosen uniformly at random, and then an α-far string x' is chosen adversarially in order to minimize $s_P(x, x')$. We will be interested in predicates P for which, except with negligible probability, the value of $s_P(x, x')$ in the above experiment will be relatively high (as a function of α). To analyze this property we make several simple observations. By symmetry, the strategy of the adversary boils down to selecting the fraction $\alpha_{0,1}$ of 0's which are flipped to 1, and the fraction $\alpha_{1,0}$ of 1's which are flipped to 0's (where $\alpha = \alpha_{0,1} + \alpha_{1,0}$). Furthermore, it suffices to analyze a simpler experiment in which x is a random string of Hamming weight $n/2$ and the set S (from Eq. 1) is chosen by selecting d indices uniformly at random from $[n]$ *with replacement* (i.e., the tuple may not be distinct). We will show (in Lemma 1) that, with all but negligible probability over x, these simplifications add a small $o(1)$ error to the value of the experiment.

The above observations motivate a new quantitative measure of sensitivity which refines the standard notion of *noise sensitivity.* For $\alpha_{0,1}, \alpha_{1,0} \in [0, \frac{1}{2}]$, let $\mathcal{D}(\alpha_{0,1}, \alpha_{1,0})$ be a distribution over pairs $w, w' \in \{0,1\}^d$ where w is chosen uniformly at random and the i-th bit of w' is obtained by flipping the i-th bit of w with probability $2\alpha_{0,1}$ if $w_i = 0$, and with probability $2\alpha_{1,0}$ if $w_i = 1$. (Hence, $\Pr[(w_i, w'_i) = (01)] = \alpha_{01}$, and $\Pr[(w_i, w'_i) = (00)] = \frac{1}{2} - \alpha_{01}$, etc.) For $\alpha \in [0,1]$ let $s_P(\alpha)$ denote the infimum of $\Pr_{(w,w') \overset{R}{\leftarrow} \mathcal{D}(\alpha_0,\alpha_1)}[P(w) \neq P(w')]$ taken over all $\alpha_{0,1}$ and $\alpha_{1,0}$ which sum-up to α. Call x *typical* if its Hamming weight is $n/2 \pm n^{2/3}$. By Chernoff bound, a random string is typical with all but negligible probability. The following lemma, whose proof is deferred to the full version, relates $s_P(x, x')$ to $s_P(\alpha)$.

Lemma 1. *For every predicate P, the function $s_P(\alpha)$ is well defined and continuous. Also, for every typical x and every string x' $s_P(x, x') \geq s_P(\Delta(x, x')) - o(1)$.*

Good Predicates. We say that P is (β, γ) *good* if: (1) The value of $s_P(\cdot)$ is lower-bounded by γ in the interval $[\beta, 1]$; and (2) P has a *sensitive coordinate* meaning that $P(w) = w_1 \oplus P'(w_2, \ldots, w_d)$ for some $(d-1)$-local predicate P'. Observe that the latter condition implies that P is balanced and that $s_P(\frac{1}{2}) = \frac{1}{2}$.

In the next section we will use (β, γ)-good predicate to construct β-RTCRs with shrinkage $1 - \varepsilon$ where $\varepsilon \in (0, \frac{1}{2})$ satisfies the inequality

$$\varepsilon < 1 - \frac{1}{2(1 - H_2(\frac{1}{2} - \gamma))}, \tag{2}$$

where H_2 is the binary entropy function. In general, we would like to have a small value of $\beta > 0$ and a large value of $\gamma \leq \frac{1}{2}$ (which leads to a larger ε and better shrinkage). It turns out that by increasing the locality, one can simultaneously push β arbitrarily close to 0 and γ arbitrarily close to $\frac{1}{2}$. This is illustrated by the following family of predicates which generalizes the predicate from [20]. Let MST_{d_1,d_2} be the $(d_1 + d_2)$ local predicate $(x_1 \wedge \ldots \wedge x_{d_1}) \oplus (y_1 \oplus \ldots \oplus y_{d_2})$. In the full version we will prove the following lemma.

Lemma 2. *For every constants $\gamma < \frac{1}{2}$, $\beta > 0$ and integer $d_1 \geq 2$ there exists a constant d_2 for which MST_{d_1,d_2} is (β, γ)-good.*

4 Random Local Functions Are (δ, β)-RTCR

Let P be (β, γ) good predicate. Assume that Eq 2 holds for some $\varepsilon > 0$ and let $m = (1-\varepsilon)n$. In Section 4.1 we prove the following.

Theorem 2. *For every $\delta_1, \delta_2 \in (0,1)$ there exists a constant $\mu > 0$ such that if $\mathcal{F}_{P,n,2m}$ is both δ_1-pseudorandom and $(\delta_2, \frac{1}{2}-\mu)$ one-way then $\mathcal{F}_{P,n,m}$ is δ'-secure β-RTCR where $\delta' = 2(\delta_1+\delta_2)+\mathrm{neg}(n)$.*

It was shown in [9, Thm. 1.3] and [3, Prop. 3.4] that if $\mathcal{F}_{n,m}$ is one-way for sufficiently long output length m, then it is also approximate one-way and pseudorandom for shorter output lengths. Together with Theorem 2, we get:

Corollary 1. *For every constant $\delta > 0$, there exists a constant c such that if $\mathcal{F}_{P,n,cn^3}$ is one-way then $\mathcal{F}_{P,n,(1-\varepsilon)n}$ is δ-secure β-RTCR. Furthermore, if $\mathcal{F}_{P,n,(1-\varepsilon)n}$ is δ-secure β-RTCR then for every constant $\eta > 0$ there exists a δ-secure $\frac{\beta}{1-\eta}$-RTCR $\mathcal{H}$ with constant input and constant output locality and shrinkage factor of $\frac{1-\varepsilon}{1-\eta}$.*

The "furthermore" part is obtained by randomly fixing a small fraction of the inputs of $\mathcal{F}_{P,n,m}$ (the ones with maximal influence). See full version for details.

4.1 Proof of Theorem 2

Assume, towards a contradiction, that $\mathcal{F}_{P,n,m}$ is not δ'-secure β-RTCR. Namely, there exists an efficient adversary $\mathcal{A}$ which, given a random target $w \overset{R}{\leftarrow} \{0,1\}^n$ and a random graph $G \overset{R}{\leftarrow} \mathcal{G}_{n,m,d}$, finds, with probability δ', a string z which is a β-far sibling of w under f_G. Assume that $\mathcal{F}_{n,2m}$ is δ_1-pseudorandom. We construct an attacker $\mathcal{B}$ who breaks the $(\delta_2, \frac{1}{2}-\mu)$ one-wayness of $\mathcal{F}_{n,2m}$ for some constant μ whose value will be determined later. Given a graph $G = (S_1, \ldots, S_{2m})$ and a string $y \in \{0,1\}^{2m}$, the algorithm $\mathcal{B}$ is defined as follows:

1. Randomly choose $w \overset{R}{\leftarrow} \{0,1\}^n$ and let $r = f_{G,P}(w) \oplus y$.
2. *Fail*, if the number of 0's in r is smaller than m or larger than $m + m^{2/3}$.
3. Let I_0 be the set of the first m indices i for which $r_i = 0$, and $I_1 = \{i : r_i = 1\}$. Let $G_0 = \{S_i : i \in I_0\}$ and $G_1 = \{S_i : i \in I_1\}$. (Note that $f_{G_0,P}(w) = y_{I_0}$ and that $f_{G_1,P}(w) = \mathbf{1} \oplus y_{I_1}$.)
4. Apply $\mathcal{A}$ to (G_0, w) and let $z \in \{0,1\}^n$ denote the resulting output.
5. If $P(z_{S_i}) = y_i$ for at least $m(1+\gamma) - 2m^{2/3}$ of indices $i \in [2m]$ output z; Otherwise, *Fail*.

We begin by bounding the failure probability of the algorithm. Intuitively, the algorithm does not fail due to the following reasoning. Assuming that z is a collision, we have that $P(z_{S_i}) = y_i$ for all the m indices $i \in I_0$. In addition, if z is β-far from w and statistically independent of G_1 then (since P is (β, γ) good), the outputs $f_{G_1,P}(w)$ and $f_{G_1,P}(z)$ are expected to disagree on a set of γm coordinates. Since $f_{G_1,P}(w) = \mathbf{1} \oplus y_{I_1}$, this translates to γm indices in I_1 for which $P(z_{S_i}) = y_i$. The above analysis is inaccurate as the random variables z and G_1 are statistically dependent (via the random variable (w, G_0)). Still the above approach can be used when the input y (as well as the graph G) is truly random.

Claim 3. $\Pr_{G \overset{R}{\leftarrow} \mathcal{G}_{n,2m,d}, y \overset{R}{\leftarrow} \{0,1\}^{2m}}[\mathcal{B}(G,y)$ *does not fail*$] > \delta'/2 - \operatorname{neg}(n)$.

Proof. When the pair (G, y) is uniformly chosen, the process $\mathcal{B}(G, y)$ can be equivalently described as follows. In the first step, we choose $S_1, \ldots, S_{2m}$ uniformly at random, choose a random string $w \overset{R}{\leftarrow} \{0,1\}^n$, and a random string $r \overset{R}{\leftarrow} \{0,1\}^{2m}$. We let $y = f_{G,P}(w) \oplus r$. Then steps 2–5 are performed exactly as before. This process is clearly equivalent to $\mathcal{B}(G, y)$, but easier to analyze. The main observation is that the string w is *statistically independent* of the graphs G_0 and G_1 which are just random graphs (whose size is determined by the random variable r).

Specifically, consider the following event: (1) The number of zeroes in r is larger than $m/2$; (2) The number of zeroes in r is smaller than $m/2 + m^{2/3}$; (3) $\mathcal{A}$ outputs β-far collision z with w under $f_{G_0,P}$; (4) The Hamming weight of w is $n/2 \pm n^{2/3}$; (5) $P(z_{S_i}) = y_i$ for at least $m(1+\gamma) - 2m^{2/3}$ of indices $i \in [2m]$.

Event (1) happens with probability $\frac{1}{2}$ (this follows from the "mean in the median" result for the binomial distribution, cf. [19]), and Event (2) happens with all but negligible probability due to a Chernoff bound. Hence, by a union bound (1) and (2) happen together with probability $\frac{1}{2} - \operatorname{neg}(n)$. Fix some r which satisfies both (1) and (2) and let $m_1 \geq m - m^{2/3}$ be the Hamming weight of r. Now, w is a random string and $G_0 \overset{R}{\leftarrow} \mathcal{G}_{n,m,d}$, hence, $\mathcal{A}$ is invoked on the "right" probability distribution and (3) happens with probability δ'. By a Chernoff bound, (4) happens with all but negligible probability. Therefore, by union bound, (3) and (4) happen simultaneously (conditioned on (1,2)) with probability $\delta' - \operatorname{neg}(n)$. Fix w and G_0 which satisfy (3) and (4), and let us move to (5).

Since w and z form a collision under $f_{G_0,P}$, we have that $f_{G_0,P}(z) = y_{I_0}$ and therefore $P(z_{S_i}) = y_i$ for all the m indices $i \in I_0$. Hence, it suffices to show that $P(z_{S_i}) = y_i$ for at least

$$(\gamma - m^{-1/3})m_1 \geq \gamma m - 2m^{2/3}$$

of the indices in I_1. (Recall that $m_1 > m - m^{2/3}$.) We claim that this happens with all but negligible probability (taken over the random choice of $G_1 \overset{R}{\leftarrow} \mathcal{G}_{n,m_1,d}$). To see this, define for every $i \in I_1$ a random variable ξ_i which equals to one if $P(z_{S_i}) = y_i$. Equivalently, $\xi_i = 1$ if $P(z_{S_i}) \neq P(w_{S_i})$. Furthermore, since the sets S_i are distributed uniformly and independently, each ξ_i takes the value 1 independently with probability at least

$$s_P(w, z) \geq s_P(\Delta(w, z)) - o(1) > \gamma$$

where the first inequality follows from Lemma 1 and the fact that w is "typical" (of Hamming weight $n/2 \pm n^{2/3}$); and the second inequality follows from the goodness of P and the fact that $\Delta(w, z) \geq \beta$. Therefore, by Chernoff's bound,

$$\Pr\left[\sum \xi_i < (\gamma - m^{-1/3})m_1\right] < 2^{-D_2(\gamma - m^{-1/3}\|\gamma)m_1} < e^{-\Omega(m^{1/3})},$$

which is negligible in n and so the claim follows. □

Moving back to the case where y is an image of a random string x, we show that when $\mathcal{B}$ does not fail its output is likely to be correlated with x.

Claim 4. *There exists a constant μ such that the following holds. With all but negligible probability over the choice of $x \xleftarrow{R} \{0,1\}^n$ and $G \xleftarrow{R} \mathcal{G}_{n,2m,d}$, there is no string z such that $f_{G,P}(x)$ and $f_{G,P}(z)$ agree on at least $m(1+\gamma) - 2m^{2/3}$ coordinates but $\Delta(x,z) \in (\frac{1}{2} \pm \mu)$.*

Proof. Let $\mu > 0$ be a small constant for which the value of $s_P(\cdot)$ in the interval $(\frac{1}{2} \pm \mu)$ is lower bounded by a constant η which satisfies $\eta > \frac{1}{2} - \gamma$ and

$$2(1-\varepsilon)D_2(\frac{1}{2} - \gamma\|\eta) > 1. \tag{3}$$

Observe that for $\mu = 0$ we can take $\eta = \frac{1}{2}$ (as $s_P(\frac{1}{2}) = \frac{1}{2}$) and so Eq 3 translates to $2(1-\varepsilon)H_2(\frac{1}{2} - \gamma) > 1$ which follows from Eq 2. Since s_P is a continuous function, and the LHS of Eq 3 is also continuous in η, we conclude that Eq 3 also holds for sufficiently small constant $\mu > 0$.

Let us condition on the event that x is typical (as in Lemma 1), which, by a Chernoff bound, happens with all but negligible probability. Fix some string z for which $\Delta(x,z) \in (\frac{1}{2} \pm \mu)$. For a random d size set S we have, by Lemma 1, that $\Pr[P(x_S) \neq P(z_S)] \geq s_P(\Delta(x,z)) > \eta - o(1) > \frac{1}{2} - \gamma$. Let $G = (S_1, \ldots, S_m) \xleftarrow{R} \mathcal{G}_{n,2m,d}$. Since each set S_i is chosen independently and uniformly at random, we can upper-bound (via Chernoff) the probability that $f_{G,P}(x)$ and $f_{G,P}(z)$ *disagree* on less than $2m - (m(1+\gamma) - 2m^{2/3}) = (1-\gamma)m + 2m^{2/3}$ of the coordinates by

$$p = 2^{-2mD_2(\frac{1}{2} - \gamma + o(1)\|s(x,z))} \leq 2^{-2(1-\varepsilon)D_2(\frac{1}{2} - \gamma + o(1)\|\eta - o(1))n}.$$

By a union bound over all z's, we get that the claim holds with probability $p \cdot 2^n$ which is negligible since Eq.3 holds. □

We can now complete the proof of the theorem. Let $G \xleftarrow{R} \mathcal{G}_{n,2m,d}$ and $y = f_{G,P}(x)$ where $x \xleftarrow{R} \{0,1\}^n$. Consider the event that: (1) G and x satisfy Claim 4; and (2) $\mathcal{B}(G,y)$ does not fail and outputs the string z. In this case, either the string z or its negation has a non-trivial agreement of $\frac{1}{2} + \mu$ with x, which may happen with probability at most δ_2 due to the approximate one-wayness of $\mathcal{F}_{n,2m}$. Hence, it suffices to show that the above event happens with probability at least $\delta'/2 - \delta_1 - \text{neg}(n)$. Indeed, (1) happens with all but negligible probability (due to Claim 4), and (2) happens with probability $\delta'/2 - \delta_1 - \text{neg}(n)$ due to Claim 3 and the fact that (G,y) is δ_1-indistinguishable from (G,y') for truly random $y' \xleftarrow{R} \{0,1\}^{2m}$. □

5 From (δ, β)-RTCR to TCR

In this section we will start with δ-secure β-RTCR with shrinkage factor of $1-\varepsilon$ and gradually amplify each of the parameters via locally computable transformations (described in Sections 5.1–5.3). Formally, we prove the following theorem.

Theorem 5. *For every $\varepsilon \in (0,1)$ there exist universal constants $\delta, \beta \in (0,1)$ such that for every desired constant shrinkage factor $\varepsilon' \in (0,1)$ the following holds. Any locally computable δ-secure β-RTCR with shrinkage factor of $1-\varepsilon$ can be transformed into a locally computable TCR with shrinkage factor of ε'.*

We note that the proof of the theorem can be adopted to the setting of collision resistance hash functions. Namely, it allows to locally transform a δ-secure β-collision resistance hash function with shrinkage factor $1-\varepsilon$ into a standard collision resistance hash function with arbitrary constant shrinkage.

Observe that our main theorem (Theorem 1) follows by combining Theorem 5 with Corollary 1 instantiated with (β, γ)-good predicate P where $\beta < \beta^*$ and $\gamma > \gamma^*$ for some universal constants $\beta^* > 0$ and $\gamma^* < \frac{1}{2}$. The exact values of β^* and γ^* are determined by the quality of LDPC codes. (See section 5.2.)

5.1 Standard Transformations

We begin with two standard transformations.

Claim 6 (RTCR to TCR). *Let $\mathcal{H} = \{h_k\}$ be δ-secure β-RTCR with shrinkage factor of $1-\varepsilon$. Then the collection $\mathcal{H}' = \left\{h'_{k,y}\right\}$ defined by $h'_{k,y}(x) = h_k(x \oplus y)$ is δ-secure β-TCR.*

Assume that we already have δ-secure standard-TCR ($\beta = 0$) with shrinkage factor of $1-\varepsilon$. A standard way to amplify the shrinkage factor from $1-\varepsilon$ to $(1-\varepsilon)^t$ is via iterated self-composition [21]. We note that when $t = O(1)$ the locality remains constant.

Claim 7 (Amplifying the Shrinkage Factor). *Let $\mathcal{H} = \{h_k\}$ be a δ-secure TCR with shrinkage factor of $1-\varepsilon$ and key sampler $\mathcal{K}$. For any constant integer $t \geq 1$, the collection $\mathcal{H}^t$ (defined below) is $t\delta$-secure TCR with shrinkage factor of $(1-\varepsilon)^t$. The collection $\mathcal{H}^t$ is defined recursively, via*

$$\mathcal{H}^t = \{h_{k_1,\ldots,k_t}\}, \quad h_{k_1,\ldots,k_t}(x) = h_{k_t}(h_{k_1,\ldots,k_{t-1}}(x)), \quad \textit{where } k_i \overset{R}{\leftarrow} \mathcal{K}(1^{n(1-\varepsilon)^{i-1}}).$$

A proof for $t = 2$ follows from [8, Lemma 3.2]. The case of arbitrary constant t follows by induction (or can be proven directly via a similar argument).

5.2 Reducing the Distance Parameter β

In this section we transform β-TCR to standard TCR (with some loss in hardness and shrinkage). Such a transformation can be easily obtained (non-locally) by encoding the input x via an error-correcting code. Here we provide a local alternative which employs low-density parity-check matrices (LDPC). Such matrices will also be used to amplify the hardness parameter δ in the next section.

LDPC. In order to amplify the distance parameter β we will need *sparse parity check matrices* of a good code. Let $m < n$ be an integer. We say that a matrix $M \in \mathbb{Z}_2^{m \times n}$ has a *dual (relative) distance* of $\beta \in (0,1)$ if the Hamming weight of every non-zero codeword $x \in \ker(M) = \{x | Mx = 0\}$ is larger than βn. We say that a family $\mathcal{M}_{m(n)\times n}$ of efficiently samplable distributions over matrices in $\{0,1\}^{m(n)\times n}$ is a low-density parity check code with error δ and distance β (in short, (δ, β)-LDPC) if (1) with probability at least $1-\delta$ a matrix $M \xleftarrow{R} \mathcal{M}_{m(n)\times n}$ has dual distance of β and (2) all matrices M in the support of $\mathcal{M}$ are *sparse* in the sense that the number of ones in each row and each column is bounded by some absolute constant d which does not depend on n. We will make use of the following proposition due to [12, Thm. 7.1].

Proposition 1. *For every $\varepsilon \in (0,1)$ there exists an efficiently samplable distribution $\mathcal{M}_{\varepsilon n \times n}$ of $(0, \beta(\varepsilon))$-LDPC for some $\beta = \varepsilon/\mathrm{polylog}(1/\varepsilon)$.*

Lemma 3 (β-TCR to TCR). *Let $\varepsilon' < \varepsilon$ and let $\mathcal{M}_{\varepsilon' n \times n}$ be an (δ', β)-LDPC. Let $\mathcal{H} = \{h_k\}$ be δ-secure β-TCR with shrinkage factor of $1-\varepsilon$ and key sampler $\mathcal{K}$, and define*

$$\mathcal{H}' = \{h'_{k,M}\} \quad h'_{k,M} = (h_k(x), Mx), \qquad \text{where } (k, M) \xleftarrow{R} (\mathcal{K}(1^n), \mathcal{M}_{\varepsilon' n \times n})$$

Then, $\mathcal{H}'$ is $(\delta + \delta')$-secure TCR with shrinkage factor of $1 - \varepsilon + \varepsilon'$.

Proof. We need the following observation: when $M \xleftarrow{R} \mathcal{M}$ has a dual distance of β, any pair of distinct strings x and x' which collide under $h'_{k,M}$ must be β-far. Indeed, if this is not the case then, since $Mx = Mx'$, the vector $x \oplus x'$ is a non-zero vector in the kernel of M whose Hamming weight is smaller than βn, in contrast to our assumption. The lemma now follows easily.

Let $\mathcal{A}_2$ be an TCR adversary that, given $(x, r) \xleftarrow{R} \mathcal{A}_1(1^n)$ and $h_{k,M} \xleftarrow{R} \mathcal{H}'$, finds a collision x' with x under $h_{k,M}$ with probability $\delta_{\mathcal{A}}$. To prove the lemma we define an adversary $\mathcal{B}$ that finds a β-close collision x' with $x \xleftarrow{R} \mathcal{A}_1(1^n)$ under $h_k \xleftarrow{R} \mathcal{H}$ with probability $\delta_{\mathcal{B}} \geq \delta_{\mathcal{A}} - \delta'$. Given a key $k \xleftarrow{R} \mathcal{K}(1^n)$ and a target/state pair $(x, r) \xleftarrow{R} \mathcal{A}_1(1^n)$, the adversary $\mathcal{B}$ samples $M \xleftarrow{R} \mathcal{M}$ and call $\mathcal{A}_2$ with $h_{k,M}$. Let good be the set of matrices whose dual distance is β and let us say that $\mathcal{A}$ *wins* if it outputs a valid collision x' with x under $h_{k,M}$, i.e., $x' \neq x, h_k(x) = h_k(x')$ and $Mx = Mx'$. Then we can write

$$\begin{aligned}
\delta_{\mathcal{A}} = &\Pr_{k,M,x,r}[\mathcal{A}_1(k, M, x, r) \text{ wins } | M \in \text{good}] \cdot \Pr_M[M \in \text{good}] \\
&+ \Pr_{k,M,x,r}[\mathcal{A}_1(k, M, x, r) \text{ wins } | M \notin \text{good}] \cdot \Pr_M[M \notin \text{good}] \\
\leq &\Pr_{k,M,x,r}[\mathcal{A}_1(k, M, x, r) \text{ wins } | M \in \text{good}] \cdot (1 - \delta') + \delta' \\
\leq &\ \delta_{\mathcal{B}} + \delta',
\end{aligned}$$

where the last inequality follows from the observation. □

Observe that the above transformation is local since the family $\mathcal{M}$ is sparse.

5.3 Hardness Amplification

We move on to amplify the hardness parameter δ from constant to negligible. In addition to LDPCs (i.e., sparse shrinking linear transformations), we employ *Distance Amplifiers* (i.e., sparse linear transformations which expands the input) which has the property of mapping any pair (x, x') of far-apart inputs to a pair of far apart outputs (y, y'). This can be seen as a relaxation of standard error-correcting codes which amplify the distance between any pair of *distinct* inputs.

Distance Amplifiers. Let $m > n$ be an integer and $\beta, \gamma \in (0, 1)$ be constants. We say that a matrix $T \in \mathbb{Z}_2^{m\times n}$ is $(\beta \to \gamma)$-*distance amplifying* if for every pair $x, x' \in \{0,1\}^n$ of β-far strings the m-bit strings Tx and Tx' are γ-far. (Jumping ahead, we note that γ is allowed to be smaller than β as long as it is larger than the hardness parameter δ.) We say that a family $\mathcal{T}_{m(n)\times n}$ of efficiently samplable distributions over matrices in $\{0,1\}^{m(n)\times n}$ is a $(\beta \to \gamma)$ sparse distance amplifier (in short, $(\beta \to \gamma)$-SDA) if (1) with all but negligible probability a matrix $T \overset{R}{\leftarrow} \mathcal{T}_{m(n)\times n}$ is $(\beta \to \gamma)$-distance amplifying and (2) all matrices T in the support of $\mathcal{T}$ are sparse, meaning that the number of ones in each row and each column is bounded by some absolute constant d which does not depend on n. In the full version we prove the following proposition.

Proposition 2. *For every constant $\beta \in (0, 1)$ and constant $\gamma \in (0, \frac{1}{2})$ there exists an efficiently samplable $(\beta \to \gamma)$-SDA $\mathcal{T}_{cn\times n}$ where $c = c(\beta, \gamma)$ is a constant.*

Let $T \in \{0,1\}^{cn^2\times n^2}$. In the following we think of the linear mapping $x \mapsto Tx$ as a mapping from n^2-bit strings to a tuple of cn strings of length n each. Accordingly, for $i \in [cn]$ we let $(Tx)_i \in \{0,1\}^n$ denote the i-th entry of Tx.

Lemma 4 (Hardness Amplification). *Let $\mathcal{H} = \{h_k : \{0,1\}^n \to \{0,1\}^{\varepsilon_1 n}\}$ be δ-secure β-TCR with key sampler $\mathcal{K}$, let $\mathcal{M}_{\varepsilon_0 n^2\times n^2}$ be a β-LDPC, and $\mathcal{T}_{cn^2\times n^2}$ be a $(\beta \to \gamma)$-SPA, where the constants $\varepsilon_0, \varepsilon, \gamma, \delta \in (0, 1)$ and $c > 1$ satisfy $\varepsilon_0 + \varepsilon c < 1$ and $\delta < \gamma$. Then, the following collection $\mathcal{H}'$ which shrinks n^2-bit strings by a factor of $\varepsilon_0 + \varepsilon c$ is a standard TCR:*

$$h'_{(k_1,\ldots,k_{cn}),M,T} : x \mapsto (Mx, h_{k_1}((Tx)_1), \ldots, h_{k_{cn}}((Tx)_{cn})),$$

where $M \overset{R}{\leftarrow} \mathcal{M}_{\varepsilon_0 n^2\times n^2}, T \overset{R}{\leftarrow} \mathcal{T}_{cn^2\times n^2}$ and $k_i \overset{R}{\leftarrow} \mathcal{K}(1^n)$ for $i \in [cn]$.

Proof. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary that breaks $\mathcal{H}'$ with probability $\delta_{\mathcal{A}}$. We construct an adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ that given cn independent samples of $\mathcal{H}$ finds collisions on γ fraction of them with probability $\delta_{\mathcal{B}}$. Namely, let $\delta_{\mathcal{B}}$ be

$$\Pr_{\substack{\boldsymbol{k}\overset{R}{\leftarrow}\mathcal{K}^{cn}(1^n)\\(\boldsymbol{y},R)\overset{R}{\leftarrow}\mathcal{B}_1(1^n)}} [\mathcal{B}_2(\boldsymbol{k}, \boldsymbol{y}, R) = \boldsymbol{y'} \text{ s.t. } |\{i : (y_i \neq y'_i) \wedge (h_{k_i}(y_i) = h_{k_i}(y'_i))\}| \geq \gamma cn],$$

where $\boldsymbol{k} = (k_1, \ldots, k_{cn}), \boldsymbol{y} = (y_1, \ldots, y_{cn})$, and $\boldsymbol{y'} = (y'_1, \ldots, y'_{cn})$. A general *threshold direct product theorem* of Impagliazzo and Kabanets [16, Thm 5.2] shows that the advantage $\delta_{\mathcal{B}}$ is upper-bounded by $2^{-cnD(\gamma\|\delta)} + \text{neg}(n) = \text{neg}(n)$. Hence, to prove the lemma it suffices to show that

$$\delta_{\mathcal{A}} - \text{neg}(n) \leq \delta_{\mathcal{B}}.$$

Let us define $\mathcal{B}$. The target sampler $\mathcal{B}_1(1^n)$ samples $M \overset{R}{\leftarrow} \mathcal{M}_{\varepsilon_0 n^2 \times n^2}, T \overset{R}{\leftarrow} \mathcal{T}_{cn^2 \times n^2}$ and $(x, r) \overset{R}{\leftarrow} \mathcal{A}_1(1^n)$. It outputs the state $R = (M, T, x, r)$ and the target vector $\boldsymbol{y} = (y_1, \ldots, y_{cn})$ where $y_i = (Tx)_i$. Given $(\boldsymbol{k}, \boldsymbol{y}, R = (x, r, M, T))$, the collision-finder $\mathcal{B}_2$ passes to $\mathcal{A}_2$ the key $(\boldsymbol{k}, M, T)$, the target x, and the state r, and asks for a collision x' under $h'_{\boldsymbol{k},M,T}$. The output of $\mathcal{B}_2$ is $\boldsymbol{y'} = (y'_1, \ldots, y'_{cn})$ where $y'_i = (Tx')_i$. We say that $\mathcal{A}_1(\boldsymbol{k}, M, T, x, r)$ *wins* if its output x' collide with x under $h'_{\boldsymbol{k},M,T}$ and $x \neq x'$. A pair (M, T) is good if M has dual distance of β and T is $(\beta \to \gamma)$ distance amplifying. We claim that

$$\delta_{\mathcal{A}} - \text{neg}(n) \leq \Pr_{\boldsymbol{k},M,T,x,r}[\mathcal{A}_1(\boldsymbol{k}, M, T, x, r) \text{ wins } |(M, T) \in \text{good}] \leq \delta_{\mathcal{B}}.$$

The first inequality follows from Bayes' law together with $\Pr[\text{good}] > 1 - \text{neg}(n)$. As for the second inequality, observe that if $\mathcal{A}$ wins and (M, T) are good then the collision x and x' must be β-far (as $Mx = Mx'$) and therefore Tx and Tx' must disagree on at least γcn^2 coordinates. Hence, for at least γ fraction of $i \in [cn]$ we have that $(Tx)_i \neq (Tx')_i$. Furthermore, $h_{k_i}((Tx)_i) = h_{k_i}((Tx')_i)$ for all $i \in [cn]$ since $\mathcal{A}$ wins. Hence, in this case $\mathcal{B}$ wins as well and the claim follows. □

Theorem 5 follows by combining Claims 6, 7 and Lemmas 3, 4 with properly chosen parameters. See full version for details.

Acknowledgement. We thank Uri Feige and Danny Vilenchik for valuable discussions.

References

1. Achlioptas, Ricci-Tersenghi: On the solution-space geometry of random constraint satisfaction problems. In: STOC: ACM Symposium on Theory of Computing (STOC) (2006)
2. Alekhnovich, M., Hirsch, E.A., Itsykson, D.: Exponential lower bounds for the running time of DPLL algorithms on satis_able formulas. J. Autom. Reasoning 35(1-3), 51–72 (2005)
3. Applebaum, B.: Pseudorandom generators with long stretch and low locality from random local one-way functions. In: STOC, pp. 805–816 (2012)
4. Applebaum, B., Bogdanov, A., Rosen, A.: A dichotomy for local small-bias generators. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 600–617. Springer, Heidelberg (2012)
5. Applebaum, B., Ishai, Y., Kushilevitz, E.: Computationally private randomizing polynomials and their applications. Journal of Computational Complexity 15(2), 115–162 (2006)
6. Applebaum, B., Ishai, Y., Kushilevitz, E.: Cryptography in NC^0. SIAM Journal on Computing 36(4), 845–888 (2006)

7. Applebaum, B., Ishai, Y., Kushilevitz, E.: Cryptography with constant input locality. Journal of Cryptology 22(4), 429–469 (2009)
8. Bellare, M., Rogaway, P.: Collision-resistant hashing: Towards making UOWHFs practical. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 470–484. Springer, Heidelberg (1997)
9. Bogdanov, A., Qiao, Y.: On the security of goldreich's one-way function. In: Dinur, I., Jansen, K., Naor, J., Rolim, J. (eds.) APPROX and RANDOM 2009. LNCS, vol. 5687, pp. 392–405. Springer, Heidelberg (2009)
10. Bogdanov, A., Rosen, A.: Input locality and hardness amplification. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 1–18. Springer, Heidelberg (2011)
11. Canetti, R., Rivest, R., Sudan, M., Trevisan, L., Vadhan, S.P., Wee, H.M.: Amplifying collision resistance: A complexity-theoretic treatment. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 264–283. Springer, Heidelberg (2007)
12. Capalbo, Reingold, Vadhan, Wigderson: Randomness conductors and constant-degree lossless expanders. In: STOC: ACM Symposium on Theory of Computing (STOC) (2002)
13. Cook, J., Etesami, O., Miller, R., Trevisan, L.: Goldreich's one-way function candidate and myopic backtracking algorithms. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 521–538. Springer, Heidelberg (2009)
14. Goldreich, O.: Candidate one-way functions based on expander graphs. Electronic Colloquium on Computational Complexity (ECCC) 7(090), `citeseer.nj.nec.com/382413.html` (2000)
15. Haitner, I., Holenstein, T., Reingold, O., Vadhan, S.P., Wee, H.: Universal one-wayhash functions via inaccessible entropy. IACR Cryptology ePrint Archive 2010, 120 (2010)
16. Impagliazzo, R., Kabanets, V.: Constructive proofs of concentration bounds. In: Serna, M., Shaltiel, R., Jansen, K., Rolim, J. (eds.) APPROX and RANDOM 2010. LNCS, vol. 6302, Springer, Heidelberg (2010)
17. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Cryptography with constant computational overhead. In: Proc. of 40th STOC, pp. 433–442 (2008)
18. Knudsen, Preneel: Construction of secure and fast hash functions using nonbinary error-correcting codes. IEEETIT: IEEE Transactions on Information Theory 48 (2002)
19. Lord, N.: Binomial averages when the mean is an integer. The Mathematical Gazette 94, 331–332 (2010)
20. Mossel, E., Shpilka, A., Trevisan, L.: On _-biased generators in NC^0. Proc. 44th FOCS, 136–145 (2003)
21. Naor, Yung: Universal one-way hash functions and their cryptographic applications. STOC: ACM Symposium on Theory of Computing (STOC) (1989)
22. Panjwani, S.K.: An experimental evaluation of goldreich's one-way function. Tech.rep., IIT, Bombay oded/PS/ow-report.ps (2001), `http://www.wisdom.weizmann.ac.il/`
23. Rogaway, P., Shrimpton, T.: Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In: Roy, B., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 371–388. Springer, Heidelberg (2004)
24. Rompel: One-way functions are necessary and su_cient for secure signatures. STOC: ACM Symposium on Theory of Computing (STOC) (1990)

Amplification of Chosen-Ciphertext Security

Huijia Lin[1,2] and Stefano Tessaro[2]

[1] Boston University
[2] MIT
{huijia,tessaro}@csail.mit.edu

Abstract. A central question in the theory of public-key cryptography is to determine which minimal assumptions are sufficient to achieve security against chosen-ciphertext attacks (or CCA-security, for short). Following the large body of work on hardness and correctness amplification, we investigate how far we can *weaken* CCA security and still be able to efficiently transform any scheme satisfying such a weaker notion into a fully CCA-secure one.

More concretely, we consider a weak CCA-secure bit-encryption scheme with decryption error $(1-\alpha)/2$ where an adversary can distinguish encryptions of different messages with possibly large advantage $\beta < 1 - 1/\mathsf{poly}$. We show that whenever $\alpha^2 > \beta$, the weak correctness and security properties can be simultaneously amplified to obtain a fully CCA-secure encryption scheme with negligible decryption error. Our approach relies both on a new hardcore lemma for CCA security as well as on revisiting the recently proposed approach to obtain CCA security due to Hohenberger *et al* (EUROCRYPT '12).

We note that such amplification results were only known in the simpler case of security against chosen-*plaintext* attacks.

1 Introduction

1.1 Public-Key Encryption and CCA Security

The seminal work of Goldwasser and Micali [1] introduced *semantic security* as the basic security notion for public-key encryption. Semantic security demands that no polynomial-time adversary, given only the public key, can distinguish encryptions of any two messages m_0 and m_1 of its choice, except with negligible advantage. Often, this notion is also referred to as *security against a chosen plaintext attack*, or CPA security, for short. This is in contrast to the stronger notion of *(adaptive) chosen-ciphertext security* (CCA security, for short) [2], where the above indistinguishability requirement must hold true even for adversaries with the additional ability to query a decryption oracle; as CCA security is required by many applications, it is by now considered to be the golden standard for secure public-key encryption.

In contrast to the case of CPA security, where simple constructions from generic assumptions (such as trapdoor permutations (TDP)) can be given, delivering CCA-secure public-key encryption from general assumptions proved itself to be a much more challenging problem. In particular, determining whether

CCA-secure public-key encryption can be achieved solely from CPA-secure public-key encryption remains a major longstanding open question. Constructions additionally relying on non-interactive zero-knowledge proof systems (NIZKs) are known [3,4,2,5]. But, so far, all constructions of NIZKs require the existence of (enhanced) TDPs, which are not known to be implied by CPA-secure encryption; furthermore, known constructions based on NIZKs are all non-black-box. It is in fact likely that no black-box construction of a CCA-secure scheme from a CPA-secure one exists, as confirmed at least for a certain natural class of constructions [6]. For this reason, efficient constructions have been instead given from more concrete families of assumptions, such as hash proof systems and variants thereof [7,8], lossy TDFs [9], correlated-product secure TDFs [10], adaptive TDFs [11], or using random oracles [12,13].

1.2 Our Results: From Weak to Strong CCA Security

In this paper, we ask and answer the following question:

> *"How far can we weaken CCA security and still provide a black-box construction of a CCA-secure encryption scheme from a scheme only satisfying the weaker notion?"*

Our approach follows the one of the large body of works on *security amplification*, which has considered a wide range of cryptographic primitives and was initiated by Yao [14] in the context of one-way functions. Interestingly, limited work has been devoted to amplification of *public-key encryption*. The problem was first considered by Dwork, Naor, and Reingold [15] for CPA-secure public-key encryption. Constructions achieving better parameters were later proposed by Holenstein [16] and by Holenstein and Renner [17]. However, the question of amplifying *CCA* security has remained wide open. This is the question that we tackle and solve in this work.[1]

Modeling Weak CCA Encryption. Our model of weak CCA encryption extends naturally the model of weak CPA encryption considered in [15,17]. We start from a bit-encryption[2] scheme with key generation algorithm Gen, encryption algorithm Enc, and decryption algorithm Dec, and weaken it in two different directions, allowing both for *non-negligible decryption errors* as well as for *non-negligible adversarial advantage* in a chosen-ciphertext attack. More concretely, for two given parameters $0 < \alpha, \beta \leq 1$, where $\alpha \geq 1/p(\kappa)$ and $\beta < 1 - 1/q(\kappa)$ for some polynomials p and q, we assume the following two conditions:

[1] Note that in the *secret*-key setting, amplification of CCA security is, at least in principle, known to be feasible, as any weak form of CCA security implies weak one-way functions, and these are sufficient to build CCA-secure symmetric-key encryption via standard techniques.

[2] As every meaningful encryption scheme has at least the ability to encrypt a binary value, this is the weakest possible assumption in terms of message space of the basic scheme.

(i) **α-weak decryptability:** The decryption error over a random key-pair and a random bit is at most $\frac{1-\alpha}{2}$. We stress that this is a very weak guarantee, as it is taken over *random* choices of the keys and of the bit b, as well as of the coins used to encrypt b.

(ii) **β-weak security:** We consider the usual CCA-security game where an adversary obtains first the public key, and later a challenge ciphertext encrypting a random bit b. Moreover, the adversary can ask arbitrary decryption queries, with the sole exception that after the adversary obtains the challenge ciphertext, it cannot ask for its decryption. The task of the adversary is to output a guess b', and we are going to require that $\mathsf{Pr}[b' = b] \leq \frac{1+\beta}{2}$ for all polynomial-size adversaries.

Justifying weak CCA security. There are several reasons why assuming the existence of such a weak scheme is reasonable. Let us mention some natural examples.

- Within the general agenda of achieving CCA security from general assumptions, we may envision that a construction of a weak CCA scheme is potentially much easier to find than a construction of a full-fledged CCA-secure encryption scheme.
- An existing scheme designed to be CCA-secure may end up being less secure than expected due to the discovery of a better concrete attack or due to implementation errors, as in the recently discussed case of faulty key generation for RSA-based systems [18,19].
- It may be generally easier to build a CCA-secure scheme with large decryption errors. For example, as pointed out in [20], an encryption scheme with a simple, easily learnable, decryption algorithm must have large decryption error. In contrast to CPA encryption, reducing the decryption error turns out to be a major challenge in the case of CCA-secure encryption, *even* if the scheme is already fully CCA secure.

Our main result. The question we are going to ask is whether for certain α and β, there exists a transformation which delivers a CCA-secure encryption scheme from any scheme which has α-weak decryptability and β-weak security. We provide an affirmative answer to this question.

Theorem 1 (Main theorem, informal). *If $\alpha^2 > \beta$, there exists a black-box construction transforming any scheme with α-weak decryptability and β-weak security into a CCA-secure encryption scheme with negligible decryption error.*

Unfortunately, we cannot rule out constructions achieving a wider range of parameters α and β. In fact, we remark that the problem of determining the optimal parameters is open even in the simpler case of amplifying weak CPA security. While the constraint $\alpha^2 > \beta$ is shown [17] to be necessary for a restricted class of CPA black-box amplifiers, we see little value in extending this result to CCA security, as our amplifier itself is not within this class.

1.3 Our Techniques

We now turn to a high-level overview of our techniques. In particular, our approach builds upon a number of previous works [17,21,22], which we first review. Then, we will move to a description of our two main new tools, namely hardcore lemmas for CCA-security and heavy-ciphertext pre-sampling, and of their use.

Amplification of CPA Encryption. Given a bit-encryption scheme PKE with α-weak decryptability and β-weak security with respect to chosen-plaintext attacks, the Holenstein-Renner (HR) construction [17] produces a fully CPA-secure encryption scheme with negligible decryption error. To encrypt each message m, the HR construction invokes the basic bit-encryption scheme PKE to encrypt several fresh random bits $b_1, \cdots, b_n$ under n public keys $\mathsf{pk}_1, \cdots, \mathsf{pk}_n$, producing ciphertexts $c_1, \cdots, c_n$; the bits $b_1, \cdots, b_n$ are then carefully "combined" to generate a one-time-pad k for hiding the actual message m, as well as some additional ciphertext component c'; the additional component c' is used by the legitimate receiver, given the secret keys, to reconstruct the one-time pad, but it should not leak any information about k to the adversary. The final ciphertext is $c = (c_1, \ldots, c_n, c', m \oplus k)$.

The reason why such a combiner can exist is that the probability that the legitimate receiver, given the secret keys, can learn each individual bit b_i from c_i is $(1 + \alpha)/2$, which we expect to be sufficiently larger than the probability that the adversary learns b_i from c_i *without* the secret keys. To make this intuition sound, one uses Impagliazzo's hardcore lemma [23] and its tighter version by Holenstein [16]: The lemma implies that if PKE is β-weakly CPA secure, then, for each i, with probability $1 - \beta$ (over the choice of b_i, the randomness for sampling pk_i and encrypting b_i), the encryption of b_i is a "hard instance", meaning that given its encryption c_i, the bit b_i is (computationally) indistinguishable from a random independent bit. This gap between what an honest decryptor and an eavesdropper can recover can be leveraged by an information-theoretically secure one-way key-agreement protocol as in the setting of Maurer [24], which turns out to provide directly the right type of combiner.

From bit CCA Encryption to String CCA Encryption. It is well known that a CPA-secure string encryption scheme can be built from a CPA-secure bit-encryption scheme via simple parallel encryption of each bit. However, this approach does not lift to extending the message space of CCA-secure bit encryption, as an adversary can easily maul a challenge ciphertext $c_1 \cdots c_i \cdots c_n$ of a n-bit string $b_1 \cdots b_i \cdots b_n$ into another ciphertext $c_1 \cdots c'_i \cdots c_n$ of a related string $b_1 \cdots 0 \cdots b_n$, and thus win in the CCA security game—additional structure is needed to retain CCA security. Myers and shelat [21] showed that although this approach is not CCA secure, it satisfies a weaker adaptive security property—called UCCA security—which requires indistinguishability to hold for adversaries that can query a decryption oracle on any ciphertext $c_1, \cdots, c_n$ of their choice, except those that "quote" the challenge ciphertext, denoted as $c^*_1, \cdots, c^*_n$, at any of its components, that is $c_i = c^*_i$ for some i. Myers and shelat, and later Hohenberger, Lewko, and Waters (HLW) [22], showed how to construct a string CCA-secure scheme $\overline{\mathsf{PKE}}$ from such a UCCA-secure string encryption scheme

PKE_s.[3] We briefly review the HLW construction: It uses PKE_s as an *inner* encryption scheme $\mathsf{PKE}_{\mathsf{in}} = \mathsf{PKE}_s$ and two *outer* schemes $\mathsf{PKE}_{\mathsf{out},1}$, $\mathsf{PKE}_{\mathsf{out},2}$ that are CCA-1 and CPA secure respectively. To encrypt a message m, the encryption algorithm proceeds by encrypting m together with two random strings $r_{\mathsf{out},1}$ and $r_{\mathsf{out},2}$ into an *inner ciphertext* $c_{\mathsf{in}} = \mathsf{Enc}_{\mathsf{in}}(\mathsf{pk}_{\mathsf{in}}, (m, r_{\mathsf{out},1}, r_{\mathsf{out},2}))$; it then encrypts the inner ciphertext into two outer ciphertexts $(c_{\mathsf{out},1}, c_{\mathsf{out},2})$ using $r_{\mathsf{out},1}$ and $r_{\mathsf{out},2}$ respectively as the randomness for encryption, that is, $c_{\mathsf{out},i} = \mathsf{Enc}_{\mathsf{out},i}(\mathsf{pk}_{\mathsf{out},i}, c_{\mathsf{in}}; r_{\mathsf{out},i})$ for $i = 1, 2$; the final ciphertext is simply $(c_{\mathsf{out},1}, c_{\mathsf{out},2})$. At a high level, the two outer schemes prevent the adversary from issuing a decryption query for a ciphertext whose embedded inner ciphertext "quotes" that in the challenge ciphertext, thus reducing CCA to UCCA security.

Our Approach. A seemingly plausible attempt for constructing a CCA-secure encryption scheme from a weak scheme PKE with α-decryptability and β-weak CCA-security is to first try to show that the HR construction PKE', when instantiated with PKE as the basic bit-encryption scheme, is UCCA secure, and subsequently plugging PKE' as the inner encryption scheme into the HLW construction $\overline{\mathsf{PKE}}$, and show that it yields a CCA-secure encryption scheme.

Unfortunately, we encounter the following two challenges: First, it is unclear whether the weak CCA security of PKE is amplified through the construction of PKE' to UCCA security; in particular, known hardcore lemmas [23,16] only hold for games where the challenger is stateless, but the challenger in the CCA security game is stateful (it changes its behavior before and after the challenge ciphertext is generated). Second, it turns out that the security proof of the HLW construction requires the basic scheme PKE to have "unpredictability"—that is, a random cipheretxt (of a random bit) of PKE has high entropy and is almost impossible to blindly guess—which holds trivially for any fully-secure CPA encryption scheme with negligible decryption error, but is not satisfied by a weak CCA encryption scheme.

Overcoming these two difficulties turns out to be quite challenging and requires the adoption of new techniques, which we now illustrate.

Step 1: The Hardcore Lemma for CCA security and XCCA security. To overcome the first difficulty, we prove a variant of Impagliazzo's hardcore lemma which applies to CCA security (Theorem 2 below): It implies that if a scheme is weakly β-CCA-secure, then with probability $1 - \beta$ (over the randomness for choosing a random plaintext bit, for key generation, and for encryption), given an encryption of a random bit b, b is indistinguishable from a random *independent* bit even to adversaries with access to the decryption oracle. Our new hardcore lemma can be used to prove that PKE' satisfies an even stronger adaptive security property than UCCA, called XCCA (read as "cross"-CCA), which guarantees indistinguishability even for adversaries with access to decryption oracles that decrypts ciphertext of the basic scheme PKE under each individual

[3] In fact, [22] showed a more general construction of string CCA encryption schemes from any encryption scheme that is DCCA secure and unpredictable. In particular, UCCA security is a special case of DCCA security.

component key of PKE', subject to the restriction that the decryption oracle for the i-th component does not answer queries that "quote" the corresponding component in the challenge ciphertext. As we will see shortly, this stronger security guarantee is quintessential for overcoming the second difficulty.

Finally, rather than presenting a direct proof of the hardcore lemma for CCA security, we provide a general characterization of games for which hardcore lemmas exist, which extends beyond games for which such lemmas are known [23,16,25]. Our hardcore lemma for CCA-security is then simply derived as a special case. We believe this step to be of independent interest.

Step 2: From XCCA security to CCA security. We prove that the CCA security of $\overline{\mathsf{PKE}}$ can be based on the stronger XCCA security of the inner encryption PKE', even if the underlying basic scheme PKE is not sufficiently "unpredictable" – in contrast to the proof in [22]. This requires a substantially different analysis than the one of [22], and in particular a new reduction. Concretely, we overcome lack of unpredictability by introducing a new technique called *heavy-ciphertext pre-sampling.* Roughly speaking, this technique allows the security reduction (from CCA security of $\overline{\mathsf{PKE}}$ to XCCA security of PKE') to proactively predict and decrypt all highly likely ciphertexts of PKE, and the challenging task is to prove that these are the only components of the inner challenge ciphertext an adversary may indeed easily "quote" after seeing the challenge ciphertext.

2 Preliminaries

2.1 Basic Concepts and Notation

The probability distribution of a random variable X is usually denoted as P_X, and we occasionally use the shorthand $\mathsf{P}_X(x)$ for $\mathsf{Pr}[X = x]$. Adversaries are going to be modeled as non-uniform families of (randomized) circuits for ease of exposition, but all results extend with some work to the uniform setting.

2.2 Weak and Strong CCA-secure Encryption

A *public-key encryption scheme* with message space $\mathcal{M} \subseteq \{0,1\}^*$ is a triple $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, where (i) Gen is the (randomized) *key generation algorithm*, outputting a pair $(\mathsf{pk}, \mathsf{sk})$ consisting of a *public-* and a *secret-key*, respectively (ii) Enc is the (randomized) *encryption algorithm* outputting a ciphertext $c = \mathsf{Enc}(\mathsf{pk}, m)$ for any message $m \in \mathcal{M}$ and a valid public key pk; and (iii) Dec is the deterministic *decryption algorithm* such that $\mathsf{Dec}(\mathsf{sk}, c) \in \mathcal{M} \cup \{\perp\}$. All algorithms additionally take (implicitly) as input the security parameter 1^κ in unary form, and the message space $\mathcal{M}$ may also depend on the security parameter κ. Whenever $\mathcal{M} = \{0,1\}$, we say that the scheme is a *bit-encryption* scheme. We sometimes need to make the randomness used by Gen and Enc explicit: In these cases, we write $\mathsf{Gen}(r)$ and $\mathsf{Enc}(\mathsf{pk}, m; r)$ to highlight the fact that random coins r are used to generate keys by Gen and to encrypt the message m, respectively.

Correctness of PKE. Throughout this paper, we say that the encryption scheme PKE with message space $\mathcal{M}$ has *decryption error* δ if

$$\Pr\left[(\mathsf{pk},\mathsf{sk}) \xleftarrow{\$} \mathsf{Gen},\ m \xleftarrow{\$} \mathcal{M}:\ \mathsf{Dec}(\mathsf{sk},\mathsf{Enc}(\mathsf{pk},m)) \neq m\right] \leq \delta,$$

where the probability is additionally over the random coins of Enc. Moreover, we say that a scheme is *almost perfectly correct*, if for an overwhelming fraction of randomness r used by the key generation algorithm, for $(\mathsf{pk},\mathsf{sk}) = \mathsf{Gen}(r)$, and all messages $m \in \mathcal{M}$, we have $\Pr[\mathsf{Dec}(\mathsf{sk},\mathsf{Enc}(\mathsf{pk},m)) = m] = 1$.

Security of PKE. In general, security of the scheme $\mathsf{PKE} = (\mathsf{Gen},\mathsf{Enc},\mathsf{Dec})$ is defined via the following security game involving a *challenger* $\mathcal{C}_{\mathsf{CCA2}}$ and an adversary $\mathcal{A}$:

Game $\mathsf{CCA2}^{\mathcal{A}}_{\mathsf{PKE}}$:

(i) $\mathcal{C}_{\mathsf{CCA2}}$ generates $(\mathsf{pk},\mathsf{sk}) \xleftarrow{\$} \mathsf{Gen}$ and $b \xleftarrow{\$} \{0,1\}$, and gives pk to $\mathcal{A}$.
(ii) $\mathcal{A}$ asks decryption queries c, which are answered with $\mathsf{Dec}(\mathsf{sk},c)$.
(iii) $\mathcal{A}$ outputs (m_0,m_1) with $|m_0| = |m_1|$; $\mathcal{C}_{\mathsf{CCA2}}$ sends $\mathcal{A}$ $c^* \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk},m_b)$.
(iv) $\mathcal{A}$ asks decryption queries $c \neq c^*$, which are answered with $\mathsf{Dec}(\mathsf{sk},c)$.
(v) The adversary $\mathcal{A}$ outputs a bit b', and *wins* the game if $b' = b$.

We refer to decryption queries in phase **(ii)** and **(iv)** as *before-the-fact* and *after-the-fact* decryption queries, respectively. Moreover, in the case that PKE is a bit-encryption scheme we assume without loss of generality that $(m_0,m_1) = (0,1)$, and hence $\mathsf{Enc}(\mathsf{pk},b)$ is the challenge ciphertext. We also define the $\mathsf{CCA2}$-*advantage* of the adversary $\mathcal{A}$ as $\mathbf{Adv}^{\mathsf{CCA2}}_{\mathsf{PKE}}(\mathcal{A}) = 2\cdot\Pr[b' = b] - 1$. We say that an encryption scheme is *CCA-secure* if $\mathbf{Adv}^{\mathsf{CCA2}}_{\mathsf{PKE}}(\mathcal{A})$ is negligible for all polynomial-size adversaries $\mathcal{A}$. We say it is *q-CCA-secure* if this holds for adversaries making at most q decryption queries, whereas it is *CPA-secure* if it is 0-CCA-secure. The following notation will also be convenient.

Definition 1. *For $\alpha,\beta \in [0,1]$, a bit-encryption scheme* PKE *is* (α,β)-CCA-secure *if the following two properties hold:* **(i)** PKE *has decryption error* $(1-\alpha)/2$*, and* **(ii)** *For any polynomial-size adversary $\mathcal{A}$, we have $\mathbf{Adv}^{\mathsf{CCA2}}_{\mathsf{PKE}}(\mathcal{A}) \leq \beta$.*

In passing, we point out that CPA-secure encryption with negligible decryption error implies one-way functions [26], and in turn implies pseudorandom generators [27], all in a black-box way.

3 The Hardcore Lemma for CCA Security

Impagliazzo's Hardcore Lemma [23] asserts that if it is mildly hard to compute $P(x)$ for a predicate P on a random input x given side information $f(x)$ (i.e., say this can be done with probability at most $\frac{1+\varepsilon}{2}$), then there exists a sufficiently large subset $\mathcal{S}$ (the "hardcore set") of the inputs such that when sampling x' from $\mathcal{S}$, it is infeasible to predict $P(x')$ from $f(x')$ noticeably better than by random guessing. A tight proof where the set $\mathcal{S}$ contains a $(1-\varepsilon)$-fraction of the inputs is due to Holenstein [16]. The main contribution of this section is to derive a similar statement for (weak) CCA-secure encryption to be used below.

In particular, we present a new abstraction of existing proofs of hardcore lemmas, which is of independent interest. Not only we apply it to derive the hardcore lemma for CCA security of bit-encryption, but it also yields previous more restricted statements [23,25] as special cases.

Bit-guessing games. We consider games (such as the CCA-security game) where the adversary is asked to guess a bit. Formally, a *bit-guessing game* is a tuple $G = (\mathsf{P}_X, \mathcal{C}, P)$, where P_X is a probability distribution with support $\mathcal{X}$, $\mathcal{C}$ is an interactive *stateful* machine taking an auxiliary input $x \in \mathcal{X}$, and $P : \mathcal{X} \to \{0,1\}$ is a predicate. Combined with an adversary $\mathcal{A}$, G defines the following random experiment: First, an input $x \xleftarrow{\$} \mathsf{P}_X$ is sampled. Then $\mathcal{A}$ interacts with the challenger $\mathcal{C}(x)$ and outputs a bit $b \xleftarrow{\$} \mathcal{A}^{\mathcal{C}(x)}$ (the oracle $\mathcal{C}(x)$ keeps state). The *G-advantage of $\mathcal{A}$ relative to a distribution* P is

$$\mathbf{Adv}^{G}_{\mathsf{P}}(\mathcal{A}) = 2 \cdot \mathsf{Pr}\left[x \xleftarrow{\$} \mathsf{P},\ b \xleftarrow{\$} \mathcal{A}^{\mathcal{C}(x)} :\ b = P(x)\right] - 1\ . \qquad (1)$$

We say that G is (s,ε)*-hard* if $\mathbf{Adv}^{G}_{\mathsf{P}_X}(\mathcal{A}) \leq \varepsilon$ for all s-size adversaries $\mathcal{A}$.

Hardcore lemmas and measures. A *measure* $\mathcal{M}$ for a bit-guessing game G is a mapping $\mathcal{M} : \mathcal{X} \to [0,1]$, and its *density* is $\mu(\mathcal{M}) = \sum_{x \in \mathcal{X}} \mathsf{P}_X(x) \cdot \mathcal{M}(x)$. We associate with $\mathcal{M}$ the probability distribution $\mathsf{P}_{\mathcal{M}}$ such that $\mathsf{P}_{\mathcal{M}}(x) := \mathsf{P}_X(x) \cdot \mathcal{M}(x)/\mu(\mathcal{M})$ for all $x \in \mathcal{X}$. The role of a measure is that of adjoining an event $\mathcal{E}$ to the sampling of $x \xleftarrow{\$} \mathsf{P}_X$ such that $\mathsf{Pr}\left[\mathcal{E} \,\middle|\, X = x\right] = \mathcal{M}(x)$; then in particular $\mathsf{Pr}\left[\mathcal{E}\right] = \mu(\mathcal{M})$, and $\mathsf{Pr}\left[X = x \,\middle|\, \mathcal{E}\right] = \mathsf{P}_{\mathcal{M}}(x)$.

We ask the question of which bit-guessing games admit a *hardcore measure*: Assuming the game G is ε-hard for some $\varepsilon \in [0,1]$, we seek for a measure $\mathcal{M}$ with large density (e.g. $\mu(\mathcal{M}) \geq 1-\varepsilon$) such that conditioned on the associated event $\mathcal{E}$, the game G is very hard to win. In [25], a proof that this is true for the case where $\mathcal{C}(x)$ is stateless for each x was given. Our new approach extends this to possibly stateful challengers, as in the case of CCA security.

Abstract hardcore lemmas. We give a sufficient condition on a bit-guessing game $G = (\mathsf{P}_X, \mathcal{C}, P)$ to admit a hardcore lemma – informally, this condition corresponds to the ability, for any given and possibly unknown x, to estimate the probability that a binary-output adversary for G, sampled according to a given distribution over circuits, outputs one when run on $\mathcal{C}(x)$. In particular, we call an oracle O a *size s circuit sampler* for G if, upon each invocation, it returns the description of a valid adversary $\mathcal{A}$ for G of size s. For each such O, we define $p_1^{G,\mathsf{O}}(x)$ as the probability that a randomly sampled adversary $\mathcal{A} \xleftarrow{\$} \mathsf{O}$ outputs one when run with $\mathcal{C}(x)$, i.e., $p_1^{G,\mathsf{O}}(x) := \mathsf{Pr}\left[\mathcal{B} \xleftarrow{\$} \mathsf{O}, b' \xleftarrow{\$} \mathcal{B}^{\mathcal{C}(x)} : b' = 1\right]$. The following definition captures the notion of a good estimation algorithm for $p_1^{G,\mathsf{O}}(x)$ which can only interact with $\mathcal{C}(x)$ and obtain samples from O, but does not learn x and must be equally successful on all such x.

Definition 2 (p_1-estimator). *A (s, s', q, γ, η)-p_1-*estimator *for a bit-guessing game $G = (\mathsf{P}_X, \mathcal{C}, P)$ is a size s circuit $\mathcal{E}$ with output in* $[0, 1]$ *such that*

$$\Pr\left[\mathcal{B}_1, \ldots, \mathcal{B}_q \xleftarrow{\$} \mathsf{O},\ \overline{p_1} \xleftarrow{\$} \mathcal{E}^{\mathcal{C}(x)}(\mathcal{B}_1, \ldots, \mathcal{B}_q) :\ \left|\overline{p_1} - p_1^{G,\mathsf{O}}(x)\right| > \gamma\right] < \eta$$

for all size-s' circuit samplers O *and for all x.*

Note that in particular $q \cdot s' \leq s$. The following theorem relates the existence of a hardcore lemma for a certain game G with the existence of a p_1-sampler for G. Its proof abstracts the ones of [16,25] and is found in the full version.

Proposition 1 (The Abstract Hardcore Lemma). *Let $s \in \mathbb{N}$ and $\varepsilon \in [0, 1]$. Let $G = (\mathsf{P}_X, \mathcal{C}, P)$ be a bit-guessing game which is (s, ε)-hard. Then, for all $\gamma > 0$, if for some $s' = s'(\gamma)$ there exists an $(s, s', q, \gamma(1-\varepsilon)/4, \gamma(1-\varepsilon)/4)$-$p_1$-estimator for G, then there exists a measure $\mathcal{M} = \mathcal{M}_\gamma$ such that:*

(i) $\mu(\mathcal{M}) \geq 1 - \varepsilon$, **(ii)** $\mathbf{Adv}^G_{\mathsf{P}_\mathcal{M}}(\mathcal{B}) \leq \gamma$ *for all s'-size $\mathcal{B}$.*

The hardcore lemma for CCA-security. We are now going to show a hardcore lemma for CCA-security as an application of Proposition 1. Let $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a public-key *bit* encryption scheme such that Gen and Enc take randomness of lengths ρ_{Gen} and ρ_{Enc}, respectively. Formally, we consider the bit-guessing game $\mathsf{CCA2}[\mathsf{PKE}] = (\mathsf{P}_X, \mathcal{C}_{\mathsf{CCA2}}, P)$ where P_X is the uniform distribution on $\{0,1\}^{\rho_{\mathsf{Gen}}} \times \{0,1\}^{\rho_{\mathsf{Enc}}} \times \{0,1\}$, whereas $\mathcal{C}_{\mathsf{CCA2}}(r_{\mathsf{Gen}}, r_{\mathsf{Enc}}, b)$ is the challenger for the CCA-security game for PKE with challenge bit b, public key and secret key $(\mathsf{pk}, \mathsf{sk}) = \mathsf{Gen}(r_{\mathsf{Gen}})$, and challenge ciphertext $c^* = \mathsf{Enc}(\mathsf{pk}, b; r_{\mathsf{Enc}})$. Moreover, we define $P(r_{\mathsf{Gen}}, r_{\mathsf{Enc}}, b) = b$. The following lemma gives an appropriate p_1-estimator for $\mathsf{CCA2}[\mathsf{PKE}]$.

Lemma 1. *For all* $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *with message space* $\{0, 1\}$*, and all $s' \in \mathbb{N}$, $\gamma, \eta \in (0, 1]$, there exists a (s, s', q, γ, η)-p_1-*estimator for $\mathsf{CCA2}[\mathsf{PKE}]$ with $q = O(\log(1/\eta)/\gamma^2)$ and $s = s' \cdot q + O(1)$.

Proof. The estimator $\mathcal{E}$, given pk from $\mathcal{C}_{\mathsf{CCA2}}$, runs sequentially each of $\mathcal{B}_1, \ldots, \mathcal{B}_q$ on input pk until they output their query $(0, 1)$. All before-the-fact decryption queries are answered using the challenger $\mathcal{C}_{\mathsf{CCA2}}$. It then obtains a challenge ciphertext c^*, and then resumes the execution of $\mathcal{B}_i$'s from the last state before outputting $(0, 1)$, again using the challenger to reply to decryption queries. Finally, let b'_i be the output of $\mathcal{B}_i$; the estimator $\mathcal{E}$ outputs the average $z = (1/q) \cdot \sum_{i=1}^{q} b'_i$. The error is at most γ with probability at most η by the Chernoff bound. □

The above proof crucially relies on the scheme encrypting one-bit messages: For a larger set of messages, each $\mathcal{B}_i$ could ask a different message pair, and the above estimation technique would fail.

The following theorem is a simple combination of Proposition 1 and Lemma 1.

Theorem 2 (Hardcore Lemma for CCA Security). *Let $\alpha, \beta \in [0, 1]$, and let $s \in \mathbb{N}$. Moreover, let* $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *be a public-key encryption* *scheme with message space* $\{0,1\}$*, and assume that* $\mathbf{Adv}^{\mathsf{CCA2}}_{\mathsf{PKE}}(\mathcal{A}) \leq \beta$ *for all* s*-size adversaries* $\mathcal{A}$*. Then, for all* $\gamma > 0$*, there exists a measure* $\mathcal{M}$ *such that* $\mu(\mathcal{M}) \geq 1 - \beta$*, and* $\mathbf{Adv}^{\mathsf{CCA2[PKE]}}_{\mathsf{P}_{\mathcal{M}}}(\mathcal{B}) \leq \gamma$ *for all adversaries* $\mathcal{B}$ *with size* s'*, where* $s = O(s' \cdot \log(1/\gamma(1-\varepsilon))/\gamma^2(1-\varepsilon)^2)$*.*

In the full version, we provide a more detailed discussion about related results and extensions to the uniform setting, which we here omit due to lack of space.

4 From Weak to Strong CCA Security

We present our construction to transform an (α, β)-CCA encryption scheme into a fully CCA-secure encryption scheme. First, we review some tools underlying our construction, before turning to its description and security.

4.1 Information-Theoretically Secure Key-Agreement

We consider the problem of two parties, Alice and Bob, agreeing on a secret key with *unconditional security* in a setting where they each hold values $X_1, \ldots X_n$ and $Y_1 \ldots, Y_n$, respectively, in presence of an adversary obtaining correlated values $Z_1, \ldots, Z_n$; in particular, (X_i, Y_i, Z_i) are sampled independently from a given tripartite probability distributions P_{XYZ} for all $1 \leq i \leq n$. That is, (X_i, Y_i, Z_i) are correlated for each i, but independent across distinct indices $i \neq j$. Moreover, Alice and Bob are connected via an authenticated channel, allowing them to exchange messages, which is however wiretapped by the adversary. Secret-key agreement in this setting was first considered by Maurer [24]. Here, we consider the special case where the channel only allows *one-way* communication from Alice to Bob. The following definition captures protocols for this setting.

Definition 3 (One-way key-agreement). *Let* $\varepsilon, \delta : \mathbb{N} \to [0,1]$*, and let* $n, \ell : \mathbb{N} \to \mathbb{N}$ *be monotonically increasing. Also, let* $\mathcal{P} = \{\mathcal{P}_\kappa\}_{\kappa \in \mathbb{N}}$ *be a family of sets of probability distribution* P_{XYZ}*. A* $(\mathcal{P}, \varepsilon, \delta, n, \ell)$-one-way key-agreement (OKA) protocol *is a pair of probabilistic polynomial-time algorithms* $\mathsf{OKA} = (\mathsf{KAEnc}, \mathsf{KADec})$ *such that for all* $\kappa \in \mathbb{N}$ *and* $\mathsf{P}_{XYZ} \in \mathcal{P}_\kappa$*, the following two properties hold when sampling* $(X_1, Y_1, Z_1), \ldots, (X_n, Y_n, Z_n) \xleftarrow{\$} \mathsf{P}_{XYZ}$ *(where* $n = n(\kappa)$*),* $(C, K) \xleftarrow{\$} \mathsf{KAEnc}(1^\kappa, X_1, \ldots, X_n)$*,* $K' \xleftarrow{\$} \mathsf{KADec}(1^\kappa, Y_1, \ldots, Y_n; C)$*, and* $K'' \xleftarrow{\$} \{0,1\}^{\ell(\kappa)}$*:* **(1)** $K = K'$ *with probability at least* $1 - \delta(\kappa)$*, and* **(2)** $(C, K, Z_1, \ldots, Z_n)$ *and* $(C, K'', Z_1, \ldots, Z_n)$ *have statistical distance at most* $\varepsilon(\kappa)$*.*

The following set of distributions was introduced in [17].

Definition 4 ([17]). *Let* $\alpha, \beta : \mathbb{N} \to [0,1]$*. Let* $\mathcal{D}(\alpha, \beta) = \{\mathcal{D}_\kappa(\alpha, \beta)\}_{\kappa \in \mathbb{N}}$ *be such that for all* $\kappa \in \mathbb{N}$*,* $\mathsf{P}_{XYZ} \in \mathcal{D}_\kappa(\alpha, \beta)$ *if* $(X, Y, Z) \xleftarrow{\$} \mathsf{P}_{XYZ}$ *satisfies* **(i)** $\mathsf{Pr}[X = 0] = \mathsf{Pr}[X = 1] = \frac{1}{2}$*, i.e.,* X *is uniform,* **(ii)** $\mathsf{Pr}[X = Y] \geq \frac{1+\alpha(\kappa)}{2}$*,* **(iii)** *there exists an event* $\mathcal{E}$*, defined on* (X, Z)*, such that* $\mathsf{Pr}\left[X = 0 \,\middle|\, Z = z, \mathcal{E}\right] = \mathsf{Pr}\left[X = 1 \,\middle|\, Z = z, \mathcal{E}\right] = \frac{1}{2}$ *for all* z*, and* $\mathsf{Pr}[\mathcal{E}] \geq 1 - \beta(\kappa)$*.*

The following two propositions show feasibility of OKA protocols for $\mathcal{D}(\alpha, \beta)$ for certain values of α and β. The first proposition was proved by Holenstein and Renner [17], the second is proved in the full version. We note that there is no a-priori reason why α^2 and β could not be closer, yet no better gap can be proven given existing constructions of capacity-achieving error-correcting codes.

Proposition 2. *Let $\alpha, \beta : \mathbb{N} \to [0, 1]$ be such that $\alpha^2 > \beta + \Omega(1)$, and let $\ell : \mathbb{N} \to \mathbb{N}$ be a polynomial function. Then, there exists a polynomial-time $(\mathcal{D}(\alpha, \beta), \varepsilon, \delta, n, \ell)$-OKA protocol such that $n(\kappa) = \frac{1}{7} \cdot \ell(\kappa) \cdot (\alpha^2 - \beta - O(1))$ and moreover, $\varepsilon(\kappa)$ is negligible in $n(\kappa)$, and $\delta(\kappa) = 2^{-\Theta(n(\kappa))}$.*

Proposition 3. *Let $p, \ell : \mathbb{N} \to \mathbb{N}$ be polynomially bounded and let $\varepsilon' : \mathbb{N} \to [0, 1]$. Then, there exists a $\mathcal{D}(1, 1 - \frac{1}{p(\kappa)}, \varepsilon, \delta, n, \ell)$-OKA protocol where $n(\kappa) = 2/(1 - \beta(\kappa)) \cdot (\ell(\kappa) + 2\log(1/\varepsilon'(\kappa)) + O(1))$, $\varepsilon(\kappa) \leq O(\sqrt{\varepsilon'(\kappa)})$, and $\delta(\kappa) = 0$.*

4.2 The Construction

Let $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a bit-encryption scheme which is (α, β)-secure. Assuming the existence of an information-theoretically secure one-way key agreement protocol for $\mathcal{D}(\alpha, \beta)$, we present a construction of a CCA-secure public-key encryption scheme $\overline{\mathsf{PKE}} = (\overline{\mathsf{Gen}}, \overline{\mathsf{Enc}}, \overline{\mathsf{Dec}})$, with message length $\ell = \ell(\kappa)$ and negligible decryption error, which makes black-box use of the basic scheme PKE.

At the highest level, our construction $\overline{\mathsf{PKE}}$ follows the paradigm recently proposed by Hohenberger, Lewko, and Waters [22]. In particular, it consists of an inner scheme $\mathsf{PKE}_{\mathsf{in}} = (\mathsf{Gen}_{\mathsf{in}}, \mathsf{Enc}_{\mathsf{in}}, \mathsf{Dec}_{\mathsf{in}})$ and two outer schemes $\mathsf{PKE}_{\mathsf{out},1} = (\mathsf{Gen}_{\mathsf{out},1}, \mathsf{Enc}_{\mathsf{out},1}, \mathsf{Dec}_{\mathsf{out},1})$ and $\mathsf{PKE}_{\mathsf{out},2} = (\mathsf{Gen}_{\mathsf{out},2}, \mathsf{Enc}_{\mathsf{out},2}, \mathsf{Dec}_{\mathsf{out},2})$, all three of which will be built from PKE, and specified below. For $\star \in \{\mathsf{in}, (\mathsf{out}, 1), (\mathsf{out}, 2)\}$, let us further denote by $\ell_\star, \rho_\star$ and $t_\star$ the message, randomness, and ciphertext lengths of $\mathsf{PKE}_\star$, respectively. We are going to require $\ell_{\mathsf{in}} = \ell + \rho_{\mathsf{out},1} + \rho_{\mathsf{out},2}$ as well as $\ell_{\mathsf{out},1} = \ell_{\mathsf{out},2} = t_{\mathsf{in}}$. A formal description of $\overline{\mathsf{PKE}}$ is given in Figure 1, on top: We encrypt the message m, together with two random values $r_{\mathsf{out},1}$ and $r_{\mathsf{out},2}$, obtaining an *inner* ciphertext c_{in}, which is then encrypted twice with the two outer schemes, using $r_{\mathsf{out},1}$ and $r_{\mathsf{out},2}$ as the respective random coins. Decryption recovers the message by decrypting the ciphertext via $\mathsf{Dec}_{\mathsf{out},1}$ and $\mathsf{Dec}_{\mathsf{in}}$ using the corresponding secret keys, and then checks validity of the ciphertext by re-encrypting the inner ciphertext using the public keys and the recovered random coins.

We now turn to describing the construction of the component schemes $\mathsf{PKE}_{\mathsf{in}}$, $\mathsf{PKE}_{\mathsf{out},1}$ and $\mathsf{PKE}_{\mathsf{out},2}$ from the basic scheme PKE.

The Inner Scheme. Let $\mathsf{OKA} = (\mathsf{KAEnc}, \mathsf{KADec})$ be a $(\mathcal{D}(\alpha, \beta), \varepsilon, \delta, n, \ell_{\mathsf{in}})$-one-way key agreement protocol such that ε and δ are negligible, and known (recall that PKE is (α, β)-CCA secure). We define $\mathsf{PKE}_{\mathsf{in}} = (\mathsf{Gen}_{\mathsf{in}}, \mathsf{Enc}_{\mathsf{in}}, \mathsf{Dec}_{\mathsf{in}})$ as in Figure 1, at the bottom: It encrypts random bits $b_1, \ldots, b_n$ with the basic scheme, and then generates a session key k via $\mathsf{KAEnc}(b_1, \ldots, b_n)$, and a ciphertext c', and uses the key k as an one-time pad. Decryption via KADec is then obvious. It is easy to see that the decryption error of this scheme is inherited from OKA, i.e., it is upper bounded by exactly δ.

Scheme $\overline{\mathsf{PKE}} = (\overline{\mathsf{Gen}}, \overline{\mathsf{Enc}}, \overline{\mathsf{Dec}})$:

Key generation $\overline{\mathsf{Gen}}(1^\kappa)$: Sample $(\mathsf{pk}_{\mathsf{in}}, \mathsf{sk}_{\mathsf{in}}) \overset{\$}{\leftarrow} \mathsf{Gen}_{\mathsf{in}}(1^\kappa)$ and for $i = 1, 2$, $(\mathsf{pk}_{\mathsf{out},i}, \mathsf{sk}_{\mathsf{out},i}) \overset{\$}{\leftarrow} \mathsf{Gen}_{\mathsf{out},i}(1^\kappa)$. Return $(\overline{\mathsf{pk}} = (\mathsf{pk}_{\mathsf{in}}, \mathsf{pk}_{\mathsf{out},1}, \mathsf{pk}_{\mathsf{out},2}), \overline{\mathsf{sk}} = (\mathsf{sk}_{\mathsf{in}}, \mathsf{sk}_{\mathsf{out},1}, \mathsf{pk}_{\mathsf{out},1}, \mathsf{pk}_{\mathsf{out},2}))$.

Encryption $\overline{\mathsf{Enc}}(\overline{\mathsf{pk}}, m)$, $m \in \{0,1\}^\ell$: Sample $r_{\mathsf{out},i} \overset{\$}{\leftarrow} \{0,1\}^{\rho_{\mathsf{out},i}}$ for $i = 1, 2$. Generate $c_{\mathsf{in}} \overset{\$}{\leftarrow} \mathsf{Enc}_{\mathsf{in}}(\mathsf{pk}_{\mathsf{in}}, m \,\|\, r_{\mathsf{out},1} \,\|\, r_{\mathsf{out},2})$ and $c_{\mathsf{out},i} \leftarrow \mathsf{Enc}_{\mathsf{out},i}(\mathsf{pk}_{\mathsf{out},i}, c_{\mathsf{in}}; r_{\mathsf{out},i})$ for $i = 1, 2$. Output ciphertext $c_{\mathsf{out},1} \,\|\, c_{\mathsf{out}_2}$.

Decryption $\overline{\mathsf{Dec}}(\overline{\mathsf{sk}}, c = c_{\mathsf{out},1} \,\|\, c_{\mathsf{out}_2})$: Decrypt $c'_{\mathsf{in}} \leftarrow \mathsf{Dec}_{\mathsf{out},1}(\mathsf{sk}_{\mathsf{out},1}, c_{\mathsf{out},1})$ and $m' \,\|\, r'_{\mathsf{out},1} \,\|\, r'_{\mathsf{out},2} \leftarrow \mathsf{Dec}_{\mathsf{in}}(\mathsf{sk}_{\mathsf{in}}, c'_{\mathsf{in}})$. If $\mathsf{Enc}_{\mathsf{out},i}(\mathsf{pk}_{\mathsf{out},i}, c'_{\mathsf{in}}; r'_{\mathsf{out},i}) = c_{\mathsf{out},i}$ for $i = 1, 2$ then return m, else return $\perp$.

Scheme $\mathsf{PKE}_{\mathsf{in}} = (\mathsf{Gen}_{\mathsf{in}}, \mathsf{Enc}_{\mathsf{in}}, \mathsf{Dec}_{\mathsf{in}})$:

Key generation $\mathsf{Gen}_{\mathsf{in}}(1^\kappa)$: Sample $(\mathsf{pk}_1, \mathsf{sk}_1), \ldots, (\mathsf{pk}_n, \mathsf{sk}_n) \overset{\$}{\leftarrow} \mathsf{Gen}(1^\kappa)$. Return $(\mathbf{pk} = (\mathsf{pk}_1, \ldots, \mathsf{pk}_n), \mathbf{sk} = (\mathsf{sk}_1, \ldots, \mathsf{sk}_n))$.

Encryption $\mathsf{Enc}_{\mathsf{in}}(\mathbf{pk}, m)$, $m \in \{0,1\}^{\ell_{\mathsf{in}}}$: For all $i \in [n]$, sample $b_i \overset{\$}{\leftarrow} \{0,1\}$ and generate $c_i \overset{\$}{\leftarrow} \mathsf{Enc}(\mathbf{pk}[i], b_i)$. Compute $(k, c') \overset{\$}{\leftarrow} \mathsf{KAEnc}(b_1, \ldots, b_n)$. Return ciphertext $(c_1, \ldots, c_n, c', m \oplus k)$.

Decryption $\mathsf{Dec}_{\mathsf{in}}(\mathbf{sk}, c = (c_1, \ldots, c_n, c', c''))$: Decrypt $b'_i \leftarrow \mathsf{Dec}(\mathbf{sk}[i], c_i)$ for $i = [n]$ and $k' \leftarrow \mathsf{KADec}(b'_1, \ldots, b'_n; c')$. Return plaintext $m' = c'' \oplus k'$.

Fig. 1. Descriptions of public-key encryption schemes $\overline{\mathsf{PKE}}$ and $\mathsf{PKE}_{\mathsf{in}}$

THE OUTER SCHEMES. We now instantiate the two outer schemes. The following description is fairly high-level, but sufficient to fully specify the construction. We refer the reader unfamiliar with the basic components to the full version for a more detailed description.

We first derive a CPA-secure public-key encryption scheme $\mathsf{PKE}^{\ell,\rho}_{\mathsf{out}}$ with message length $\ell = \mathsf{poly}(\kappa)$ and randomness length $\rho = \omega(\log(\kappa))$ from the basic scheme PKE which also enjoys almost-perfect correctness:[4]

1. We use the same construction as in $\mathsf{PKE}_{\mathsf{in}}$ to achieve a CPA-secure scheme $\mathsf{PKE}'_{\mathsf{out}}$, with message length truncated to 1-bit. CPA-security of the resulting scheme follows from the proof in [17] or from the stronger Lemma 2 below. Let ρ be the randomness length of $\mathsf{PKE}'_{\mathsf{out}}$.
2. We apply the transformation by Dwork, Naor, and Reingold [15] to enhance correctness of $\mathsf{PKE}'_{\mathsf{out}}$ with negligible decryption error to almost-perfect correctness, via sparsification of the randomness space. Let δ be the decryption error of $\mathsf{PKE}'_{\mathsf{out}}$. The transformation of [15] reduces randomness length to $\rho' = \frac{1}{4} \cdot \log(1/\delta(\kappa)) = \omega(\log(\kappa))$ via a PRG $G : \{0,1\}^{\rho'} \to \{0,1\}^\rho$, whose existence is implied by the existence of $\mathsf{PKE}'_{\mathsf{out}}$ in a black-box fashion [26,27].

[4] In the following, we are not going to optimize the complexity of the scheme; it is clear that some modifications can be done to save on complexity.

3. We then use parallel repetition of ℓ copies of $\mathsf{PKE}''_{\mathsf{out}}$ to obtain $\mathsf{PKE}^{\ell,\rho}_{\mathsf{out}}$, possibly using a PRG again to shorten the overall randomness length to ρ.

We let $\mathsf{PKE}_{\mathsf{out},2} = \mathsf{PKE}^{\ell_{\mathsf{out},2},\rho_{\mathsf{out},2}}_{\mathsf{out}}$. To obtain the first outer scheme $\mathsf{PKE}_{\mathsf{out},1}$, we rely on the result by Cramer *et al* [28] that transforms a CPA-secure encryption scheme into a 1-CCA secure one in a black-box way, which preserves the almost perfect correctness property of the underlying CPA-secure scheme. By applying their transformation to $\mathsf{PKE}^{\ell_{\mathsf{out},1},\rho}_{\mathsf{out}}$ (for some $\rho = \mathsf{poly}(\kappa)$), and then finally using a PRG to reduce the randomness length to $\rho_{\mathsf{out},1}$, we obtain a 1-CCA secure encryption scheme that is almost-perfectly correct.

4.3 CCA Security of $\overline{\mathsf{PKE}}$

We turn to our main result and show that our construction $\overline{\mathsf{PKE}}$ is CCA secure.

Theorem 3. *Let ε and δ be two negligible functions. Assume that* PKE *is (α, β)-CCA-secure, and* OKA *is a $(\mathcal{D}(\alpha, \beta), \varepsilon, \delta, n, \ell_{\mathsf{in}})$-one-way key-agreement protocol. Then,* $\overline{\mathsf{PKE}}$ *is a CCA-secure encryption scheme with negligible decryption error.*

In particular, by Propositions 2 and 3, we achieve amplification whenever $\alpha^2 > \beta + \Omega(1)$, and whenever $\alpha = 1$ and $\beta < 1 - \frac{1}{p(\kappa)}$ for some polynomial p.

Overview of the Security Proof. Towards showing the CCA security of $\overline{\mathsf{PKE}}$, we first show that it follows from Theorem 2 that the inner encryption scheme $\mathsf{PKE}_{\mathsf{in}}$ satisfies a strong adaptive security property, which we refer to as XCCA (to be read as "cross"-CCA) security. We are then going to reduce the CCA security of $\overline{\mathsf{PKE}}$ to the XCCA security of $\mathsf{PKE}_{\mathsf{in}}$ using the 1-CCA security of $\mathsf{PKE}_{\mathsf{out},1}$ and the CPA security of $\mathsf{PKE}_{\mathsf{out},2}$, combined with their almost perfect correctness. This second step resembles the proof of [22] only at a first glance, as it will require a completely different technique to handle the fact that ciphertexts of the basic scheme PKE are not sufficiently unpredictable.

Before proceeding to describing the two steps in more details, we first describe the XCCA security game. For simplicity, here we only define the XCCA game w.r.t. the concrete scheme $\mathsf{PKE}_{\mathsf{in}}$; one can easily generalize the definition to a larger class of encryption schemes whose ciphertext contains multiple component ciphertexts of a base encryption scheme, similarly to [21]; we omit the details here. The game proceeds almost identically to the CCA game except that instead of having access to the decryption oracle for the whole encryption scheme, the adversary has access to the decryption oracles of the basic encryption scheme PKE using each of the component secret keys; the i'th decryption oracle using the i'th component secret key is denoted as $\mathsf{Dec}(\mathbf{sk}[i], \cdot)$. As a result, the adversary cannot make any after-the-fact decryption queries that is the same as any of the component ciphertexts encrypted using one of the component public keys $\mathbf{pk}[i]$ in the challenge ciphertext. Similar to the CCA game, we define the XCCA-*advantage* of the adversary $\mathcal{A}$ as $\mathbf{Adv}^{\mathsf{XCCA}}_{\mathsf{PKE}_{\mathsf{in}}}(\mathcal{A}) = 2 \cdot \Pr[b' = b] - 1$.

We say that $\mathsf{PKE}_{\mathsf{in}}$ is XCCA-secure if no polynomial sized adversary can achieve a non-negligible advantage in the XCCA game.

We remark that the XCCA game is closely related to the notion of UCCA security defined in [21], and the similar notion of DCCA security in [22]: In comparison, in the UCCA security game w.r.t. $\mathsf{PKE}_{\mathsf{in}}$, the adversary only has access to the decryption oracle of the *whole encryption scheme*, but is not allowed to make any after-the-fact query that quotes any of the component ciphertexts in the challenge ciphertext (in DCCA a more fine grained control on disallowed queries is considered). As we will see shortly, the stronger security guarantee given by XCCA is crucial for our proof to succeed.

With the definition of the XCCA game in mind, the remainder of the proof proceeds via the following two lemmas, for which we give a proof sketch. (A formal proof is given in the full version.)

Lemma 2. *Let ε and δ be two negligible functions. Assume that* PKE *is (α, β)-secure, and* OKA *is a $(\mathcal{D}(\alpha, \beta), \varepsilon, \delta, n, \ell_{\mathsf{in}})$-one-way KA protocol. Then,* $\mathsf{PKE}_{\mathsf{in}}$ *is XCCA-secure.*

Lemma 3. *Assume that* $\mathsf{PKE}_{\mathsf{in}}$*,* $\mathsf{PKE}_{\mathsf{out},1}$ *and* $\mathsf{PKE}_{\mathsf{out},2}$ *are respectively XCCA, 1-CCA and CPA secure, and* $\mathsf{PKE}_{\mathsf{out},1}$ *and* $\mathsf{PKE}_{\mathsf{out},2}$ *have almost-perfect correctness, then* $\overline{\mathsf{PKE}}$ *is CCA secure.*

Proof Sketch of Lemma 2: We are going to use the hardcore lemma for CCA-security (Theorem 2) to show that $\mathsf{PKE}_{\mathsf{in}}$ is XCCA secure. Informally speaking, in the XCCA game, with respect to each random bit b_i used to generate the component c_i of the challenge ciphertext, the adversary is participating in an independently and randomly executed CCA game for PKE: Indeed, each random bit b_i is encrypted using an independently and randomly chosen public key $\mathbf{pk}[i]$ and random coins, and the adversary has access to the decryption oracle $\mathsf{Dec}(\mathbf{sk}[i], \cdot)$. Thus, by the hardcore lemma, each of these CCA games has probability $1 - \beta$ of delivering an "hard instance", and thus the corresponding bit b_i remains hidden to the adversary, i.e., it looks (pseudo-)random with probability $1 - \beta$. More precisely, each triple $(b_i, \mathsf{Dec}(\mathbf{sk}[i], c_i), c_i)$, with $c_i \overset{\$}{\leftarrow} \mathsf{Enc}(\mathbf{pk}[i], b_i)$ is computationally indistinguishability from a sample from a valid distribution from $\mathcal{D}(\alpha, \beta)$. In this case, then it simply follows from the fact that OKA is a $(\mathcal{D}(\alpha, \beta), \varepsilon, \delta, n, \ell_{\mathsf{in}})$-one-way key agreement scheme that the key k output by $\mathsf{KAEnc}(b_1, \cdots, b_n)$ remains random and thus the message m_b is hidden.

Proof Sketch of Lemma 3: We base the CCA security of $\overline{\mathsf{PKE}}$ on the XCCA security of $\mathsf{PKE}_{\mathsf{in}}$ via a black-box reduction. The reduction $\mathcal{B}$ participates in the XCCA game for $\mathsf{PKE}_{\mathsf{in}}$ and internally emulates the CCA game for $\overline{\mathsf{PKE}}$ to a CCA-adversary $\mathcal{A}$ succeeding with non-negligible advantage γ as follows:

- It receives the public key $\mathbf{pk}$ in the XCCA game and internally generates the public key $\overline{\mathsf{pk}}$ by sampling key pairs $(\mathsf{pk}_{\mathsf{out},1}, \mathsf{sk}_{\mathsf{out},1})$ and $(\mathsf{pk}_{\mathsf{out},2}, \mathsf{sk}_{\mathsf{out},2})$ for the two outer schemes and gives $\overline{pk} = (\mathbf{pk}, \mathsf{pk}_{\mathsf{out},1}, \mathsf{pk}_{\mathsf{out},2})$ to $\mathcal{A}$.

- To emulate the challenge ciphertext c^* of $\overline{\mathsf{PKE}}$ that encrypts either m_0 or m_1 chosen by $\mathcal{A}$ in the emulated CCA game, $\mathcal{B}$ first chooses random $r_{\mathsf{out},1}$ and $r_{\mathsf{out},2}$, and obtains the challenge ciphertext c^*_{in} of $\mathsf{PKE}_{\mathsf{in}}$ that encrypts $m_b \| r_{\mathsf{out},1} \| r_{\mathsf{out},2}$ for a random $b \in \{0,1\}$ chosen in the XCCA game. It then produces c^* honestly by encrypting $c^*_{\mathsf{out},1} = \mathsf{Enc}_{\mathsf{out},1}(c^*_{\mathsf{in}}; r_{\mathsf{out},1})$ and $c^*_{\mathsf{out},2} = \mathsf{Enc}_{\mathsf{out},2}(c^*_{\mathsf{in}}; r_{\mathsf{out},2})$.
- Finally, it emulates the decryption oracle $\overline{\mathsf{Dec}}(\overline{\mathsf{sk}}, \cdot)$ for $\mathcal{A}$ by using the secret key $\mathsf{sk}_{\mathsf{out},1}$ and the decryption oracles $\{\mathsf{Dec}(\mathsf{sk}[i], \cdot)\}_{i \in [n]}$ in the XCCA game.

It is easy to see that as long as $\mathcal{A}$ does not ask any after-the-fact queries whose inner ciphertext (embedded in the first outer ciphertext) "quotes" the inner challenge ciphertexts c^*_{in}, i.e., it does not share a common component ciphertext, $\mathcal{B}$ always decrypts queries from $\mathcal{A}$ perfectly and consequently also emulates the view of $\mathcal{A}$ perfectly.

It is therefore tempting to try to show that the probability that $\mathcal{A}$ "quotes" is negligible. Indeed, this is the approach taken by [21,22]. The rationale in their proof is that if the basic scheme PKE has unpredictability — a random ciphertext of a random bit has high entropy and is hard to blindly guess — then the fact that $\mathcal{A}$ manages to quote would violate the 1-CCA security of the first outer scheme or the CPA-security of the second outer scheme. In [22], a series of hybrids is used to remove the circular dependence between the inner challenge ciphertext and the randomness used in its two outer encryptions, and move to a setting where $\mathcal{A}$'s view is *statistically* independent from the inner challenge ciphertext, but the quoting probability is negligibly close to the original one. One can then easily show that unpredictability of PKE yields that quoting occurs with negligible probability only.

Unfortunately, this approach fails completely in our setting, as our basic encryption scheme PKE does not ensure unpredictability; in fact, it is possible to build an (α, β)-CCA-secure scheme where ciphertexts have very low min-entropy. We address this via a new technique, called *heavy ciphertext pre-sampling*: We observe that if $\mathcal{A}$ can blindly guess some component ciphertext c_i in c^*_{in}, then c_i is a ciphertext value which appears with sufficiently large probability when encrypting a random bit under $\mathbf{pk}[i]$. Hence, we can hope that the same value is hit by the reduction $\mathcal{B}$ by simply generating a large number of random encryptions (of random bits) of PKE under $\mathbf{pk}[i]$; call these pre-sampled ciphertexts. Since the component ciphertexts in c^*_{in} are generated identically to the pre-sampled ciphertexts, the probability that $\mathcal{A}$'s guess collides with the former is the same as the probability it collides with any of the pre-sampled ciphertexts. Setting the size of the pre-sampling large enough, say $\mathsf{poly}(1/\varepsilon)$, the reduction can exhaust all the component ciphertexts that $\mathcal{A}$ may "quote" with probability $1-\varepsilon$, for any ε. Furthermore, due to the strong security provided by the XCCA game, the reduction $\mathcal{B}$, with access to the decryption oracles of the component ciphertexts, can obtain the decrypted values of these pre-sampled ciphertexts before-the-fact. This is crucial, since even if *we know* that a ciphertext is obtained by encrypting some bit d, its actual decryption could well be equal $1-d$ due to the weak α-correctness.

Intuitively this solves the problem, as whenever $\mathcal{A}$ makes an after-the-fact query that "quotes" c^*_{in}, $\mathcal{B}$ can still decrypt by using either the external decryption oracles (for components that do not quote) *or* the decrypted values of the pre-sampled ciphertexts (for these that quote). This will allow us to show that $\mathcal{B}$ succeeds in emulating the view of $\mathcal{A}$ with high probability, and thus the CCA security of $\overline{\mathsf{PKE}}$ reduces to the XCCA security of $\mathsf{PKE}_{\mathsf{in}}$.

Acknowledgments. We wish to thank Russell Impagliazzo and Thomas Ristenpart for insightful discussions at early stages of this work.

This work was partially supported by NSF grant CCF-1018064. This material is based on research sponsored by DARPA under agreement number FA8750-11-2-0225. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA or the U.S. Government.

References

1. Goldwasser, S., Micali, S.: Probabilistic encryption. Journal of Computer and System Sciences 28(2), 270–299 (1984)
2. Rackoff, C., Simon, D.R.: Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 433–444. Springer, Heidelberg (1992)
3. Naor, M., Yung, M.: Public-key cryptosystems provably secure against chosen ciphertext attacks. In: 22nd ACM STOC. ACM Press (May 1990)
4. Dolev, D., Dwork, C., Naor, M.: Non-malleable cryptography. In: 23rd ACM STOC, pp. 542–552. ACM Press (May 1991)
5. Sahai, A.: Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In: 40th FOCS, pp. 543–553. IEEE Computer Society Press (October 1999)
6. Gertner, Y., Malkin, T., Myers, S.: Towards a separation of semantic and CCA security for public key encryption. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 434–455. Springer, Heidelberg (2007)
7. Cramer, R., Shoup, V.: Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 45–64. Springer, Heidelberg (2002)
8. Wee, H.: Efficient chosen-ciphertext security via extractable hash proofs. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 314–332. Springer, Heidelberg (2010)
9. Peikert, C., Waters, B.: Lossy trapdoor functions and their applications. In: Ladner, R.E., Dwork, C. (eds.) 40th ACM STOC, pp. 187–196. ACM Press (May 2008)
10. Rosen, A., Segev, G.: Chosen-ciphertext security via correlated products. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 419–436. Springer, Heidelberg (2009)
11. Kiltz, E., Mohassel, P., O'Neill, A.: Adaptive trapdoor functions and chosen-ciphertext security. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 673–692. Springer, Heidelberg (2010)

12. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Ashby, V. (ed.) ACM CCS 1993, pp. 62–73. ACM Press (November 1993)
13. Bellare, M., Rogaway, P.: Optimal asymmetric encryption. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 92–111. Springer, Heidelberg (1995)
14. Yao, A.C.: Theory and applications of trapdoor functions. In: 23rd FOCS, pp. 80–91. IEEE Computer Society Press (November 1982)
15. Dwork, C., Naor, M., Reingold, O.: Immunizing encryption schemes from decryption errors. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 342–360. Springer, Heidelberg (2004)
16. Holenstein, T.: Key agreement from weak bit agreement. In: Gabow, H.N., Fagin, R. (eds.) 37th ACM STOC, pp. 664–673. ACM Press (May 2005)
17. Holenstein, T., Renner, R.S.: One-way secret-key agreement and applications to circuit polarization and immunization of public-key encryption. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 478–493. Springer, Heidelberg (2005)
18. Lenstra, A.K., Hughes, J.P., Augier, M., Bos, J.W., Kleinjung, T., Wachter, C.: Public keys. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 626–642. Springer, Heidelberg (2012)
19. Heninger, N., Durumeric, Z., Wustrow, E., Halderman, J.A.: Mining your ps and qs: Detection of widespread weak keys in network devices. In: Proceedings of the 21st USENIX Security Symposium (2012)
20. Kearns, M.J., Valiant, L.G.: Cryptographic limitations on learning boolean formulae and finite automata. J. ACM 41(1), 67–95 (1994)
21. Myers, S., Shelat, A.: Bit encryption is complete. In: 50th FOCS, pp. 607–616. IEEE Computer Society Press (October 2009)
22. Hohenberger, S., Lewko, A., Waters, B.: Detecting dangerous queries: A new approach for chosen ciphertext security. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 663–681. Springer, Heidelberg (2012)
23. Impagliazzo, R.: Hard-core distributions for somewhat hard problems. In: FOCS 1995, pp. 538–545 (1995)
24. Maurer, U.M.: Protocols for secret key agreement by public discussion based on common information. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 461–470. Springer, Heidelberg (1993)
25. Tessaro, S.: Security amplification for the cascade of arbitrarily weak pRPs: Tight bounds via the interactive hardcore lemma. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 37–54. Springer, Heidelberg (2011)
26. Impagliazzo, R., Luby, M.: One-way functions are essential for complexity-based cryptography. In: 30th FOCS, pp. 230–235. IEEE Computer Society Press (October / November 1989)
27. Håstad, J., Impagliazzo, R., Levin, L.A., Luby, M.: A pseudorandom generator from any one-way function. SIAM Journal on Computing 28(4), 1364–1396 (1999)
28. Cramer, R., Hanaoka, G., Hofheinz, D., Imai, H., Kiltz, E., Pass, R., Shelat, A., Vaikuntanathan, V.: Bounded CCA2-secure encryption. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 502–518. Springer, Heidelberg (2007)

Circular Chosen-Ciphertext Security with Compact Ciphertexts

Dennis Hofheinz*

Karlsruhe Institute of Technology

Abstract. A key-dependent message (KDM) secure encryption scheme is secure even if an adversary obtains encryptions of messages that depend on the secret key. Such key-dependent encryptions naturally occur in scenarios such as harddisk encryption, formal cryptography, or in specific protocols. However, there are not many provably secure constructions of KDM-secure encryption schemes. Moreover, only one construction, due to Camenisch, Chandran, and Shoup (Eurocrypt 2009) is known to be secure against active (i.e., CCA) attacks.

In this work, we construct the first public-key encryption scheme that is KDM-secure against active adversaries and has compact ciphertexts. As usual, we allow only circular key dependencies, meaning that encryptions of arbitrary *entire* secret keys under arbitrary public keys are considered in a multi-user setting.

Technically, we follow the approach of Boneh, Halevi, Hamburg, and Ostrovsky (Crypto 2008) to KDM security, which however only achieves security against passive adversaries. We explain an inherent problem in adapting their techniques to active security, and resolve this problem using a new technical tool called "lossy algebraic filters" (LAFs). We stress that we significantly deviate from the approach of Camenisch, Chandran, and Shoup to obtain KDM security against active adversaries. This allows us to develop a scheme with compact ciphertexts that consist only of a constant number of group elements.

Keywords: key-dependent messages, chosen-ciphertext security, public-key encryption.

1 Introduction

KDM Security. An encryption scheme is key-dependent message (KDM) secure if it is secure even against an adversary who has access to encryptions of messages that depend on the secret key. Such a setting arises, e.g., in harddisk encryption [10], computational soundness results in formal methods [7, 2], or specific protocols [13]. KDM security does not follow from standard security [1, 15], and there are indications [19, 5] that KDM security (at least in its most

* Supported by DFG grant GZ HO 4534/2-1.

general form) cannot be proven using standard techniques; it seems that dedicated constructions and proof techniques are necessary.[1]

The BHHO Approach to KDM-CPA Security. Boneh, Halevi, Hamburg, and Ostrovsky [10] (henceforth BHHO) were the first to construct and prove a public-key encryption (PKE) scheme that is KDM secure under chosen-plaintext attacks (KDM-CPA-secure) in the standard model, under the Decisional Diffie-Hellman (DDH) assumption. While they did not prove their scheme secure under messages that *arbitrarily* depend on the secret key, their result encompasses the important case of *circular (CIRC-CPA) security.* Loosely speaking, a PKE scheme is circular secure if it is secure even in a multi-user setting where encryptions of arbitrary secret keys under arbitrary public keys are known. This notion is sufficient for certain applications [13], and can often be extended to stronger forms of KDM security [5, 12]. Inspired by BHHO, KDM-CPA-secure PKE schemes from other computational assumptions followed [4, 11, 25].

Since we will be using a similar approach, we give a high-level intuition of BHHO's approach. The crucial property of their scheme is that it is *publicly* possible to construct encryptions of the secret key (under the corresponding public key). Thus, encryptions of the secret key itself do not harm the (IND-CPA) security of that scheme. Suitable homomorphic properties of both keys and ciphertexts allow to extend this argument to circular security (for arbitrarily many users/keys), and to affine functions of all keys.

Why the BHHO Approach Fails to Achieve KDM-CCA Security. When considering an *active* adversary, we require a stronger form of KDM security. Namely, KDM-CCA, resp. CIRC-CCA security requires security against an adversary who has access to key-dependent encryptions *and* a decryption oracle. (Naturally, to avoid a trivial notion, the adversary is not allow to submit any of those given KDM encryptions to its decryption oracle.) Now if we want to extend BHHO's KDM-CPA approach to an adversary with a decryption oracle, the following problem arises: since it is publicly possible to construct (fresh) encryptions of the secret key, an adversary can generate such an encryption and then submit it to its decryption oracle, thus obtaining the full secret key. Hence, the very property that BHHO use to prove KDM-CPA security seemingly contradicts chosen-ciphertext security.

Our Technical Tool: Lossy Algebraic Filters (LAFs). Before we describe our approach to KDM-CCA security, let us present the core technical tool we use. Namely, a *lossy algebraic filter* (LAF) is a family of functions, indexed by a public key and a tag. A function from that family takes a vector $X = (X_i)_{i=1}^{\mathtt{n}}$ as input. Now if the tag is *lossy*, then the output of the function reveals only a linear combination of the X_i. If the tag is *injective*, however, then so is the function. We require that there are many lossy tags, which however require a special trapdoor to be found. On the other hand, lossy and injective tags are computationally

[1] We mention, however, that there are semi-generic transformations that *enhance* the KDM security of an already "slightly" KDM-secure scheme [5, 12, 3].

indistinguishable. This concept is very similar to (parameterized) lossy trapdoor functions [27], and in particular to all-but-many lossy trapdoor functions (ABM-LTFs [20]). However, we do not require efficient inversion, but we do require that lossy functions always reveal *the same* linear combination about the input. In particular, evaluating the same input under many lossy tags will still leave the input (partially) undetermined.

We give a construction of LAFs under the Decision Linear (DLIN) assumption in pairing-friendly groups. Similar to ABM-LTFs, lossy tags correspond to suitably blinded signatures. (This in particular allows to release many lossy tags, while still making the generation of a fresh lossy tag hard for an adversary.) However, unlike with ABM-LTFs, functions with lossy tags always release the same information about its input. Our construction has compact tags with $\mathbf{O}(1)$ group elements, which will be crucial for our KDM-CCA secure encryption scheme.[2]

Our Approach to KDM-CCA Security. We can now describe our solution to the KDM-CCA dilemma explained above. We will start from a hybrid between the BHHO-like PKE schemes of Brakerski and Goldwasser [11], resp. Malkin et al. [25]. This scheme has compact ciphertexts ($\mathbf{O}(1)$ group elements), and its KDM-CPA security can be proved under the Decisional Composite Residuosity (DCR) assumption. As with the BHHO scheme, the scheme's KDM-CPA security relies on the fact that encryptions of its secret key can be publicly generated. Essentially, our modification consists of adding a suitable authentication tag to each ciphertext. This authentication tag comprises the (encrypted) image of the plaintext message under an LAF. During decryption, a ciphertext is rejected in case of a wrong authentication tag.

In our security proof, all authentication tags for the key-dependent encryptions the adversary gets are made with respect to lossy filter tags. This means that information-theoretically, little information about the secret key is released (even with many key-dependent encryptions, resp. LAF evaluations). However, any decryption query the adversary makes must refer (by the LAF properties) to an injective tag. Hence, in order to place a valid key-dependent decryption query, the adversary would have to guess the whole (hidden) secret key.[3]

Thus, adding a suitable authentication tag allows us to leverage the techniques by BHHO, resp. Brakerski and Goldwasser, Malkin et al. to chosen-ciphertext attacks. In particular, we obtain a CIRC-CCA-secure PKE scheme with compact ciphertexts (of $\mathbf{O}(1)$ group elements). We prove security under the conjunction of the following assumptions: the DCR assumption (in $\mathbb{Z}_{N^3}^*$), the DLIN assumption

[2] The size of the LAF public key depends on the employed signature scheme. In our main construction, we use Waters signatures, which results in very compact tags, but public keys of $\mathbf{O}(k)$ group elements, where k is the security parameter. Alternatively, at the end of Section 3.1, we sketch an LAF with constant-size (but larger than in our main construction) tags *and* constant-size public keys.

[3] We will also have to protect against a re-use of (lossy) authentication tags, and "ordinary", key-independent chosen-ciphertext attacks. This will be achieved by a combination of one-time signatures and 2-universal hash proof systems [16, 24, 22].

(in a pairing-friendly group), and the DDH assumption (somewhat curiously, in the subgroup of order $(P-1)(Q-1)/4$ of $\mathbb{Z}_{N^3}^*$, where $N = PQ$).[4]

Relation to Camenisch et al.'s CIRC-CCA-secure Scheme. Camenisch, Chandran, and Shoup [14] present the only other known CIRC-CCA-secure PKE scheme in the standard model. They also build upon BHHO techniques, but instead use a Naor-Yung-style double encryption technique [26] to achieve chosen-ciphertext security. As an authentication tag, they attach to each ciphertext a non-interactive zero-knowledge proof that *either* the encryption is consistent (in the usual Naor-Yung sense), *or* that they know a signature for the ciphertext. Since they build on the original, DDH-based BHHO scheme, they can use Groth-Sahai proofs [18] to prove consistency. Compared to our scheme, their system is less efficient: they require $\mathbf{O}(k)$ group elements per ciphertext, and the secret key can only be encrypted bitwise. However, their sole computational assumption to prove circular security is the DDH (or, more generally, k-Linear) assumption in pairing-friendly groups. One thing to point out is their implicit use of a signature scheme. Their argument is conceptually not unlike our LAF argument. However, since they can apply a hybrid argument to substitute all key-dependent encryptions with random ciphertexts, they only require one-time signatures. Furthermore, the meaning of "consistent ciphertext" and "proof" in our case is very different. (Unlike Camenisch et al., we apply an argument that rests on the *information* that the adversary has about the secret key.)

Note about Concurrent Work. In a work concurrent to ours, Galindo, Herranz, and Villar [17] define and instantiate a strong notion of KDM security for identity-based encryption (IBE) schemes. Using the IBE$\to$PKE transformation of Boneh, Canetti, Halevi, and Katz [9], they derive a KDM-CCA-secure PKE scheme. Their concrete construction is entropy-based and achieves only a bounded form of KDM security, much like the KDM-secure SKE scheme from [23]. Thus, while their ciphertexts are very compact, they can only tolerate a number of (arbitrary) KDM queries that is linear in the size of the secret key. In particular, it is not clear how to argue that the encryption of a full secret key in their scheme is secure.

2 Preliminaries

Notation. For $n \in \mathbb{N}$, let $[n] := \{1, \ldots, n\}$. Throughout the paper, $k \in \mathbb{N}$ is the security parameter. For a finite set $\mathcal{S}$, $s \leftarrow \mathcal{S}$ denotes the process of sampling s uniformly from $\mathcal{S}$. For a probabilistic algorithm A, $y \leftarrow A(x; R)$ denotes the process of running A on input x and with randomness R, and assigning y the

[4] Very roughly, we resort to the DDH assumption since we release *partial* information about our secret keys. Whereas the argument of [11, 25] relies on the fact that the secret key *sk* is completely hidden modulo N, where computations take place in $\mathbb{Z}_N$, we cannot avoid to leak some information about *sk* mod N by releasing LAF images of *sk*. However, using a suitable message encoding, we *can* argue that *sk* is completely hidden modulo the coprime order $(P-1)(Q-1)/4$ of quadratic residues modulo N, which enables a reduction to the DDH assumption.

result. We write $y \leftarrow A(x)$ for $y \leftarrow A(x; R)$ with uniformly chosen R. If A's running time is polynomial in k, A is called probabilistic polynomial-time (PPT).

Standard Definitions. Due to lack of space, we postpone some standard definitions to the full version [21]. These include definitions of PKE and signature schemes, (one-time/strong) EUF-CMA security, IND-CPA security, (chameleon) hash functions, and the DCR, DDH, and DLIN assumptions.

Key-unique SKE Schemes. A secret-key encryption (SKE) scheme (E, D) consists of two PPT algorithms. Encryption $\mathsf{E}(K, M)$ takes a key K and a message M, and outputs a ciphertext C. Decryption $\mathsf{D}(K, C)$ takes a key K and a ciphertext C, and outputs a message M. For correctness, we want $\mathsf{D}(K, C) = M$ for all M, all K, and all $C \leftarrow \mathsf{E}(K, M)$. We say that (E, D) is *key-unique* if for every ciphertext C, there is at most one key K with $\mathsf{D}(K, C) \neq \bot$. For instance, ElGamal encryption can be interpreted as a key-unique SKE scheme through $\mathsf{E}(x, M) := (g^x, g^y, g^{xy} \cdot M)$ (and the obvious D). This example assumes a publicly known group $\mathbb{G} = \langle g \rangle$ in which the DDH assumption holds.[5] If a larger message space (e.g., $\{0,1\}^*$) is desired, hybrid encryption techniques (which are easily seen to preserve key-uniqueness) can be employed.

Pairings. A (symmetric) pairing is a map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ between two cyclic groups $\mathbb{G}$ and $\mathbb{G}_T$ that satisfies $e(g, g) \neq 1$ and $e(g^a, g^b) = e(g, g)^{ab}$ for all generators g of $\mathbb{G}$ and all $a, b \in \mathbb{Z}$.

Waters signatures. In [28], Waters proves the following signature scheme EUF-CMA secure:[6]

- $\mathsf{Gen}(1^k)$ chooses groups $\mathbb{G}, \mathbb{G}_T$ of prime order p, along with a pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, a generator $g \in \mathbb{G}$, and uniform group elements $g^\omega, H_0, \ldots, H_k \in \mathbb{G}$. Output is $vk = (\mathbb{G}, \mathbb{G}_T, e, p, g, (H_i)_{i=0}^k, e(g, g)^\omega)$ and $sigk = (vk, g^\omega)$.
- $\mathsf{Sig}(sigk, M)$, for $M = (M_i)_{i=1}^k \in \{0,1\}^k$, picks $r \leftarrow \mathbb{Z}_p$, and outputs $\sigma := (g^r, g^\omega \cdot (H_0 \prod_{i=1}^k H_i^{M_i})^r)$.
- $\mathsf{Ver}(vk, M, (\sigma_0, \sigma_1))$, outputs 1 iff $e(g, \sigma_1) = e(g, g)^\omega \cdot e(\sigma_0, H_0 \prod_{i=1}^k H_i^{M_i})$.

KDM-CCA and CIRC-CCA security. Let $n = n(k)$ and let PKE be a PKE scheme with message space $\mathcal{M}$. PKE is chosen-ciphertext secure under key-dependent message attacks (n-KDM-CCA secure) iff

$$\mathsf{Adv}^{\mathsf{kdm\text{-}cca}}_{\mathsf{PKE},n,A}(k) \;:=\; \Pr\left[\mathsf{Exp}^{\mathsf{kdm\text{-}cca}}_{\mathsf{PKE},n,A}(k) = 1\right] - 1/2$$

is negligible for all PPT A, where experiment $\mathsf{Exp}^{\mathsf{kdm\text{-}cca}}_{\mathsf{PKE},n,A}$ is defined as follows. First, the experiment tosses a coin $b \leftarrow \{0,1\}$, and samples public parameters $pp \leftarrow \mathsf{Pars}(1^k)$ and n keypairs $(pk_i, sk_i) \leftarrow \mathsf{Gen}(pp)$. Then A is invoked with input pp and $(pk_i)_{i=1}^n$, and access to two oracles:

- a KDM oracle $\mathcal{KDM}_b(\cdot, \cdot)$ that maps $i \in [n]$ and a function $f : (\{0,1\}^*)^n \to \{0,1\}^*$ to a ciphertext $C \leftarrow \mathsf{Enc}(pp, pk_i, M)$. If $b = 0$, then $M = f((sk_i)_{i=1}^n)$; else, $M = 0^{|f((sk_i)_{i=1}^n)|}$.

[5] In our application, $\mathbb{G}$ can be made part of the public parameters.

[6] In fact, our description is a slight folklore variant of Waters's scheme. The original scheme features elements g^α, g^β in vk, so that $e(g^\alpha, g^\beta)$ takes the role of $e(g, g)^\omega$.

– a decryption oracle $\mathcal{DEC}(\cdot,\cdot)$ that takes as input an index $i \in [n]$ and a ciphertext C, and outputs $\mathsf{Dec}(pp, sk_i, C)$.

When A finally generates an output $b' \in \{0,1\}$, the experiment outputs 1 if $b = b'$ (and 0 else). We require that (a) A never inputs a ciphertext C to $\mathcal{DEC}$ that has been produced by $\mathcal{KDM}_b$ (for the same index i), and (b) A only specifies PPT-computable functions f that always output messages of the same length. As a relevant special case, PKE is *n-CIRC-CCA-secure* if it is n-KDM-CCA secure against all A that only query $\mathcal{KDM}_b$ with functions $f \in \mathcal{F}$ for

$$\mathcal{F} := \{f_j : f_j((sk_i)_{i=1}^n) = sk_j\}_{j\in[n]} \cup \{f_M : f_M((sk_i)_{i=1}^n) = M\}_{M\in\mathcal{M}} .$$

(Technically, what we call "circular security" is called "clique security" in [10]. However, our notion of circular security implies that of [10].) Our main result will be a PKE scheme that is n-CIRC-CCA-secure for all polynomials $n = n(k)$.

3 Lossy Algebraic Filters

Informal Description. An $(\ell_{\mathsf{LAF}}, \mathfrak{n})$-lossy algebraic filter (LAF) is a family of functions indexed by a public key Fpk and a tag t. A function $\mathsf{LAF}_{Fpk,t}$ from the family maps an input $X = (X_i)_{i=1}^{\mathfrak{n}} \in \mathbb{Z}_p^{\mathfrak{n}}$ to an output $\mathsf{LAF}_{Fpk,t}(X)$, where p is an ℓ_{LAF}-bit prime contained in the public key.

The crucial property of an LAF is its lossiness. Namely, for a given public key Fpk, we distinguish *injective* and *lossy* tags.[7] For an injective tag t, the function $\mathsf{LAF}_{Fpk,t}(\cdot)$ is injective, and thus has an image of size $p^{\mathfrak{n}}$. However, if t is lossy, then $\mathsf{LAF}_{Fpk,t}(\cdot)$ only depends on a linear combination $\sum_{i=1}^{\mathfrak{n}} \omega_i X_i \bmod p$ of its input. In particular, different X with the same value $\sum_{i=1}^{\mathfrak{n}} \omega_i X_i \bmod p$ are mapped to the same image. Here, the coefficients $\omega_i \in \mathbb{Z}_p$ only depend on Fpk (but not on t). For a lossy tag t, the image of $\mathsf{LAF}_{Fpk,t}(\cdot)$ is thus of size at most p. Note that the modulus p is public, while the coefficients ω_i may be (and in fact will have to be) computationally hidden.

For this concept to be useful, we require that (a) lossy and injective tags are computationally indistinguishable, (b) lossy tags can be generated using a special trapdoor, but (c) new lossy (or, rather, non-injective) tags cannot be found efficiently without that trapdoor, even when having seen polynomially many lossy tags before. In view of our application, we will work with structured tags: each tag $t = (t_{\mathsf{c}}, t_{\mathsf{a}})$ consists of a *core tag* t_{c} and an *auxiliary tag* t_{a}. The auxiliary tag will be a ciphertext part that is authenticated by a filter image.

Definition 1. *An* $(\ell_{\mathsf{LAF}}, \mathfrak{n})$-lossy algebraic filter (LAF) LAF *consists of three PPT algorithms:*

Key generation. $\mathsf{FGen}(1^k)$ *samples a keypair* (Fpk, Ftd)*. The public key* Fpk *contains an* ℓ_{LAF}*-bit prime* p *and the description of a tag space* $\mathcal{T} = \mathcal{T}_{\mathsf{c}} \times \{0,1\}^*$ *for efficiently samplable* $\mathcal{T}_{\mathsf{c}}$*. A tag* $t = (t_{\mathsf{c}}, t_{\mathsf{a}})$ *consists of a core tag* $t_{\mathsf{c}} \in \mathcal{T}_{\mathsf{c}}$ *and an auxiliary tag* $t_{\mathsf{a}} \in \{0,1\}^*$*. A tag may be injective, or lossy, or neither.* Ftd *is a trapdoor that will allow to sample lossy tags.*

[7] Technically, there may also be tags that are neither injective nor lossy.

Evaluation. $\mathsf{FEval}(Fpk, t, X)$, *for a public key* Fpk *and a tag* $t = (t_\mathsf{c}, t_\mathsf{a}) \in \mathcal{T}$, *maps an input* $X = (X_i)_{i=1}^{\mathfrak{n}} \in \mathbb{Z}_p^{\mathfrak{n}}$ *to a unique output* $\mathsf{LAF}_{Fpk,t}(X)$.

Lossy tag generation. $\mathsf{FTag}(Ftd, t_\mathsf{a})$, *for a trapdoor* Ftd *and* $t_\mathsf{a} \in \{0,1\}^*$, *samples a core tag* t_c *such that* $t = (t_\mathsf{c}, t_\mathsf{a})$ *is lossy.*

We require the following:

Lossiness. *The function* $\mathsf{LAF}_{Fpk,t}(\cdot)$ *is injective if* t *is injective. If* t *is lossy, then* $\mathsf{LAF}_{Fpk,t}(X)$ *depends only on* $\sum_{i=1}^{\mathfrak{n}} \omega_i X_i \bmod p$ *for* $\omega_i \in \mathbb{Z}_p$ *that only depend on* Fpk.

Indistinguishability. *Lossy tags are indistinguishable from random tags:*

$$\mathsf{Adv}^{\mathsf{ind}}_{\mathsf{LAF},A}(k) := \Pr\left[A(1^k, Fpk)^{\mathsf{FTag}(Ftd,\cdot)} = 1\right] - \Pr\left[A(1^k, Fpk)^{\mathcal{O}_{\mathcal{T}_\mathsf{c}}(\cdot)} = 1\right]$$

is negligible for all PPT A, *where* $(Fpk, Ftd) \leftarrow \mathsf{FGen}(1^k)$, *and* $\mathcal{O}_{\mathcal{T}_\mathsf{c}}(\cdot)$ *is the oracle that ignores its input and samples a random core tag* t_c.

Evasiveness. *Non-injective (and in particular lossy) tags are hard to find, even given multiple lossy tags:*

$$\mathsf{Adv}^{\mathsf{eva}}_{\mathsf{LAF},A}(k) := \Pr\left[t \text{ non-injective} \,\middle|\, t \leftarrow A(1^k, Fpk)^{\mathsf{FTag}(Ftd,\cdot)}\right]$$

is negligible with $(Fpk, Ftd) \leftarrow \mathsf{FGen}(1^k)$, *and for any PPT algorithm* A *that never outputs a tag obtained through oracle queries (i.e.,* A *never outputs* $t = (t_\mathsf{c}, t_\mathsf{a})$ *when* t_c *has been obtained by an oracle query* $\mathsf{FTag}(Ftd, t_\mathsf{a})$*).*

3.1 Construction

Intuition. We present a construction based on the DLIN problem in a group $\mathbb{G}$ of order p with symmetric pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. Essentially, each tag corresponds to $\mathfrak{n}$ DLIN-encrypted Waters signatures. If the signatures are valid, the tag is lossy. The actual filter maps an input $X = (X_i)_{i=1}^{\mathfrak{n}} \in \mathbb{Z}_p^{\mathfrak{n}}$ to the tuple

$$\mathsf{LAF}_{Fpk,t}(X) := \mathbf{M} \circ X := \Big(\prod_{j=1}^{\mathfrak{n}} \mathbf{M}_{i,j}^{X_j}\Big)_{j=1}^{\mathfrak{n}} \in \mathbb{G}_T^{\mathfrak{n}}, \tag{1}$$

where the matrix $\mathbf{M} = (\mathbf{M}_{i,j})_{i,j \in [\mathfrak{n}]} \in \mathbb{G}_T^{\mathfrak{n}\times\mathfrak{n}}$ is computed from public key and tag. Note that this mapping is lossy if and only if the matrix

$$\widetilde{\mathbf{M}} := (\widetilde{\mathbf{M}}_{i,j}) := (\mathrm{dlog}_{e(g,g)}(\mathbf{M}_{i,j}))_{i,j} \in \mathbb{Z}_p^{\mathfrak{n}\times\mathfrak{n}} \tag{2}$$

of discrete logarithms (to some arbitrary basis $e(g,g) \in \mathbb{G}_T$) is non-invertible.

For a formal description, let $\ell_{\mathsf{LAF}}(k), \mathfrak{n}(k)$ be two functions.

Key generation. $\mathsf{FGen}(1^k)$ generates cyclic groups $\mathbb{G}, \mathbb{G}_T$ of prime order p (where p has bitlength $\lfloor \log_2(p) \rfloor = \ell_{\mathsf{LAF}}(k)$), and a symmetric pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. Then FGen chooses

- a generator $g \in \mathbb{G}$ and a uniform exponent $\omega \leftarrow \mathbb{Z}_p$,
- uniform group elements $U_1, \ldots, U_{\mathfrak{n}} \leftarrow \mathbb{G}$, $H_0, \ldots, H_k \leftarrow \mathbb{G}$, and
- a keypair (Hpk, Htd) for a chameleon hash $\mathsf{CH} : \{0,1\}^* \to \{0,1\}^k$.

FGen finally outputs

$$\begin{aligned} \mathit{Fpk} &:= (\mathbb{G}, \mathbb{G}_T, e, p, g, (H_i)_{i=0}^k, (U_i)_{i=1}^{\mathfrak{n}}, W := e(g,g)^\omega, \mathit{Hpk}) \\ \mathit{Ftd} &:= (\mathit{Fpk}, g^\omega, \mathit{Htd}). \end{aligned}$$

For convenience, write $U_i = g^{u_i}$ for suitable (unknown) exponents u_i.

Tags. (Core) tags are of the form

$$t_\mathsf{c} := (R, (\widetilde{S_i})_{i=1}^{\mathfrak{n}}, (S_{i,j})_{i,j=1}^{\mathfrak{n}}, R_\mathsf{CH}) \in \mathbb{G} \times \mathbb{G} \times \mathbb{G}^{\mathfrak{n}\times\mathfrak{n}} \times \mathcal{R}_\mathsf{CH}$$

(for CH's randomness space $\mathcal{R}_\mathsf{CH}$), where we require $e(U_{j'}, S_{i,j}) = e(U_j, S_{i,j'})$ whenever $i \notin \{j, j'\}$. This means we can write $R = g^r$, $\widetilde{S_i} = g^{\widetilde{s_i}}$, and $S_{i,j} = U_j^{s_i}$ (for $i \neq j$) for suitable $r, s_i, \widetilde{s_i}$. To a tag $t = (t_\mathsf{c}, t_\mathsf{a})$ (with auxiliary part $t_\mathsf{a} \in \{0,1\}^*$), we associate the matrix $\mathbf{M} = (\mathbf{M}_{i,j})_{i,j=1}^{\mathfrak{n}} \in \mathbb{G}_T^{\mathfrak{n}\times\mathfrak{n}}$ with

$$\begin{aligned} \mathbf{M}_{i,j} &= e(U_j, \widetilde{S_i}) \cdot e(g, S_{i,j}) = e(g,g)^{u_j(\widetilde{s_i}+s_i)} \qquad (i \neq j) \\ \mathbf{M}_{i,i} &= \frac{e(g, S_{i,i})}{W \cdot e(H_0 \prod_{i=1}^k H_i^{T_i}, R)} \end{aligned} \tag{3}$$

for $(T_i)_{i=1}^k := \mathsf{CH}_{\mathit{Hpk}}(R, (\widetilde{S_i})_{i=1}^{\mathfrak{n}}, (S_{i,j})_{i,j=1}^{\mathfrak{n}}, t_\mathsf{a}; R_\mathsf{CH})$. If the matrix $\widetilde{\mathbf{M}}$ of discrete logarithms (see (2)) is invertible, we say that t is injective; if $\widetilde{\mathbf{M}}$ has rank 1, then t is lossy. Thus, for lossy tags, $\mathbf{M}_{i,j} = e(g,g)^{u_j(\widetilde{s_i}+s_i)}$ for *all* i, j.

Evaluation. $\mathsf{FEval}(\mathit{Fpk}, t, X)$, for $t = (t_\mathsf{c}, t_\mathsf{a})$, $t_\mathsf{a} \in \{0,1\}^*$, $X = (X_i)_{i=1}^{\mathfrak{n}} \in \mathbb{Z}_p^{\mathfrak{n}}$, and Fpk and t_c as above, computes $\mathbf{M}$ as in (3) and then $(Y_i)_{i=1}^{\mathfrak{n}} := \mathsf{LAF}_{\mathit{Fpk},t}(X) \in \mathbb{G}_T^{\mathfrak{n}}$ as in (1).

Lossiness. If we write $Y_i = e(g,g)^{y_i}$, the definition of FEval implies $(y_i)_{i=1}^{\mathfrak{n}} = \widetilde{\mathbf{M}} \cdot X$. Since injective tags satisfy that $\widetilde{\mathbf{M}}$ is invertible, they lead to injective functions $\mathsf{LAF}_{\mathit{Fpk},t}(\cdot)$. But for a lossy tag, $\widetilde{\mathbf{M}}_{i,j} = u_j(\widetilde{s_i} + s_i)$, so that

$$y_i = \sum_{j=1}^{\mathfrak{n}} u_j(\widetilde{s_i}+s_i)X_j = (\widetilde{s_i}+s_i) \cdot \sum_{j=1}^{\mathfrak{n}} u_j X_j \mod p.$$

Specifically, $\mathsf{LAF}_{\mathit{Fpk},t}(X)$ depends only on $\sum_i \omega_i X_i \bmod p$ for $\omega_i := u_i$.

Lossy tag generation. $\mathsf{FTag}(\mathit{Ftd}, t_\mathsf{a})$, for Ftd as above and $t_\mathsf{a} \in \{0,1\}^*$, first chooses a random CH-image $T = (T_i)_{i=1}^k \in \{0,1\}^k$ that can later be explained, using Htd, as the CH-image of an arbitrary preimage. FTag then chooses uniform $r, s_i, \widetilde{s_i} \leftarrow \mathbb{Z}_p$ and sets (for $i \neq j$)

$$\begin{aligned} R &:= g^r, \qquad \widetilde{S_i} := g^{\widetilde{s_i}}, \\ S_{i,j} &:= U_j^{s_i}, \quad S_{i,i} := U_i^{\widetilde{s_i}+s_i} \cdot g^\omega \cdot \left(H_0 \prod_{i=1}^k H_i^{T_i}\right)^r. \end{aligned} \tag{4}$$

Finally, FTag chooses R_CH with $\mathsf{CH}_{\mathit{Hpk}}(R, (\widetilde{S_i})_{i=1}^{\mathfrak{n}}, (S_{i,j})_{i,j=1}^{\mathfrak{n}}, t_\mathsf{a}; R_\mathsf{CH}) = T$ and outputs $t_\mathsf{c} = (R, (\widetilde{S_i})_{i=1}^{\mathfrak{n}}, (S_{i,j})_{i,j=1}^{\mathfrak{n}}, R_\mathsf{CH})$. Intuitively, t_c consists of $\mathfrak{n}$ DLIN encryptions (with correlated randomness $s_i, \widetilde{s_i}$) of Waters signatures $(g^r, g^\omega \cdot (H_0 \prod_{i=1}^k H_i^{T_i})^r)$ for message T. Indeed, substituting into (3) yields

$$\mathbf{M}_{i,i} := \frac{e(g,g)^{u_i(\widetilde{s_i}+s_i)} \cdot W \cdot e(g,(H_0 \prod_{i=1}^k H_i^{T_i})^r)}{W \cdot e(g^r, H_0 \prod_{i=1}^k H_i^{T_i})} = e(g,g)^{u_i(\widetilde{s_i}+s_i)}.$$

Hence, $\widetilde{\mathbf{M}}_{i,j} = u_j(\widetilde{s_i} + s_i)$ for *all* i, j, and thus the resulting tag $t = (t_\mathsf{c}, t_\mathsf{a})$ is lossy.

A Generalization. In the full paper [21], we also show how to generalize the above construction to achieve constant-size tags *and* evaluation keys.

Other Instances and Further Applications of LAFs. Since LAFs can be seen as "disguised signature schemes", it seems interesting to try to convert other signature schemes (and in particular schemes that do not require pairing-friendly groups) to LAFs. Besides, LAFs would seem potentially useful in other settings, specifically in settings with inherently many challenges (e.g., the selective-opening setting [6]).

Theorem 1. *If the DLIN assumption holds in* $\mathbb{G}$*, and* CH *is a chameleon hash function, then the LAF construction* LAF *from Section 3.1 satisfies Definition 1.*

The lossiness of LAF has already been discussed in Section 3.1. We prove indistinguishability and evasiveness separately.

Lemma 1. *For every adversary A on* LAF*'s indistinguishability, there exists a DLIN distinguisher B such that* $\mathsf{Adv}^{\mathsf{ind}}_{\mathsf{LAF},A}(k) = \mathfrak{n} \cdot \mathsf{Adv}^{\mathsf{dlin}}_{B}(k)$.

Intuitively, to see Lemma 1, observe that lossy tags differ from random tags only in their $S_{i,i}$ components, and in how the CH randomness R_CH is generated. For lossy tags, the $S_{i,i}$ are (parts of) DLIN ciphertexts, which are pseudorandom under the DLIN assumption. Furthermore, the uniformity property of CH guarantees that the distribution of R_CH is the same for lossy and random tags. We formally prove Lemma 1 in the full version [21].

Lemma 2. *For every adversary A on* LAF*'s evasiveness, there exist adversaries B, C, and F with* $\mathsf{Adv}^{\mathsf{eva}}_{\mathsf{LAF},A}(k) \leq \left|\mathsf{Adv}^{\mathsf{ind}}_{\mathsf{LAF},B}(k)\right| + \mathsf{Adv}^{\mathsf{cr}}_{\mathsf{CH},C}(k) + \mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\mathsf{Sig}_{\mathsf{Wat}},F}(k)$.

Intuitively, Lemma 2 holds because lossy (or, rather, non-injective) tags correspond to DLIN-encrypted Waters signatures. Hence, even after seeing many lossy tags (i.e., encrypted signatures), an adversary cannot produce a fresh encrypted signature. We note that the original Waters signatures from [28] are re-randomizable and thus not *strongly* unforgeable. To achieve evasiveness, we have thus used a chameleon hash function, much like Boneh et al. [8] did to make Waters signatures strongly unforgeable. We give a formal proof in [21].

Combining Lemma 1, Lemma 2, and the fact that Waters signatures are EUF-CMA secure already under the CDH assumption, we obtain Theorem 1.

4 CIRC-CCA-Secure Encryption Scheme

Setting and Ingredients. First, we assume an algorithm GenN that outputs ℓ_N-bit Blum integers $N = PQ$ along with their prime factors P and Q. If N

is clear from the context, we write $\mathbb{G}_{\mathsf{rnd}}$ and $\mathbb{G}_{\mathsf{msg}}$ for the unique subgroups of $\mathbb{Z}_{N^3}^*$ of order $(P-1)(Q-1)/4$, resp. N^2. We also write $h := 1 + N \bmod N^3$, so $\langle h \rangle = \mathbb{G}_{\mathsf{msg}}$. Note that it is efficiently possible to compute $\mathrm{dlog}_h(X) := x$ for $X := h^x \in \mathbb{G}_{\mathsf{msg}}$ and $x \in \mathbb{Z}_{N^2}$. Specifically, it is efficiently possible to test for membership in $\mathbb{G}_{\mathsf{msg}}$. In our scheme, $\mathbb{G}_{\mathsf{msg}}$ will be used to embed a suitably encoded message, and $\mathbb{G}_{\mathsf{rnd}}$ will be used for blinding. We require that

- P and Q are safe primes of bitlength between $\ell_N/2 - k$ and $\ell_N/2 + k$,
- $\gcd((P-1)(Q-1)/4, N) = 1$ (as, e.g., for uniform P, Q of a certain length),
- $\ell_N \geq 25k + 8$ (e.g., $k = 80$ and $\ell_N = 2048$)
- the DCR assumption holds in $\mathbb{Z}_{N^3}^*$, and the DDH assumption holds in $\mathbb{G}_{\mathsf{rnd}}$.

We also assume an $(\ell_{\mathsf{LAF}}, \mathfrak{n})$-lossy algebraic filter LAF for $\mathfrak{n} = 6$ and $\ell_{\mathsf{LAF}} = (\ell_N + k + 1)/(\mathfrak{n} - 2)$. Our scheme will encrypt messages from the domain

$$\mathcal{M} := \mathbb{Z}_{2^{3k}} \times \mathbb{Z}_{p \cdot 2^k} \times \mathbb{Z}_{N \cdot 2^{k-2}},$$

where p is the modulus of the used LAF. (The reason for this weird-looking message space will become clearer in the proof.) During encryption, we will have to treat a message $M = (a, b, c) \in \mathcal{M}$ both as an element of $\mathbb{Z}_{N^2}$ and as an LAF-input from $\mathbb{Z}_p^{\mathfrak{n}}$. In these cases, we can encode

$$\begin{aligned} [M]_{\mathbb{Z}} &:= a + 2^{3k} \cdot b + p \cdot 2^{4k} \cdot c \in \mathbb{Z}, \\ [M]_{\mathbb{Z}_p^{\mathfrak{n}}} &:= (a, b \bmod p, c_0, \ldots, c_{\mathfrak{n}-3}) \in \mathbb{Z}_p^{\mathfrak{n}} \end{aligned} \qquad (5)$$

for the natural interpretation of $\mathbb{Z}_i$-elements as integers between 0 and $i-1$, and c's p-adic representation $(c_i)_{i=0}^{\mathfrak{n}-3} \in \mathbb{Z}_p^{\mathfrak{n}-2}$ with $c = \sum_{i=0}^{\mathfrak{n}-3} c_i \cdot p^i$. By our choice of ℓ_N and ℓ_{LAF}, we have $0 \leq [M]_{\mathbb{Z}} < N^2 - 2^k$. However, the encoding $[M]_{\mathbb{Z}_p^{\mathfrak{n}}}$ is *not* injective, since it only depends on $b \bmod p$ (while $0 \leq b < p \cdot 2^k$).

Finally, we assume a strongly OT-EUF-CMA secure signature scheme $\mathsf{Sig} = (\mathsf{SGen}, \mathsf{Sig}, \mathsf{Ver})$ with k-bit verification keys, and a key-unique IND-CPA secure symmetric encryption scheme (E, D) (see Section 2) with k-bit symmetric keys K and message space $\{0,1\}^*$.

Now consider the following PKE scheme PKE:

Public Parameters. $\mathsf{Pars}(1^k)$ first runs $(N, P, Q) \leftarrow \mathsf{GenN}(1^k)$. Recall that this fixes the groups $\mathbb{G}_{\mathsf{rnd}}$ and $\mathbb{G}_{\mathsf{msg}}$. Then, Pars selects two generators g_1, g_2 of $\mathbb{G}_{\mathsf{rnd}}$. Finally, Pars runs $(\mathit{Fpk}, \mathit{Ftd}) \leftarrow \mathsf{FGen}(1^k)$ and outputs $pp = (N, g_1, g_2, \mathit{Fpk})$. In the following, we denote with p the LAF modulus contained in Fpk.

Key Generation. $\mathsf{Gen}(pp)$ uniformly selects four messages $s_j = (a_j, b_j, c_j) \in \mathcal{M}$ (for $1 \leq j \leq 4$) as secret key, and sets $pk := \left(u := g_1^{[s_1]_{\mathbb{Z}}} g_2^{[s_2]_{\mathbb{Z}}}, v := g_1^{[s_3]_{\mathbb{Z}}} g_2^{[s_4]_{\mathbb{Z}}}\right)$ and $sk := (s_j)_{j=1}^4$.

Encryption. $\mathsf{Enc}(pp, pk, M)$, for pp and pk as above, and $M \in \mathcal{M}$, uniformly selects exponents $r, \widetilde{r} \leftarrow \mathbb{Z}_{N/4}$, a random filter core tag t_{c}, a Sig-keypair $(vk, sigk) \leftarrow \mathsf{SGen}(1^k)$, and a random symmetric key $K \in \{0,1\}^k$ for (E, D), and computes

$$\begin{aligned} (G_1, G_2) &:= (g_1^r, g_2^r) & C_{\mathsf{E}} &\leftarrow \mathsf{E}(K, \mathsf{LAF}_{Fpk,t}([M]_{\mathbb{Z}_p^{\mathfrak{n}}})) \\ (\widetilde{G}_1, \widetilde{G}_2) &:= (g_1^{\widetilde{r}}, g_2^{\widetilde{r}}) & \sigma &\leftarrow \mathsf{Sig}(sigk, ((G_j, \widetilde{G}_j)_{j=1}^2, Z, \widetilde{Z}, C_{\mathsf{E}}, t_{\mathsf{c}})) \\ Z &:= (u^{vk} v)^{r \cdot N^2} & C &:= ((G_j, \widetilde{G}_j)_{j=1}^2, Z, \widetilde{Z}, C_{\mathsf{E}}, t_{\mathsf{c}}, vk, \sigma) \\ \widetilde{Z} &:= (u^{vk} v)^r u^{\widetilde{r}} h^{K+2^k \cdot [M]_{\mathbb{Z}}} \end{aligned}$$

for the auxiliary tag $t_{\mathsf{a}} := vk$, and the resulting filter tag $t := (t_{\mathsf{c}}, t_{\mathsf{a}})$.

Decryption. $\mathsf{Dec}(pp, sk, C)$, for pp, sk and C as above, first checks the signature σ and rejects with $\perp$ if $\mathsf{Ver}(vk, ((G_j, \widetilde{G}_j)_{j=1}^2, Z, \widetilde{Z}, C_{\mathsf{E}}, t_{\mathsf{c}}), \sigma) = 0$, or if

$$Z \neq \left(G_1^{[s_1]_{\mathbb{Z}} \cdot vk + [s_3]_{\mathbb{Z}}} G_2^{[s_2]_{\mathbb{Z}} \cdot vk + [s_4]_{\mathbb{Z}}} \right)^{N^2} .$$

Then Dec computes

$$Z' := G_1^{[s_1]_{\mathbb{Z}} \cdot vk + [s_3]_{\mathbb{Z}}} G_2^{[s_2]_{\mathbb{Z}} \cdot vk + [s_4]_{\mathbb{Z}}} \widetilde{G}_1^{[s_1]_{\mathbb{Z}}} \widetilde{G}_2^{[s_2]_{\mathbb{Z}}}$$

and then $K \in \{0,1\}^k, M \in \mathcal{M}$ with $K + 2^k \cdot [M]_{\mathbb{Z}} := \mathrm{dlog}_h(\widetilde{Z}/Z')$. If $\widetilde{Z}/Z' \notin \mathbb{G}_{\mathsf{msg}}$, or no such M exists, or $\mathsf{D}(K, C_{\mathsf{E}}) \neq \mathsf{LAF}_{Fpk,t}([M]_{\mathbb{Z}_p^{\mathfrak{n}}})$ (for $t = (t_{\mathsf{c}}, t_{\mathsf{a}})$ computed as during encryption), then Dec rejects with $\perp$. Else, Dec outputs M.

Secret Keys as Messages. Our scheme has secret keys $s = (s_j)_{j=1}^4 \in \mathcal{M}^4$; hence, we can only encrypt one quarter s_j of a secret key at a time. In the security proof below, we will thus only consider KDM queries that ask to encrypt a specific secret key *part*. Alternatively, we can change our scheme, so that 4-tuples of $\mathcal{M}$-elements are encrypted. To avoid malleability (which would destroy CCA security), we of course have to use only one LAF tag for this. Our CIRC-CCA proof below applies to such a changed scheme with minor syntactic changes.

Efficiency. When instantiated with our DLIN-based LAF construction from Section 3, and taking $\mathfrak{n} = 6$ as above, our scheme has ciphertexts with 43 $\mathbb{G}$-elements, 6 $\mathbb{Z}_{N^3}$-elements, plus chameleon hash randomness, a one-time signature and verification key, and a symmetric ciphertext (whose size could be in the range of one $\mathbb{Z}_{N^2}$-element plus some encryption randomness). The number of group elements in the ciphertext is constant, and does not grow in the security parameter. The public parameters contain $\mathbf{O}(k)$ group elements[8] (most of them from $\mathbb{G}$), and public keys contain two $\mathbb{Z}_{N^3}$-elements; secret keys consist of four $\mathbb{Z}_{N^2}$-elements. While these parameters are not competitive with current non-KDM-secure schemes, they are significantly better than those from the circular-secure scheme of Camenisch et al. [14].[9]

Security Proof (single-user user). It is instructive to first treat the single-user case. Here, we essentially only require that PKE is IND-CCA secure, even if

[8] Using the generalized LAF mentioned at the end of Section 3.1, public parameters with $\mathbf{O}(1)$ group elements are possible, at the cost of a (constant) number of extra group elements per tag.

[9] For instance, Section 7 of the full version of [14] implies that their scheme has a public key, resp. ciphertext of about 500, resp. 1000 $\mathbb{G}$-elements (for $\log_2(|\mathbb{G}|) = 160$).

encryptions of its secret key are made public. For the multi-user case (see [21]), we can then proceed like [10, 11] and re-randomize keys and ciphertexts of a single PKE instance. This enables an analysis analogous to the single-user case.

Theorem 2. *Assume the DCR assumption holds in* $\mathbb{Z}_{N^3}$*, the DDH assumption holds in* $\mathbb{G}_{\mathsf{rnd}}$*,* LAF *is an LAF,* Sig *is a strongly OT-EUF-CMA secure signature scheme,* H *is collision-resistant, and* (E,D) *is a key-unique IND-CPA secure SKE scheme. Then* PKE *is* 1*-CIRC-CCA-secure.*

Proof. Assume a PPT adversary A on PKE's 1-CIRC-CCA security. Say that A always makes $q = q(k)$ KDM queries. We proceed in games. Let out_i denote the output of Game i.

Game 1 is the 1-KDM-CCA experiment with PKE and A. By definition, $\Pr[out_1 = 1] - 1/2 \;=\; \mathsf{Adv}^{\mathsf{kdm\text{-}cca}}_{\mathsf{PKE},A}(k)$.

In **Game** 2, we modify the way KDM queries are answered. Namely, in each ciphertext prepared for A, we set up Z and $\widetilde{Z}$ up as

$$\begin{aligned} Z \;&:=\; \left(G_1^{[s_1]_{\mathbb{Z}}\cdot vk+[s_3]_{\mathbb{Z}}} G_2^{[s_2]_{\mathbb{Z}}\cdot vk+[s_4]_{\mathbb{Z}}}\right)^{N^2} \\ \widetilde{Z} \;&:=\; G_1^{[s_1]_{\mathbb{Z}}\cdot vk+[s_3]_{\mathbb{Z}}} G_2^{[s_2]_{\mathbb{Z}}\cdot vk+[s_4]_{\mathbb{Z}}} \widetilde{G}_1^{[s_1]_{\mathbb{Z}}} \widetilde{G}_2^{[s_2]_{\mathbb{Z}}} \cdot h^{K+2^k\cdot[M]_{\mathbb{Z}}}. \end{aligned} \tag{6}$$

for the already prepared $(G_j, \widetilde{G}_j) = (g_j^r, g_j^{\widetilde{r}})$. This change is only conceptual by our setup of u, v, so $\Pr[out_2 = 1] \;=\; \Pr[out_1 = 1]$.

In **Game** 3, we again change how KDM ciphertexts are prepared. Intuitively, our goal is now to prepare the G_j and $\widetilde{G}_j$ with additional $\mathbb{G}_{\mathsf{msg}}$-components, such that $\widetilde{Z}$, as computed in (6), is of the form $g \cdot h^K$ for some $g \in \mathbb{G}_{\mathsf{rnd}}$. (That is, we want the $\mathbb{G}_{\mathsf{msg}}$-components of the $G_j, \widetilde{G}_j$ to cancel out the $h^{2^k\cdot[M]_{\mathbb{Z}}}$ term in (6).) To do so, we prepare $G_j = g_j^r / h^{\alpha_j \cdot 2^k}$ and $\widetilde{G}_j = g_j^r / h^{\widetilde{\alpha}_j \cdot 2^k}$ for $j \in \{1,2\}$ and suitable $\alpha_j, \widetilde{\alpha}_j$ to be determined. $\widetilde{Z}$ is still computed as in (6), so

$$\widetilde{Z} \;=\; g \cdot h^{K+2^k\cdot[M]_{\mathbb{Z}} - 2^k(\alpha_1([s_1]_{\mathbb{Z}}\cdot vk+[s_3]_{\mathbb{Z}})+\alpha_2([s_2]_{\mathbb{Z}}\cdot vk+[s_4]_{\mathbb{Z}})+\widetilde{\alpha}_1[s_1]_{\mathbb{Z}}+\widetilde{\alpha}_2[s_2]_{\mathbb{Z}})}$$

for $g = g_1^{r\cdot([s_1]_{\mathbb{Z}}\cdot vk+[s_3]_{\mathbb{Z}})+\widetilde{r}[s_1]_{\mathbb{Z}}} g_2^{r\cdot([s_2]_{\mathbb{Z}}\cdot vk+[s_4]_{\mathbb{Z}})+\widetilde{r}[s_2]_{\mathbb{Z}}} = \left(u^{vk}v\right)^r u^{\widetilde{r}} \in \mathbb{G}_{\mathsf{rnd}}$. So to prepare a KDM encryption of s_{j^*} with $\widetilde{Z} = g \cdot h^K$, we can set $(\alpha_1, \alpha_2, \widetilde{\alpha}_1, \widetilde{\alpha}_2)$ to $(0,0,1,0)$ for $j^* = 1$, to $(1, 0, -vk, 0)$ for $j^* = 2$, to $(0,0,1,0)$ for $j^* = 3$, and to $(0, 1, 0, -vk)$ for $j^* = 4$. (vk can be chosen independently in advance.) The remaining parts of C are prepared as in Game 2. We claim

$$\Pr[out_3 = 1] - \Pr[out_2 = 1] \;\leq\; 4 \cdot \mathsf{Adv}^{\mathsf{dcr}}_{\mathbb{Z}^*_{N^3},B}(k) + \mathbf{O}(2^{-k}) \tag{7}$$

for a suitable DCR distinguisher B that simulates Game 2, resp. Game 3. Concretely, B gets as input a value $\widetilde{W} \in \mathbb{Z}^*_{N^3}$ of the form $\widetilde{W} = \tilde{g}^{N^2} \cdot h^b$ for $b \in \{0,1\}$. Note that if we set $W := \widetilde{W}^{-2^k}$, we have $W = g^{\widehat{r}} / h^{b\cdot 2^k} \in \mathbb{Z}^*_{N^3}$, with uniform $g^{\widehat{r}} \in \mathbb{G}_{\mathsf{rnd}}$. First, B guesses a value of $j^* \in [4]$. (This gives a very small hybrid argument, in which in the j^*-th step, only encryptions of s_{j^*} are changed.) We only detail B's behavior for the case $j^* = 3$; the other cases are easier or analogous. First, B sets up $g_1 := W^{N^2}$ and $g_2 := W^{\gamma N^4}$ for uniform $\gamma \in \mathbb{Z}_{N/4}$. To prepare an encryption of s_3, B chooses uniform $\rho, \widetilde{\rho} \in \mathbb{Z}_{N^2/4}$ and sets

$$\begin{aligned} G_1 &:= W^{\rho \cdot (\rho^{-1})} & G_2 &:= W^{\gamma \cdot \rho \cdot (\rho^{-1}) \cdot N^2} \\ \widetilde{G}_1 &:= W^{vk \cdot \widetilde{\rho} \cdot (\widetilde{\rho}^{-1})} & \widetilde{G}_2 &:= W^{\gamma \cdot vk \cdot \widetilde{\rho} \cdot (\widetilde{\rho}^{-1}) \cdot N^2}, \end{aligned}$$

where the values $\rho^{-1}, \widetilde{\rho}^{-1}$ are computed modulo N^2. This implicitly sets $r = \rho \cdot (\rho^{-1})/N^2 \bmod |\mathbb{G}_{\mathsf{rnd}}|$ and $\widetilde{r} = vk \cdot \widetilde{\rho} \cdot (\widetilde{\rho}^{-1})/N^2 \bmod |\mathbb{G}_{\mathsf{rnd}}|$, both of which are statistically close to uniform. Furthermore, $G_j = g_j^r / h^{b \cdot \alpha_j \cdot 2^k}$ and $\widetilde{G}_j = g_j^{\widetilde{r}} / h^{b \cdot \widetilde{\alpha_j} \cdot 2^k}$; so, depending on B's challenge, encryptions of s_3 are prepared as in Game 2 or Game 3. Similar arguments work for $j^* = 1, 2, 4$, and (7) follows. (The $\mathbf{O}(2^{-k})$ term in (7) accounts for the statistical defect caused by choosing $\mathbb{G}_{\mathsf{rnd}}$-exponents from $\mathbb{Z}_{N/4}$, resp. $\mathbb{Z}_{N^2/4}$.)

Using the definition of u and v, our change in Game 3 implies $\widetilde{Z} = (u^{vk} v)^r \cdot u^{\widetilde{r}} \cdot h^K$ when a key part s_j is to be encrypted. (However, note that we still have $Z = (u^{vk} v)^{r \cdot N^2}$ in any case.) This means that A still obtains information about the s_j (beyond what is public from pk) from its KDM queries, but this information is limited to values $\mathsf{LAF}_{Fpk,t}([s_j]_{\mathbb{Z}_p^n})$. We will now further cap this leaked information by making $\mathsf{LAF}_{Fpk,t}(\cdot)$ lossy. Namely, in **Game** 4, we use the LAF trapdoor Ftd initially sampled along with Fpk. Concretely, when preparing a ciphertext C for A, we sample t_{c} using $t_{\mathsf{c}} \leftarrow \mathsf{FTag}(Ftd, t_{\mathsf{a}})$ for the corresponding auxiliary tag $t_{\mathsf{a}} = vk$. A simple reduction shows

$$\Pr[out_4 = 1] - \Pr[out_3 = 1] \;=\; \mathsf{Adv}^{\mathsf{ind}}_{\mathsf{LAF},C_2}(k)$$

for a suitable adversary C_2 on LAF's indistinguishability.

In **Game** 5, we reject all decryption queries of A that re-use a verification key vk from one of the KDM ciphertexts. To show that this change does not significantly affect A's view, assume a decryption query C that re-uses a key $vk = vk^*$ from a KDM ciphertext C^*. Recall that C contains a signature σ of $X := ((G_j, \widetilde{G}_j)_{j=1}^2, Z, \widetilde{Z}, C_{\mathsf{E}}, t_{\mathsf{c}})$ under an honestly generated Sig-verification-key $vk = t_{\mathsf{a}} = t_{\mathsf{a}}^* = vk^*$. Since we assumed $t = (t_{\mathsf{c}}, t_{\mathsf{a}}) = (t_{\mathsf{c}}^*, t_{\mathsf{a}}^*) = t^*$, and A is not allowed to query unchanged challenge ciphertexts for decryption, we must have $(X, \sigma) \neq (X^*, \sigma^*)$ for the corresponding message X^* and signature σ^* from C^*. Hence, Game 4 and Game 5 only differ when A manages to forge a signature. A straightforward reduction to the strong OT-EUF-CMA security of Sig yields

$$\Pr[out_5 = 1] - \Pr[out_4 = 1] \;=\; q(k) \cdot \mathsf{Adv}^{\mathsf{seuf\text{-}cma}}_{\mathsf{LAF},F}(k)$$

for a forger F against Sig that makes at most one signature query.

In **Game** 6.i (for $0 \leq i \leq q$), the first i challenge ciphertexts are prepared using $Z = \widehat{g}^{N^2}$ and $\widetilde{Z} = \widehat{g} \cdot u^{\widetilde{r}} \cdot h^K$ (if a key component s_j is to be encrypted), resp. $\widetilde{Z} = \widehat{g} \cdot u^{\widetilde{r}} \cdot h^{K + 2^k [M]_{\mathbb{Z}}}$ (if a constant $M \in \mathcal{M}$ is to be encrypted) for an independently uniform $\widehat{g} \leftarrow \mathbb{G}_{\mathsf{rnd}}$ drawn freshly for each ciphertext. Obviously, Game 6.0 is identical to Game 5: $\Pr[out_{6.0} = 1] \;=\; \Pr[out_5 = 1]$. We will move from Game 6.i to Game 6.$(i+1)$ in several steps. During these steps, we denote with $C = ((G_j, \widetilde{G}_j)_{j=1}^2, Z, \widetilde{Z}, C_{\mathsf{E}}, t_{\mathsf{c}}, vk, \sigma)$ the $(i+1)$-st KDM ciphertext.

In **Game** 6.i.1, we change the $\mathbb{G}_{\mathsf{rnd}}$ parts of G_1, G_2 from a Diffie-Hellman tuple (with respect to g_1, g_2) to a random tuple. Concretely, if an s_j is to be encrypted, we set $(G_1, G_2) = (g_1^{r_1}/h^{\alpha_1 \cdot 2^k}, g_2^{r_2}/h^{\alpha_2 \cdot 2^k})$; if a constant M is encrypted, we set $(C_1, C_2) = (g_1^{r_1}, g_2^{r_2})$, in both cases for independently uniform $r_1, r_2 \leftarrow \mathbb{Z}_{N/4}$. The $\mathbb{G}_{\mathsf{msg}}$ parts of G_1, G_2 are thus unchanged compared to Game 6.i. Note that the $\widetilde{G}_j$ are still prepared as $\widetilde{G}_j = g_j^{\widetilde{r}}/h^{\widetilde{\alpha} \cdot 2^k}$, resp. $\widetilde{G}_j = g_j^{\widetilde{r}}$. A straightforward reduction to the DDH assumption in $\mathbb{G}_{\mathsf{rnd}}$ yields

$$\sum_{i=1}^{q(k)} \left(\Pr\left[out_{6.i} = 1\right] - \Pr\left[out_{6.i.1} = 1\right]\right) \;=\; q(k) \cdot \mathsf{Adv}^{\mathsf{ddh}}_{\mathbb{G}_{\mathsf{rnd}}, D_1}(k) + \mathbf{O}(2^{-k})$$

for a suitable D_1. The $\mathbf{O}(2^{-k})$ error term accounts for the statistical difference caused by the choice of exponents $r_j \leftarrow \mathbb{Z}_{N/4}$, which induces an only almost-uniform distribution on group elements g^{r_j}. Note that at this point, Z and $\widetilde{Z}$ are still computed as in (6), even if an s_j is to be encrypted.

In **Game** 6.i.2, we compute Z and $\widetilde{Z}$ as $Z = \widehat{g}^{N^2}$ and $\widetilde{Z} = \widehat{g} \cdot u^{\widetilde{r}} \cdot h^K$, resp. $\widetilde{Z} = \widehat{g} \cdot u^{\widetilde{r}} \cdot h^{K + 2^k [M]_{\mathbb{Z}}}$ for a fresh $\widehat{g} \leftarrow \mathbb{G}_{\mathsf{rnd}}$. Thus, the difference to Game 6.i.1 is that we substitute a $\mathbb{G}_{\mathsf{rnd}}$-element computed as $G_1^{[s_1]_{\mathbb{Z}} \cdot vk + [s_3]_{\mathbb{Z}}} G_2^{[s_2]_{\mathbb{Z}} \cdot vk + [s_4]_{\mathbb{Z}}}$ with a fresh random $\widehat{g}$. To show that this change affects A's view only negligibly, it suffices to show that A's statistical information about

$$\begin{aligned} X \;:=\; & \mathrm{dlog}_g \left(G_1^{[s_1]_{\mathbb{Z}} \cdot vk + [s_3]_{\mathbb{Z}}} G_2^{[s_2]_{\mathbb{Z}} \cdot vk + [s_4]_{\mathbb{Z}}} \right) \\ & = \gamma_1 r_1 ([s_1]_{\mathbb{Z}} \cdot vk + [s_3]_{\mathbb{Z}}) + \gamma_2 r_2 ([s_2]_{\mathbb{Z}} \cdot vk + [s_4]_{\mathbb{Z}}) \bmod |\mathbb{G}_{\mathsf{rnd}}| \end{aligned}$$

(for some arbitrary generator g of $\mathbb{G}_{\mathsf{rnd}}$ and $\gamma_j = \mathrm{dlog}_g(g_j)$) is negligible. This part will be rather delicate, since we will have to argue that both A's KDM queries and A's decryption queries yield (almost) no information about X.

First, observe that A gets the following information about the s_j:

- pk reveals (through u and v) precisely the two linear equations $\gamma_1 [s_1]_{\mathbb{Z}} + \gamma_2 [s_2]_{\mathbb{Z}} \bmod |\mathbb{G}_{\mathsf{rnd}}|$ and $\gamma_1 [s_3]_{\mathbb{Z}} + \gamma_2 [s_4]_{\mathbb{Z}} \bmod |\mathbb{G}_{\mathsf{rnd}}|$ about the s_j, where the γ_j are as above. For $r_1 \neq r_2$, these equations are linearly independent of the equation that defines X. Hence, for uniform r_1, r_2, X is (almost) independent of pk.
- By LAF's lossiness, KDM queries yield (through $C_{\mathsf{E}} = \mathsf{E}(K, \mathsf{LAF}_{Fpk,t}([s_j]_{\mathbb{Z}_p^{\mathfrak{n}}}))$ in total at most one equation $\omega_1 a_j + \omega_2 b_j + \sum_{i=0}^{\mathfrak{n}-2} \omega_{3+i} c_{j,i} \bmod p$ for each j, where $(a_j, b_j, c_{j,0}, \ldots, c_{j,\mathfrak{n}-3}) := [s_j]_{\mathbb{Z}_p^{\mathfrak{n}}}$, and the ω_i are the (fixed) coefficients from LAF's lossiness property. (Recall the encodings $[s_j]_{\mathbb{Z}}, [s_j]_{\mathbb{Z}_p^{\mathfrak{n}}}$ of the $s_j = (a_j, b_j, c_j)$ from (5).) Hence, the $b_j \in \mathbb{Z}_{p \cdot 2^k}$ fully blind the information released about the $c_j \in \mathbb{Z}_{2^{k-2}N}$ through the KDM ciphertexts. Thus, KDM ciphertexts reveal no information about $c_j \bmod |\mathbb{G}_{\mathsf{rnd}}|$ and hence also about $[s_j]_{\mathbb{Z}} \bmod |\mathbb{G}_{\mathsf{rnd}}|$.

Consequently, even given pk and the KDM ciphertexts, X is statistically close to independently uniform. This already shows that our change from Game 6.i.2

affects A's view only negligibly if A makes no decryption queries. It remains to show that decryption queries yield no additional information about the s_j.

To do so, let us say that a ciphertext C is *consistent* iff there exist $r, \widetilde{r}$ with $(G_j, \widetilde{G}_j) = (g_j^r, g_j^{\widetilde{r}})$ for both $j \in \{1,2\}$. Note that the decryption of a consistent ciphertext yields no information about the s_j beyond pk. (pk and $r, \widetilde{r}$ determine the values Z, Z' computed during decryption; everything else follows from Z' and C.) So it suffices to prove the following lemma (which we do in [21]):

Lemma 3. *In the situation of Game 6.$i.\ell$ (for $\ell \in \{1,2\}$), let $\mathsf{bad}_{\mathsf{query.i.}\ell}$ be the event that A places an inconsistent decryption query that is not rejected. Then*

$$\sum_{i=1}^{q(k)} \left(\Pr\left[\mathsf{bad}_{\mathsf{query.i.1}}\right] + \Pr\left[\mathsf{bad}_{\mathsf{query.i.2}}\right]\right) \;\le\; 2 \cdot q(k) \cdot \mathsf{Adv}^{\mathsf{eva}}_{\mathsf{LAF},F}(k) + \mathbf{O}(2^{-3k}).$$

for a suitable evasiveness adversary F on LAF.

By our discussion above and Lemma 3, we obtain that

$$\sum_{i=1}^{q(k)} \left|\Pr\left[out_{6.i.2} = 1\right] - \Pr\left[out_{6.i.1} = 1\right]\right| \;\le\; 2 \cdot q(k) \cdot \mathsf{Adv}^{\mathsf{eva}}_{\mathsf{LAF},F}(k) + \mathbf{O}(2^{-3k}).$$

In **Game** 6.i.3, we reverse the change from Game 6.i.1. Concretely, we prepare the G_j as $G_j = g_j^r / h^{\alpha_j \cdot 2^k}$, resp. $G_j = g_j^r$ for a single $r \leftarrow \mathbb{Z}_{N/4}$. Another straightforward reduction to the DDH assumption in $\mathbb{G}_{\mathsf{rnd}}$ yields that

$$\sum_{i=1}^{q(k)} \left(\Pr\left[out_{6.i.3} = 1\right] - \Pr\left[out_{6.i.2} = 1\right]\right) \;=\; q(k) \cdot \mathsf{Adv}^{\mathsf{ddh}}_{\mathbb{G}_{\mathsf{rnd}},D_2}(k) + \mathbf{O}(2^{-k})$$

for a suitable D_2. To close the hybrid argument, note that Games 6.i.3 and 6.(i+1) are identical.

In **Game** 7, we clear the $\mathbb{G}_{\mathsf{msg}}$-component of $\widetilde{Z}$ in all ciphertexts prepared for A. That is, instead of computing $\widetilde{Z} = \widehat{g} \cdot u^{\widetilde{r}} \cdot h^K$, resp. $\widetilde{Z} = \widehat{g} \cdot u^{\widetilde{r}} \cdot h^{K + [M]_{\mathbb{Z}}}$ for a freshly uniform $\widehat{g} \leftarrow \mathbb{G}_{\mathsf{rnd}}$, we set $\widetilde{Z} = \widehat{g} \cdot u^{\widetilde{r}}$. (We stress that we still compute $Z = \widehat{g}^{N^2}$.) Since all $\widetilde{Z}$ already have an independently uniform $\mathbb{G}_{\mathsf{rnd}}$-component, a straightforward reduction to the DCR assumption yields

$$\Pr\left[out_{6.q} = 1\right] - \Pr\left[out_7 = 1\right] \;=\; \mathsf{Adv}^{\mathsf{dcr}}_{\mathbb{Z}^*_{N^3},E}(k) + \mathbf{O}(2^{-k})$$

for a DCR distinguisher E. Note that because of the re-randomizability of DCR, there is no factor of $q(k)$, even though we substitute many group elements at once. However, since the precise order of $\mathbb{G}_{\mathsf{rnd}}$ is not known, this re-randomization costs us an error term of $\mathbf{O}(2^{-k})$.

In **Game** 8, we substitute the symmetric ciphertexts C_{E} in all KDM ciphertexts by encryptions of random messages. By our change in Game 7, we do not use the symmetric keys K used to produce C_{E} anywhere else. Thus, a reduction to the IND-CPA security of (E, D) gives

$$\Pr\left[out_7 = 1\right] - \Pr\left[out_8 = 1\right] \;=\; q(k) \cdot \mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{(\mathsf{E},\mathsf{D}),G}(k)$$

for an IND-CPA adversary G. Note that in Game 8, A's view is independent of the challenge bit b initially selected by the KDM challenger. Hence, we have $\Pr[out_8 = 1] = 1/2$. Taking things together yields the theorem.

Acknowledgements. I would like to thank Hoeteck Wee for pointing out to me the connection between hash proof systems and KDM security. His interpretation of the schemes of [10] and [11] was the starting point for this work. I am grateful to Tibor Jager for very useful feedback and discussions. I am indebted to Baodong Qin, who pointed out two flaws (along with very helpful suggestions on how to fix them). Finally, I am thankful to the Crypto 2012 referees who found an annoying cyclic dependency in the PKE construction, and to the Eurocrypt 2013 referees for helpful comments.

References

[1] Acar, T., Belenkiy, M., Bellare, M., Cash, D.: Cryptographic agility and its relation to circular encryption. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 403–422. Springer, Heidelberg (2010)

[2] Adão, P., Bana, G., Herzog, J., Scedrov, A.: Soundness of formal encryption in the presence of key-cycles. In: De Capitani di Vimercati, S., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 374–396. Springer, Heidelberg (2005)

[3] Applebaum, B.: Key-dependent message security: Generic amplification and completeness. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 527–546. Springer, Heidelberg (2011)

[4] Applebaum, B., Cash, D., Peikert, C., Sahai, A.: Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 595–618. Springer, Heidelberg (2009)

[5] Barak, B., Haitner, I., Hofheinz, D., Ishai, Y.: Bounded key-dependent message security. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 423–444. Springer, Heidelberg (2010)

[6] Bellare, M., Hofheinz, D., Yilek, S.: Possibility and impossibility results for encryption and commitment secure under selective opening. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 1–35. Springer, Heidelberg (2009)

[7] Black, J., Rogaway, P., Shrimpton, T.: Encryption-scheme security in the presence of key-dependent messages. In: Nyberg, K., Heys, H.M. (eds.) SAC 2002. LNCS, vol. 2595, pp. 62–75. Springer, Heidelberg (2003)

[8] Boneh, D., Shen, E., Waters, B.: Strongly unforgeable signatures based on computational Diffie-Hellman. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.) PKC 2006. LNCS, vol. 3958, pp. 229–240. Springer, Heidelberg (2006)

[9] Boneh, D., Canetti, R., Halevi, S., Katz, J.: Chosen-ciphertext security from identity-based encryption. SIAM Journal on Computing 36(5), 1301–1328 (2007)

[10] Boneh, D., Halevi, S., Hamburg, M., Ostrovsky, R.: Circular-secure encryption from decision diffie-hellman. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 108–125. Springer, Heidelberg (2008)

[11] Brakerski, Z., Goldwasser, S.: Circular and leakage resilient public-key encryption under subgroup indistinguishability - (or: Quadratic residuosity strikes back). In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 1–20. Springer, Heidelberg (2010)

[12] Brakerski, Z., Goldwasser, S., Kalai, Y.T.: Black-box circular-secure encryption beyond affine functions. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 201–218. Springer, Heidelberg (2011)

[13] Camenisch, J.L., Lysyanskaya, A.: An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 93–118. Springer, Heidelberg (2001)

[14] Camenisch, J., Chandran, N., Shoup, V.: A public key encryption scheme secure against key dependent chosen plaintext and adaptive chosen ciphertext attacks. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 351–368. Springer, Heidelberg (2009)

[15] Cash, D., Green, M., Hohenberger, S.: New definitions and separations for circular security. Cryptology ePrint Archive, Report 2010/144 (2010), `http://eprint.iacr.org/`

[16] Cramer, R., Shoup, V.: Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 45–64. Springer, Heidelberg (2002)

[17] Galindo, D., Herranz, J., Villar, J.: Identity-based encryption with master key-dependent message security and leakage-resilience. In: Foresti, S., Yung, M., Martinelli, F. (eds.) ESORICS 2012. LNCS, vol. 7459, pp. 627–642. Springer, Heidelberg (2012)

[18] Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 415–432. Springer, Heidelberg (2008)

[19] Haitner, I., Holenstein, T.: On the (im)Possibility of key dependent encryption. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 202–219. Springer, Heidelberg (2009)

[20] Hofheinz, D.: All-but-many lossy trapdoor functions. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 209–227. Springer, Heidelberg (2012)

[21] Hofheinz, D.: Circular chosen-ciphertext security with compact ciphertexts. Cryptology ePrint Archive, Report 2012/150 (2012), `http://eprint.iacr.org/`

[22] Hofheinz, D., Kiltz, E.: Secure hybrid encryption from weakened key encapsulation. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 553–571. Springer, Heidelberg (2007)

[23] Hofheinz, D., Unruh, D.: Towards key-dependent message security in the standard model. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 108–126. Springer, Heidelberg (2008)

[24] Kurosawa, K., Desmedt, Y.: A new paradigm of hybrid encryption scheme. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 426–442. Springer, Heidelberg (2004)

[25] Malkin, T., Teranishi, I., Yung, M.: Efficient circuit-size independent public key encryption with KDM security. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 507–526. Springer, Heidelberg (2011)

[26] Naor, M., Yung, M.: Public-key cryptosystems provably secure against chosen ciphertext attacks. In: 22nd ACM STOC. ACM Press (May 1990)

[27] Peikert, C., Waters, B.: Lossy trapdoor functions and their applications. In: Ladner, R.E., Dwork, C. (eds.) 40th ACM STOC, pp. 187–196. ACM Press (May 2008)

[28] Waters, B.: Efficient identity-based encryption without random oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)

MiniLEGO: Efficient Secure Two-Party Computation from General Assumptions

Tore Kasper Frederiksen, Thomas Pelle Jakobsen, Jesper Buus Nielsen, Peter Sebastian Nordholt, and Claudio Orlandi*

Department of Computer Science, Aarhus University
{jot2re,tpj,jbn,psn,orlandi}@cs.au.dk

Abstract. One of the main tools to construct secure two-party computation protocols are Yao garbled circuits. Using the cut-and-choose technique, one can get reasonably efficient Yao-based protocols with security against malicious adversaries. At TCC 2009, Nielsen and Orlandi [28] suggested to apply cut-and-choose at the gate level, while previously cut-and-choose was applied on the circuit as a whole. This idea allows for a speed up with practical significance (in the order of the logarithm of the size of the circuit) and has become known as the "LEGO" construction. Unfortunately the construction in [28] is based on a specific number-theoretic assumption and requires public-key operations per gate of the circuit. The main technical contribution of this work is a new XOR-homomorphic commitment scheme based on oblivious transfer, that we use to cope with the problem of connecting the gates in the LEGO construction. Our new protocol has the following advantages:

1. It maintains the efficiency of the LEGO cut-and-choose.
2. After a number of seed oblivious transfers linear in the security parameter, the construction uses only primitives from Minicrypt (i.e., private-key cryptography) per gate in the circuit (hence the name MiniLEGO).
3. MiniLEGO is compatible with all known optimization for Yao garbled gates (row reduction, free-XORs, point-and-permute).

1 Introduction

Secure two-party computation allows two parties to compute a function of their inputs while ensuring security properties such as the privacy of the inputs and the correctness of the outputs. The first protocol for secure two-party computation is Yao's garbled circuit [21, 32]. In recent years there has been a significant effort to bring secure computation into practice. These efforts resulted in terrific improvements in terms of algorithmic complexity, efficiency of implementations etc. (see e.g., [1, 2, 5, 7, 11–13, 15–20, 22–24, 27–31] and references therein). Perhaps the most interesting problem is

* Partially supported by the European Research Commission Starting Grant 279447 and the Danish National Research Foundation and The National Science Foundation of China (grant 61061130540) for the Sino-Danish Center for the Theory of Interactive Computation. Tore and Thomas are supported by Danish Council for Independent Research Starting Grant 10-081612. Claudio is supported by The Danish Council for Independent Research (DFF) Grant 11-116416/FTP.

how to achieve protocols with security against malicious adversaries that are efficient enough to be used in practice.

In a nutshell, Yao's protocol works as follows: A constructs an encrypted version of the circuit to be computed (the "garbled circuit") and sends it to B who evaluates the encrypted circuit on encrypted inputs, thus learning nothing but the output of the computation. One of the main problems of this protocol is that if A is malicious she can encrypt a circuit different than the one B agreed on computing, with dramatic consequences for the correctness of the result and the privacy of B's input. One of the main tools to cope with this is the so called *cut-and-choose* technique: A prepares many copies of the encrypted circuit and B checks some of them for correctness. This induces a probability on the unopened circuits to be correct. Nielsen and Orlandi [28] presented a twist on this approach, known as LEGO: their approach consists of performing a cut-and-choose test at the gate level (instead of at circuit level), and allows to save a factor $O(\log(s))$ with s being the circuit size, in the computation and communication complexity w.r.t., "standard" cut-and-choose at the circuit level. However, the approach did not have a practical impact for the efficiency of Yao-based protocols, for several reasons:

1. *LEGO uses public-key primitives for each gate in the circuit:* Each gate has associated three commitments to its input/output keys. Those commitments are used for the "soldering" and need to be homomorphic. For this purpose LEGO uses Pedersen commitments. This is a drawback for the efficiency of the protocol (group operations, even in an elliptic curve, are orders of magnitude slower than symmetric primitives such as hash functions or private-key encryption). Moreover, it limits LEGO to discrete logarithm computational assumptions.
2. *LEGO is not compatible with known optimization for Yao's protocol:* Keys in LEGO are element of $\mathbb{Z}_p$ for some prime p, while using binary strings $\{0,1\}^t$ is more natural and standard. Therefore, it is not possible to use the "free-XOR" trick with LEGO, nor many of the others optimizations that are tailored for bit-string keys.
3. *LEGO has too many bricks:* there are many different kind of objects in LEGO (key-filters, not-two gates, etc.) that make the use of LEGO complex to understand and implement.

Contributions. In this paper, we present a generalization and a simplification of the LEGO approach. The main technical difference is to replace the Pedersen commitments with some XOR-homomorphic commitments based on oblivious transfer (OT) which we believe is of independent interest and might be used in other applications. We take this direction as OT can be efficiently extended (both with passive [14] and active security [9,27]), the price is only a few private-key operations per OT (together with a small number of "real" seed OTs that use public-key technology to bootstrap the process). Doing so allows us to:

1. Maintain LEGO's good complexity and achieve statistical security 2^{-k} when the replication factor is only $\rho = O(k/\log(s))$ against a replication factor of $\rho = O(k)$ for standard cut-and-choose such as the one in [22]
2. Implement a variant on LEGO whose security only relies on generic, symmetric primitives (except for the few seed OTs needed to bootstrap the OT extension).

3. Achieve a variant of LEGO that uses "standard" garbled gates (ANDs and free XORs), compatible with garbled gates optimization.

Whether our proposed protocol will be more efficient in practice than protocols with standard cut-and-choose [7, 18, 20, 29, 31] will only be decided by performing a serious comparison of similar implementations running on the same hardware-network configuration of our and other approaches. This is on-going work.

Technical Overview. The main idea of the protocol we present here is the same as in [28]: A prepares many garbled gates (NANDs in [28], while here we use ANDs) together with commitments to the input and output garbled keys. If A prepares a gate dishonestly we view it as a faulty gate, i.e., one that does not give the correct output on some inputs. B asks A to open a random subset of the AND gates and checks them for correctness. If the check goes through, B randomly permutes the unopened gates into buckets representing a redundant AND gate. He solders the gates within a given bucket together and then solders the buckets together to form a circuit that computes the function even in the presence of a minority of faulty gates within each bucket. As part of this soldering NOT gates can be injected thus the garbled AND gates and the soldering alone provides a universal set of Boolean gates.

As the gates have been generated independently, the output keys of the gates in one layer of the circuit cannot be directly fed as input to the next layer. Therefore, A reveals the XOR difference between the output keys in the first layer with the corresponding input keys in the second layer (using the XOR-homomorphic properties of the commitment scheme). This allows B to "align" the input keys of the gates in one layer with the output keys of the gates in the previous layer. He then evaluates all ρ garbled gates in a bucket on the same input key and take the output of the bucket to be any output key agreed upon by more than $\lfloor \rho/2 \rfloor$ of the replicated garbled gates it contains. The main intuition for the security of LEGO cut-and-choose is as follows: If A had sent B k faulty gates, B would detect this with probability $1 - 2^{-k}$. Therefore, if B accepts the test, with very high probability there are only a few faulty gates among the unopened ones. As all gates are permuted at random and placed in random buckets in the circuit, only very little redundancy is needed to correct for all faulty gates.

Because we use XOR homomorphic commitments, our construction can be instantiated with essentially any free-XOR compatible garbled gate scheme and is compatible with various state of the art optimizations (such as free-XOR, row-reduction, point-and-permute).

Organization. We start with preliminaries and background in Section 2. We then continue to go through the overall description of the secure-two party computation protocol in Section 3. This is followed by Section 4 where we describe the main technical contribution of this paper; the XOR homomorphic commitments.

2 Background

In this section we formalize our goal in the UC framework (refer to textbooks such as [8, 10] for definitions). We furthermore list the basic building blocks of our protocol and quickly review their individual constructions.

The Ideal Functionality. In Fig. 1 the ideal functionality for secure function evaluation is presented (taken almost verbatim from [28]). Note that the functionality is insecure in the sense that A can try to guess B's input bits, but if her guess is wrong B is told that A is cheating. This models a standard problem in Yao based protocols known as "selective failure attack", that can be solved by modifying the circuit to be evaluated. For instance, to evaluate a circuit $\mathcal{C}\left((a_i)_{i\in[\ell]},(b_i)_{i\in[\ell]}\right)$ securely one could instead evaluate $\mathcal{C}'\left((a_i)_{i\in[\ell]},(b_{i,j})_{i\in[\ell],j\in[k]}\right) = \mathcal{C}\left((a_i)_{i\in[\ell]},(\oplus_{i\in[k]}b_{i,j})_{j\in[\ell]}\right)$ i.e., B encodes his real input bit in the parity of a k bit-long string, and the modified circuit first reconstructs the real input and then evaluates the original circuit. Now, in order to guess one of B's real input bits A needs to guess correctly the k random bits, so she will fail with probability $1-2^{-k}$. As our construction allows to compute XOR gates for free, this only marginally increases the complexity. Better encodings can be used (See [20]) to reduce the size of the encoded input from $\ell \cdot k$ bits to $\max(4\ell, 8k)$ bits.

Circuit and inputs: On input $(\texttt{init}, A, k)$ from A and input $(\texttt{init}, B, k)$ from B where $A = (\mathbf{a}, \mathcal{C}_A)$, $B = (\mathbf{b}, \mathcal{C}_B)$ proceed as follows:

1. Let k be a statistical security parameter and let $\mathcal{C}_A$ and $\mathcal{C}_B$ be descriptions of Boolean circuits consisting of NOT, XOR and AND gates computing the corresponding Boolean functions f_A, respectively f_B.
2. Leak $\mathcal{C}_A$, $\mathcal{C}_B$ and k to the adversary.
3. If $\mathcal{C}_A \neq \mathcal{C}_B$, then the ideal functionality outputs `disagreement!` to both parties and terminates. Otherwise, let $\mathcal{C} = \mathcal{C}_A$ and parse $\mathcal{C}$ as $(\ell, \mathcal{C}')$, where $\ell \in \mathbb{N}$ and $\mathcal{C}'$ is a circuit with 2ℓ input wires and ℓ output wires. I.e., we potentially add blank wires to make sure that the size of A's input, B's input and the output are the same. Thus $\mathcal{C}'$ computes the Boolean function $f = f_A$.
4. Finally parse $\mathbf{a}$ as $\mathbf{a} \in \{0,1\}^\ell$ and $\mathbf{b} \in \{0,1\}^\ell$ and return $(\ell, \mathcal{C}')$ to both A and B.

Corrupt A: On input $(\texttt{corrupt})$ from A, let her be corrupt. She can then specify a set $\{(i,\beta_i)\}_{i\in I}$, where $I \subseteq \{1,\ldots,k\}$ and $\beta_i \in \{0,1\}$. If $\beta_i = b_i$ for $i \in I$, then output `correct!` to A. Otherwise, output `You were nicked!` to A and output `Alice cheats!` to B.

Evaluation: If both parties are honest or A was not caught above, then on input $(\texttt{evaluate})$ from both A and B the ideal functionality computes $z = f(a,b)$ and outputs z to A. The adversary decides the time of delivery.

Fig. 1. The ideal functionality, $\mathcal{F}_{\mathsf{SFE}}$, for secure function evaluation for two parties

Building Blocks. We here review the main building blocks of our protocol.

Generic Free-XOR Yao Gate. Our protocol is compatible with every "free-XOR compatible" garbling schemes. In particular, it is possible to use very optimized garbling schemes. We now describe such a garbling scheme that combines the state of the art optimizations for Yao Gates i.e., free XOR [17], permutation bits [26], garbled row-reduction [26] in the same way as [1].

In particular this means that to garble a gate 4 evaluations of AES are needed, and a garbled gate consists of only 3 ciphertexts (therefore saving on communication complexity). The evaluation of the gate consists of a single AES evaluation.

- We have a (possibly randomized) algorithm $\mathsf{Yao}(L_0, R_0, \Delta, id)$ with id a unique gate identifier, a left input *zero-key* $L_0 \in \{0,1\}^t$, a right input zero-key $R_0 \in \{0,1\}^t$ and a global difference $\Delta \in \{0,1\}^t$ outputs a garbled gate gg and a output zero-key $O_0 \in \{0,1\}^t$.
- We have a (possibly randomized) algorithm $\mathsf{Eval}(gg, L', R')$ that on input a garbled gate gg, a left key $L' \in \{0,1\}^t$ and a right key $R' \in \{0,1\}^t$ outputs an output key $O' \in \{0,1\}^t \cup \{\perp\}$.
- We define the *one-keys* L_1,R_1,O_1 s.t. $L_0 \oplus L_1 = R_0 \oplus R_1 = O_0 \oplus O_1 = \Delta$.

The idea is that a garbled AND gate gg has a zero- and a one-key associated with each of its wires (left input, right input and output wire), and that these keys represent the bit values on those wires. E.g., if gg is a garbled AND gate generated as $(gg, O_0) \leftarrow \mathsf{Yao}(L_0, R_0, \Delta, id)$ then $\mathsf{Eval}(gg, L_a, R_b)$ for any $a, b \in \{0,1\}$ should output $O_{a \wedge b}$.

Note that if A samples Δ and a zero-key, say L_0, at random and give the key L_a to B then there is no way for B to infer the bit a from L_a. Furthermore, even if B learns a he cannot guess the key L_{1-a} with better probability than guessing Δ. For a garbling scheme to be secure we want that even if B learns gg and keys L_a and R_b for $a, b \in \{0,1\}$, and is able to evaluate $O_{a\wedge b} \leftarrow \mathsf{Eval}(gg, L_a, R_b)$, then he cannot guess L_{1-a}, R_{1-b} or $O_{1-a\wedge b}$ with better probability then guessing the random string Δ, even if he knows a and/or b.

Thus B can evaluate the garbled gate gg without knowing anymore about the output than he can infer from his knowledge of a and b. Furthermore, B cannot evaluate the gate on any other inputs. Thus if B sends back $O_{a\wedge b}$ to A, A can learn $a \wedge b$ (as she knows O_0 and Δ) and be confident that this is the correct result. We formalize this intuition about correctness and security of a garbled gate in Def. 1.

Definition 1. *We say that* (Yao, Eval) *is a* Yao free-XOR garbling scheme *if the following holds:*

Correctness: *Let* $(gg, O_0) \leftarrow \mathsf{Yao}(L_0, R_0, \Delta, id)$*. Then for all* $a, b \in \{0,1\}$*,* $\mathsf{Eval}(gg, L_a, R_b) = O_{a\wedge b}$*, with overwhelming probability over the choices of* L_0*,* R_0*,* Δ *and the random coins of* Yao *and* Eval.

Secrecy: *Consider the following indistinguishability under chosen input attack game for a stateful adversary* $\mathcal{A}$*: The adversary outputs two pairs of bit vectors* $\left(a_0^i, b_0^i\right)_{i\in[k]}, \left(a_1^i, b_1^i\right)_{i\in[k]} \in \{0,1\}^{2k}$*. The game picks a uniformly random challenge* $c \in_{\mathrm{R}} \{0,1\}$*, samples* $\Delta \in_{\mathrm{R}} \{0,1\}^t$ *and for* $i = 1, \ldots, k$ *it samples* $L^i, R^i \in_{\mathrm{R}} \{0,1\}^t$*, samples* $(gg^i, O_0^i) \leftarrow \mathsf{Yao}\left(L_0^i, R_0^i, \Delta, i\right)$ *and then inputs* $\left(gg^i, L^i_{a^i_c}, R^i_{b^i_c}\right)_{i\in[k]}$ *to* $\mathcal{A}$*. Finally* $\mathcal{A}$ *outputs a bit* $d \in \{0,1\}$ *and wins if* $d = c$*. We say that the scheme is IND-CIA if for all PPT* $\mathcal{A}$*,* $\mathcal{A}$ *wins the* IND-CIA *game with at most negligible advantage in* t*.*

In the full version [6] we describe an optimized garbling scheme that can be used with our protocol. See also [1].

Soldering. The idea for this component is the same as in [28], however, slightly changed to support a general gate garbling scheme. When a garbled gate gg^1 has the same zero-key (and therefore also one-key) associated to one of its wires, as is associated with one of gg^2's wires, we say that the given wire of gg^1 is *soldered* to the given wire of gg^2. This is a useful concept when we want to build circuits of garbled gates. To see this consider a garbled gate gg^1 with its left input wire soldered to the output of gg^2, and its right input wire soldered to the output of gg^3. This means that if gg^2 and gg^3 has output zero-keys O_0^2 and O_0^3 respectively, then gg^1 has left and right zero-keys $L_0^1 = O_0^2$ and $R_0^1 = O_0^3$. Thus if we evaluate gg^2 and gg^3 on some input and obtain output keys O_a^2 and O_b^3 we can use this to further evaluate gg^1 on these outputs. The resulting output would be some output key $O_{a \wedge b}^1$.

Alternatively notice that if gg^1 has, e.g., left, input zero-key $L_0^1 = O_0^2 \oplus O_0^3$ then

$$O_a^2 \oplus O_b^3 = O_0^2 \oplus O_0^3 \oplus (a \oplus b)\Delta = L_0^1 \oplus (a \oplus b)\Delta = L_{a \oplus b}^1 \ .$$

In this case we say that the left input wire of gg^1 is soldered to the XOR of the output of gg^2 and gg^3. This is because by XOR'ing the outputs keys of gg^2 and gg^3 we get the left input key of gg^1 corresponding to XOR of the outputs of gg^2 and gg^3. This is also why we call the garbling *free-XOR*: we do not need to garble XOR gates, since this is handled by the soldering.

In our protocol we will first generate garbled gates where all zero-keys are picked independently, and then in a later stage we will *solder* the wires of the garbled gates to each other to form a garbled circuit. For this purpose, we introduce a function $\mathsf{Shift}\left(gg, L^d, R^d, O^d\right)$ that on input a garbled gate $(gg, O_0) \leftarrow \mathsf{Yao}\left(L_0, R_0, \Delta, id\right)$, and three *differences* $L^d, R^d, O^d \in \{0,1\}^t$, outputs a new *shifted gate sgg*. The shifted gate sgg is the gate gg modified to have have input zero-keys $\left(L_0 \oplus L^d\right)$ and $\left(R_0 \oplus R^d\right)$ and output zero-key $\left(O_0 \oplus O^d\right)$.

This can be implemented by letting Shift output the concatenation of its inputs i.e., $sgg = \left(gg, L^d, R^d, O^d\right)$ and let the evaluation of a shifted gate sgg be defined by:

$$\mathsf{ShiftEval}\left(sgg, \hat{L}, \hat{R}, \hat{O}\right) = \mathsf{Eval}\left(gg, \hat{L} \oplus L^d, \hat{R} \oplus R^d\right) \oplus O^d$$

where for all K we define $\perp \oplus K = \perp$. It is clear that a shifted gate is correct (with respect to the shifted zero-keys) if and only if a standard gate is correct, and clearly shifting a gate does not threaten its security property. A shifted gate can be shifted again: The Shift function will just update the values L^d, R^d, O^d accordingly.

Consider two garbled gates $\left(gg^1, O_0^1\right) \leftarrow \mathsf{Yao}\left(L_0^1, R_0^1, \Delta, 1\right)$ and $\left(gg^2, O_0^2\right) \leftarrow \mathsf{Yao}\left(L_0^2, R_0^2, \Delta, 2\right)$. The shifted gate $sgg^2 = \mathsf{Shift}\left(gg^2, \left(O_0^1 \oplus L_0^2\right), 0, 0\right)$ then becomes a garbled gate with left zero-key $L_0^2 \oplus \left(O_0^1 \oplus L_0^2\right) = O_0^1$. I.e. the output wire of gg^1 is now soldered to the left input wire of sgg^2.

Similarly we could have used the Shift function to solder the input of sgg^2 to the XOR of some other garbled gates.

If one wish to use NOT gates then these can be implemented as part of this shifting by a simply change in the the difference, i.e., to add a NOT gate to the soldering to the left wire of a gate we simply use $\neg L^d = L^d \oplus \Delta$ instead of just L^d.

To see this assume we want to put a NOT into the soldering between gg^1 and gg^2. In this case we would have $\neg {L^1}^d = {L^1}^d \oplus \Delta = O_0^1 \oplus L_0^2 \oplus \Delta$, i.e., $sgg^2 = \mathsf{Shift}\left(gg^2, \left(O_0^1 \oplus L_0^2 \oplus \Delta\right), 0, 0\right)$. Thus when the evaluator does

$$\mathsf{ShiftEval}\left(sgg^2, L^2, 0, 0\right) = \mathsf{Eval}\left(gg^2, L_a^2 \oplus (O_0^1 \oplus L_0^2 \oplus \Delta), R^2\right) \ .$$

If $a = 0$ we get the left input key for gg^2 is $L_0^2 \oplus (O_0^1 \oplus L_0^2 \oplus \Delta) = O_0^1 \oplus \Delta = O_1^1$ and similarly for $a = 1$ we get $L_1^2 \oplus (O_0^1 \oplus L_0^2 \oplus \Delta) = (L_0^2 \oplus \Delta) \oplus O_0^1 \oplus (L_0^2 \oplus \Delta) = O_0^1$. Thus we clearly see that the bit represented by the left input key (along with the key itself) for gg^2 has been flipped.

Initialization

On input $(\texttt{init}, ID, W)$ from the adversary, with $|ID| = \mu$, $|W| \leq \kappa$ and $W \subset ID$, output ID to both A and B and let $\mathcal{J} = \emptyset$. If A is honest, then $W = \emptyset$.

Commit

On input $(\texttt{commit}, j \in ID, m_j)$ with $m_j \in \{0,1\}^h$ from A, and where no value of the form $(j, \cdot)$ is stored, store (j, m_j). If $j \in ID \setminus W$, add $J = \{j\}$ to $\mathcal{J}$ and associate with J the equation $\mathtt{X}_j = m_j$. Then output $(\texttt{commit}, j)$ to B.

Open

On input $(\texttt{open}, J \subset ID)$ from A, where for all $j \in J$ a pair (j, m_j) is stored do the following:

- If A is honest, output $(\texttt{open}, J, \oplus_{j \in J} m_j)$ to B.
- If A is corrupted wait for A to input $(\texttt{corrupt-open}, J, m_J)$. Then add J to $\mathcal{J}$, associate the equation $\oplus_{j\in J}\mathtt{X}_j = m_J$ to J, and check that the equation system $\{\oplus_{j\in J}\mathtt{X}_j = m_J\}_{J \in \mathcal{J}}$ has a solution. If so, output $(\texttt{open}, J, m_J)$ to B. Otherwise, output `Alice cheats` to B and terminate.

Oblivious Opening

On input $(\texttt{OT-choose}, \mathit{otid}, b)$ with $b \in \{0,1\}$ from B output $(\texttt{OT-choose}, \mathit{otid})$ to A. On input $(\texttt{OT-open}, \mathit{otid}, J_0, J_1)$ from A with $J_0, J_1 \subset ID$ where for all $j \in J_0, J_1$ a pair (j, m_j) is stored and $(\texttt{OT-choose}, \mathit{otid}, *)$ was input before by B do the following:

- If A is honest, output $(\texttt{OT-open}, \mathit{otid}, J_b, \oplus_{j\in J_b} m_j)$ to B (Note that B does not learn the set of ids J_{1-b}).
- If A is corrupted, wait for A to input $(\texttt{guess}, g)$ with $g \in \{0, 1, \perp\}$. If $g \in \{0,1\}$ and $g \neq b$ output `Alice cheats` to B and terminate. Otherwise, proceed to wait for A to input $(\texttt{corrupt-open}, J_0, J_1, m_{J_0}, m_{J_1})$. Add J_b to $\mathcal{J}$ and associate the equation $\oplus_{j\in J_b}\mathtt{X}_j = m_{J_b}$ to J_b. Check that the equation system still has a solution as described above. If so, output $(\texttt{OT-open}, J_b, m_{J_b})$ to B. Otherwise output `Alice cheats` to B.

OR Open

For up to ω Or-Openings, that must all occur before the first Oblivious-Opening, do the following: On input $(\texttt{OR-open}, J_0, J_1, a)$ from A, with $J_0, J_1 \subset ID, a \in \{0,1\}$ where for all $j \in J_0, J_1$ a pair (j, m_j) is stored do the following:

- If A is honest, output $(\texttt{OR-open}, J_0, J_1, \oplus_{j\in J_a} m_j)$ to B.
- If A is corrupted, and if $J_a \cap W \neq \emptyset$, wait for corrupt A to input $(\texttt{corrupt-open}, J_a, m_{J_a})$, add J_a to $\mathcal{J}$ and associate $\oplus_{j\in J_a}\mathtt{X}_j = m_{J_a}$ to J_a. Check if the equation system still has a solution as described above. If so, output $(\texttt{OR-open}, J_0, J_1, m_{J_a})$ to B. Otherwise output `Alice cheats` to B.

before the first Oblivious-Opening.

Fig. 2. The ideal functionality, $\mathcal{F}_{\mathsf{COM}}$, for the commitment scheme used by π_{LEGO}

Homomorphic Commitments. To securely implement the soldering described above, we cannot simply have (potentially malicious) A send the differences needed to shift the gates. Instead we will have A give homomorphic commitments to all zero-keys of each gate, and then have her open the differences of the committed keys. Therefore we need a homomorphic commitment scheme. In Fig. 2 we state the ideal functionality $\mathcal{F}_{\mathsf{COM}}$ for the homomorphic commitments that we implement in Section 4. As we are now going to use this functionality to implement $\mathcal{F}_{\mathsf{SFE}}$ we will briefly recap the features of this functionality.

The functionality allows A to commit to messages and to later reveal those messages. In addition the functionality allows to reveal the XOR of two or more committed messages to B (without revealing any extra information).

The functionality is "insecure", in the sense that A can choose a set of up to κ wildcard commitments where she can change her mind about the committed value at opening time. However, openings need to be consistent. More specifically, the $\mathcal{F}_{\mathsf{COM}}$ functionality stores a system of linear equations. Initially these equations simply specify that non-wildcard commitments must be opened to the value, they were commitments to. Every time A performs an opening involving wildcard commitments this defines a new linear equation, which is stored in the ideal functionality. For an opening of a wildcard commitment to be successful the set of linear equations stored in the ideal functionality must be consistent.

If the set of equations stored in the ideal functionality restricts the opening of a commitment in such a way that it can only be opened to *one* value, we say that the commitment is *fixed* to that value. Note, that all non-wildcard commitments are fixed, and a fixed wildcard commitment can essentially be viewed as a non-wildcard commitment.

As we are treating the commitments as an ideal functionality, we need to push into the ideal functionalities two extra commands (in a way similar to the commit-and-prove functionality in [3])): apart from the regular openings the functionality allows to open (the XOR of) committed messages in two alternative ways: In an *Oblivious-Opening*, B can choose between two sets of committed messages and learn the XOR of the messages in one of them. In an *Or-Opening* we allow A to open the XOR of one out of two sets of committed messages without revealing which one. For technical reasons there can only be a total of ω Or-Openings and all Or-Openings must be done before the first Oblivious-Opening. Also, note that there is a build-in selective failure attack in the Oblivious-Opening. However, this is not a problem as we will only use this type of opening to handle B's input where, as discussed above, the $\mathcal{F}_{\mathsf{SFE}}$ functionality already allows a selective failure attack.

Commitment from B to A. Additional to the $\mathcal{F}_{\mathsf{COM}}$ functionality we are going to use an extractable commitment Com. This commitment is used only once by B to commit to his challenge in the cut-and-choose phase and extraction is needed for simulation (to avoid selective opening issues). Since this commitment does not need to be homomorphic it can be easily implemented in the $\mathcal{F}_{\mathrm{OT}}$-hybrid model.

3 The MiniLEGO Protocol

We now show how to use the ingredients outlined in the previous section in order to construct the MiniLEGO protocol.

We denote by $\mathcal{C}'$ the Boolean circuit to be evaluated. We assume $\mathcal{C}'$ to be composed of NOT, XOR and AND gates. The XOR gates are allowed to have unbounded fan-in while the AND gates have fan-in 2. With each AND gate in $\mathcal{C}'$ we associate a unique label and we let gates be the set of all these labels. A subset $\mathsf{inputGates} \subset \mathsf{gates}$ of size 2ℓ are specially marked as input gates. The AND gates in $\mathsf{inputGates}$ should be given the same bit on both input wires, so that the gate simply computes the identity function. A subset in $\mathsf{Ainputs} \subset \mathsf{inputGates}$ of size ℓ are taken to be A's inputs. The remaining ℓ gates in $\mathsf{Binputs} = \mathsf{inputGates} \setminus \mathsf{Ainputs}$ are B's inputs (for convenience assume that $\mathsf{Binputs} = [\ell]$). A has input bits $(a_1, \ldots, a_\ell)$, while B has input bits $(b_1, \ldots, b_\ell)$.

A subset $\mathsf{outputGates} \subset \mathsf{gates}$ of size ℓ are marked as output gates. The output of these gates are taken to be the output of the circuit. Note that this means that all output gates are AND gates. This is without loss of generality: a circuit with XOR gates as output gates can be modified to an equivalent circuit with AND gates as output gates by adding at most ℓ AND gates. The ℓ output bits are denoted $(z_1, \ldots, z_\ell)$.

The wiring of the circuit $\mathcal{C}'$ is described by two functions $\mathsf{lp}, \mathsf{rp} : \mathsf{gates} \setminus \mathsf{inputGates} \to 2^{\mathsf{gates} \cup \{\mathbb{1}\}}$. We call $\mathsf{lp}(j)$ the left parents of j (resp. $\mathsf{rp}(j)$ the right parents of j), and take the left (resp. right) input of j to be the XOR of the output bits of all gates in $\mathsf{lp}(j)$ (resp. $\mathsf{rp}(j)$). Thus, the XOR gates of $\mathcal{C}'$ are implicitly defined by the lp and rp functions. If the special symbol $\mathbb{1}$ is included in the set returned by lp, rp, this means that a NOT gate is inserted between gate j and its parent gate (i.e., the input is XORed with the constant 1). We assume that $\max(\mathsf{lp}(j) \cup \mathsf{rp}(j)) < j$ for all j.

Garbled Circuit. Let $\Gamma = 2\rho s$ for $s = |\mathsf{gates}|$ and some replication factor $\rho \in \mathbb{N}$. For our protocol A will construct Γ garbled gates. She constructs twice as many garbled gates as is needed to build the garbled circuit, because half the gates are going to be checked during the cut-and-choose phase. We choose to check exactly half for the sake of presentation but, as in [31], this could be changed to any fraction in order to optimize concrete efficiency.

Bucket Notation. In the protocol individual garbled gates are combined together in "buckets" of gates. We introduce here some convenient notation that allow us to address the gates in a bucket, the bucket of a gate etc. Let $\mathcal{B}$ be the family of ρ-to-1, ρ-wise independent functions from a set $U \subset [\Gamma]$ of size ρs to gates. For a function $\mathsf{BucketOf} \in \mathcal{B}$ let Bucket be the function that, for all $j \in \mathsf{gates}$ outputs the set $\{i \in U | \mathsf{BucketOf}(i) = j\}$. Let $\mathsf{BucketHead}(j)$ be the function that returns the "first" (in lexicographic order) element of $\mathsf{Bucket}(j)$.

There are $\Gamma' = 3\Gamma + 1$ keys in the protocol, because every constructed AND gate has a left, right and output key and in addition there is a global difference Δ. The key index is written as a superscript while subscripts are in $\{0, 1\}$ and describe the value carried by the key i.e., $K^i_b = K^i \oplus (b\Delta)$. Let id be a function that on input a key $K^j_0 \in \{0, 1\}^t$ returns a unique label for that key. We will sometimes abuse notation and write $\mathsf{id}\left(K^j_1\right)$ to denote the set $\left\{\mathsf{id}\left(K^j_0\right), \mathsf{id}\left(\Delta\right)\right\}$. This will simplify the notation when using the $\mathcal{F}_{\mathsf{COM}}$ functionality.

Protocol Specification. The protocol π_{LEGO} in Fig.s 3 and 4 progresses in six phases: **Setup**, **Garbling**, **Cut-and-choose**, **Soldering**, **Input** and **Evaluation**. Here we describe these phases one by one.

During **Setup**, A or B initialize the $\mathcal{F}_{\mathsf{COM}}$ functionality by calling $(\texttt{init}, ID, W)$ with $ID = \Gamma'$ and $|W| \leq k$. For the remainder of the protocol if $\mathcal{F}_{\mathsf{COM}}$ outputs `Alice cheats` , B will abort the protocol. Then A samples the global difference Δ and commits to it using the `commit` command in $\mathcal{F}_{\mathsf{COM}}$. B samples his challenge for the cut-and-choose phase and the $\mathsf{BucketOf}$ function as described above, and commits to both using the extractable commitment Com. B also "commits" to his input using the `OT-choose` command of the $\mathcal{F}_{\mathsf{COM}}$ functionality. These commitments of B's are needed to avoid selective opening issues in the cut-and-choose phase and reduce the security of the protocol to the $\mathsf{IND\text{-}CIA}$ game.

In **Garbling**, A constructs the candidate garbled gates $(gg^i)_{i\in[\Gamma]}$ and commits to the input/output zero-keys of each garbled gate using $\mathcal{F}_{\mathsf{COM}}$.

In **Cut-and-choose**, B reveals his challenge. The challenge consists of a set of indices $T \subset [\Gamma]$ of size ρs and a sequence of bits $(u_i, v_i)_{i\in T}$, indicating that B wants to test garbled gate gg^i on input (u_i, v_i). A opens the corresponding input and output keys for the test gates, allowing B to check for correctness. Note that B only tests one set of inputs for each gate – otherwise he will learn Δ.

In the remainder of the protocol the garbled gates that are not checked in **Cut-and-choose**, those with indices in $U = [\Gamma] \setminus T$, are used to build a garbled circuit according to the following fault tolerant circuit design: With each gate $j \in \mathsf{gates}$ we associate a *bucket* of ρ AND gates. To evaluate gate j we will evaluate each gate in the bucket of j on the inputs given to j. If more than $\lfloor \rho/2 \rfloor$ of the gates in the bucket agree on their output bit, we take this bit to be the output of j (otherwise the output is $\perp$). Clearly if there are more than $\lfloor \rho/2 \rfloor$ non-faulty gates in each bucket the output is correct.

To build such a garbled circuit the gates that were not checked $(gg^i)_{i\in U}$ are put into buckets using the $\mathsf{BucketOf}$ function. Then B uses the Shift function as described in Section 2 to solder the wires of the garbled gates. Since A may be malicious we cannot simply have her sent the XOR's of zero-keys that B needs for soldering. Instead A reveals the XOR's by opening the corresponding commitments to the zero-keys.

The garbled circuit is constructed in **Soldering** in three different soldering steps: For all $j \in \mathsf{gates}$ **Horizontal Soldering** solders all wires of all gates in $(gg^i)_{i\in\mathsf{Bucket}(j)}$ to the corresponding wires of $gg^{\mathsf{BucketHead}(j)}$. This allows to evaluate all the gates in the same bucket on the same input keys and get the same output keys. I.e., if A is honest, after the horizontal soldering all the gates in one bucket have exactly the same keys. For all $j \in \mathsf{gates}$ **Vertical Soldering** solders the left input wire of $gg^{\mathsf{BucketHead}(j)}$ to the XOR of the output wires of $\left(gg^{\mathsf{BucketHead}(i)}\right)_{i\in\mathsf{lp}(j)}$, and the right input wire of $gg^{\mathsf{BucketHead}(j)}$ to the XOR of the output wires of $\left(gg^{\mathsf{BucketHead}(i)}\right)_{i\in\mathsf{rp}(j)}$ (and we use the convention $O^{\mathbb{1}} = \Delta$ to deal with NOT gates – note that Δ can be seen as the 1 key of a special wire with zero-key equal to 0^t). Note that since **Horizontal Soldering** made all garbled gates in a bucket have the same input keys, this essentially means soldering *all* the gates in the bucket to the output wires of gates in $(\mathsf{Bucket}(i))_{i\in\mathsf{lp}(j)}$ and $(\mathsf{Bucket}(i))_{i\in\mathsf{rp}(j)}$. I.e., vertical soldering is "functional", in the sense that it ensures that the garbled circuit computes the right circuit, $\mathcal{C}'$. For all $j \in \mathsf{inputGates}$ **Input Soldering** simply solders the left and right input wire of garbled gates in $\mathsf{Bucket}(j)$ to each other, i.e., the gates in $\mathsf{inputGates}$ compute the identity function.

Setup Choose $\rho = O(k/\log(s))$ and $\Gamma = 2\rho s$ where $s = |\mathsf{gates}|$ in $\mathcal{C}'$. Let $\Gamma' = 3\Gamma + 1$ and proceed as follows:

1. A and B initialize a $\mathcal{F}_{\mathsf{COM}}$ functionality by having either of them call $(\mathtt{init}, ID, W)$ with $|ID| = \Gamma'$ and $|W| \leq k$.
2. A samples $\Delta \in_R \{0,1\}^t$ and inputs $(\mathtt{commit}, \mathsf{id}(\Delta), \Delta)$ to $\mathcal{F}_{\mathsf{COM}}$.
3. B samples a random $T \subset [\Gamma]$ of size ρs, and for all $i \in T$ samples $u_i, v_i \in_R \{0,1\}$. Let $U = [\Gamma] \setminus T$.
4. B samples $\mathsf{BucketOf} \in \mathcal{B}$ as described in Section 2.
5. B sends $C_T = \mathsf{Com}(T, (u_i, v_i)_{i \in T}, \mathsf{BucketOf}, r_T)$ to A.
6. For each $j \in \mathsf{Binputs}$, B inputs $(\mathtt{OT\text{-}choose}, j, b_j)$ to $\mathcal{F}_{\mathsf{COM}}$.

Garbling

1. For all $i \in [\Gamma]$, A samples $L_0^i, R_0^i \in_R \{0,1\}^t$, computes $(gg^i, O_0^i) \leftarrow \mathsf{Yao}(L_0^i, R_0^i, \Delta, i)$ and sends $GG = (gg^i)_{i \in [\Gamma]}$ to B.
2. A inputs $(\mathtt{commit}, \mathsf{id}(L_0^i), L_0^i)$, $(\mathtt{commit}, \mathsf{id}(R_0^i), R_0^i)$ and $(\mathtt{commit}, \mathsf{id}(O_0^i), O_0^i)$ to $\mathcal{F}_{\mathsf{COM}}$.

Cut-and-choose

1. B sends T, $(u_i, v_i)_{i \in T}$, $\mathsf{BucketOf}$ and randomness r_T to A.
2. If this is not a valid opening of C_T A aborts. Otherwise, for all $i \in T$ A inputs to $\mathcal{F}_{\mathsf{COM}}$ $(\mathtt{open}, \mathsf{id}(L_{u_i}^i))$, $(\mathtt{open}, \mathsf{id}(R_{v_i}^i))$, $(\mathtt{open}, \mathsf{id}(O_{u_i \wedge v_i}^i))$. Let $\hat{L}^i, \hat{R}^i, \hat{O}^i$ be the values output to B by $\mathcal{F}_{\mathsf{COM}}$.
3. B aborts if there is an $i \in T$ so that $\hat{O}^i \neq \mathsf{Eval}(gg^i, \hat{L}^i, \hat{R}^i)$.

Soldering

1. **Horizontal Soldering:** For all $j \in \mathsf{gates}$, let $h = \mathsf{BucketHead}(j)$: For all $i \neq h \in \mathsf{Bucket}(j)$ A inputs $(\mathtt{open}, \{\mathsf{id}(L^h), \mathsf{id}(L^i)\})$, $(\mathtt{open}, \{\mathsf{id}(R^h), \mathsf{id}(R^i)\})$, and $(\mathtt{open}, \{\mathsf{id}(O^h), \mathsf{id}(O^i)\})$ to $\mathcal{F}_{\mathsf{COM}}$. Let $L^{i^d}, R^{i^d}, O^{i^d}$ be the keys output to B from $\mathcal{F}_{\mathsf{COM}}$ and $sgg^i = \mathsf{Shift}(gg^i, L^{i^d}, R^{i^d}, O^{i^d})$.
2. **Vertical Soldering:** For all $j \in \mathsf{gates} \setminus \mathsf{inputGates}$, let $h = \mathsf{BucketHead}(j)$: A inputs $\left(\mathtt{open}, \{\mathsf{id}(L^h)\} \cup \left\{\mathsf{id}\left(O^{\mathsf{BucketHead}(i)}\right)\right\}_{i \in \mathsf{lp}(j)}\right)$ and $\left(\mathtt{open}, \{\mathsf{id}(R^h)\} \cup \left\{\mathsf{id}\left(O^{\mathsf{BucketHead}(i)}\right)\right\}_{i \in \mathsf{rp}(j)}\right)$ to $\mathcal{F}_{\mathsf{COM}}$. Let L^{h^d}, R^{h^d} be the keys output to B by $\mathcal{F}_{\mathsf{COM}}$ and $sgg^h = \mathsf{Shift}(gg^h, L^{h^d}, R^{h^d}, 0^t)$.
3. **Input Soldering:** For all $j \in \mathsf{inputGates}$, let $h = \mathsf{BucketHead}(j)$: A inputs $(\mathtt{open}, \{\mathsf{id}(L^h), \mathsf{id}(R^h)\})$ to $\mathcal{F}_{\mathsf{COM}}$. Let R^{h^d} be the key output to B by $\mathcal{F}_{\mathsf{COM}}$ and $sgg^h = \mathsf{Shift}(sgg^h, 0^t, R^{h^d}, 0^t)$.

Fig. 3. The Protocol π_{LEGO} implementing $\mathcal{F}_{\mathsf{SFE}}$ (Part 1)

In **Input**, for all $j \in \mathsf{Ainputs}$ A uses the Or-Opening of $\mathcal{F}_{\mathsf{COM}}$ to open the input key to the garbled gates in $\mathsf{Bucket}(j)$ corresponding to her input bit. For all $j \in \mathsf{Binputs}$ B also learns the input key to the garbled gates in $\mathsf{Bucket}(j)$ corresponding to his input bit, using the Oblivious-Opening.

Given the initial input keys in **Evaluation** B evaluates each bucket of garbled gates in the following way: He evaluates each gate in the bucket on the left and right input keys for that bucket. If a key appears more than $\lfloor \rho/2 \rfloor$ times as the output key of the garbled gates in the bucket, he takes this to be the output key of the bucket. If no such key exists B aborts. Note that by the way we soldered the garbled circuit, this corresponds exactly to the fault tolerant circuit we described above. Finally B provides A with the output keys. Knowing Δ, A can decipher the output keys and obtain the output values.

Input

1. For all $j \in \mathsf{Ainputs}$ let $h = \mathsf{BucketHead}(j)$, A inputs $\left(\texttt{OR-open}, \mathsf{id}\left(L_0^h\right), \mathsf{id}\left(L_1^h\right), a_j\right)$ to $\mathcal{F}_{\mathsf{COM}}$.
2. For all $j \in \mathsf{Binputs}$ let $h = \mathsf{BucketHead}(j)$, A inputs $\left(\texttt{OT-open}, j, \mathsf{id}\left(L_0^h\right), \mathsf{id}\left(L_1^h\right)\right)$ to $\mathcal{F}_{\mathsf{COM}}$.
3. In both cases let $\hat{L}^h$ be the key output from $\mathcal{F}_{\mathsf{COM}}$ to B. B computes a list of candidate keys $\mathsf{Cand}_j = \left(\mathsf{ShiftEval}\left(sgg^i, \hat{L}^h, \hat{L}^h\right)\right)_{i \in \mathsf{Bucket}(j)}$. If any key appears more than $\rho/2$ times in Cand_j name it $\hat{O}^j$, otherwise B aborts.

Evaluate

1. For each gate $j \in \mathsf{gates} \setminus \mathsf{inputGates}$, B computes:
   (a) Left and right keys $\hat{L}^j = \bigoplus_{l \in \mathsf{lp}(j)} \hat{O}^l$ and $\hat{R}^j = \bigoplus_{l \in \mathsf{rp}(j)} \hat{O}^l$.
   (b) The list of output keys $\mathsf{Cand}_j = \left(\mathsf{ShiftEval}\left(sgg^i, \hat{L}^j, \hat{R}^j\right)\right)_{i \in \mathsf{Bucket}(j)}$.
   (c) If a key appears more than $\rho/2$ times in Cand_j name it $\hat{O}^j$ and proceed, otherwise abort.
2. For all $j \in \mathsf{outputGates}$, B sends $\hat{O}^j$ to A.
3. For all $j \in \mathsf{outputGates}$, A outputs $z_j = 0$ if $\hat{O}^j = O_0^{\mathsf{BucketHead}(j)}$, $z_j = 1$ if $\hat{O}^j = O_1^{\mathsf{BucketHead}(j)}$ and aborts otherwise.

Fig. 4. The Protocol π_{LEGO} implementing $\mathcal{F}_{\mathsf{SFE}}$ (Part 2)

Theorem 1. *Let k be the security parameter, $\rho = O(k/\log(s))$. If* $(\mathsf{Yao}, \mathsf{Eval})$ *is an* IND-CIA *secure* Yao free-XOR garbling scheme *then the protocol* π_{LEGO} *in Fig.s 3 and 4 UC, active, static securely implements* $\mathcal{F}_{\mathsf{SFE}}$ *in the* $(\mathcal{F}_{\mathsf{COM}})$*-hybrid model (initialized with* $(\texttt{init}, ID, W)$ *for* $|ID| = 3\Gamma + 1$ *and* $|W| \leq k$*).*

Analysis. We sketch the idea of the proof. Details are in the full version. First considering a corrupted B and then considering a corrupted A.

Corrupted B. B does not receive any output nor has any real way of cheating in the protocol (in the output phase, if B changes the output key in a way that makes A accept, then he must have guessed Δ, thus breaking the IND-CIA game). Essentially, we only need to argue that his view does not leak any information, thanks to the IND-CIA security of the garbling scheme. Note that in the protocol B starts by committing to his input and challenge for the cut-and-choose phase. This allows the simulator S to extract all this information at the beginning of the simulation (and provide input on behalf of corrupted B to the ideal functionality). Then we reduce the security of the protocol to the IND-CIA security of the garbling scheme: the simulator knows in fact T and U before it sends the gates to B, therefore S will place honestly constructed gates in T

(for which it knows the openings and therefore can easily simulate the cut-and-choose test – remember that the simulator fully controls $\mathcal{F}_{\mathsf{COM}}$) and the challenge garbled gates from the $\mathsf{IND\text{-}CIA}$ game in U: that is, the simulator produces a view such that distinguishing between a real and a simulated execution is equivalent to winning the $\mathsf{IND\text{-}CIA}$ game.

Corrupted A. Essentially, the proof of security boils down to proving correctness. By the design of the garbled circuit, correctness follows when there is more than $\lfloor \rho/2 \rfloor$ correct gates in each of the buckets.

The LEGO approach ensures that if A passes the cut-and-choose test, then with overwhelming probability there are at most k faulty gates left in U. Those faulty gates are then randomly assigned into buckets, and this means that with overwhelming probability each bucket will have a majority of correct gates. However, as opposed to [28] where all commitments were binding, here we have also k wildcard commitments to deal with. This is in problematic, as wildcard commitments can be opened to anything, and we need make sure that this does not break correctness.

To be more specific we say that a garbled gate gg^i is *faulty* if the commitments to its input and output zero-keys are fixed to values L_0^i, R_0^i and O_0^i respectively, and there exists some $a, b \in \{0,1\}$ so that $\mathsf{Eval}\left(gg^i, L_a^i, R_b^i\right)$ does not output $O_{a \wedge b}^i$ with overwhelming probability. If a gate gg^i has a wire where the commitment to the associated zero-key is not fixed, then we say that this wire is faulty, and gg^i has *faulty wiring*. We say that gg^i is *fault free* if it is neither faulty nor has faulty wiring. If a garbled gate gg^i is faulty, fault free or has faulty wiring, we say the same of any shifted gate sgg^i resulting from shifting gg^i.

Gates gg^i with faulty wiring are problematic for the cut-and-choose test: If $i \in T$ A can choose to let gg^i act as a fault free gate by opening the wildcard commitments consistently with the actual zero-keys used to generate gg^i. On the other hand, if $i \in U$ A can make sgg^i faulty by opening the commitment inconsistently in **Soldering**[1].

In the full version we show that, with overwhelming probability, there will be a majority of fault free gates in $\left(gg^i\right)_{i \in \mathsf{Bucket}(j)}$ for all $j \in \mathsf{gates}$. It is easy to verify that this means that after **Horizontal Soldering** all commitments to zero-keys are fixed. I.e., the commitment to the zero-key of a faulty wire will be fixed to open as one specific value. If this value is not consistent with the zero-keys used to generate the associated garbled gate, then that gate becomes faulty.

Note however, that for all $j \in \mathsf{gates}$ all fault free shifted gates $\left(sgg^i\right)_{i \in \mathsf{Bucket}(j)}$ resulting from **Horizontal Soldering** will have identical input and output keys, as required of the garbled circuit, even if some gates in $\left(gg^i\right)_{i \in \mathsf{Bucket}(j)}$ had faulty wiring. I.e., the effect of a garbled gate gg^i having faulty wiring is *at worst* that shifted gate sgg^i after **Soldering** is faulty. Since we use a $\mathcal{F}_{\mathsf{COM}}$ functionality with at most k wildcard commitments we still have at most k faulty gates in **Evaluation**. Since these faulty gates are placed in random buckets we can still guarantee correctness with overwhelming probability.

[1] By *inconsistently* we mean inconsistent with the actual keys used for gg^i, not inconsistent with the equations stored in $\mathcal{F}_{\mathsf{COM}}$.

Note that the faulty wires are also the reason for gate replication on the input layer, to not let A change her or B's input by using the wildcard commitments.

4 Commitments

In this section we present our novel construction of XOR-homomorphic commitment based solely on OT. We also describe how to transform this general construction of XOR-homomorphic commitments into the specific type we need in Fig.s 3 and 4.

The Ideal Functionality. We name our ideal functionality $\mathcal{F}_{\mathsf{WCOM}}$ and describe it in Fig. 5. The functionality allows A to commit to up to μ messages and to later reveal those messages. In addition the $\mathcal{F}_{\mathsf{WCOM}}$ allows to reveal the XOR of two or more committed messages to B (without revealing any extra information). In the context of Fig.s 3 and 4. this is the same as $\mathcal{F}_{\mathsf{COM}}$ except without the methods for **Oblivious Opening**, **OR open** and (which will become apparent later on) contains more wildcards than we can handle in Fig.s 3 and 4.

We formalize $\mathcal{F}_{\mathsf{WCOM}}$ in the following way: First we let the adversary specify a set of identifiers ID of size κ used to identify each of the μ commitments (for technical reasons ID will be a subset of $[2\mu]$). In addition the adversary gives a set $W \subset ID$, of size at most κ, to identify the wildcard commitments. $\mathcal{F}_{\mathsf{WCOM}}$ stores a set of linear equations on μ variables $(\mathtt{X}_i)_{i \in ID}$ (one for each commitment). Initially this set is empty. Each time A commits to a message m_j using a non-wildcard commitment (i.e., $j \in ID \setminus W$) $\mathcal{F}_{\mathsf{WCOM}}$ will store the equation $\mathtt{X}_j = m_j$. For wildcard commitments no such equation are stored when A makes the commitment. If A instructs $\mathcal{F}_{\mathsf{WCOM}}$ to open the XOR of the set of commitments $J \subset ID$ we let corrupted A input a message $m_J \in \{0,1\}^t$ she wants to open to. The functionality then adds the equation $\bigoplus_{j \in J} \mathtt{X}_j = m_J$ to the set of stored equations and checks that this set of equations has a solution, i.e., if there is an assignment of values in $\{0,1\}^t$ to each variable $\mathtt{X}_j$ such that all equations are satisfied. If so, the functionality permanently stores the equation $\bigoplus_{j \in J} \mathtt{X}_j = m_J$ and opens the XOR of the set of commitments J as m_J. Otherwise, $\mathcal{F}_{\mathsf{WCOM}}$ will output `Alice cheats` to B and terminate. Note that if $J \cap W = \emptyset$ then for all $j \in J$ the functionality has stored the equation $\mathtt{X}_j = m_j$, and therefore a corrupt A can only open the commitment successfully if $m_J = \bigoplus_{j \in J} m_j$. Note also that if, e.g., A has made commitments $i \in W$ and $j \in ID \setminus W$, and opens the XOR of commitments i and j as m' then for all later openings A can only successfully open the wildcard commitment i as $m'_i = m_j \oplus m'$. In these cases, when a commitment can only successfully be opened to one value, we say that the commitment is *fixed* to that value. Non-wildcard commitments are always fixed; when A opens the XOR of a wildcard commitments and non-wildcard commitments, a wildcard commitment can become fixed. When a wildcard commitment has been fixed it can essentially be viewed as a non-wildcard commitment.

Notice that the terminology can become a little confusing because of the wildcard commitments: when we say that A opens the XOR of some set of commitments $J \subset ID$ to a value m_J, then we cannot guarantee that $m_J = \bigoplus_{j \in J} m_j$, when $J \cap W \neq \emptyset$.

Initialization: On input $(\texttt{init}, ID, W)$ with $|ID| = \mu$, $|W| \leq \kappa$ and $W \subset ID$ from the adversary output ID to both parties and let $\mathcal{J} = \emptyset$. If A is honest, then $W = \emptyset$.

Commit On input $(\texttt{commit}, j \in ID, m_j)$ with $m_j \in \{0,1\}^t$ from A, and where no value of the form $(j, \cdot)$ is stored, store (j, m_j). If $j \in ID \backslash W$, add $J = \{j\}$ to $\mathcal{J}$ and associate with J the equation $\mathtt{X}_j = m_j$. Then output $(\texttt{commit}, j)$ to B.

Open On input $(\texttt{open}, J \subset ID)$ from A, where for all $j \in J$ a pair (j, m_j) is stored do:

- If A is honest, output $(\texttt{open}, J, \oplus_{j \in J} m_j)$ to B.
- If A is corrupted wait for A to input $(\texttt{corrupt-open}, J, m_J)$. Then add J to $\mathcal{J}$, associate the equation $\oplus_{j \in J} \mathtt{X}_j = m_J$ to J, and check that the equation system $\{\oplus_{j \in J} \mathtt{X}_j = m_J\}_{J \in \mathcal{J}}$ has a solution[a]. If so, output $(\texttt{open}, J, m_J)$ to B. Otherwise, output `Alice cheats` to B and terminate.

[a] I.e., there should be an assignment of values to the wildcard commitments such that all stored openings can be explained by this assignment.

Fig. 5. The ideal functionality, $\mathcal{F}_{\mathsf{WCOM}}$, for our basic commitment scheme consisting

Building blocks. Here we give the building blocks from which we implement $\mathcal{F}_{\mathsf{WCOM}}$.

Oblivious Transfer. We use a $\binom{n}{u}$-Oblivious Transfer functionality with message strings of length 2μ. We denote this functionality $\binom{n}{u}$-$\mathcal{F}_{\mathrm{OT}}(2\mu)$. On input `start` from both A and B the $\binom{n}{u}$-$\mathcal{F}_{\mathrm{OT}}(2\mu)$ functionality picks n message strings $S_1, \ldots, S_n \in_{\mathrm{R}} \{0,1\}^\mu$, a uniformly random set $I \subseteq \{1, \ldots, n\}$ with $|I| = u$ and outputs $(S_i)_{i \in [n]}$ to A and $(I, (S_i)_{i \in I})$ to B. We can implement $\binom{n}{u}$-$\mathcal{F}_{\mathrm{OT}}(2\mu)$ for any $2\mu = \mathrm{poly}(k)$ using a $\mathcal{F}_{\mathrm{OT}}$ functionality and a pseudo random generator (prg), where k is the security parameter, using the same technique as in [27]: Simply use the $\mathcal{F}_{\mathrm{OT}}(k)$ functionality to send seeds to the prg and then use the prg to expand those seeds to 2μ bits. One can then construct a $\binom{n}{u}$-$\mathcal{F}_{\mathrm{OT}}(2\mu)$ functionality from $\binom{2}{1}\mathcal{F}_{\mathrm{OT}}(2\mu)$ e.g., as described in [25].

Error Correcting Codes. We also need an error correcting code (ECC), which encodes an t-bit string as an n-bit string with minimal distance at least d using some ϕ-bits of randomness. It should at the same time be a secret sharing scheme in that seeing u random positions of a random codeword does not leak information on the message. We denote this scheme by $\mathrm{ssecc}^{t,n,d,u}$. We use $\mathrm{enc}^{t,n,d,u}$ to denote the encoding function and we use $\mathrm{dec}^{t,n,d,u}$ to denote the decoding function. Both should be PPT and we drop parameters for notational convenience. The code should have the following properties.

Error correction. For all $m \in \{0,1\}^t$, $r \in \{0,1\}^\phi$ and error vectors $e \in \{0,1\}^n$ with $\mathrm{hw}(e) < d/2$ it holds that $\mathrm{dec}(\mathrm{enc}(m; r) \oplus e) = m$, where hw is the Hamming weight in $\{0,1\}^n$. We assume that $\mathrm{dec}(C) = \perp$ when C has distance more than $d/2$ to all codewords and we assume that there exists an efficient algorithm ncw (nearest codeword) such that $\mathrm{ncw}(\mathrm{enc}(m; r) \oplus e) = \mathrm{enc}(m; r)$ when $\mathrm{hw}(e) < d/2$.

Privacy. There exists a PPT function xpl which can explain any codeword as being a codeword of any message to anyone who knows at most u positions of the codeword. Formally, for all $I \subset [n]$, $|I| = u$ and all $m, m' \in \{0,1\}^t$ the distributions D_0 and D_1 described below are statistically close. The distribution D_0 is generated as follows: sample $r \in_{\mathrm{R}} \{0,1\}^\phi$, let $c = \mathrm{enc}(m; r)$ and output $((c_i)_{i \in I}, m, r)$. The distribution D_1 is generated as follows: sample $r' \in_{\mathrm{R}} \{0,1\}^\phi$, let $c = \mathrm{enc}\,(m'; r')$, sample $r \leftarrow \mathrm{xpl}(I, m', r', m)$ and output $((c_i)_{i \in I}, m, r)$.

Linearity. For all $m, m' \in \{0,1\}^t$ and $r, r' \in \{0,1\}^\phi$ it holds that $\mathrm{enc}(m;r) \oplus \mathrm{enc}(m';r') = \mathrm{enc}(m \oplus m'; r \oplus r')$.

Note that **Error correction** implies that the minimal distance is at least d, i.e., for all $m \neq m' \in \{0,1\}^t$ and $r, r' \in \{0,1\}^\phi$, $c = \mathrm{enc}(m;r)$ and $c' = \mathrm{enc}(m';r')$ it holds that $\mathrm{ham}(c, c') \geq d$ where ham is the Hamming distance.

We further require of the parameters of ssecc that: $n = \Theta(k)$, $u = \Theta(n)$ and $d = \Theta(n)$. I.e., both the privacy and minimum distance of ssecc must be a constant fraction of the length of codewords, and the code should have constant rate. Codes that satisfy the desired properties can be found in [4].

Protocol Specification. Here we describe the ideas behind the protocol π_{WCOM} implementing $\mathcal{F}_{\mathsf{WCOM}}$ described in Fig. 6.

Let $v \in \{0,1\}^n$ and $I = \{i_1, i_2, \ldots, i_u\} \subseteq [n]$. We define the function $w_I : \{0,1\}^n \to \{0,1\}^u$ so that $w_I(v) = (v_{i_1}, v_{i_2}, \ldots, v_{i_u}) \in \{0,1\}^u$, i.e., $w_I(v)$ is the u-bit string consisting of the u bits in v indexed by I.

In the protocol a commitment to a message m is a one-time pad of m with some key T. Clearly this is hiding but not binding. To make the commitment binding we allow the receiver of the commitment (B) to learn $w_I(m)$ for some secret set $I \subseteq [n]$. We denote $w_I(m)$ the *watch bits* of the commitment. To open the commitment to m A sends m' to B and B checks if $w_I(m') = w_I(m)$. If this is not the case B rejects the opening.

The watch bits give some degree of binding since A can only open the commitment to some message $m' \neq m$ if $w_I(m') = w_I(m)$. I.e., if $u = |I|$ is large enough A can only hope to change a few bits of m without getting caught. On the other hand the watch bits clearly compromises the hiding property of the commitment. To avoid this we use the code ssecc to encode the message m and commit to the encoded m instead. I.e., a commitment to m becomes $\mathrm{enc}(m;r) \oplus T$. By privacy of ssecc m is now hidden.

The encoding additionally strengthens the binding of the commitment: codewords c and c' encoding to two different messages m and m' must be different in many bit positions. Thus for A to open a commitment to m to m' none of these positions must be in the watch bits.

More precisely let $d = 2w + 1$ be the minimum distance of ssecc for some $w < \frac{n}{2}$. Suppose a corrupt A gives the commitment, $c \oplus T$. Note that when A is corrupt c does not have to be a codeword. In that case we have that $c = \mathrm{ncw}(c) \oplus e$ for some *error vector* $e \in \{0,1\}^n$, and we say the commitment has $\mathrm{hw}(e)$ *errors*.

Regardless of the number of errors, consider what it takes for A to be able to open this commitment to two different messages m' and m'', with codewords c' and c'' respectively: for any two different codewords c' and c'' one of them has distance at least w to c, say c'. In other words c' has at least w bit positions different from c. If A tries to open the commitment to m', B only accepts the opening if none of these bit positions are in his u watch bits for the commitment. Thus for any commitment (possibly with errors) the probability that a cheating A can open the commitment to two different messages m' and m'' is at most

$$\binom{n-u}{w}\binom{n}{w}^{-1} = \prod_{i=0}^{w-1} \frac{n-u-i}{s-i} \prod_{i=0}^{w-1} \frac{w-i}{n-i} = \prod_{i=0}^{w-1} \frac{n-u-i}{n-i} = \prod_{i=0}^{w-1} 1 - \frac{u}{n-i} .$$

Assume that $w = w'n$ and $u = u'n$ for constants $0 < u', w' < 1$, then the probability of A breaking the binding property is at most

$$\prod_{i=0}^{w-1}\left(1 - \frac{u}{n-i}\right) \leq \prod_{i=0}^{w-1}\left(1 - \frac{u'n}{n}\right) = (1-u')^{w'n} \ .$$

Thus with k being the security parameter, $n = \Theta(k)$ and for any positive constants u', w' with $0 < u', w' < 1$, A will have negligible probability of breaking binding.

Notice, that while c' will have distance at least w to c it could be that c'' is much closer to c. E.g., c'' could be the nearest codeword to c. In this case, we have that, if the commitment has *very few* errors, a cheating A *could* open the commitment to m'' with noticeable probability (say, if the number of errors were constant in k). This is not a problem since such a "slightly wrong" commitment can be seen as a commitment to the message m'' encoded by the nearest codeword c''.

To get XOR-homomorphic commitments, more work has to be done. The problem being that the XOR of several commitments with errors, may become a commitment that breaks binding, even if the individual commitments only have a few errors. Consider a number of commitments made with non-codewords c_i with nearest codewords c_i'. The XOR the non-codewords $c = \bigoplus_i c_i$ may then be very far from the XOR of their nearest codewords $c' = \bigoplus_i c_i'$. In fact c might be so far away from c' that it gets very close to some other codeword c''. Hence the XOR of the commitments can be opened to a message different from the XOR of the message associated with the individual commitments. This would break the binding property.

To deal with this problem, the protocol initialization π_{WCOM} starts by letting A commit to 2μ random messages. We then do a cut-and-choose test to check that half of these commitments can be opened correctly. If A passes the test we have that, with overwhelming probability, the remaining commitments only have a few errors. Additionally, those errors must be isolated to a few common bit positions. Thus the result of XOR'ing these commitments will at most have a few errors, namely in these few positions.

Thus if A passes the cut-and-choose test we use the un-tested random commitments to implement the actual commitments. The resulting commitment will have exactly the same errors as the random commitment (if any).

Theorem 2. *Let k be the security parameter and use a code with $n = \Theta(k)$, $u = \Theta(n)$, $d = \Theta(n)$ and $k < d/2$ as, e.g., given by [4]. Then the protocol in Fig. 6 UC, active, static securely implements $\mathcal{F}_{\mathsf{WCOM}}$ in the $\left(\binom{n}{u}\text{-}\mathcal{F}_{\mathrm{OT}}(2\mu)\right)$-hybrid model when initialized on $(\mathtt{init}, ID, W)$ with $|ID| = \mu$ and $|W| \leq nk + k$.*

Analysis. Simulating when no party is corrupted or both parties are corrupted is straight forward. Simulating when B is corrupted is also quite simple, and can be done using standard techniques from simulation in secure multi-party computation based on secret sharing. Thus we will only sketch the proof for corrupted B, and focus on the case of corrupted A.

Corrupted B. The simulator commits to 0^t in all commitments. When asked to open such a commitment U_j to a given $m_j \in \{0,1\}^t$ it uses the efficient algorithm xpl to explain the commitment as $U_j = \mathrm{enc}(x_j; r_j) \oplus T_j$ for $x_j = y_j \oplus m_j$. The only non-trivial detail is that if the simulator is asked to open a commitment, where the

Setup To set up the scheme A and B run the following.

1. A and B run a $\binom{n}{u}$-$\mathcal{F}_{\text{OT}}(2\mu)$ functionality and get as output $(R_i)_{i\in[n]}$ and $(I, (R_i)_{i\in I})$ respectively where $R_1, \ldots, R_n \in_{\text{R}} \{0,1\}^{2\mu}$ and I is a uniformly random subset of $[n]$ with $|I| = u$.
2. A lets $R \in \{0,1\}^{n\times 2\mu}$ be the matrix with R_i as the i'th row and lets $T_j \in \{0,1\}^n$ be the j'th column of R.[a]
3. A for $j = 1, \ldots, 2\mu$, samples $x_j \in_{\text{R}} \{0,1\}^t$, $r_j \in_{\text{R}} \{0,1\}^\phi$ and sends the commitment $c_j = (\text{enc}(x_j; r_j) \oplus T_j, j)$. Let $c_j = (U_j, j)$ the value received by B.
4. B sends a uniformly random subset $C \subset [2\mu]$. This also defines $ID = \bar{C}$.
5. For $j \in C$, A opens c_j by sending $o_j = (x_j, r_j, j)$.
6. For $j \in C$, B receives (x'_j, r'_j, j) and checks that $w_I\left(\text{enc}\left(x'_j; r'_j\right)\right) = w_I(U_j) \oplus w_I(T_j)$, if not B terminates the protocol.

Commit To commit to m_j for $j \in ID$ A sends (y_j, j) to B where y_j is the *correction value* $y_j = x_j \oplus m_j$.

Open To open the XOR of commitments $J \subset ID$ the parties do the following.

1. For $j \in J$, let $c_j = (\text{enc}(x_j; r_j) \oplus T_j, j)$ be the commitments sent in initialization and y_j the value sent during commitment. A computes the opening of $\bigoplus_{j\in J} m_j$ as $o_J = \left(\bigoplus_{j\in J} x_j, \bigoplus_{j\in J} r_j, J\right)$, and sends it to B.
2. If an opening of J was done previously, B uses the previous m_J, otherwise he proceeds as follows: Let $c_j = (U_j, j)$ be the commitments received during **Setup**. B accepts $o_J = (x_J, r_J, J)$ iff

$$w_I(\text{enc}(x_J; r_J)) = w_I\left(\bigoplus_{j\in J} U_j\right) \oplus w_I\left(\bigoplus_{j\in J} T_j\right), \text{ where}$$

$$w_I\left(\bigoplus_{j\in J} U_j\right) = \bigoplus_{j\in J} w_I(U_j) \text{ and } w_I\left(\bigoplus_{j\in J} T_j\right) = \bigoplus_{j\in J} w_I(T_j)\,.$$

If B accepts he outputs $x_J \oplus y_J$, where $y_J = \bigoplus_{j\in J} y_j$. Otherwise, B rejects the opening and terminates the protocol.

[a] Notice B can use $(R_i)_{i\in I}$ to compute $w_I(T_j)$ for all $j \in [2\mu]$.

Fig. 6. The protocol π_{WCOM} implementing $\mathcal{F}_{\mathsf{WCOM}}$

value of the opening follows from previous openings (i.e., using some linear equation), it computes the opening as a linear combination of the previous simulated openings. As an example, if the simulator opened U_j as $U_j = \text{enc}(x_j; r_j) \oplus T_j$ and opened U_i as $U_i = \text{enc}(y_i; r_i) \oplus T_i$. Then it will open $U_j \oplus U_i$ as $U_j = \text{enc}(x_j \oplus x_i; r_j \oplus r_i) \oplus T_j \oplus T_i$.
Corrupted A. Intuition of the proof when A is corrupted is that the cut-and-choose test will catch A if there are many indices i for which there exists a commitment that has an error in position i. This is because if the errors of the commitments are very spread out, with high probability, many of them will be in the watch bits positions. As mentioned above, this means that almost all errors must be isolated in a few positions. Therefore XOR's of commitments will also have errors only in these position, so the XOR's will also be close to their "correct" codeword. The formal proof is complicated by the fact that a few commitments with many errors, or errors outside isolated few positions, may

pass the cut-and-choose. These commitments will be the wildcards. It can be shown that not even a commitment with many errors can be opened to two different values, as it would give a codeword encoding a non-zero value which is 0 in all the watch bits, which happens with negligible probability by the watch bits being random and the minimal distance high. This translates into it being impossible to make any combination of openings of linear equations yielding inconsistent outputs.

Completing the Construction. There is a gap between the ideal functionality $\mathcal{F}_{\mathsf{WCOM}}$ that we just implemented and the functionality $\mathcal{F}_{\mathsf{COM}}$ used in the protocol π_{LEGO}. The gap can be closed using standard techniques, as sketched now. There are many more details on this in the full version. We can reduce the number of wildcard commitments by opening random pairs of commitments and discarding one of the commitments. This fixes any wildcard commitment not lucky enough to be paired with another wildcard commitment. We can implement **Oblivious Opening** by sending both openings through an oblivious transfer: note that we allow selective errors in the ideal functionality, so it is not an issue that the adversary can send one correct and one incorrect opening. Finally, we can implement **OR Open** using a standard technique where the committer commits to many pairs of values, each pair being a random permutation of the values in the two commitments of which he wants to open one. Then for each pair he is randomly challenged by the receiver to either uses the XOR homomorphism to show that the correct two messages were committed, or to open one of the two commitments to the claimed value.

References

1. Bellare, M., Hoang, V.T., Rogaway, P.: Foundations of garbled circuits. In: ACM Conference on Computer and Communications Security, pp. 784–796 (2012)
2. Ben-David, A., Nisan, N., Pinkas, B.: FairplayMP: a system for secure multi-party computation. In: ACM Conference on Computer and Communications Security, pp. 257–266 (2008)
3. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: STOC, pp. 494–503 (2002)
4. Chen, H., Cramer, R.: Algebraic geometric secret sharing schemes and secure multi-party computations over small fields. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 521–536. Springer, Heidelberg (2006)
5. Damgård, I., Keller, M., Larraia, E., Miles, C., Smart, N.P.: Implementing AES via an actively/covertly secure dishonest-majority MPC protocol. In: Visconti, I., De Prisco, R. (eds.) SCN 2012. LNCS, vol. 7485, pp. 241–263. Springer, Heidelberg (2012)
6. Frederiksen, T., Jakobsen, T.P., Jesper Buus Nielsen, P.S.N., Orlandi, C.: Minilego: Efficient secure two-party computation from general assumptions (full version). Cryptology ePrint Archive, Report (2013), `http://eprint.iacr.org/`
7. Frederiksen, T.K., Nielsen, J.B.: Fast and maliciously secure two-party computation using the gpu. Cryptology ePrint Archive, Report 2013/046 (2013), `http://eprint.iacr.org/`
8. Goldreich, O.: Foundations of Cryptography: Basic Applications. Cambridge University Press (2004)
9. Harnik, D., Ishai, Y., Kushilevitz, E., Nielsen, J.B.: OT-combiners via secure computation. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 393–411. Springer, Heidelberg (2008)
10. Hazay, C., Lindell, Y.: Efficient Secure Two-Party Protocols: Techniques and Constructions. Springer (2010)
11. Henecka, W., Kögl, S., Sadeghi, A.-R., Schneider, T., Wehrenberg, I.: TASTY: tool for automating secure two-party computations. In: ACM Conference on Computer and Communications Security, pp. 451–462 (2010)

12. Huang, Y., Evans, D., Katz, J., Malka, L.: Faster secure two-party computation using garbled circuits. In: USENIX Security Symposium (2011)
13. Huang, Y., Katz, J., Evans, D.: Quid-pro-quo-tocols: Strengthening semi-honest protocols with dual execution. In: IEEE Symposium on Security and Privacy, pp. 272–284 (2012)
14. Ishai, Y., Kilian, J., Nissim, K., Petrank, E.: Extending oblivious transfers efficiently. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 145–161. Springer, Heidelberg (2003)
15. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Cryptography with constant computational overhead. In: STOC, pp. 433–442 (2008)
16. Ishai, Y., Prabhakaran, M., Sahai, A.: Founding cryptography on oblivious transfer – efficiently. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 572–591. Springer, Heidelberg (2008)
17. Kolesnikov, V., Schneider, T.: Improved garbled circuit: Free XOR gates and applications. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 486–498. Springer, Heidelberg (2008)
18. Kreuter, B., Shelat, A., Shen, C.: Billion-gate secure computation with malicious adversaries. USENIX Security. Available at Cryptology ePrint Archive, Report 2012/179 (2012), `http://eprint.iacr.org/`
19. Lindell, Y., Oxman, E., Pinkas, B.: The IPS compiler: Optimizations, variants and concrete efficiency. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 259–276. Springer, Heidelberg (2011)
20. Lindell, Y., Pinkas, B.: An efficient protocol for secure two-party computation in the presence of malicious adversaries. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 52–78. Springer, Heidelberg (2007)
21. Lindell, Y., Pinkas, B.: A proof of security of Yao's protocol for two-party computation. J. Cryptology 22(2), 161–188 (2009)
22. Lindell, Y., Pinkas, B.: Secure two-party computation via cut-and-choose oblivious transfer. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 329–346. Springer, Heidelberg (2011)
23. Lindell, Y., Pinkas, B., Smart, N.P.: Implementing two-party computation efficiently with security against malicious adversaries. In: Ostrovsky, R., De Prisco, R., Visconti, I. (eds.) SCN 2008. LNCS, vol. 5229, pp. 2–20. Springer, Heidelberg (2008)
24. Malkhi, D., Nisan, N., Pinkas, B., Sella, Y.: Fairplay - secure two-party computation system. In: USENIX Security Symposium, pp. 287–302 (2004)
25. Naor, M., Pinkas, B.: Oblivious transfer with adaptive queries. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 573–590. Springer, Heidelberg (1999)
26. Naor, M., Pinkas, B., Sumner, R.: Privacy preserving auctions and mechanism design. In: ACM Conference on Electronic Commerce, pp. 129–139 (1999)
27. Nielsen, J.B., Nordholt, P.S., Orlandi, C., Burra, S.S.: A new approach to practical active-secure two-party computation. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 681–700. Springer, Heidelberg (2012)
28. Nielsen, J.B., Orlandi, C.: LEGO for two-party secure computation. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 368–386. Springer, Heidelberg (2009)
29. Pinkas, B., Schneider, T., Smart, N.P., Williams, S.C.: Secure two-party computation is practical. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 250–267. Springer, Heidelberg (2009)
30. Schoenmakers, B., Tuyls, P.: Practical two-party computation based on the conditional gate. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 119–136. Springer, Heidelberg (2004)
31. shelat, A., Shen, C.-H.: Two-output secure computation with malicious adversaries. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 386–405. Springer, Heidelberg (2011)
32. Yao, A.C.-C.: How to generate and exchange secrets (extended abstract). In: FOCS, pp. 162–167 (1986)

How to Hide Circuits in MPC an Efficient Framework for Private Function Evaluation

Payman Mohassel and Saeed Sadeghian

University of Calgary

Abstract. We revisit the problem of general-purpose *private function evaluation* (PFE) wherein a single party P_1 holds a circuit $\mathcal{C}$, while each P_i for $1 \leq i \leq n$ holds a private input x_i, and the goal is for a subset (or all) of the parties to learn $\mathcal{C}(x_1, \ldots, x_n)$ but nothing else. We put forth a general framework for designing PFE where the task of hiding the circuit and securely evaluating its gates are addressed independently: First, we reduce the task of hiding the circuit topology to oblivious evaluation of a mapping that encodes the topology of the circuit, which we refer to as *oblivious extended permutation* (OEP) since the mapping is a generalization of the permutation mapping. Second, we design a subprotocol for private evaluation of a single gate (PFE for one gate), which we refer to as *private gate evaluation* (PGE). Finally, we show how to naturally combine the two components to obtain efficient and secure PFE.

We apply our framework to several well-known general-purpose MPC constructions, in each case, obtaining the most efficient PFE construction to date, for the considered setting. Similar to the previous work we only consider semi-honest adversaries in this paper.

1 Introduction

In a *private function evaluation* (PFE) protocol, a party P_1 holds a function f, and its corresponding circuit C_f, while every party P_i holds a private input x_i; their goal is for a subset (or all) of the parties to learn $f(x_1, \ldots, x_n)$ without learning any information beyond this. In particular, besides the size of the circuit, and the length of P_1's inputs and outputs, P_i $(i \geq 2)$ should not learn anything else about the circuit. This is in contrast to the standard setting for secure multi-party computation where the function f and the corresponding circuit C_f are publicly known to all the participants. PFE is particularly useful in scenarios where learning the function compromises privacy, reveals security vulnerabilities, or when service providers need to hide the function or a specific implementation of it to protect their Intellectual Property. A number of papers in the literature have considered the design of efficient general-purpose private function evaluation protocols [1,2,3,4].

Solutions Based on Universal Circuits. Most general-purpose PFE solutions reduce the problem to secure computation of a *universal circuit* U_g that takes as input the circuit C_f (with at most g gates), and the parties' private inputs $x_1, \ldots, x_n$, and outputs $f(x_1, \ldots, x_n)$. The main objective of this line of work is to design smaller size universal circuits, and to optimize their implementation using existing MPC constructions such as Yao's garbled circuit protocol [2,5,3].

The Universal circuit approach works with any secure MPC protocol for evaluating boolean circuits and is applicable to both the two-party and the multi-party settings. Its main disadvantage, and the main motivation for other alternatives is the additional overhead in efficiency due to the size of universal circuits and the complexity of designing and implementing such circuits. Valiant [6] showed a construction of a boolean universal circuit achieving an optimal circuit size of $|U_g| \approx 19 g \log g$. Kolesnikov and Schneider [2] gave an alternative construction of universal circuits. They obtain a worse asymptotic bound of $|U_g| \approx 1.5 g \log^2 g$, but their techniques lead to smaller constant factors and seem to yield smaller universal circuits than Valiant's construction for circuit sizes less than 5000. Furthermore, the universal circuit approach does not provide a satisfactory solution in case of arithmetic circuits. While universal arithmetic circuits exist (e.g. see [7] and [8]), their sizes are too large for any practical purpose (e.g. as high as $O(g^5)$).

Solutions Based on Homomorphic Encryption. It is relatively easy to design a PFE based on a fully homomorphic encryption scheme [9]. While asymptotically optimal, this solution is not practical due to its high computational cost. Recently, Katz and Malka [4] designed a novel two-party PFE protocol based on a *singly homomorphic encryption*. Complexity of the resulting protocol is linear in the size of the circuit but the number of public-key operations is also linear in the size of the circuit. Standard techniques for reducing public-key operations (e.g. OT extension) do not seem applicable either. Given the significant gap between the efficiency of public- vs. symmetric-key operations, this new approach improves over the universal circuit only when dealing with large circuits. Finally, this solution only works in the two-party setting.

Our Contribution. Practical design and implementation of MPC has been the subject of active research in the last few years. As discussed above, however, when it comes to PFE the situations is not the same. The existing solutions are considerably less scalable and more expensive compared to their MPC counterparts, and *no good solution exists for the multiparty case, or when considering arithmetic circuits.* We revisit private function evaluation with the intention of designing more practical two-party and multi-party constructions. In particular, we put forth a general framework for designing PFE and show how it enables us to construct more efficient PFE variants of the well-known MPC protocols.

Our Framework for Designing PFE. In order to fully hide a circuit $\mathcal{C}$, one needs to hide two types of information about it: (i) the *topology* of the circuit, and (ii) the *function of the gates* in the circuit (AND, OR, XOR). Note that these are in addition to what is already hidden in a MPC setting. Following this observation we divide the task of private function evaluation into two different functionalities:

(1) the Circuit Topology Hiding (CTH) functionality, and (2) the Private Gate Evaluation (PGE) functionality. Next, we describe these two functionalities in more detail:

CTH *Functionality.* We observe that the topology of a circuit $\mathcal{C}$ can be fully described using a mapping $\pi_{\mathcal{C}} : \{1 \dots |\mathsf{OW}|\} \rightarrow \{1 \dots |\mathsf{IW}|\}$ where OW (outgoing wires) is the union of the set of input wires $\{\mathsf{ow}_1 = x_1, \dots, \mathsf{ow}_n = x_n\}$, and the output wires for each non-output gate in the circuit $\{\mathsf{ow}_{n+1}, \dots, \mathsf{ow}_{n+g-o}\}$ (g is the circuit size and o is the number of output gates), and IW (incoming wires) is the set of input wires to all the gates in the circuit $\{\mathsf{iw}_1, \dots, \mathsf{iw}_{2g}\}$. $\pi_{\mathcal{C}}$ maps i to j ($\pi_{\mathcal{C}}(i) = j$) if and only if wire $\mathsf{ow}_i \in \mathsf{OW}$ is connected to $\mathsf{iw}_j \in \mathsf{IW}$, in the circuit $\mathcal{C}$. Note that since the fan-out for each gate can be more than one, $\pi_{\mathcal{C}}$ is not always a function, but it is easy to check that its inverse $\pi_{\mathcal{C}}^{-1}$ is. *Note that the party who knows the function f and the corresponding circuit $\mathcal{C}$ can efficiently compute $\pi_{\mathcal{C}}$.* Figure 1 demonstrates an example circuit and its corresponding mapping. Intuitively the $\mathcal{F}_{\mathcal{CTH}}$ functionality provides a mechanism for obliviously applying the mapping π_C to the n input values and the $(g - o)$ values for intermediate outgoing wires (i.e. mapping them to incoming wires) in an on-demand fashion, and as the MPC protocol proceeds.

PGE *Functionality.* The PGE functionality can be seen as a PFE protocol where the function is a single gate. P_1 provides the gate's functionality, while all parties including P_1 provide their shares of the two inputs to the gate. The functionality returns to each party, his share of the gate's output.

These two functionalities can be naturally composed to obtain a complete PFE protocol as described in Figure 4. A visual demonstration of the steps appears in Figure 2.

Efficient Realizations of $\mathcal{F}_{\mathcal{CTH}}$. We refer to the mapping $\pi_{\mathcal{C}} : \{1 \dots |\mathsf{OW}|\} \rightarrow \{1 \dots |\mathsf{IW}|\}$ discussed above as an *extended permutation* (EP) since it not only permutes the elements in $\{1 \dots |\mathsf{OW}|\}$, but also can replicate them as many times as needed. A main component of our $\mathcal{F}_{\mathcal{CTH}}$ realization is a protocol for *oblivious evaluation* of this extended permutation (OEP) on a vector of inputs:

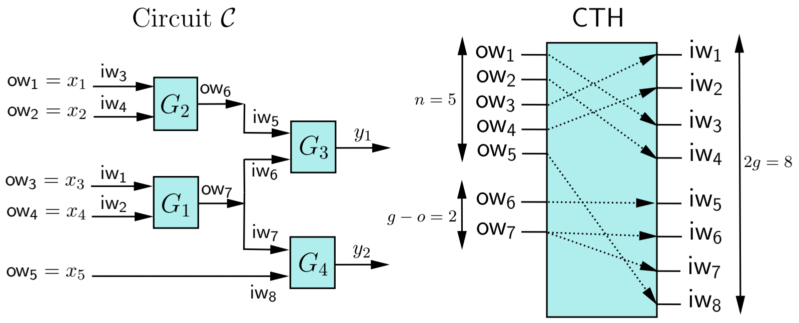


Fig. 1. An example circuit and the corresponding mapping

the first party holds π_C and a blinding[1] vector $\boldsymbol{t}$ of size $|\mathsf{IW}|$, while the second party holds an input vector $\boldsymbol{x}$ of size $|\mathsf{OW}|$. Their goal is to let the second party learn the output of π_C applied to $\boldsymbol{x}$, blinded by $\boldsymbol{t}$. Neither party should learn anything else. OEP can be instantiated using a singly homomorphic encryption, or any general-purpose 2PC. As discussed in the Full version, however, neither solution is efficient enough for use in practice. We introduce a new and efficient construction for OEP based on *generalized switching networks* and oblivious transfer.

OEP via Generalized Switching Networks. First, we show how to efficiently implement an extended permutation using a generalized switching network SN. Once the EP is represented using a SN, we solve the OEP problem by designing a new OT-based protocol for *Oblivious Switching Network evaluation* (OSN) where one party P_1 holds the selection bits to SN, and a blinding vector $\boldsymbol{t}$, while the other party P_2 holds the input vector $\boldsymbol{x}$ to the SN. The goal is for P_2 to learn the output of SN applied to the input vector $\boldsymbol{x}$, blinded by $\boldsymbol{t}$. Our OSN protocol runs in a *constant number of rounds* and requires $O(g)$ oblivious transfers where g is the number of switches in the network. We also need a *multiparty variant of our OEP protocol* where the mapping is known to a single party while the input vector $\boldsymbol{x}$ and the blinding vector $\boldsymbol{t}$ are shared among the players. We show how to construct such an m-party OEP protocol via m invocations of the two-party version.

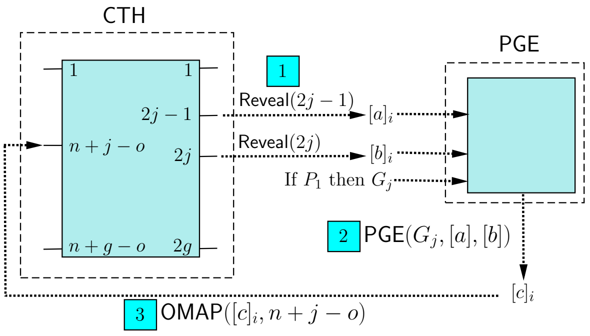


Fig. 2. Steps of framework for party i and the jth gate in a topological order

Improved Oblivious Shuffling. Digressing from the main topic of this paper, we note that OSN is a generalization of the previously studied problems such as oblivious shuffling [10] (a subprotocol used for private set intersection), or secure two-party permutation [11,12]. Our new construction yields more efficient solutions to these problems as well, improving on the previous proposals based

[1] The nature of blinding is intentionally left unspecified as different protocols may use different blinding functions. Our constructions use XOR or addition in a finite Ring for this purpose.

Table 1. Comparison of oblivious shuffling protocols. N is number of shuffled elements, ℓ is the length of each, and k is the security paremeter.

Oblivious Shuffling Protocols	Asymptotic Complexity
HE-Based	$O(N)$ Asym.
Garbled Circuit-Based [10]	$(\frac{4\ell(N\log N-N+1)}{3}+2N\ell)$ Sym. + $O(k)$ Asym.
OSN-Based (our paper)	$(2N\log N-2N+2)$ Sym. + $O(k)$ Asym.

on garbled circuit implementation of sorting networks, permutation networks, or randomize shell sort [10,11,12]. See Table 1 for efficiency comparison with previous work.

Applying our Framework to Existing MPC. We apply the above framework to the GMW protocol [13], Yao's garbled circuit protocol [14], and secure computation of arithmetic circuits via homomorphic encryption [15]. In each case we obtain the most efficient PFE construction to date, for the considered setting.

Linear Multi-party PFE. We apply our framework to the seminal GMW protocol [13] to obtain a multiparty PFE against a dishonest majority. The CTH component can be instantiated using either the HE-based or the SN-based OEP discussed above. We also design a simple and efficient multiparty PGE functionality given a multiparty OT as in [16]. To the best of our knowledge, this is the first multiparty PFE besides the generic solutions of applying MPC to universal circuits. When instantiated using a HE-based OEP, it yields the *first multiparty PFE with linear complexity* (in the circuit size) and when instantiated using our new SN-based OEP, it yields a black-box construction based solely on OT. What makes the second instantiation desirable from a practical point of view, as demonstrated in some recent GMW implementations [17,18], is that it only uses *oblivious transfers*. As a result, one can use OT extension [19] and pre-processing techniques [20] to significantly reduce the number of public-key operations, and to shift the bulk of the computation to an *offline phase*. Table 2 compares the efficiency of these two constructions with the only other alternative, i.e. using GMW with universal circuits.

Table 2. Comparison of m-party PFE protocols. g denotes the number of gates.

Multi-Party PFE	Complexity
[2] Universal Circuits	$O(m^2 g\log^2 g)$ Sym. + $O(k)$ Asym.
[6] Universal Circuits	$O(m^2 g\log g)$ Sym. + $O(k)$ Asym.
GMW-PFE (SN-OEP)	$O(m^2 g+mg\log g)$ Sym. + $O(k)$ Asym.
GMW-PFE (HE-OEP)	$O(m^2 g)$ Sym. + $O(mg)$ HE. + $O(k)$ Asym.

More Efficient Two-party PFE. We also design a constant round two-party PFE based on Yao's garbled circuit protocol [14]. Once again, the $\mathcal{F}_{\mathcal{CTH}}$

functionality is realized using our OEP constructions and for the $\mathcal{F}_{\mathcal{PGE}}$ functionality we use Yao's garbling/ungarbling algorithms. To ensure that functions of the gates are hidden, we build the circuit entirely out of NAND gates. As we will see in Section 5.3, multiple subtleties need to be addressed for this work and in particular to guarantee that the circuit evaluator can unblind garbled keys during the evaluation of the garbled circuit without learning the values for the intermediate wires.

We note that the construction of [4] also fits in the general framework described above (though not presented in this way). However, our new abstraction helps us gain more efficiency improvements. When using our HE-OEP, we obtain a two-party PFE with linear complexity that is simpler and more efficient than that of [4] (see Full Version for details), and when implemented using our SN-OEP, the resulting protocol is concretely more efficient for most circuit sizes, since the number of public-key operations can be made independent of the circuit size (via OT extension). Our construction is both asymptotically and concretely more efficient than the previous work of [2] based on universal circuits. It is concretely more efficient than Valiant's construction [6]. Table 3 summarizes efficiency comparison of our two-party PFE with all previous constructions. In the full version of this paper [21], we show (thorough operations counting) that our construction concretely improves over previous constructions for benchmark circuits such as AES, RSA and Edit-distance.

Linear 2PC for Arithmetic Circuits. We also apply our framework to the construction for secure computation of arithmetic circuits based on a homomorphic encryption [15], and obtain the *first two-party PFE for arithmetic circuits with linear complexity*. Besides utilizing our $\mathcal{F}_{\mathcal{CTH}}$ realizations, we instantiate the $\mathcal{F}_{\mathcal{PGE}}$ functionality by designing a secure gate evaluation protocol wherein only one party knows/learns the functionality (multiplication or addition) but both parties learn their share of the output (product or sum).

Table 3. Comparison of 2-party PFE protocols. (HM: Homomorphic Multiplication, HA: Homomorphic Addition, HE: Homomorphic Encryption). Last column shows concrete gain over universal circuit approaches for benchmark circuits, AES, RSA and Edit-distance (refer to Full version for detailed discussion). g denotes the number of gates.

2-Party PFE	Complexity	Gain
[2]	$1.5g\log^2 g$ sym. + $O(k)$ Asym.	3-6
[6]	$19g\log g$ sym. + $O(k)$ Asym.	2
[4]	$O(g)$ Sym. + $O(g)$ (HE+HM+HA) + $O(k)$ Asym.	-
Yao-PFE (HE-OEP)	$O(g)$ Sym. + $O(g)$ (HE+HA) + $O(k)$ Asym.	-
Yao-PFE (SN-OEP)	$O(g\log g)$ Sym. + $O(k)$ Asym.	1

2 Preliminaries

Notations. For a set D, we denote its size by $|D|$. We use the same notation to show the size (number of gates) of a circuit C. We denote a vector by $\boldsymbol{v}$. We use $[a]$ to denote secret sharing of a value a among multiple parties. We intentionally do not specify the sharing scheme used. In our constructions we use a number of different schemes such as XOR sharing, and additive sharing over a finite ring. We denote the ith party's shared by $[a]_i$. We use $\{1...n\}$ to denote the set of positive integers less than equal to n.

Generalized Switching Networks. A switching network SN is a set of interconnected switches that takes N inputs and a set of selection bits, and outputs N values. Each *switch* in the network accepts two ℓ-bits strings as input and outputs two ℓ-bit strings. In our generalized notion of a switch, each of the two output strings can take the value of each of the two input strings. Therefore, assuming input values (x_0, x_1), and output values (y_0, y_1), four different switch types are possible. The two selection bits s_0 and s_1 determine the switch type. In particular, the output of the switch will be $y_1 = x_{s_1}$,and $y_0 = x_{s_0}$. In the rest of the paper, we drop the term generalized and simply refer to these networks as switching networks.

Definition 1 (Mapping for a Switching Network). *The* mapping $\pi : \{1...N\} \rightarrow \{1...N\}$ *corresponding to a switching network SN is defined such that* $\pi(i) = j$ *if and only if after evaluation of SN on the N inputs, the value of the input wire i is assigned to the output wire j (assuming a standard numbering of the input/output wires).*

Note that the mapping π need not be a function since the value for each input wire maybe mapped to multiple output wires in the network. On the other hand, π^{-1} is always a function.

Permutation Networks. A permutation network PN is a switching network for which the mapping is a permutation. In constructing a permutation network, one only needs to use two of the four switch types described above. Particularly, for each switch (also called a permutation cell) with inputs I_0 and I_1, one selection bit is sufficient to select between the two possible outputs (I_0, I_1) and (I_1, I_0).

An optimal construction for a permutation network was proposed by Waksman [22]. The main theorem of [22] states that for any N power of 2, there exists a permutation network with $N \log N - N + 1$ switches, and depth of $2 \log N - 1$. We refer the reader to [22] for the details of the construction which can be efficiently implemented with $O(N \log N)$ complexity.

In the remainder of the paper, if a switch takes two selection bits, we refer to it as a *2-switch*, and otherwise we use the term *1-switch*.

Security Definitions. Security definitions are the standard notions of security against semi-honest adversaries (see Full version).

3 Our Framework for Designing PFE Protocols

Similar to the previous work on private function evaluation, we assume that the following information about the circuit is publicly known: the number of gates in the circuit, the number of each party's input wires, and the number of output wires. Everything else about the circuit is considered private information. We aim to hide the circuit through the CTH and PGE functionalities discussed earlier. In this section we formally describe these functionalities and explain how they can be combined to obtain a PFE.

Our interpretation of sharing (denote using []) in the following discussion is very general. In the GMW-based PFE we use XOR sharing, for arithmetic circuits we use additive shares over a finite ring, and in Yao's garbled circuit, one party holds one random key (in a key pair) while the other party holds the mapping of each key to its actual bit value.

The $\mathcal{F}_{\mathcal{CTH}}$ functionality with circuit parameters n (number of input wires), g (number of gates), o (number of output wires), and internal variables $\mathsf{Out}[i,j]$ for $1 \le i \le m$ and $1 \le j \le 2g$ where m is the number of parties, and $\mathsf{Out}[i,j]$ denote P_i's share for the value of the j-th incoming wire in the circuit.

Parties Setup: P_1 computes the mapping $\pi_{\mathcal{C}}$ corresponding to circuit $\mathcal{C}$. He also generates m random vectors $\boldsymbol{t}_i$, $1 \le i \le m$, where $\boldsymbol{t}_i =< t_i[1], \ldots, t_i[2g] >$. P_i for $2 \le i \le m$ generates a random key vector $\boldsymbol{k}_i =< k_i[1], \ldots, k_i[2g] >$.

On Queries:

$\mathsf{OMAP}([x], j)$:

- P_1**'s Input:** $\pi_{\mathcal{C}}$, $\boldsymbol{t}_1, \ldots, \boldsymbol{t}_m$.
- P_i**'s** $(1 \le i \le m)$ **Input:** $[x]_i$, $\boldsymbol{k}_i$, index j for outgoing wire ow_j.

It sends to P_1, $\mathsf{Out}[i,l] = [x]_i \otimes k_i[l] \otimes t_i[l]$ for all l where $\pi_{\mathcal{C}}(j) = l$. Other parties do not receive any output.

$\mathsf{Reveal}(j)$:

- P_i**'s** $(1 \le i \le m)$ **Input:** index j for the incoming wire iw_j.

It reveals $\mathsf{Out}[i,j]$ to P_i for $i \ge 2$. (Note that P_i can unblinds $\mathsf{Out}[i,j]$ using $k_i[j]$ and recover his fresh random share of $[x]_i \otimes t_i[j]$.)

Fig. 3. The Circuit-Topology Hiding Functionality ($\mathcal{F}_{\mathcal{CTH}}$)

CTH *Functionality.* As described in the introduction, the interconnection of wires in the circuit can be represented by a mapping $\pi_{\mathcal{C}}$. The CTH functionality is responsible for obliviously applying this mapping to the values of the input wires and the intermediate wires in the circuit, in an *on-demand* fashion. Our definition of the CTH functionality captures this useful property refered to as *on-demand mapping* via use of the OMAP/Reveal queries. The OMAP queries allow the participants in the CTH to feed their shares of the values for each outgoing wire to the mapping (individually) and obtain the mapped/blinded outcomes for each incoming wire through the Reveal queries. Our new realization of the CTH functionality as well as the existing constructions all possess the on-demand property (see full version). Figure 3 describes the CTH functionality more formally.

The role of vectors $\boldsymbol{k}$ is to prevent P_1 from learning the other parties' shares and the role of vectors $\boldsymbol{t}$ is to hide P_1's mapping $\pi_{\mathcal{C}}$ from the other parties. The operator $\otimes$ is used to denote a blinding operation. Depending on the CTH realization, the blinding operation can be XORing, modular addition, or homomorphic addition using an additively homomorphic encryption.

The PGE *Functionality.*The PGE functionality can be seen as a PFE protocol where the function is a single gate. A formal description is as follows.

Inputs: P_1's input is G, $[a]_1$, $[b]_1$. P_i's input ($i \geq 2$) is $[a]_i$, $[b]_i$.
Output: P_i's output is fresh random shares of $G(a,b)$, i.e. $[c]_i = [G(a,b)]_i$

Our PFE Framework. These two functionalities can be naturally composed to obtain a complete PFE protocol as described in Figure 4. Our framework can be seen as a way to extend a PFE protocol for one gate (PGE) to a PFE protocol for the complete circuit (by employing the CTH functionality). We give an overview next. In the initialization phase, P_1, knowing the circuit $\mathcal{C}$, sorts the gates topologically and computes the mapping $\pi_{\mathcal{C}}$ corresponding to it. Next, each party distributes shares of its input to all parties. The idea is for the parties to send the value of each outgoing wire to the CTH functionality as soon as it is ready. Hence, at the start of the protocol they send shares of their input values to $\mathcal{F}_{\mathcal{CTH}}$ (the input wires are the first set of outgoing wires in the circuit). The $\mathcal{F}_{\mathcal{CTH}}$ maps these values to the corresponding incoming wires (through OMAP queries). This ends the initialization phase. Parties then individually evaluate the gates. For the current gate being evaluated, parties obtain their shares for the two input values using two Reveal queries to the $\mathcal{F}_{\mathcal{CTH}}$. Next, parties invoke the PGE functionality to receive fresh random shares for the output of the current gate. Parties send these newly learnt shares to the CTH functionality and repeat the process until all gates are evaluated.A visual demonstration of the steps appears in Figure 2.

Theorem 1. *Given secure realizations of $\mathcal{F}_{\mathcal{CTH}}$ and $\mathcal{F}_{\mathcal{PGE}}$ against semi-honest adversaries, the above PFE framework is secure against semi-honest adversaries.*

4 Realizing the CTH Functionality via OEP

What is an Extended Permutation? Before describing our construction in more detail, we need to explain the notion of an *extended permutation.* Recall that a mapping $\pi : \{1...N\} \to \{1...N\}$ is a permutation if it is a bijection (i.e. one-to-one and onto). An extended permutation generalizes this notion as follows:

Definition 2 (Extended Permutation). *For positive integers M and N, we call a mapping $\pi : \{1...M\} \to \{1...N\}$ an extended permutation (EP) if for* *every $y \in \{1...N\}$ there is exactly one $x \in \{1...M\}$ such that $\pi(x) = y$. We often denote x by $\pi^{-1}(y)$.*

Note that in an extended permutation, unlike a standard permutation mapping, the mapping can also replicate/omit elements (as many times as needed) hence allowing the range to be larger or smaller than the domain.

CTH and the OEP Problem. To realize the CTH functionality we have to implement $n+g-o$ OMAP queries, one for each outgoing wire, and $2g$ Reveal queries, one for each incoming wire. When combined, these OMAP/Reveal queries naturally form a problem we refer to as *oblivious evaluation of the extended permutation* (OEP). We define the two-party OEP problem here. In the full version, we describe a natural generalization of the problem to the m-party case and show how to efficiently realize it using m invocations of the two-party variant (wee need the multiparty variant for our GMW-based PFE).

P_1's Inputs: The circuit $\mathcal{C}$ with g gates, n input wires, and o output gates. Denote the corresponding mapping by $\pi_{\mathcal{C}}$.
P_i's Input ($1 \leq i \leq m$): x_j for all input wires j in the circuit belonging to P_i.
Outputs: For $1 \leq i \leq m$, P_i learns his share of the values for the output wires.
Initialization:

1. P_1 sort the gates in the circuit, topologically. Denote the ordered gates by $G_1, \ldots, G_g$.
2. For $1 \leq i \leq m$, P_i distributes shares of his inputs among all parties.
3. For $1 \leq j \leq n$, parties make the query $\mathsf{OMAP}([x_j], j)$ to the $\mathcal{F}_{\mathcal{CTH}}$.

Private Function Evaluation:
For $1 \leq j \leq g$:

1. Parties make the queries $\mathsf{Reveal}(2j-1)$, and $\mathsf{Reveal}(2j)$ to the $\mathcal{F}_{\mathcal{CTH}}$. Denote the output P_i receives by $[a]_i$ and $[b]_i$, respectively.
2. Parties invoke the $\mathcal{F}_{\mathcal{PGE}}$ where P_i's input is $([a]_i, [b]_i)$, while P_1's input also includes the gate functionality (G_j). Each party P_i receives its share of the gate's output, i.e. $[G_j(a,b)]_i$.
3. If $j < g-o$, parties send the query $\mathsf{OMAP}([G_j(a,b)], n+j)$ to $\mathcal{F}_{\mathcal{CTH}}$.

For $g-o < j \leq g$, parties reveal their shares of $[G_j(a,b)]$, and everyone reconstructs the value of the o output wires.

Fig. 4. A General Framework For m-Party PFE of Circuits

Definition 3 (The Two-party OEP Problem: $\mathsf{2\text{-}OEP}(\pi, \boldsymbol{x}, \boldsymbol{t})$). *In this problem, the first party P_1 holds an extended permutation $\pi : \{1...M\} \to \{1...N\}$ for two positive integers M and N, and a blinding vector $\boldsymbol{t} = (t_1, \ldots, t_N)$ while the second party P_2 holds a vector of inputs $\boldsymbol{x} = (x_1, \ldots, x_M)$. Both the x_is and*

t_is are ℓ-bit strings where ℓ is a positive integer. At the end of the protocol, P_2 learns $(x_{\pi^{-1}(1)} \oplus t_1, \ldots, x_{\pi^{-1}(N)} \oplus t_N)$[2]*, while P_1 does not learn anything.*

4.1 A New OEP Protocol

Next, we design a novel OEP protocol that improves on the efficiency of the above constructions. First, we show how to efficiently implement any extended permutation using a switching network. Then, we design a new and efficient protocol for oblivious evaluation of a switching network (OSN).

Building EPs out of Switching Networks. We first show how to construct an extended permutation using a switching network. Note that in a switching network, the number of inputs and outputs are the same which is in contrast to an extended permutation. Since for circuits we only deal with the case of $N \geq M$, the switching network we build for simulating an extended permutation $\pi : \{1...M\} \to \{1...N\}$, takes M real inputs of the EP and $N-M$ additional *dummy inputs.*

We divide the switching network into three components: (i) dummy-value placement, (ii) replication, and (iii) permutation (See Figure 5). Each component takes the output of the previous one as input.

Dummy-value Placement Component. takes the real and dummy values as input and for each real input that is mapped to k different outputs according to π, outputs the real value followed by $k-1$ dummy values. This is repeated for each real value. This process can be efficiently implemented using a Waksman permutation network.

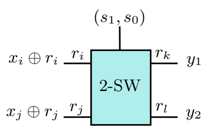
Replication Component. takes the output of the previous component as input. It directly outputs each real value but replaces each dummy input with the real input that precedes it. Each replacement can be implemented using a 1-switch (with a single selection bit) choosing between rows 1 and 3 of Figure 5 (a), as discussed in Section 2. The entire replication phase can be implemented using $N-1$ such switches. At the end of this step, we have the necessary copies for each real input and the dummy inputs are eliminated.

Permutation Component. takes the output of the replication component as input and permutes each element to its final location as prescribed by π. Once again, this can be efficiently implemented using a Waksman permutation network.

Size of the Switching Network for an EP. Adding up the three components, the total number of 1-switches needed to implement the extended permutation described above is $2(N \log N - N + 1) + N - 1 = 2N \log N - N + 1$.

Oblivious Evaluation of Switching Networks (OSN). Next, we design a new and efficient protocol for oblivious evaluation of a generalized switching network. In this problem, P_2 holds the input vector $\boldsymbol{x}$ while P_1 holds the selection bits

[2] For simplicity we use XOR as the blinding function but one can replace XOR with any other natural blinding function.



(s_1, s_0)	y_1	y_2
$(0,0)$	$x_i \oplus r_k$	$x_i \oplus r_l$
$(0,1)$	$x_i \oplus r_k$	$x_j \oplus r_l$
$(1,0)$	$x_j \oplus r_k$	$x_i \oplus r_l$
$(1,1)$	$x_j \oplus r_k$	$x_j \oplus r_l$

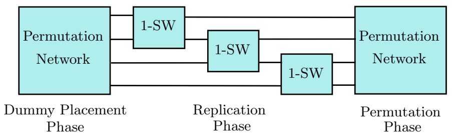


Fig. 5. (a) A 2-Switch (Left), (b) A Switching Network for an EP (Right)

into the switching network, and a blinding vector $\boldsymbol{t}$. P_2 learns the output of the network on his vector $\boldsymbol{x}$ blinded using vector $\boldsymbol{t}$. We start with a high level overview. A complete description appears in the full version.

Secure Evaluation of a Single 2-Switch. The idea can be best explained by describing the procedure for secure evaluation of a single 2-switch u in the network. Consider a 2-switch with input wires w_i and w_j and output wires w_k and w_l. P_2 assigns four uniformly random values r_i, r_j, r_k, r_l to the four wires. P_1 holds the blinded values $x_i \oplus r_i$ and $x_j \oplus r_j$ for the two input wires. The goal is to let P_1 learn the blinded values for the output wires (see Figure 5). Particularly, depending on the value of his two selection bits $s_0(u)$ and $s_1(u)$, P_1 learns one of the four possible output pairs: $(x_i \oplus r_k, x_j \oplus r_l)$, $(x_i \oplus r_k, x_i \oplus r_l)$, $(x_j \oplus r_k, x_i \oplus r_l)$, or $(x_j \oplus r_k, x_j \oplus r_l)$.

To implement this step, P_2 creates a table with four rows: $(r_i \oplus r_k, r_j \oplus r_l)$, $(r_i \oplus r_k, r_i \oplus r_l)$, $(r_j \oplus r_k, r_i \oplus r_l)$, and $(r_j \oplus r_k, r_j \oplus r_l)$. Then, P_1 and P_2 engage in a 1-out-of-4 oblivious transfer in which P_2's input is the four rows of the table he just created, and P_1's input is his two selection bits for the switch u. Without loss of generality suppose that P_1's selection bits are 0, and 0. Hence, P_1 retrieves the first row in the table, i.e. $(r_i \oplus r_k, r_j \oplus r_l)$. He then XORs $x_i \oplus r_i$ and $r_i \oplus r_k$ to recover $x_i \oplus r_k$ and XORs $x_j \oplus r_j$ and $r_j \oplus r_l$ to recover $x_j \oplus r_l$, i.e. the blinded values for the output wires.

Evaluating the Entire Switching Network. The above protocol can be extended to securely evaluate the entire switching network in constant round. In an *offline stage*, P_2 generates a set of random values for every wire in the network, and computes a table for each as described above. Then, P_1 and P_2 engage in a series of parallel 1-out-of-4 oblivious transfers, one for each switch, where P_1 learns a single row of each table according to his selection bits.

In the *online stage*, P_2 blinds his input vector using the randomness for the input wires, and sends them to P_1. P_1 now has all the information necessary to evaluate the switches in the network in a topological order, and recover the blinded values for the output wires (at this stage, P_1 locally performs a sequence of XORs discussed above). He then applies an additional layer of blinding using his random vector $\boldsymbol{t}$, and returns the result to P_2. P_2 can remove his own blinding (i.e. the randomness he generated for the output wires in the network) to learn the output of the switching network blinded only with P_1's vector $\boldsymbol{t}$.

The above OSN protocol runs in a constant number of rounds and requires one invocation of an oblivious transfer per switch in the network. We omit the proof of the following theorem.

Theorem 2. *In the OT-hybrid model, the above OSN protocol (and the resulting OEP) is secure against semi-honest adversaries.*

Efficiency of the New OEP. We can now evaluate the efficiency of the OEP protocol that results from applying our OSN construction to the switching network corresponding to an EP. As discussed earlier, the total number of switches needed to implement an extended permutation $\pi : \{1...M\} \rightarrow \{1...N\}$ is $2N \log N - N + 1$. Furthermore, we only need to use 1-switches to implement an EP which means we only need 1-out-of-2 OT as opposed 1-out-of-4 OT. This yields an OEP protocol with $O(k)$ public-key operations and $4N \log N - 2N + 2$ symmetric-key operations. The communication of the protocol is dominated by $O(N \log N)$ hash values.

How OSN Realizes $\mathcal{F}_{\mathcal{CTH}}$ Queries. It remains to show how our OSN implementation of OEP realizes the queries in $\mathcal{F}_{\mathcal{CTH}}$. While it is obvious that our OSN protocol securely performs all the OMAP/Reveal queries combined, for it to fully satisfy the CTH, we need the ability to make these queries on-demand (see full version for details).

5 Efficient PFEs from MPC

5.1 Multi-Party Private Function Evaluation

In this section we apply our framework to the seminal GMW protocol to obtain a multi-party PFE variant. In particular, we need to describe how the CTH and the PGE functionalities are designed and then plug them into the framework to obtain the desired multiparty PFE. We implement the PGE functionality by means of a *multi-party private gate evaluation* (m-XOR-PGE(G, a, b)) protocol. In such a protocol, only P_1 knows the functionality of the gate G while each party holds his XOR share of the input bits a and b and obtains his XOR share of the output bit $G(a, b)$. See full version, for an efficient instantiation based on oblivious transfer. The protocol requires the same number of OTs as a single gate evaluation in the standard GMW. Hence, making the gate functionality private comes for free in terms of computation or communication.

For the CTH functionality, we can use the multiparty variant of either the HE-OEP or the SN-OEP constructions discussed earlier, where each party uses his XOR shares of the outgoing wires as input to the OEP and obtains his share of the value for the incoming wires.

The following theorem is implied by the security of our framework (Theorem 1), secure instantiations of the OEP and the PGE functionalities and a standard sequential composition theorem [23].

Theorem 3. *Given that the OEP and* m-XOR-PGE *protocols are secure against semi-honest adversaries, the Multi-Party PFE protocol based on our framework is also secure against semi-honest adversaries.*

Efficiency. The resulting protocol requires a single invocation of the m-OEP protocol (even though the protocol is executed in an on-demand fashion), and one invocation of the m-XOR-PGE per gate. Using the HE-OPE instantiation, we obtain a protocol with linear complexity (linear number of exponentiations), and using the SN-OPE, we obtain a protocol that uses $O(m^2g+mg\log g)$ invocations of OT ($O(m^2g)$ for the PGE and $O(mg\log g)$ for the OEP). The number of rounds is equal to the number of gates since they are evaluated sequentially.

5.2 Private Function Evaluation for Arithmetic Circuits

PGE *for Arithmetic Circuits.* Let $E = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a semantically secure and additively homomorphic encryption scheme. Suppose $a = [a]_1 + [a]_2$ and $b = [b]_1 + [b]_2$ are the inputs to the gate, and $c = [c]_1 + [c]_2$ is the output of the gate (where the addition occurs over the domain of plaintexts for the encryption scheme). $[a]_i, [b]_i, [c]_i$ are the shares of P_i. In order to hide the functionality of the gate, we design a PGE protocol in which P_2's actions are independent of the functionality of the gate (i.e. addition or multiplication). To achieve this, P_2 sends to P_1 encryption of $[a]_2, [b]_2$, and $[a]_2[b]_2$. Given these three ciphertexts, P_1 can compute an encryption of both the sum and the product of a and b using homomorphic properties of the scheme. He then sends an encrypted random shares of the outcome to P_1 to decrypt (See Full version for details). It is easy to see that the protocol is secure again semi-honest adversaries if the encryption scheme is semantically secure. We omit the proof of the following theorem.

Theorem 4. *Given $E = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ a semantically secure encryption scheme,* 2-Arith-PGE *protocol is secure against semi-honest adversaries.*

We plug in the above PGE and our HE-OEP protocols in our general framework to obtain an efficient and secure 2PC for arithmetic circuits with *linear complexity.* The following theorem is implied by the security of our framework (Theorem 1), secure instantiations of the OEP and the PGE functionalities and a standard sequential composition theorem [23].

Theorem 5. *Given that the OEP and* 2-Arith-PGE *protocols are secure against semi-honest adversaries, the 2-Party Arithmetic PFE protocol based on our framework is secure against semi-honest adversaries.*

Efficiency. Each PGE invocation requires a constant number of public-key operations adding up to a total of $O(g)$ public-key operations. The HE-OEP has a linear complexity leading to a PFE protocol with similar complexity. The number of rounds is equal to the number of gates since they are evaluated sequentially.

5.3 A Constant-Round Two-Party PFE

In this section we apply the PFE framework to Yao's garbled circuit protocol. We only describe the high level ideas here. A full description of the protocol (2-PFE) appears in the Full version of the paper [21]. At first sight, it may not be obvious how to interpret the sharing mechanism in Yao's protocol. But a closer look at the garbling and evaluation steps reveals that the bit value a for a wire in the circuit is shared by having P_2 (garbler) hold the mapping of a pair of random keys to their bit value ($k^0 \rightarrow [a]_2$,$k^1 \rightarrow \overline{[a]_2}$), and P_1 (the evaluator) holding one of the two keys ($k^{[a]_1}$). Note that one may wonder why we do not simplify the sharing scheme by always letting $[a]_2 = 0$. But such a sharing would indeed be insecure in our PFE framework, and more specifically would allow the evaluator to learn values for the intermediate wires as he evaluates the circuit (since he creates and knows the mapping of keys). Making the CTH component work with this sharing scheme turns out to be the main technical difficulty in designing an efficient Yao-base PFE.

General Idea. Recall Yao's garbled circuit protocol in the semi-honest case. In our construction, the evaluator is the party who holds the circuit, while we intend to hide the circuit from the garbler. We need to hide the topology of the circuit from him using the CTH functionality: first, the Garbler generates his own random shares for the output wires of all the gates in the circuit (i.e. the permuted garbled key pairs for all those wires). Next, he sends all his shares to the CTH functionality, and receives his output which are his shares for the input wires to all the gates in the circuit (i.e. garbled key pairs for all those wires). The garbler now has all garbled keys he needs to garble the circuit. If we assume that all the gates are NAND, there would be no need to hide the gates functionalities. Therefore, our $\mathcal{F}_{\mathcal{PGE}}$ functionality realization consists of the normal garbling of the gates by the garbler and the standard evaluation of the gates by the evaluator. Next, we go into the details of each component and address some of the subtleties that arise.

PGE *Realization.* Realization of the $\mathcal{F}_{\mathcal{PGE}}$ functionality is simple. Lets assume that the inputs are shared using the above sharing scheme. P_2 first randomly generates his own share of the output wire for the current gate, which is basically generating two random keys and assigning them to bits zero and one. He then sends his share to CTH functionality. Upon receiving his shares for input wires to the gates, from CTH functionality, P_2 garbles each gate using his shares for the input and output wires of the gate. He then sends the garbled gates to P_1 who can use his own share of the input wires to ungarble a single row and learn his own share of the output wire.

We now need to integrate our CTH realization with the above PGE construction. For this to work, we need to modify our standard CTH realization, particularly to make sure that its outputs are fresh shares based on the sharing scheme above (i.e. $[a]_1$ and $[a]_2$, and the key pair are fresh and random).

CTH *Realization.* During the evaluation, P_1 needs to XOR his share with its corresponding blinding value(s) to obtain his correct input share for evaluating the next garbled gate. But observing which blinding value enables correct decryption of the next garbled gate (potentially) reveals the value of that intermediate wire. To avoid this issue, we need to ensure that the shares generated by the CTH are truly random. In particular, we need to ensure that P_1 cannot associate the first blinding with key 0 and the second blinding with key 1. As a first solution, P_2 randomly swaps the key pairs to prevent such association by P_1.

P_2 Swaps Each Key Pair Randomly. We solve this problem by having P_2 swap each key pair randomly and independently (using a random bit-vector $\boldsymbol{v}$) before using them in the OMAP queries (for the CTH). Each pair should be swapped using a different bit since using the same bit would reveal whether the bit values for certain intermediate wires are the same or not. If the first(second) blinding is used for two or more wires we learn that their value is the same, though we don't know if it zero or one. This solves the issue above, but undermines correctness of the protocol. When P_2 sends the swapped key pairs to P_1, he gets back an extended permuted (and blinded) set of key pairs. As a result, P_2 does not know the correct order for each pair, and will not be able to perform the garbling of the gates without knowing which key is for 0 and which is for 1.

P_1 and P_2 Jointly Swap Each Key Pair Into Its Original Form. A naive fix would be to attach each "swapping bit" to its corresponding key pair as it goes through the CTH, and reveal the bit to P_2 as part of the output of the CTH, who then uses it to swap the key pair back to its original order. But this would allow P_2 to learn some information about π_C (and the topology of the circuit) by comparing the swapping bits in the input and output key pairs for the CTH.

To address this issue, P_1 and P_2 perform this step together, each holding an XOR share of swapping bits. In particular, the random bit vector $\boldsymbol{v}$ will be fed to the CTH, but P_2 only learns a blinded version, i.e. $v''_i = v_{\pi^{-1}(i)} \oplus v'_i$ for $1 \leq i \leq 2g$, where the blinding vector $\boldsymbol{v'} = (v'_1, \ldots, v'_{2g})$ is only known to P_1. To swap each key pair back to its original order, P_1 first swaps the pair using v'_i, and sends it to P_2. P_2 then swaps it one more time using v''_i which puts the key pair back in its original order. Of course, at this point, the key shares are fresh and random.

If we use a homomorphic-based OEP, this solution is sufficient, but when using the CTH functionality in a black-box way, and particularly when using our SN-OEP construction, there is one more issue to address. The described solution does not use the OEP in a black-box fashion, since P_1 needs to swap the outcome using $\boldsymbol{v'}$, before sending it to P_2. But if the pair is swapped using a random bit vector not known to P_2, he cannot use the appropriate random values to unblind the result (recall the final step of the OEP where P_2 removes his blinding from the output).

P_1 Does his Swapping Using an OSN Protocol. To handle this problem, we require that P_1's swapping procedure based on the bit-vector $\boldsymbol{v'}$ takes place as part of an oblivious switching network evaluation where the v'_is are P_1's selection bits to the network. This requires the use of an additional layer of switches attached to the original switching network for the OEPs. This also has the advantage of making the usage of the OEP and the OSN protocols black-box.

When using our SN-OEP in the above construction, the total number of symmetric operations required for the protocol is $8g \log 2g + 5g + 2$. We discuss our efficiency in detail in the Full version [21] where we also prove the following theorem.

Theorem 6. *Given that the OSN and the OEP protocols are secure against semi-honest adversaries, and that Yao's protocol uses a symmetric-key encryption with related-key security, the* 2-PFE *protocol is secure against semi-honest adversaries.*

References

1. Abadi, M., Feigenbaum, J.: Secure circuit evaluation. Journal of Cryptology 2, 1–12 (1990)
2. Kolesnikov, V., Schneider, T.: A practical universal circuit construction and secure evaluation of private functions. In: Tsudik, G. (ed.) FC 2008. LNCS, vol. 5143, pp. 83–97. Springer, Heidelberg (2008)
3. Sadeghi, A.R., Schneider, T.: Generalized universal circuits for secure evaluation of private functions with application to data classification. In: Lee, P.J., Cheon, J.H. (eds.) ICISC 2008. LNCS, vol. 5461, pp. 336–353. Springer, Heidelberg (2009)
4. Katz, J., Malka, L.: Constant-round private function evaluation with linear complexity. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 556–571. Springer, Heidelberg (2011)
5. Schneider, T.: Practical secure function evaluation (2008)
6. Valiant, L.: Universal circuits (preliminary report). In: Proceedings of the Eighth Annual ACM STOC, pp. 196–203 (1976)
7. Shpilka, A., Yehudayoff, A.: Arithmetic circuits: A survey of recent results and open questions (2010)
8. Raz, R.: Elusive functions and lower bounds for arithmetic circuits. In: Proceedings of the 40th Annual ACM STOC (2008)
9. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proceedings of the 41st Annual ACM, STOC 2009, pp. 169–178. ACM (2009)
10. Huang, Y., Evans, D., Katz, J.: Private set intersection: Are garbled circuits better than custom protocols? In: Proceedings of 19th NDSS Conference (2012)
11. Wang, G., Luo, T., Goodrich, M.T., Du, W., Zhu, Z.: Bureaucratic protocols for secure two-party sorting, selection, and permuting. In: Proceedings of the 5th ACM ASIACCS, pp. 226–237 (2010)
12. Du, W.: A Study of Several Specific Secure Two-party Computation Problems. PhD thesis, Department of Computer Sciences, Purdue University (2001)
13. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game. In: Proceedings of the Nineteenth Annual ACM, STOC 1987, pp. 218–229. ACM (1987)
14. Yao, A.C.C.: How to generate and exchange secrets. In: 27th Annual Symposium on Foundations of Computer Science, pp. 162–167 (October 1986)
15. Cramer, R., Damgård, I., Nielsen, J.B.: Multiparty computation from threshold homomorphic encryption. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 280–299. Springer, Heidelberg (2001)
16. Franklin, M., Gondree, M., Mohassel, P.: Multi-party indirect indexing and applications. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 283–297. Springer, Heidelberg (2007)
17. Choi, S.G., Hwang, K.-W., Katz, J., Malkin, T., Rubenstein, D.: Secure multi-party computation of boolean circuits with applications to privacy in on-line marketplaces. In: Dunkelman, O. (ed.) CT-RSA 2012. LNCS, vol. 7178, pp. 416–432. Springer, Heidelberg (2012)

18. Nielsen, J.B., Nordholt, P.S., Orlandi, C., Burra, S.S.: A new approach to practical active-secure two-party computation. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 681–700. Springer, Heidelberg (2012)
19. Ishai, Y., Kilian, J., Nissim, K., Petrank, E.: Extending oblivious transfers efficiently. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 145–161. Springer, Heidelberg (2003)
20. Beaver, D.: Precomputing oblivious transfer. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 97–109. Springer, Heidelberg (1995)
21. Mohassel, P., Sadeghian, S.: How to hide circuits in mpc: An efficient framework for private function evaluation (2013), `http://eprint.iacr.org/`
22. Waksman, A.: A permutation network. J. ACM 15, 159–163 (1968)
23. Canetti, R.: Security and composition of multiparty cryptographic protocols. Journal of Cryptology 13(1), 143–202 (2000)

Multi-party Computation of Polynomials and Branching Programs without Simultaneous Interaction

S. Dov Gordon[1,⋆], Tal Malkin[2,⋆⋆], Mike Rosulek[3,⋆⋆⋆], and Hoeteck Wee[4,†]

[1] Applied Communication Sciences
[2] Columbia University
[3] University of Montana
[4] George Washington University

Abstract. Halevi, Lindell, and Pinkas (CRYPTO 2011) recently proposed a model for secure computation that captures communication patterns that arise in many practical settings, such as secure computation on the web. In their model, each party interacts only once, with a single centralized server. Parties do not interact with each other; in fact, the parties need not even be online simultaneously.

In this work we present a suite of new, simple and efficient protocols for secure computation in this "one-pass" model. We give protocols that obtain optimal privacy for the following general tasks:

– Evaluating any multivariate polynomial $F(x_1, \ldots, x_n)$ (modulo a large RSA modulus N), where the parties each hold an input x_i.
– Evaluating any read once branching program over the parties' inputs.

As a special case, these function classes include all previous functions for which an optimally private, one-pass computation was known, as well as many new functions, including variance and other statistical functions, string matching, second-price auctions, classification algorithms and some classes of finite automata and decision trees.

1 Introduction

Most of the literature on secure multi-party computation assumes that all parties remain on-line throughout the computation. Unfortunately, this assumption

⋆ Parts of this work was completed while the author was a postdoctoral researcher at Columbia University.

⋆⋆ Supported in part by NSF grant CCF-1116702 and by the the Intelligence Advanced Research Project Activity (IARPA) via Department of Interior National Business Center (DoI / NBC) contract Number D11PC20194. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoI/NBC, or the U.S. Government.

⋆⋆⋆ Supported by NSF grant CCF-1149647.

† Supported by NSF CAREER Award CNS-1237429.

is problematic in many emerging environments, where the parties are often disconnected from the network due to geographic or power constraints. Moreover, the protocols typically require each party to broadcast a large number of messages to the other parties, which can be quite impractical in large distributed networks. We would like to minimize interaction to the greatest extent possible due to practical communication and bandwidth considerations — ideally, each party would need to send only one message.

We consider secure computation in a one-pass client-server model put forth in a recent work of Halevi, Lindell and Pinkas [13].[1] In this model, there is a single server and multiple clients, and the goal is for the server to securely compute some function of the inputs held by the respective clients. Each client connects to the server once (hence "one-pass") and interacts with it, without any other client necessarily being connected at the same time. In particular, there is no need for any two clients to interact. This model is applicable in settings where maintaining constant network connectivity can be problematic — for example, when deployed troops are communicating with the central command center. It is also applicable in situations where the participants cannot be coordinated for social reasons. Imagine trying to get thirty program committee members across different time zones online at the same time to cast a vote. Instead, in the one-pass model, each will receive an email instructing them to login to the server at their leisure. When all participants have done so, the server can compute the output and post the data to a website (or email it out). Similarly, if a website would like to gather data from its visitors, it is unreasonable to ask that they remain logged-in to the site for the duration of the computation. Instead, as they login, they can upload the relevant data according to the protocol, assured of their privacy, and the server can compute the agreed-upon function offline.

1.1 Security for the One-Pass Model

We briefly outline the security model for the one-pass client-server setting and previous results of Halevi et al. [13] — hereafter, "HLP." First, observe that secure computation in this setting is easy if the server is always honest, and is trusted with user data: each client simply sends its input to the server, encrypted under the server's public key; the server will then perform all of the computation. However, assuming that the server is completely honest is not realistic. Instead, we aim to protect the privacy of the honest parties' inputs even amidst a malicious server that may collude with some subset of the clients. Together with the requirement that the protocol be one-pass, this imposes inherent limitations on what we can securely compute in this model. To see why this is the case, consider parties $P_1, P_2, \ldots, P_n$ computing some function $f(x_1, \ldots, x_n)$, where party P_i holds x_i and the parties go in order $P_1, P_2, \ldots, P_n$. If the server colludes with the last t parties, then the correctness and one-pass nature of the protocol imply that the coalition can compute the "residual function"

[1] The ideas of "non-interactive" and "one-pass" computations can be further traced back to [19, 15]. See Section 1.3.

$f(x_1, \ldots, x_{n-t}, \cdot, \cdots, \cdot)$, on *any* choice of a t-tuple $(z_{n-t+1}, \ldots, z_n)$, and for arbitrarily many such choices. In other words, *inherent* to this one-pass model is the fact that parties $P_1, \ldots, P_{n-t}$ must disclose enough information about their inputs to allow the remaining parties to correctly evaluate the residual function $f(x_1, \ldots, x_{n-t}, \cdot, \cdots, \cdot)$. Once the last parties have this information, nothing can prevent them from using it repeatedly. This is in stark contrast to the standard interactive model for secure computation, where the adversary only learns the output of the computation on a single set of inputs, and which allows us to securely compute every efficiently computable function [20, 11].

Due to these inherent limitations of the one-pass model, the "best possible" security guarantee that one could hope for is that the protocol reveals *no more information* than what is revealed by oracle access to this residual function $f(x_1, \ldots, x_{n-t})$. Throughout this paper, this will be the notion we mean when we refer to security (following [13], we will also refer to this notion as *optimal privacy*); for completeness, we provide the formal definitions in Section 2.2. HLP [13] presented practical optimally private protocols for sum of inputs, selection, and symmetric functions like majority, and leave as an open problem whether we can obtain practical optimally private protocols for some larger classes of functions. Indeed, there is no clear candidate for such a larger class of functions as the previous protocols are somewhat ad-hoc and seem to rely on different ideas.

Even ignoring the issue of practical efficiency, the aforementioned functions are essentially the only ones for which we have optimally private protocols. The main technical challenge in designing optimally private protocols is as follows: on one hand, the view y_i of the server after interaction with party P_i should encode sufficient information about the first i inputs $x_1, \ldots, x_i$ to be able to compute the function f; on the other hand, in order to establish security, the simulator needs to be able to efficiently reconstruct the view y_i given only oracle access to the residual function $f(x_1, \ldots, x_i, \cdot, \cdots, \cdot)$. HLP formalize this via the notion of *minimum-disclosure decomposition*, which is a combinatorial property of the function itself, providing a necessary condition for the existence of an optimally private protocol. In addition, they demonstrate that every function with this combinatorial property admits *some* optimally private protocol, albeit a highly inefficient one. However, beyond the small classes of functions mentioned above, they do not demonstrate that any function has such a property. Indeed, using pseudorandom functions, they demonstrate that not all functions have a minimum-disclosure decomposition.

1.2 Our Results

We present practical, optimally private protocols for two broad classes of functions: (1) sparse polynomials over large domains, which capture many algebraic and arithmetic functions of interest, such as mean and variance, and (2) read-once branching programs, which capture symmetric functions, string matching, classification algorithms and some classes of finite automata and decision trees (c.f. [16, 15]).[2] Together, these two classes capture all of the functions addressed in the previous work of HLP, and also include many more functions of interest. One such concrete example is a second-price auction (the n-party functionality that returns the *index* of the largest value along with the second largest value). This function is asymmetric, but can be represented as a branching program. A second-price auction with n parties and discrete bids in the range $\{1, \ldots, k\}$ has an associated branching program of width nk^2.

We begin by giving a simplified exposition of the protocols (achieving security against semi-honest adversaries), and outlining the simulation strategies used in the proof of security. In particular, the simulation strategies provide a solution to the *minimum-disclosure decomposition* problem.

Computing Sparse Polynomials. Consider a sparse[3] polynomial F in n variables $X_1, \ldots, X_n$, where party P_i holds an input x_i for variable X_i. The parties go in the order $P_1, \ldots, P_n$. Consider the following polynomial:

$$F_i(X_{i+1}, \ldots, X_n) := F(x_1, \ldots, x_i, X_{i+1}, \ldots, X_n).$$

Informally, party P_i will post to the server an encryption of the coefficients of polynomial F_i. The next party P_{i+1} will homomorphically evaluate an encryption of (the coefficients of) F_{i+1} given its input x_{i+1} and the previous encryption of F_i (Figure 1). To do so, the encryption scheme must be homomorphic with respect to affine functions over the integers. We are able to realize such an encryption scheme from the DCR assumption, which leads to a one-pass protocol for computing sparse polynomials over $\mathbb{Z}_N$, where N is a RSA modulus. Overall, each party does $O(M)$ group operations and sends $O(M)$ group elements, where M is an upper bound on the number of monomials in F.

To establish security of this protocol, we must show a simulator that can efficiently reconstruct the coefficients of F_i given oracle access to appropriate residual function, which in this case is F_i itself. (For technical reasons, the simulator needs to reconstruct not just the encrypted coefficients but the coefficients themselves.) We show that by querying F_i on sufficiently many random points, the simulator can obtain the coefficients of F_i by solving a suitable system of linear equations.

Computing Branching Programs. Consider a layered read-once branching program, where party i holds the input x_i in the i'th layer. Our protocol proceeds by evaluating the branching program in a *bottom-up* manner, "percolating" output labels from the end of the branching program towards the start node. Accordingly, we label the output layer of the branching program L_0, and layers $L_1, \ldots, L_n$ proceed up from there. The parties go in order $P_1, \ldots, P_n$, and party P_i will post to the server an encryption of the output labels on all of the nodes in the i'th layer. The next party, P_{i+1}, generates an encryption of

[2] For technical reasons outlined below, our protocol for computing polynomials relies on having a large input domain (namely, $\mathbb{Z}_N$). On the other hand, the nature of branching programs makes them well-suited to functions with small input domains. Thus these two classes of functions are somewhat incomparable.

[3] That is, F can be written as the sum of poly(n) monomials.

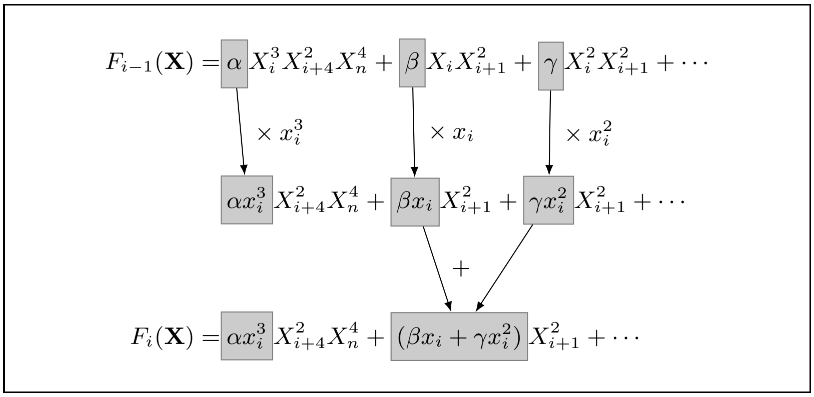

$F_{i-1}(\mathbf{X}) = \boxed{\alpha} X_i^3 X_{i+4}^2 X_n^4 + \boxed{\beta} X_i X_{i+1}^2 + \boxed{\gamma} X_i^2 X_{i+1}^2 + \cdots$
$\times x_i^3$
$\times x_i$
$\times x_i^2$
$\boxed{\alpha x_i^3} X_{i+4}^2 X_n^4 + \boxed{\beta x_i} X_{i+1}^2 + \boxed{\gamma x_i^2} X_{i+1}^2 + \cdots$
$+$
$F_i(\mathbf{X}) = \boxed{\alpha x_i^3} X_{i+4}^2 X_n^4 + \boxed{(\beta x_i + \gamma x_i^2)} X_{i+1}^2 + \cdots$

Fig. 1. Obtaining coefficients of F_i using the coefficients of F_{i-1} and the value of x_i. Shaded boxes are encrypted values. Operations on arrows are homomorphic operations possible in an additively homomorphic scheme.

labels in layer $i + 1$, given x_{i+1} and an encryption of labels in the i'th layer (Figures 2 & 3). Due to the simplicity of the percolation operation, it suffices to use an encryption scheme which is homomorphic with respect to the identity map (i.e., *re-randomizable*). Such an encryption scheme may be realized from the DCR, DDH and DLIN assumptions (the latter two instantiations are important for compatibility with Groth-Sahai proofs [12]). Overall, each party does $O(w)$ group operations and sends $O(w)$ group elements, where w is an upper bound on the width of the branching program.

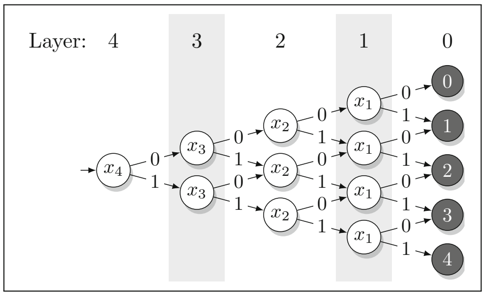


Fig. 2. A layered branching program for computing a tally among 4 parties. Output nodes are darkly shaded.

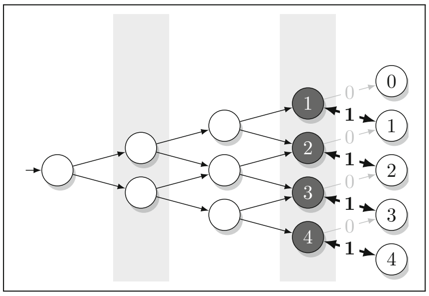


Fig. 3. How party #1 truncates the branching program, corresponding to input $x_1 = 1$

To establish security of this protocol, we must show a simulator that can efficiently compute the labels that the protocol assigns to the layer corresponding to the last honest party, given oracle access to the appropriate residual function. For each node u in the i'th layer, the simulator runs a depth-first search to find a path to u from the start node in the branching program. The path determines a set of inputs on which to query the residual function; the result of the query will be the label on the node u.

The Full-fledged Protocol: More Details. The outline above is a little over-simplified. The parties will in fact need to use a homomorphic *threshold* encryption scheme, which is also re-randomizable, in order to provide "circuit privacy" (that is, hide the homomorphic operations). Roughly speaking, the i'th party P_i's message will be encrypted under the public keys of parties $P_{i+1}, \ldots, P_n$ and the server, so that the message will be private unless all of these parties and the server are corrupted. The use of homomorphic threshold encryption here is analogous to previous constructions [13].

The protocols outlined above obtain optimal privacy against only semi-honest adversaries. To achieve security against malicious adversaries, we can use a generic GMW-style compiler via non-interactive zero-knowledge proofs in the random oracle model, in line with previous work. For our branching-program protocol, we provide an alternative method, in the standard model, that relies on Groth-Sahai proofs. The same approach does not apply to our polynomial-evaluation protocol, since it requires an additively homomorphic encryption scheme, and none are known that are compatible with Groth-Sahai proofs.

As with previous constructions, our protocols can often be extended to handle arbitrary ordering of the players (which is useful in such an asynchronous interaction setting). Indeed, this is the case for our polynomial evaluation protocols. Our branching-program protocol can also allow for arbitrary ordering if the function computed is such that the branching program can be adjusted "on the fly" based on the order in which the parties show up; this is the case for all symmetric functions, as well as some asymmetric ones such as the second-price auction mentioned above.

Finally, we note that while the previously known constructions of [13] are captured as special cases of our two protocols, our technical novelty over these previous constructions is two-fold. First, for our polynomial-evaluation protocol we provide a novel threshold homomorphic encryption scheme based on the DCR assumption. This is important for extending the expressivity from simple summations to more general polynomials while keeping the protocol practical.[4] Second, proving security for our constructions (in particular, proving that the functions admit minimum-disclosure decompositions) requires much more sophisticated simulation strategies than those required by the previous work. In particular, for the classes of functions considered previously, there is no need to solve systems of linear equations or solve s-t connectivity, as we do in this work.

1.3 Additional Related Work

Related Constructions. Surprisingly, our result statements are similar to the results of Harnik, Ishai and Kushilevitz [14, Section 4] for a very different problem. They showed how to securely compute branching programs and sparse

[4] Recall that if efficiency is not an issue, then we could instead rely on threshold fully homomorphic encryption, or a threshold variant of i-hop garbled circuits [9], as shown in [13].

polynomials[5], where every pair of parties makes a single call to an oblivious transfer channel. In their setting, as in ours, the parties incrementally maintain a succinct representation of the inputs of the first i parties. Beyond that similarity, however, the security goal and the underlying communication model are very different. Specifically, they achieve security in the standard MPC setting where the simulator calls the ideal functionality once (there is no "one pass" restriction); indeed, our simulation strategy is very different from theirs. An interesting open problem is to adapt their result on linear branching programs to our setting; the key technical obstacle appears to be solving the analogue of s-t connectivity on the computation graph for linear branching programs.

Related Models. There is a large body of work considering the general theme of secure computation with a restricted communication pattern. Sander, Young and Yung [19] were the first to put forth the notion of 'non-interactive' secure computation, but only in the context of two-party computation. Extensions to the multi-party setting were addressed recently in the work of Ishai et al. [17]. These are essentially 'two-pass' protocols, where it is still possible to securely compute any efficiently computable function. Secure computation in two passes was also recently considered by Asharov et al. [1].

The notion of one-pass computation was considered by Ibrahim, Kiayias, Yung and Zhou [15]. The notion of security is however quite different – roughly speaking, they do not allow the server to collude with the clients, which is in some sense the main source of technical difficulty in the model we study here; their main goal is to minimize server's storage. Ibrahim et al. also provided an efficient protocol for computing branching programs in their model. We note that their protocol is very different from ours: (1) the computation is done in a top-down manner, whereas ours is done in a bottom-up manner; and (2) the transitions from one layer to the next is encoded using a degree w polynomial where w is the width of the branching program, and the parties homomorphically evaluate a degree w polynomial on ciphertexts. The authors showed how to realize the latter based on only the DCR assumption, whereas our protocol may be based on either the DDH, DLIN, or DCR assumptions. The idea for evaluating branching programs in a bottom-up manner originates in a paper of Ishai and Paskin [16] in a different context; their main result exploits the DCR assumption to obtain short ciphertexts.

Other Related Works. We also point out that both classes of functions we consider in this work have been studied in several recent works in a variety of different settings [4, 3, 18, 16, 15].

Organization. We summarize the general one-pass framework [13] (including minimum-disclosure decomposition) Section 2. We provide a generic protocol

[5] They handle sparse polynomials over bits, whereas we consider sparse polynomials over $\mathbb{Z}_N$. In addition, they evaluate the branching programs top-down, whereas we do it bottom-up.

construction in Section 3, and show how to apply it to computing polynomials and branching programs in Sections 4 and 5 respectively. We provide concrete instantiations for underlying primitives in Section 6.

2 General Framework

We design our protocols in the registered public-key infrastructure (PKI) model [2]. We assume that in an initial setup phase every party registers a public and private key pair with a central authority and all the public keys are made known to everyone. We discuss the exact assumptions in the full version.

2.1 Decompositions

As described above, we prove that our protocols leak only the minimum possible information, even if the server colludes with some of the players. We assume that parties $P_1, \dots, P_n$ interact with the server in order, with P_1 going first and P_n going last.[6] As in [13], we define a decomposition of the function f that the players are computing, by a sequence of functions $f_1, \dots, f_n$.

Definition 1 (Decomposition). *For a function $f : D^n \to R$, we define a* decomposition *of f by a tuple of n functions, $f_1, \dots, f_n$, where $f_1 : D \to \{0,1\}^*$, $f_i : \{0,1\}^* \times D \to \{0,1\}^*$ for $1 < i < n$, and $f_n : \{0,1\}^* \times D \to R$, such that for all $(x_1, \dots, x_n) \in D^n$, it holds that $f_n(f_{n-1}(\cdots f_2(f_1(x_1), x_2) \cdots, x_{n-1}), x_n) = f(x_1, \dots, x_n)$. We define a partial decomposition inductively as $\tilde{f}_1(x_1) = f_1(x_1)$ and $\tilde{f}_i(x_1, \dots, x_i) = f_i(\tilde{f}_{i-1}(x_1, \dots, x_{i-1}), x_i)$.*

Minimum-Disclosure Decompositions: As in the work of Halevi et al. [13], we use the notion of a *minimum-disclosure decomposition* to argue that our protocols reveal as little information as possible. For a function f, a decomposition of f given by $f_1, \dots, f_n$, some fixed inputs $\mathbf{x} = (x_1, \dots, x_n)$, and for all $i \in [n]$, we define the residual function $g_i^{\mathbf{x}}(z_{i+1}, \dots, z_n) = f(x_1, \dots, x_i, z_{i+1}, \dots, z_n)$.

Definition 2 ([13]). *A decomposition of function f, given by $f_1, \dots, f_n$, is a* minimum-disclosure decomposition *if there exists a probabilistic, black-box simulator $\mathcal{S}$ that for any set of inputs $\boldsymbol{x} = (x_1, \dots, x_n)$ having total length m, and any $i \in [n]$, when $\mathcal{S}$ is given black-box access to an oracle computing $g_i^{\boldsymbol{x}}(\cdot)$, the output of the simulator satisfies $\mathcal{S}^{g_i^{x}(\cdot)}(m, n, i) = \tilde{f}_i(x_1, \dots, x_i)$, and the running time of $\mathcal{S}^{g_i^{x}(\cdot)}(m, n, i)$ is polynomial in m and n.*

2.2 Defining Security

Security is defined using the real/ideal world paradigm [10, 13]. In the ideal world, there is a trusted party that computes f, which is represented by some fixed

[6] As noted before, the parties can actually interact with the server in arbitrary order for our polynomial evaluation protocol and in many cases for the branching program protocol as well.

decomposition, $f_1, \ldots, f_n$. Each party P_i gives input x_i to the trusted party. If P_i is honest, or semi-honest, he simply uses the value x_i that was found on his input tape; a malicious $P_i(z)$, with auxiliary information z, may use any input of his choice. We denote the corrupted set of parties by $\mathcal{I} \subset \{P_1, \ldots, P_{n+1}\}$. If $P_{n+1} \notin \mathcal{I}$ (i.e. if the server is honest), the trusted party sends output $f(x_1, \ldots, x_n)$ to the server. If $P_{n+1} \in \mathcal{I}$, then we let i^* denote the largest index such that $P_{i^*} \notin \mathcal{I}$ (i.e. P_{i^*} is the last honest party). The trusted party ignores inputs $(x_{i^*+1}, \ldots, x_n)$ and sends $\tilde{f}_{i^*}(x_1, \ldots, x_{i^*})$ to the adversary controlling $\mathcal{I}$. In this case, we stress that the trusted party does *not* send $f(x_1, \ldots, x_n)$, although this can of course be computed by the adversary once he is given $\tilde{f}_{i^*}(x_1, \ldots, x_{i^*})$. This subtlety becomes important while proving security, because the simulator will have no way to extract the input of malicious party P_j for $j > i^*$.

In the real world, f is computed by a sequence of protocols $\pi = (\pi_1, \ldots, \pi_n)$, where π_i is a two-party protocol between the server and P_i. Each party P_i uses input x_i in π_i, and, as above, if they are honest or semi-honest, they use the input found on their input tape. The server uses his output from π_{i-1} as input to π_i. Each player is also given all $n+1$ public keys, denoted by $\widetilde{\mathrm{PK}}$, which are set up as described at the beginning of this Section.

Let $\mathcal{S}(z)$ denote an ideal-world adversary holding auxiliary input z and corrupting some set of parties $\mathcal{I}$. On input set $\mathbf{x} = (x_1, \ldots, x_n)$ and security parameter κ, we denote the output of $\mathcal{S}(z)$ and server P_{n+1} by $\mathsf{Ideal}_{\bar{f},\mathcal{S}(z),\mathcal{I}}(\mathbf{x}, z, 1^\kappa)$. Let $\mathcal{A}(z)$ denote a real-world adversary holding auxiliary input z and corrupting the set of parties $\mathcal{I}$. On input set $\mathbf{x} = (x_1, \ldots, x_n)$ and security parameter κ, we denote the output of $\mathcal{A}(z)$ and server P_{n+1} by $\mathsf{Real}_{\bar{f},\mathcal{A}(z),\mathcal{I}}(\mathbf{x}, z, \widetilde{\mathrm{PK}}, 1^\kappa)$.

Definition 3 ([13]). *We say that a protocol $\pi = (\pi_1, \ldots, \pi_n)$ securely computes a decomposition $\bar{f} = (f_1, \ldots, f_n)$ with optimal privacy, if π is a minimum decomposition for $\bar{f}$, and if for any non-uniform, PPT adversary $\mathcal{A}(z)$ corrupting some subset of parties $\mathcal{I}$ in the real-world, there exists a non-uniform, PPT adversary $\mathcal{S}(z)$ corrupting $\mathcal{I}$ in the ideal-world such that*

$$\left\{\mathsf{Ideal}_{\bar{f},\mathcal{S}(z),\mathcal{I}}(\boldsymbol{x}, z, 1^\kappa)\right\} \stackrel{c}{=} \left\{\mathsf{Real}_{\pi,\mathcal{A}(z),\mathcal{I}}(\boldsymbol{x}, z, \widetilde{\mathrm{PK}}, 1^\kappa)\right\}.$$

2.3 Homomorphic Threshold Encryption

Our constructions require a (n-out-of-n) threshold encryption scheme which supports the following properties in addition to the standard Enc, Dec, and Gen procedures: (These properties generalize the "layer re-randomizable encryption" in [13, Definition 4.1].)

- To encrypt to a set of users whose corresponding public keys comprise the set S, one simply *aggregates* their public keys via $\widetilde{\mathrm{PK}} \leftarrow \mathsf{Aggregate}(S)$, and then encrypts normally treating $\widetilde{\mathrm{PK}}$ as a normal public key.
- The scheme is *homomorphic* (with respect to a class of functions we specify later when describing our main protocols). More formally, there is a procedure Eval which takes a (possibly aggregated) public key, a ciphertext,

and a function, and outputs another ciphertext. We then require that for all valid keypairs $(\textsc{sk}, \textsc{pk})$, all supported functions f, and all ciphertexts C:

$$\mathsf{Dec}(\textsc{sk}, \mathsf{Eval}(\textsc{pk}, C, f)) = f(\mathsf{Dec}(\textsc{sk}, C))$$

– Given an encryption C under public keys $\textsc{pk}_1, \ldots, \textsc{pk}_n$, the owner of any corresponding secret key $\textsc{sk}_i$, $i \in [n]$, can transform C into a (*fresh*) encryption of the same message, under the remaining $n-1$ public keys. More formally, there is a procedure Strip which takes a (aggregated) public key, a secret key, and a ciphertext, and outputs another ciphertext. We require that, for all valid keypairs $(\textsc{sk}^*, \textsc{pk}^*)$, all $S \ni \textsc{pk}^*$, all plaintexts M, and all C in the support of $\mathsf{Enc}(\mathsf{Aggregate}(S), M)$, we have

$$\mathsf{Strip}(\mathsf{Aggregate}(S), \textsc{sk}^*, C) \approx_s \mathsf{Enc}(\mathsf{Aggregate}(S \setminus \{\textsc{pk}^*\}), M).$$

Semantic Security. For an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ we define the advantage $\mathsf{AdvThEnc}^{\mathcal{A}}(k)$ to be:

$$\left| \Pr\left[U \setminus U^* \neq \emptyset \wedge b = b' : \begin{array}{l} (\textsc{pk}_i, \textsc{sk}_i) \leftarrow \mathsf{Gen}(1^k), i = 1, \ldots, n; \\ (U, U^*, M_0, M_1) \leftarrow \mathcal{A}_1(1^k, \textsc{pk}_1, \ldots, \textsc{pk}_n); \\ b \xleftarrow{\$} \{0,1\}; \\ C \leftarrow \mathsf{Enc}(\mathsf{Aggregate}(\{\textsc{pk}_i \mid i \in U\}), M_b); \\ b' \leftarrow \mathcal{A}_2(C, \{\textsc{sk}_i \mid i \in U^*\}); \end{array} \right] - \frac{1}{2} \right|$$

A threshold encryption scheme is said to be *indistinguishable against chosen plaintext attacks* (IND-CPA) if for all PPT adversaries $\mathcal{A}$, the advantage $\mathsf{AdvThEnc}^{\mathcal{A}}(k)$ is a negligible function in k.

3 Our General Protocol

Our protocols are designed using the following high-level approach, which is essentially an abstraction of that in [13].

1. We begin with a decomposition for the class of functions we are interested in, namely sparse polynomials and read-once branching programs, as described in Sections 4 and 5 respectively. We show that our decompositions are in fact minimal, proving that our protocols are optimally private for these classes of functions.
2. We construct a semi-honest protocol by combining the decomposition with a threshold homomorphic encryption scheme. (See Section 3.1.) For our constructions, the only homomorphic operations we need to support are the identity function and affine functions. In Section 6, we provide concrete instantiations from DDH, DCR and DLIN.
3. We construct a protocol that is secure against malicious parties by having the participants first encrypt their inputs and then prove consistency using suitable NIZKs. We provide a detailed treatment in the design of NIZKs, where we completely specify the witnesses used by the honest provers. (Some of these details were omitted in [13].) These results appear in the full version.

3.1 Protocol for Semi-honest Adversaries

We consider n parties $P_1, \ldots, P_n$, with corresponding registered key pairs $\{(\text{PK}_i, \text{SK}_i)\}_{i\in[n]}$. Let $f_1, \ldots, f_n$ be a decomposition for f in which the parties go in order $1, \ldots, n$. Our protocol is as follows: At a high level, party i sends to the server the ciphertext C_i, which is an encryption of the value $y_i := f_i(y_{i-1}, x_i) = \tilde{f}_i(x_1, \ldots, x_i)$ under the aggregated public key $\widetilde{\text{PK}}_i = \mathsf{Aggregate}(\text{PK}_{i+1}, \ldots, \text{PK}_{n+1})$. Ciphertext C_i is generated by applying the encryption scheme's homomorphic properties to ciphertext C_{i-1}. In more detail:

1. Party P_1 computes $C_1 \xleftarrow{\$} \mathsf{Enc}(\widetilde{\text{PK}}_1, f_1(x_1))$ and sends C_1 to the server P_{n+1}.
2. For $i = 2, \ldots, n$: party P_i receives C_{i-1} from the server, and sends C_i to the server, where:
$$C_i \xleftarrow{\$} \mathsf{Strip}(\widetilde{\text{PK}}_i, \text{SK}_i, \mathsf{Eval}(\widetilde{\text{PK}}_i, C_{i-1}, f_i(\cdot, x_i))).$$
3. Upon receiving C_n from P_n, the server P_{n+1} decrypts the ciphertext using its secret key SK_{n+1} and outputs the result.

From the properties of Eval and Strip, it is easy to see that if all players are honest, then $C_i \approx_s \mathsf{Enc}(\widetilde{\text{PK}}_i, y_i)$ for all i. Correctness then follows from the fact that $f_1, \ldots, f_n$ is a correct decomposition.

Lemma 1 (Semi-honest security). *If* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Aggregate}, \mathsf{Eval}, \mathsf{Strip})$ *is a secure threshold encryption scheme (Section 2.3), then the above protocol is an optimally private protocol for decomposition* $(f_1, \ldots, f_n)$*, against semi-honest adversaries.*

4 Computing Sparse Multivariate Polynomials

In this section we instantiate our general framework to obtain a protocol for evaluating a multivariate polynomial on the parties' inputs. We begin with a simple lemma about learning the coefficients of a multivariate polynomial via oracle queries:

Lemma 2. *Let* $F \in \mathbb{Z}_N[X_1, \ldots, X_n]$ *be a known multivariate polynomial with total degree* d*, where* N *is square-free, and* $d \leq p/2$ *for every prime* p *dividing* N*. Let* M *be the number of monomials in* F*. Fix an (unknown) input to the polynomial* $(x_1, \ldots, x_n) \in (\mathbb{Z}_N)^n$ *and define:*

$$F_i(X_{i+1}, \ldots, X_n) := F(x_1, \ldots, x_i, X_{i+1}, \ldots, X_n)$$

Then, for each i*, it is possible to learn the coefficients of the polynomial* F_i *by making a polynomial number (in* M *and* $\log N$*) of queries to an oracle for* F_i*.*

Proof. Our approach for learning the coefficients of F_i is to simply query F_i on a sufficiently large number of random points (the number of points to be determined later). Then the coefficients of F_i can be viewed as unknowns in a linear system over $\mathbb{Z}_N$, which can be solved via Gaussian elimination. We must show that the linear system uniquely determines F_i with high probability.

Fix i and recall that F is fixed and known. Let us say that a monomial m' in variables $\{X_{i+1}, \ldots, X_n\}$ is *valid* if there exists some monomial $m \in F$ (with nonzero coefficient) such that the degree of X_j is the same in both m' and m, for all $j \in \{i+1, \ldots n\}$. Since F_i is of the form $F(\hat{x}_1, \ldots, \hat{x}_i, X_{i+1}, \ldots, X_n)$, every monomial of F_i must be valid. Then we may restrict our linear system to polynomials whose monomials are all valid, by including unknowns only for the coefficients of valid monomials. Recall that there are at most M valid monomials. Now, it suffices to show that the linear system uniquely determines F_i, among polynomials that contain only valid monomials.

Let p be a prime divisor of N. Fix any polynomial $F' \neq F_i$, where F' contains only valid monomials. Then by the Schwartz-Zippel lemma, we have that F_i and F' agree on q randomly selected points (modulo p) with probability at most $(d/p)^q \leq 1/2^q$. There are at most N^M such multivariate polynomials F', and at most $\log N$ prime divisors of N, so choose $q = Mk \log N \log\log N$. Then by a union bound, we have that F_i agrees with *some* other F' on all q random points modulo *some* prime divisor with probability at most $1/2^k$. By the Chinese Remainder Theorem, the linear system over $\mathbb{Z}_N$ uniquely determines F_i with probability at least $1 - 1/2^k$.

Function Decomposition. The preceding lemma suggests that, given a sparse polynomial F, we may compute its minimum-disclosure decomposition as follows:

> $f_i(\cdot, x_i)$ takes as input the list of coefficients for a polynomial $P(X_i, X_{i+1}, \ldots, X_n)$ and outputs the list of coefficients for the polynomial $P'(X_{i+1}, \ldots, X_n)$ where $P'(X_{i+1}, \ldots, X_n) := P(x_i, X_{i+1}, \ldots, X_n)$.

Specifically, f_i proceeds as follows:

1. For each monomial of P that contains a term of the form X_i^t, multiply that coefficient by x_i^t.
2. For each set of monomials whose degrees in $X_{i+1}, ..., X_n$ are identical, add the coefficients together.

This next Lemma follows directly from Lemma 2.

Lemma 3. *The decomposition described above is a minimum-disclosure decomposition.*

Secure, One-pass Protocols. It is easy to see that $f_i(\cdot, x_i)$ is an affine function of its inputs. Therefore, using our general framework in the preceding section, it suffices to construct a threshold homomorphic encryption scheme that supports computing affine functions on encrypted values. Indeed, we provide such an instantiation based on DCR in the full version.

Theorem 1. *Under the DCR assumption, there is a one-pass protocol, secure against a semi-honest adversary, for evaluating any $F \in \mathbb{Z}_N[X_1, \ldots, X_n]$ with M monomials, where N is a RSA modulus and M and the total degree of F satisfy the bounds given in Lemma 2. The protocol achieves optimal privacy, its runtime is polynomial in M, n, and $\log N$, and it requires $O(M)$ exponentiations per party.*

In Section 6, with further details in the full version, we demonstrate concrete instantiations of NIZKs appropriate to ensure security against malicious adversaries. This leads to the following Theorem.

Theorem 2. *Under the DCR assumption, there is a one-pass protocol in the random-oracle model, secure against malicious adversaries, for evaluating any* $F \in \mathbb{Z}_N[X_1, \ldots, X_n]$ *expressed as a sum of monomials, where* N *is as in Lemma 2. The protocol achieves optimal privacy and it requires* $O(nM|\mathcal{D}|)$ *exponentiations per party (where* $\mathcal{D}$ *denotes the input domain for each party).*

5 Computing Branching Programs

In this section we describe our protocol for computing branching programs.

Overview. A (deterministic) branching program P is defined by a directed acyclic graph in which the nodes are labeled by input variables and every nonterminal node has two outgoing edges, labeled by 0 and 1.[7] An input naturally induces a computation path from a distinguished initial node to a terminal node, whose label determines the output. We rely on a technique of Ishai and Paskin [16] for computing branching programs (BPs) in a bottom-up manner. Let $x_1, \ldots, x_n$ be the inputs to the BP. First, without loss of generality we may make the BP *layered* (defined below), incurring at most a quadratic blow-up in its size (this blow-up may be avoided in specific cases, see [16]). In a layered BP, all nodes can be partitioned into layers $L_0, \ldots, L_n$, with the property that all nodes in layer $i \in \{1, \ldots, n\}$ correspond to input variable X_i and have outgoing edges only into layer $i-1$. (Because we work in a bottom-up manner, we label the output layer L_0, and the topmost layer L_n.) Layer 0 contains only output nodes.

Imagine evaluating a layered BP by "percolating" output labels from the end of the BP towards the start node, as follows.[8] Starting at layer L_0, we do the following: For every edge (u, v) between layer L_i and L_{i-1} that is labeled with the value x_i (that is, if we are at node u and X_i assumes the value x_i, we proceed to node v), copy the output label from node v to node u (there will not be a conflict by the deterministic property of the branching program). Finally, the start node in layer L_n is labeled with the output of the computation.

This process naturally lends itself to a decomposition of the branching program's functionality. Namely, the ith phase of the decomposition outputs the labels of all nodes in layer i. To show that this decomposition is minimum-disclosure, we must argue that an adversary could also learn this information

[7] We note that our protocols work also for more general "linear branching programs", where the edges are labeled with affine functions.

[8] We note that computing branching programs in a top-down manner may also be considered in the one-pass model. Each party simply posts an encryption of the unique active node in its layer. This leads to a minimum-disclosure decomposition if the BP does not have redundant states, which can be achieved using a variant of the Myhill-Nerode algorithm. However, this top-down approach requires the threshold encryption to support the BP's transition function as a homomorphic operation, whereas our bottom-up approach requires only re-randomizability.

by corrupting the server and parties $i+1$ through n in the ideal world. To see why, first assume that all nodes in the branching program are reachable from the start node. Then a path from the start node to some node v in layer i naturally corresponds to a set of inputs that the adversary could query to the residual function. The result of the query is the label that this process would have assigned to node v.

Definitions. We proceed with the details of our protocol:

Definition 4 (Branching program). *A* branching program *on variables* $X = (X_1, \ldots, X_n)$ *with input domain* $\mathcal{D}$ *and output range* $\mathcal{R}$ *is defined by a tuple* $\{G = \{V, E\}, \mathcal{S}_{\mathsf{out}}, \phi_V, \phi_E\}$. V *contains a single start node with in-degree 0, and a set of designated leaf nodes,* $\mathcal{S}_{\mathsf{out}}$*, along with any internal nodes. The function* ϕ_V *assigns each node in* $\mathcal{S}_{\mathsf{out}}$ *with an output value from* $\mathcal{R}$*, and every other node with a variable from* X. ϕ_E *is a function that labels each edge* $(u, v) \in E$ *with values from* $\mathcal{D}$.

Definition 5 (Read-Once, Layered BP). *In a* layered *branching program,* V *can be partitioned into layers* $L_n, \ldots, L_1, L_0 = \mathcal{S}_{\mathsf{out}}$ *such that for any node* $u \in L_i$ *and* $v \in L_j$*, with* $i > j$*, the length of every path from* u *and* v *is exactly* $i - j$*. A layered branching program is read-once if every node in layer* i *is labeled with variable* X_i *(possibly after re-naming the variables).*

Informally, we can think of every node in layer i as having the same height, and the same variable assignment. Looking ahead, layer i will coincide with the input variable of player P_i. We note that any branching program can be turned into a layered branching program with at most a quadratic blowup in the size of V. For simplicity, we will assume that our branching programs are already read-once, layered branching programs.

Function Decomposition. Let $F : \mathcal{D}^n \to \mathcal{R}$ denote the function on $X = (X_1, \ldots, X_n)$ described by a read-once, layered branching program BP. Let $s_i = |\{v \in L_i\}|$ denote the size of layer i in BP. We assume some (arbitrary) ordering on the nodes in each layer: let $(v_1, \ldots, v_{s_i})$ be the ordered nodes of layer i, and $(u_1, \ldots, u_{s_{i-1}})$ the ordered nodes in layer $i-1$. We define $f_i : \mathcal{R}^{s_{i-1}} \times \{x_i\} \to \mathcal{R}^{s_i}$ as follows. Let $\mathsf{in}_j \in \mathcal{R}$ denote the jth input to f_i, and $\mathsf{out}_k \in \mathcal{R}$ denote the kth output. Then $\mathsf{out}_k = \mathsf{in}_j$ if and only if $(v_k, u_j) \in E$, and $\phi_E(v_k, u_j) = x_i$.

Intuitively, this decomposition percolates the output "up" the graph, stripping off layers as it goes. For example, $f_1(\phi_V(\mathcal{S}_{\mathsf{out}}), x_1)$ fixes the variable $X_1 = x_1$ in layer 1, and percolates the resulting output values from layer 0 up to each node in layer 1. The output nodes in $\mathcal{S}_{\mathsf{out}}$ now become irrelevant to the computation. Similarly, $\tilde{f}_i = f_i(\cdots f_2(f_1(\phi_V(\mathcal{S}_{\mathsf{out}}), x_1), x_2) \cdots, x_i)$ strips off layers 0 through $i-1$, labeling all the nodes in layer i with the correct output, and making all layers $j < i$ irrelevant. More specifically, consider two nodes $u_j \in L_i$ and $v_k \in \mathcal{S}_{\mathsf{out}}$. If there exists some path $p = (e_i, \ldots, e_1)$ from u_j to v_k such that $(\phi_E(e_i), \ldots, \phi_E(e_1)) = x_i, \ldots, x_1$, then $\tilde{f}_i(x_1, \ldots, x_i)$ assigns $\phi_V(v_k)$ to node u_j.

Lemma 4. *The decomposition of F described above is a minimum-disclosure decomposition.*

Proof. We must show that for every $i \in [n]$, there exists a simulator $\mathcal{S}^{g_i^{\mathbf{x}}(\cdot)}(m, n, i)$, that outputs $\tilde{f}_i(x_1, \ldots, x_i)$. Recall that the output of $\tilde{f}_i$ contains $s_i = |\{v \in L_i\}|$ values, $\mathsf{out}_1, \ldots \mathsf{out}_{s_i} \in \mathcal{R}$. To compute the value of out_j, the simulator takes the jth node u_j in layer L_i and runs a breadth-first-search on G to find a path from the start node to u_j. Let $x_n, \ldots, x_{i+1}$ denote the input assignments associated with the edges along this path (according to ϕ_E). $\mathcal{S}$ queries his oracle and sets $\mathsf{out}_j = g_i^{\mathbf{x}}(x_{i+1}, \ldots, x_n)$.

Secure, One-pass Protocols. To obtain a secure protocol using our framework in Section 2, we need to specify the homomorphic operation required by party P_i. It is easy to verify that we only need to re-randomize ciphertexts. By our conventions for homomorphic encryption (Section 2.3), re-randomization is performed when P_i strips his secret key's contribution from the ciphertext. We do not require any homomorphic operations beyond this. A formal description of the protocol is in Figure 4.

Branching Programs

Inputs: Player P_i holds input $x_i \in \{0,1\}$. Each also has a full description of the branching program, $BP = \{G = \{V, E\}, \mathcal{S}_{\mathsf{out}}, \phi_V, \phi_E\}$ Let $L_i = \{v_1, \ldots, v_{s_i}\}$ denote the nodes in layer i.

Protocol:

Player P_1 begins the protocol. For each $v_j \in L_1$,

- P_1 finds $u \in \mathcal{S}_{\mathsf{out}}$ such that $(u, v_j) \in E$ and $\phi_E(u, v_j) = x_1$.
- He computes $\psi_j = \mathsf{Enc}(\widetilde{\text{PK}}_2, \phi_V(u))$.

P_1 sends $C_1 := (\psi_1, \ldots, \psi_{s_1})$ to the server.

For $i = 2 \ldots n$:

- Party P_i receives ciphertexts $C_{i-1} = (\psi_1, \ldots, \psi_{s_{i-1}})$ from the server.
- For every $v_j \in L_i$,
  - P_i finds $u_k \in L_{i-1}$ such that $(u_k, v_j) \in E$ and $\phi_E(u_k, v_j) = x_i$. We let ψ_k denote the ciphertext corresponding to u_k.
  - P_i sets $\psi'_j = \mathsf{Strip}(\widetilde{\text{PK}}_i, \text{SK}_i, \psi_k)$.
- P_i sends $C_i := (\psi'_1, \ldots, \psi'_{s_i})$ to the server.

Output: Let C_n be the (single) ciphertext sent from P_n to the server. The server computes and outputs $\mathsf{Dec}(\text{SK}_{n+1}, C_n)$.

Fig. 4. A protocol secure for computing branching program BP

Theorem 3. *Assuming an encryption scheme satisfying the conditions of Section 2.3 w.r.t. the identity function, the protocol in Figure 4 is a one-pass protocol, secure against a semi-honest adversary, for evaluating any read-once, layered branching program. The protocol achieves optimal privacy. For branching programs of width w, the runtime is polynomial in w and n, and it requires $O(w)$ exponentiations per party.*

In Section 6 we provide instantiations of the NIZKs that are necessary to make this protocol secure against malicious adversaries. This gives us the following theorem as well.

Theorem 4. *Assuming an encryption scheme satisfying the conditions of Section 2.3 w.r.t. the identity function, and that the NIZK schemes mentioned above are secure, there is a one-pass protocol, secure against a malicious adversary, for evaluating any read-once, layered branching program. The protocol achieves optimal privacy. For branching programs of width w and output domain $\mathcal{D}$, the runtime is polynomial in w, n and $|\mathcal{D}|$, and it requires $O(nw|\mathcal{D}|)$ exponentiations per party.*

6 Realizing the Required Encryption and NIZK Schemes

In the full version, we present three threshold homomorphic encryption schemes. Two are based on the DDH and DLIN assumptions, respectively, and support homomorphic evaluation of the identity function (i.e., re-randomization). The third is based on the DCR assumption, and supports homomorphic evaluation of affine functions over $\mathbb{Z}_N$. We rely on the first two schemes for branching programs and the last for sparse polynomials. The full details of our malicious-secure protocol are given in the full version. There we also describe concrete and efficient NIZK proofs, consistent with our instantiations of homomorphic threshold encryption, for the statements described in the malicious-secure protocol.

In the random oracle model, it suffices to construct appropriate Σ-protocols and then apply the Fiat-Shamir technique. We additionally use techniques of Cramer et al. [7] to compose simple Σ-protocols using logical conjunction and disjunction. The main challenge then is to show how party P_i can prove that the ciphertexts C_{i-1} and C_i are consistent, in that C_i was derived from C_{i-1} according to the protocol (with the encryption scheme's Strip and Eval operations). We eventually reduce this problem to the task of proving that two ciphertexts encrypt the same value (under different aggregated public keys), for which we provide efficient Σ-protocols.

Our instantiations based on the DDH and DLIN assumptions are compatible with our protocol for evaluating branching programs. For these homomorphic threshold schemes, we describe efficient NIZK proofs in the *standard model*, using the NIZK scheme of Groth and Sahai [12].

Acknowledgements. We thank Yuval Ishai and Yehuda Lindell for helpful discussions.

References

[1] Asharov, G., Jain, A., López-Alt, A., Tromer, E., Vaikuntanathan, V., Wichs, D.: Multiparty Computation with Low Communication, Computation and Interaction via Threshold FHE. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 483–501. Springer, Heidelberg (2012)

[2] Barak, B., Canetti, R., Nielsen, J.B., Pass, R.: Universally composable protocols with relaxed set-up assumptions. In: FOCS, pp. 186–195 (2004)
[3] Benabbas, S., Gennaro, R., Vahlis, Y.: Verifiable delegation of computation over large datasets. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 111–131. Springer, Heidelberg (2011)
[4] Boneh, D., Freeman, D.M.: Homomorphic signatures for polynomial functions. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 149–168. Springer, Heidelberg (2011)
[5] Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)
[6] Camenisch, J., Shoup, V.: Practical verifiable encryption and decryption of discrete logarithms. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 126–144. Springer, Heidelberg (2003)
[7] Cramer, R., Damgård, I., Schoenmakers, B.: Proof of partial knowledge and simplified design of witness hiding protocols. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 174–187. Springer, Heidelberg (1994)
[8] Damgárd, I., Toft, T.: Trading sugar beet quotas - secure multiparty computation in practice. ERCIM News 2008(73) (2008)
[9] Gentry, C., Halevi, S., Vaikuntanathan, V.: i-hop homomorphic encryption and rerandomizable Yao circuits. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 155–172. Springer, Heidelberg (2010)
[10] Goldreich, O.: Foundations of Cryptography. Basic Applications, vol. 2. Cambridge University Press, New York (2004) ISBN 0521830842
[11] Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or a completeness theorem for protocols with honest majority. In: STOC, pp. 218–229 (1987)
[12] Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 415–432. Springer, Heidelberg (2008)
[13] Halevi, S., Lindell, Y., Pinkas, B.: Secure computation on the web: Computing without simultaneous interaction. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 132–150. Springer, Heidelberg (2011)
[14] Harnik, D., Ishai, Y., Kushilevitz, E.: How many oblivious transfers are needed for secure multiparty computation? In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 284–302. Springer, Heidelberg (2007)
[15] Ibrahim, M.H., Kiayias, A., Yung, M., Zhou, H.-S.: Secure function collection with sublinear storage. In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikoletseas, S., Thomas, W. (eds.) ICALP 2009, Part II. LNCS, vol. 5556, pp. 534–545. Springer, Heidelberg (2009)
[16] Ishai, Y., Paskin, A.: Evaluating branching programs on encrypted data. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 575–594. Springer, Heidelberg (2007)
[17] Ishai, Y., Kushilevitz, E., Ostrovsky, R., Prabhakaran, M., Sahai, A.: Efficient non-interactive secure computation. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 406–425. Springer, Heidelberg (2011)
[18] Kruger, L., Jha, S., Goh, E.-J., Boneh, D.: Secure function evaluation with ordered binary decision diagrams. In: ACM Conference on Computer and Communications Security, pp. 410–420 (2006)
[19] Sander, T., Young, A., Yung, M.: Non-interactive cryptocomputing for NC^1. In: FOCS, pp. 554–567 (1999)
[20] Yao, A.C.-C.: How to generate and exchange secrets. In: FOCS, pp. 162–167 (1986)

Quantum-Secure Message Authentication Codes

Dan Boneh and Mark Zhandry

Stanford University
{dabo,zhandry}@cs.stanford.edu

Abstract. We construct the first Message Authentication Codes (MACs) that are existentially unforgeable against a *quantum* chosen message attack. These chosen message attacks model a quantum adversary's ability to obtain the MAC on a superposition of messages of its choice. We begin by showing that a quantum secure PRF is sufficient for constructing a quantum secure MAC, a fact that is considerably harder to prove than its classical analogue. Next, we show that a variant of Carter-Wegman MACs can be proven to be quantum secure. Unlike the classical settings, we present an attack showing that a pair-wise independent hash family is insufficient to construct a quantum secure *one-time* MAC, but we prove that a four-wise independent family is sufficient for one-time security.

Keywords: Quantum computing, MAC,chosen message attacks, post-quantum security.

1 Introduction

Message Authentication Codes (MACs) are an important building block in cryptography used to ensure data integrity. A MAC system is said to be secure if an efficient attacker capable of mounting a chosen message attack cannot produce an existential MAC forgery (see Section 2.2).

With the advent of quantum computing there is a strong interest in post-quantum cryptography, that is systems that remain secure even when the adversary has access to a quantum computer. There are two natural approaches to defining security of a MAC system against a quantum adversary. One approach is to restrict the adversary to issue classical chosen message queries, but then allow the adversary to perform quantum computations between queries. Security in this model can be achieved by basing the MAC construction on a quantum intractable problem.

The other more conservative approach to defining quantum MAC security is to model the entire security game as a quantum experiment and allow the adversary to issue *quantum* chosen message queries. That is, the adversary can submit a superposition of messages from the message space and in response receive a superposition of MAC tags on those messages. Informally, a quantum chosen message query performs the following transformation on a given superposition of messages:

$$\sum_m \psi_m \left|m\right\rangle \quad \longrightarrow \quad \sum_m \psi_m \left|m, S(k,m)\right\rangle$$

where $S(k,m)$ is a tag on the message m with secret key k.

To define security, let q be the number of queries that the adversary issues by the end of the game. Clearly it can now produce q classical message-tag pairs by sampling the q superpositions it received from the MAC signing oracle. We say that the MAC system is *quantum secure* if the adversary cannot produce $q+1$ valid message-tag pairs. This captures the fact that the adversary cannot do any better than trivially sampling the responses from its MAC signing oracle and is the quantum analogue of a classical existential forgery.

1.1 Our Results

In this paper we construct the first quantum secure MAC systems. We begin with a definition of quantum secure MACs and give an example of a MAC system that is secure against quantum adversaries capable of *classical* chosen message queries, but is insecure when the adversary can issue *quantum* chosen message queries. We then present a number of quantum secure MAC systems.

Quantum Secure MACs. In the classical settings many MAC systems are based on the observation that a secure pseudorandom function gives rise to a secure MAC [BKR00, BCK96]. We begin by studying the same question in the quantum settings. Very recently Zhandry [Zha12b] defined the concept of a quantum secure pseudorandom function (PRF) which is a PRF that remains indistinguishable from a random function even when the adversary can issue *quantum* queries to the PRF. He showed that the classic GGM construction [GGM86] remains secure under quantum queries assuming the underlying pseudorandom generator is quantum secure.

The first question we study is whether a quantum secure PRF gives rise to a quantum secure MAC, as in the classical settings. To the MAC adversary a quantum secure PRF is indistinguishable from a random function. Therefore proving that the MAC is secure amounts to proving that with q quantum queries to a random oracle H no adversary can produce $q+1$ input-output pairs of H with non-negligible probability. In the classical settings where the adversary can only issue *classical* queries to H this is trivial: given q evaluations of a random function, the adversary learns nothing about the value of the function at other points. Unfortunately, this argument fails under quantum queries because the response to a *single* quantum query to $H : \mathcal{X} \to \mathcal{Y}$ contains information about all of H. In fact, with a single quantum query the adversary can produce two input-output pairs of H with probability about $2/|\mathcal{Y}|$ (with classical queries the best possible is $1/|\mathcal{Y}|$). As a result, proving that q quantum queries are insufficient to produce $q+1$ input-output pairs is quite challenging. We prove tight upper and lower bounds on this question by proving the following theorem:

Theorem 1 (informal). *Let $H : \mathcal{X} \to \mathcal{Y}$ be a random oracle. Then an adversary making at most $q < |\mathcal{X}|$ quantum queries to H will produce $q+1$ input-output pairs of H with probability at most $(q+1)/|\mathcal{Y}|$. Furthermore, when $q \ll |\mathcal{Y}|$ there is an algorithm that with q quantum queries to H will produce $q+1$ input-output pairs of H with probability $1-(1-1/|\mathcal{Y}|)^{q+1} \approx (q+1)/|\mathcal{Y}|$.*

The first part of the theorem is the crucial fact needed to build quantum secure MACs and is the harder part to prove. It shows that when $|\mathcal{Y}|$ is large any algorithm has only a negligible chance in producing $q+1$ input-output pairs of H from q quantum queries. To prove this bound we introduce a new lower-bound technique we call the *rank method* for bounding the success probability of algorithms that succeed with only small probability. Existing quantum lower bound techniques such as the polynomial method [BBC$^+$01] and the adversary method [Amb00, Aar02, Amb06, ASdW09] do not give the result we need. One difficulty with existing lower bound techniques is that they generally prove asymptotic bounds on the number of queries required to solve a problem with high probability, whereas we need a bound on the success probability of an algorithm making a limited number of queries. Attempting to apply existing techniques to our problem at best only bounds the success probability away from 1 by an inverse polynomial factor, which is insufficient for our purposes. The rank method for proving quantum lower bounds overcomes these difficulties and is a general tool that can be used in other post-quantum security proofs.

The second part of Theorem 1 shows that the lower bound presented in the first part of the theorem is tight. A related algorithm was previously presented by van Dam [vD98], but only for oracles outputting one bit, namely when $\mathcal{Y}=\{0,1\}$. For such a small range only about $|\mathcal{X}|/2$ quantum queries are needed to learn the oracle at all $|\mathcal{X}|$ points. A special case where $\mathcal{Y}=\mathcal{X}=\{0,1\}$ and $q=1$ was developed independently by Kerenidis and de Wolf [KdW03]. Our algorithm is a generalization of van Dam's result to multi-bit oracles.

Quantum Secure Carter-Wegman MACs. A Carter-Wegman MAC [WC81] signs a message m by computing $\big(r,\ h(m)\oplus F(k,r)\big)$ where h is a secret hash function chosen from an xor-universal hash family, F is a secure PRF with secret key k, and r is a short random nonce. The attractive feature of Carter-Wegman MACs is that the long message m is hashed by a fast xor-universal hash h. We show that a slightly modified Carter-Wegman MAC is quantum secure assuming the underlying PRF is quantum secure in the sense of Zhandry [Zha12b].

One-time Quantum Secure MACs. A one-time MAC is existentially unforgeable when the adversary can make only a *single* chosen message query. Classically, one-time MACs are constructed from pair-wise independent hash functions [WC81]. These MACs are one-time secure since the value of a pair-wise independent hash at one point gives no information about its value at another point. Therefore, a single classical chosen-message query tells the adversary nothing about the MAC tag of another message.

In the quantum settings things are more complicated. Unlike the classical settings, we show that pair-wise independence does not imply existential unforgeability under a one-time quantum chosen message attack. For example, consider the simple pair-wise independent hash family $\mathcal{H}=\{h(x)=ax+b\}_{a,b\in\mathbb{F}_p}$ with domain and range $\mathbb{F}_p$. We show that a quantum adversary presented with an oracle for a random function $h\in\mathcal{H}$ can find both a and b with a *single* quantum query to h. Consequently, the classical one-time MAC constructed from

$\mathcal{H}$ is completely insecure in the quantum settings. More generally we prove the following theorem:

Theorem 2 (informal). *There is a polynomial time quantum algorithm that when presented with an oracle for $h(x) = a_0 + a_1x + \ldots + a_dx^d$ for random $a_0, \ldots, a_d$ in $\mathbb{F}_p$ can recover $a_0, \ldots, a_d$ using only d quantum queries to the oracle with probability $1 - O(d/n)$.*

The $h(x) = ax + b$ attack discussed above is a special case of this theorem with $d = 1$. With classical queries finding $a_0, \ldots, a_d$ requires $d + 1$ queries, but with quantum queries the theorem shows that d queries are sufficient.

Theorem 2 is a quantum polynomial interpolation algorithm: given oracle access to the polynomial, the algorithm reconstructs its coefficients. This problem was studied previously by Kane and Kutin [KK11] who prove that $d/2$ quantum queries are insufficient to interpolate the polynomial. Interestingly, they conjecture that quantum interpolation requires $d + 1$ quantum queries as in the classical case, but Theorem 2 refutes that conjecture. Theorem 2 also applies to a quantum version of secret sharing where the shares themselves are superpositions. It shows that the classical Shamir secret sharing scheme [Sha79] is insecure if the shares are allowed to be quantum states obtained by evaluating the secret sharing polynomial on quantum superpositions. More generally, the security of secret sharing schemes in the quantum settings was analyzed by Damård et al. [DFNS11].

As for one-time secure MACs, while pair-wise independence is insufficient for quantum one-time security, we show that four-wise independence is sufficient. That is, a four-way independent hash family gives rise to an existentially unforgeable MAC under a one-time quantum chosen message attack. It is still an open problem whether three-way independence is sufficient. More generally, we show that $(q + 1)$-way independence is insufficient for a q-time quantum secure MAC, but $(3q + 1)$-way independence is sufficient.

Motivation. Allowing the adversary to issue quantum chosen message queries is a natural and conservative security model and is therefore an interesting one to study. Showing that classical MAC constructions remain secure in this model gives confidence in case end-user computing devices eventually become quantum. Nevertheless, one might imagine that even in a future where computers are quantum, the last step in a MAC signing procedure is to sample the resulting quantum state so that the generated MAC is always classical. The quantum chosen message query model ensures that even if the attacker can bypass this last "classicalization" step, the MAC remains secure.

As further motivation we note that the results in this paper are the tip of a large emerging area of research with many open questions. Consider for example signature schemes. Can one design schemes that remain secure when the adversary can issue quantum chosen message queries? Similarly, can one design encryption systems that remain secure when the the adversary can issue quantum chosen ciphertext queries? More generally, for any cryptographic primitive modeled as an interactive game, one can ask how to design primitives that remain secure when the interaction between the adversary and its given oracles is quantum.

Other Related Work. Several recent works study the security of cryptographic primitives when the adversary can issue quantum queries [BDF$^+$11, Zha12a, Zha12b]. So far these have focused on proving security of signatures, encryption, and identity-based encryption in the *quantum* random oracle model where the adversary can query the random oracle on superpositions of inputs. These works show that many, but not all, random oracle constructions remain secure in the quantum random oracle model. The quantum random oracle model has also been used to prove security of Merkle's Puzzles in the quantum settings [BS08, BHK$^+$11]. Meanwhile, Damård et al. [DFNS11] examine secret sharing and multiparty computation in a model where an adversary may corrupt a superposition of subsets of players, and build zero knowledge protocols that are secure, even when a dishonest verifier can issue challenges on superpositions.

Some progress toward identifying sufficient conditions under which classical protocols are also quantum immune has been made by Unruh [Unr10] and Hallgren et al. [HSS11]. Unruh shows that any scheme that is statistically secure in Cannetti's universal composition (UC) framework [Can01] against classical adversaries is also statistically secure against quantum adversaries. Hallgren et al. show that for many schemes this is also true in the computational setting. These results, however, do not apply to MACs.

2 Preliminaries: Definitions and Notation

Let $[n]$ be the set $\{1, ..., n\}$. For a prime power n, let $\mathbb{F}_n$ be the finite field on n elements. For any positive integer n, let $\mathbb{Z}_n$ be the ring of integers modulo n.

Functions will be denoted by capitol letters (such as F), and sets by capitol script letters (such as $\mathcal{X}$). We denote vectors with bold lower-case letters (such as $\mathbf{v}$), and the components of a vector $\mathbf{v} \in \mathcal{A}^n$ by v_i, $i \in [n]$. We denote matrices with bold capital letters (such as $\mathbf{M}$), and the components of a matrix $\mathbf{M} \in \mathcal{A}^{m\times n}$ by $M_{i,j}$, $i \in [m], j \in [n]$. Given a function $F : \mathcal{X} \to \mathcal{Y}$ and a vector $\mathbf{v} \in \mathcal{X}^n$, let $F(\mathbf{v})$ denote the vector $(F(v_1), F(v_2), ..., F(v_k))$. Let $F([n])$ denote the vector $(F(1), F(2), ..., F(n))$.

Given a vector space $\mathcal{V}$, let $\dim \mathcal{V}$ be the dimension of $\mathcal{V}$, or the number of vectors in any basis for $\mathcal{V}$. Given a set of vectors $\{\mathbf{v}_1, ..., \mathbf{v}_k\}$, let $\text{span}\{\mathbf{v}_1, ..., \mathbf{v}_k\}$ denote the space of all linear combinations of vectors in $\{\mathbf{v}_1, ..., \mathbf{v}_k\}$. Given a subspace S of an inner-product space V, and a vector $\mathbf{v} \in V$, define $\text{proj}_S \mathbf{v}$ as the orthogonal projection of $\mathbf{v}$ onto S, that is, the vector $\mathbf{w} \in S$ such that $|\mathbf{v} - \mathbf{w}|$ is minimized.

Given a matrix $\mathbf{M}$, we define the rank, denoted $\text{rank}(\mathbf{M})$, to be the size of the largest subset of rows (equivalently, columns) of $\mathbf{M}$ that are linearly independent.

Given a function $F : \mathcal{X} \to \mathcal{Y}$ and a subset $\mathcal{S} \subseteq \mathcal{X}$, the restriction of F to $\mathcal{S}$ is the function $F_\mathcal{S} : \mathcal{S} \to \mathcal{Y}$ where $F_\mathcal{S}(x) = F(x)$ for all $x \in \mathcal{S}$. A distribution D on the set of functions F from $\mathcal{X}$ to $\mathcal{Y}$ induces a distribution $D_\mathcal{S}$ on the set of functions from $\mathcal{S}$ to $\mathcal{Y}$, where we sample from $D_\mathcal{S}$ by first sampling a function F from D, and outputting $F_\mathcal{S}$. We say that D is k-wise independent if, for each set $\mathcal{S}$ of size at most k, each of the distributions $D_\mathcal{S}$ are truly random distributions on functions from $\mathcal{S}$ to $\mathcal{Y}$. A set $\mathcal{F}$ of functions from $\mathcal{X}$ to $\mathcal{Y}$ is k-wise independent if the uniform distribution on $\mathcal{F}$ is k-wise independent.

2.1 Quantum Computation

The quantum system A is a complex Hilbert space $\mathcal{H}$ with inner product $\langle\cdot|\cdot\rangle$. The state of a quantum system is given by a vector $|\psi\rangle$ of unit norm ($\langle\psi|\psi\rangle = 1$). Given quantum systems $\mathcal{H}_1$ and $\mathcal{H}_2$, the joint quantum system is given by the tensor product $\mathcal{H}_1 \otimes \mathcal{H}_2$. Given $|\psi_1\rangle \in \mathcal{H}_1$ and $|\psi_2\rangle \in \mathcal{H}_2$, the product state is given by $|\psi_1\rangle|\psi_2\rangle \in \mathcal{H}_1 \otimes \mathcal{H}_2$. Given a quantum state $|\psi\rangle$ and an orthonormal basis $B = \{|b_0\rangle, ..., |b_{d-1}\rangle\}$ for $\mathcal{H}$, a measurement of $|\psi\rangle$ in the basis B results in a value b_i with probability $|\langle b_i|\psi\rangle|^2$, and the state $|\psi\rangle$ is collapsed to the state $|b_i\rangle$. We let $b_i \leftarrow |\psi\rangle$ denote the distribution on b_i obtained by sampling $|\psi\rangle$.

A unitary transformation over a d-dimensional Hilbert space $\mathcal{H}$ is a $d \times d$ matrix $\mathbf{U}$ such that $\mathbf{U}\mathbf{U}^\dagger = \mathbf{I}_d$, where $\mathbf{U}^\dagger$ represents the conjugate transpose. A quantum algorithm operates on a product space $\mathcal{H}_{in} \otimes \mathcal{H}_{out} \otimes \mathcal{H}_{work}$ and consists of n unitary transformations $\mathbf{U}_1, ..., \mathbf{U}_n$ in this space. $\mathcal{H}_{in}$ represents the input to the algorithm, $\mathcal{H}_{out}$ the output, and $\mathcal{H}_{work}$ the work space. A classical input x to the quantum algorithm is converted to the quantum state $|x, 0, 0\rangle$. Then, the unitary transformations are applied one-by-one, resulting in the final state

$$|\psi_x\rangle = \mathbf{U}_n ... \mathbf{U}_1 |x, 0, 0\rangle \ .$$

The final state is measured, obtaining (a, b, c) with probability $|\langle a, b, c|\psi_x\rangle|^2$. The output of the algorithm is b.

Quantum-accessible Oracles. We will implement an oracle $O : \mathcal{X} \to \mathcal{Y}$ by a unitary transformation $\mathbf{O}$ where

$$\mathbf{O}|x, y, z\rangle = |x, y + O(x), z\rangle$$

where $+ : \mathcal{X} \times \mathcal{X} \to \mathcal{X}$ is some group operation on $\mathcal{X}$. Suppose we have a quantum algorithm that makes quantum queries to oracles $O_1, ..., O_q$. Let $|\psi_0\rangle$ be the state of the algorithm before any queries, and let $\mathbf{U}_1, ..., \mathbf{U}_q$ be the unitary transformations applied between queries. The final state of the algorithm will be

$$\mathbf{U}_q \mathbf{O}_q ... \mathbf{U}_1 \mathbf{O}_1 |\psi_0\rangle$$

We can also have an algorithm make classical queries to O_i. In this case, the input to the oracle is measured before applying the transformation $\mathbf{O}_i$.

Fix an oracle $O : \mathcal{X} \to \mathcal{Y}$. Let $O^{(q)} : \mathcal{X}^q \to \mathcal{Y}^q$ be the oracle that maps $\mathbf{x}$ into $O(\mathbf{x}) = (O(x_1), O(x_2), ..., O(x_q))$. Observe that any quantum query to $O^{(q)}$ can be implemented using q quantum queries to O, where the unitary transformations between queries just permute the registers. We say that an algorithm that makes a single query to $O^{(q)}$ makes q non-adaptive queries to O.

2.2 Quantum Secure MACs

A MAC system comprises two algorithms: a (possibly) randomized MAC signing algorithm $S(k, m)$ and a MAC verification algorithm $V(k, m, t)$. Here k denotes the secret key chosen at random from the key space, m denotes a message in the message space, and t denotes the MAC tag in the tag space on the message m. These algorithms and spaces are parameterized by a security parameter λ.

Classically, a MAC system is said to be secure if no attacker can win the following game: a random key k is chosen from the key space and the attacker is presented with a signing oracle $S(k, \cdot)$. Queries to the signing oracle are called chosen message queries. Let $\{(m_i, t_i)\}_{i=1}^q$ be the set of message-tag pairs that the attacker obtains by interacting with the signing oracle. The attacker wins the game if it can produce an existential forgery, namely a valid message-tag pair (m^*, t^*) satisfying $(m^*, t^*) \notin \{(m_i, t_i)\}_{i=1}^q$. The MAC system is said to be secure if no "efficient" adversary can win this game with non-negligible probability in λ.

Quantum Chosen Message Queries. In the quantum settings we allow the adversary to maintain its own quantum state and issue quantum queries to the signing oracle. Let $\sum_{m,x,y} \psi_{m,x,y} \left|m, x, y\right\rangle$ be the adversary's state just prior to issuing a signing query. The MAC signing oracle transforms this state as follows:

1. it chooses a random string r that will be used by the MAC signing algorithm,
2. it signs each "slot" in the given superposition by running $S(k, m; r)$, that is running algorithm S with randomness r. More precisely, the signing oracle performs the following transformation:

$$\sum_{m,x,y} \psi_{m,x,y} \left|m, x, y\right\rangle \quad \longrightarrow \quad \sum_{m,x,y} \psi_{m,x,y} \left|m,\ x \oplus S(k, m; r),\ y\right\rangle$$

When the signing algorithm is deterministic there is no need to choose an r. However, for randomized signing algorithms the same randomness is used to compute the tag for all slots in the superposition. Alternatively, we could have required fresh randomness in every slot, but this would make it harder to implement the MAC system on a quantum device. Allowing the same randomness in every slot is more conservative and frees the signer from this concern. At any rate, the two models are very close — if need be, the random string r can be used as a key for a quantum-secure PRF [Zha12b] which is used to generate a fresh pseudorandom value for every slot.

Existential Forgery. After issuing q quantum chosen message queries the adversary wins the game if it can generate $q + 1$ valid classical message-tag pairs.

Definition 1. *A MAC system is existentially unforgeable under a quantum chosen message attack (EUF-qCMA) if no adversary can with the quantum MAC game with non-negligible advantage in λ.*

Zhandry [Zha12b] gives an example of a classically secure PRF that is insecure under quantum queries. This PRF gives an example MAC that is classically secure, but insecure under quantum queries. Our goal for the remainder of the paper is to construct EUF-qCMA secure MACs.

3 The Rank Method

In this section we introduce the rank method which is a general approach to proving lower bounds on quantum algorithms. The setup is as follows: we give a quantum algorithm A access to some quantity $H \in \mathcal{H}$. By access, we mean that the final state of the algorithm is some fixed function of H. In this paper, $\mathcal{H}$ will be a set of functions, and A will be given oracle access to $H \in \mathcal{H}$ by allowing A to make q quantum oracle queries to H, for some q. For now, we will treat $\mathcal{H}$ abstractly, and return to the specific case where $\mathcal{H}$ is a set of functions later.

The idea behind the rank method is that, if we treat the final states of the algorithm on different H as vectors, the space spanned by these vectors will be some subspace of the overall Hilbert space. If the dimension of this subspace is small enough, the subspace (and hence all of the vectors in it) must be reasonably far from most of the vectors in the measurement basis. This allows us to bound the ability of such an algorithm to achieve some goal.

For $H \in \mathcal{H}$, let $|\psi_H\rangle$ be the final state of the quantum algorithm A, before measurement, when given access to H. Suppose the different $|\psi_H\rangle$ vectors all lie in a space of dimension d. Let $\mathbf{\Psi}_{A,\mathcal{H}}$ be the the $|\mathcal{H}| \times d$ matrix whose rows are the various vectors $|\psi_H\rangle$.

Definition 2. *For a quantum algorithm A given access to some value $H \in \mathcal{H}$, we define the* rank*, denoted* $\mathrm{rank}(A, \mathcal{H})$*, as the rank of the matrix* $\mathbf{\Psi}_{A,\mathcal{H}}$*.*

The rank of an algorithm A seemingly contains very little information: it gives the dimension of the subspace spanned by the $|\psi_H\rangle$ vectors, but gives no indication of the orientation of this subspace or the positions of the $|\psi_H\rangle$ vectors in the subspace. Nonetheless, we demonstrate how the success probability of an algorithm can be bounded from above knowing only the rank of $\mathbf{\Psi}_{A,\mathcal{H}}$.

Theorem 3. *Let A be a quantum algorithm that has access to some value $H \in \mathcal{H}$ drawn from some distribution D and produces some output $w \in \mathcal{W}$. Let $R : \mathcal{H} \times \mathcal{W} \to \{\texttt{True}, \texttt{False}\}$ be a binary relation. Then the probability that A outputs some w such that $R(H, w) = \texttt{True}$ is at most*

$$\left(\max_{w \in \mathcal{W}} \Pr_{H \leftarrow D}[R(H, w)] \right) \times \mathrm{rank}(A, \mathcal{H}) \ .$$

In other words, the probability that A succeeds in producing $w \in \mathcal{W}$ for which $R(H, w)$ is true is at most $\mathrm{rank}(A, \mathcal{H})$ *times the best probability of success of any algorithm that ignores H and just outputs some fixed w.*

Proof. The probability that A outputs a w such that $R(H, w) = \texttt{True}$ is

$$\Pr_{\substack{H \leftarrow D \\ w \leftarrow |\psi_H\rangle}}[R(H, w)] = \sum_H \Pr_D[H] \sum_{w : R(H,w)} |\langle w|\psi_H\rangle|^2 = \sum_w \sum_{H : R(H,w)} \Pr_D[H] |\langle w|\psi_H\rangle|^2$$

Now, $|\langle w|\psi_H\rangle|$ is just the magnitude of the projection of $|w\rangle$ onto the space spanned by the vector $|\psi_H\rangle$, that is, $\mathrm{proj}_{\mathrm{span}|\psi_H\rangle}(|w\rangle)$. This is at most the magnitude of the projection of $|w\rangle$ onto the space spanned by all of the $|\psi_{H'}\rangle$ for $H' \in \mathcal{H}$, or $\text{proj}_{\text{span}\{|\psi_{H'}\rangle\}}(|w\rangle)$. Thus,

$$\Pr_{\substack{H \leftarrow D \\ w \leftarrow |\psi_H\rangle}}[R(z,w)] \leq \sum_w \left(\sum_{H: R(H,w)} \Pr_D[H] \right) \left| \text{proj}_{\text{span}\{|\psi_{H'}\rangle\}}(|w\rangle) \right|^2$$

Now, we can perform the sum over H, which gives $\Pr_{H \leftarrow D}[R(H,w)]$. We can bound this by the maximum it attains over all w, giving us

$$\Pr_{\substack{H \leftarrow D \\ w \leftarrow |\psi_H\rangle}}[R(H,w)] \leq \left(\max_w \Pr_{H \leftarrow D}[R(H,w)] \right) \sum_w \left| \text{proj}_{\text{span}\{|\psi_{H'}\rangle\}}(|w\rangle) \right|^2$$

Now, let $|b_i\rangle$ be an orthonormal basis for $\text{span}\{|\psi_{H'}\rangle\}$. Then

$$\left| \text{proj}_{\text{span}\{|\psi_{H'}\rangle\}}(|w\rangle) \right|^2 = \sum_i |\langle b_i | w \rangle|^2$$

Summing over all w gives

$$\sum_w \left| \text{proj}_{\text{span}\{|\psi_{H'}\rangle\}}(|w\rangle) \right|^2 = \sum_w \sum_i |\langle b_i | w \rangle|^2 = \sum_i \sum_w |\langle b_i | w \rangle|^2$$

Since the w are the possible results of measurement, the vectors $|w\rangle$ form an orthonormal basis for the whole space, meaning $\sum_w |\langle b_i | w \rangle|^2 = |\ |b_i\rangle\ |^2 = 1$. Hence, the sum just becomes the number of $|b_i\rangle$, which is just the dimension of the space spanned by the $|\psi_{H'}\rangle$. Thus,

$$\Pr_{\substack{H \leftarrow D \\ w \leftarrow |\psi_H\rangle}}[R(H,w)] \leq \left(\max_{w \in \mathcal{W}} \Pr_{H \leftarrow D}[R(H,w)] \right) (\dim \text{span}\{|\psi_{H'}\rangle\}) \ .$$

But $\dim \text{span}\{|\psi_{H'}\rangle\}$ is exactly $\text{rank}(\mathbf{\Psi}_{A,\mathcal{H}}) = \text{rank}(A, \mathcal{H})$, which finishes the proof of the theorem. □

We now move to the specific case of oracle access. $\mathcal{H}$ is now some set of functions from $\mathcal{X}$ to $\mathcal{Y}$, and our algorithm A makes q quantum oracle queries to a function $H \in \mathcal{H}$. To use the rank method (Theorem 3) for our purposes, we need to bound the rank of such an algorithm. First, we define the following quantity:

$$C_{k,q,n} \equiv \sum_{r=0}^{q} \binom{k}{r} (n-1)^r \ .$$

Theorem 4. *Let $\mathcal{X}$ and $\mathcal{Y}$ be sets of size m and n and let H_0 be some function from $\mathcal{X}$ to $\mathcal{Y}$. Let $\mathcal{S}$ be a subset of $\mathcal{X}$ of size k and let $\mathcal{H}$ be some set of functions from $\mathcal{X}$ to $\mathcal{Y}$ that are equal to H_0 except possibly on points in $\mathcal{S}$. If A is a quantum algorithm making q queries to an oracle drawn from $\mathcal{H}$, then*

$$\text{rank}(A, \mathcal{H}) \leq C_{k,q,n} \ .$$

Proof. The proof appears in the full version [BZ13]. Here we sketch our approach. Let $|\psi_H^q\rangle$ be the final state of a quantum algorithm after q quantum oracle calls to an oracle $H \in \mathcal{H}$. We wish to bound the dimension of the space spanned by the vectors $|\psi_H^q\rangle$ for all $H \in \mathcal{H}$. We accomplish this by exhibiting a basis for this space. Our basis consists of $|\psi_{H'}^q\rangle$ vectors where H' only differs from H_0 at a maximum of q points in $\mathcal{S}$. A simple counting argument shows that there are exactly $C_{k,q,n}$ such H' oracles. We show in the full version that these $|\psi_{H'}^q\rangle$ vectors do indeed span the entire space. □

3.1 An Example

Suppose our task is to, given one quantum query to an oracle $H : \mathcal{X} \to \mathcal{Y}$, produce two distinct pairs (x_0, y_0) and (x_1, y_1) such that $H(x_0) = y_0$ and $H(x_1) = y_1$. Suppose further that H is drawn from a pairwise independent set $\mathcal{H}$. We will now see that the rank method leads to a bound on the success probability of any quantum algorithm A.

Corollary 1. *No quantum algorithm A, making a single query to a function $H : \mathcal{X} \to \mathcal{Y}$ drawn from a pairwise independent set $\mathcal{H}$, can produce two distinct input/output pairs of H, except with probability at most $|\mathcal{X}|/|\mathcal{Y}|$.*

Proof. Let $m = |\mathcal{X}|$ and $n = |\mathcal{Y}|$. Since no outputs of H are fixed, we will set $\mathcal{S} = \mathcal{X}$ in Theorem 4, showing that the rank of the algorithm A is bounded by $C_{m,1,n} = 1 + m(n-1) < mn$. If an algorithm makes no queries to H, the best it can do at outputting two distinct input/output pairs is to just pick two arbitrary distinct pairs, and output those. The probability that this zero-query algorithm succeeds is at most $1/n^2$. Then Theorem 3 tells us that A succeeds with probability at most $\text{rank}(A, \mathcal{H})$ times this amount, which equates to $\frac{m}{n}$. □

For $m > n$, this bound is trivial. However, for m smaller than n, this gives a non-trivial bound, and for m exponentially smaller than n, this bound is negligible.

4 Outputting Values of a Random Oracle

In this section, we will prove Theorem 1. We consider the following problem: given q quantum queries to a random oracle $H : \mathcal{X} \to \mathcal{Y}$, produce $k > q$ distinct pairs (x_i, y_i) such that $y_i = H(x_i)$. Let $n = |\mathcal{Y}|$ be the size of the range. Motivated by our application to quantum-accessible MACs, we are interested in the case where the range $\mathcal{Y}$ of the oracle is large, and we want to show that to produce even one extra input/output pair ($k = q + 1$) is impossible, except with negligible probability. We are also interested in the case where the range of the oracle, though large, is far smaller than the domain. Thus, the bound we obtained in the previous section (Corollary 1) is not sufficient for our purposes, since it is only non-trivial if the range is larger than the domain.

In the classical setting, when $k \leq q$, this problem is easy, since we can just pick an arbitrary set of k different x_i values, and query the oracle on each value. For $k > q$, no adversary of even unbounded complexity can solve this problem, except with probability $1/n^{k-q}$, since for any set of k inputs, at least $k-q$ of the corresponding outputs are completely unknown to the adversary. Therefore, for large n, we have have a sharp threshold: for $k \leq q$, this problem can be solved efficiently with probability 1, and for even $k = q+1$, this problem cannot be solved, even inefficiently, except with negligible probability.

In the quantum setting, the $k \leq q$ case is the same as before, since we can still query the oracle classically. However, for $k > q$, the quantum setting is more challenging. The adversary can potentially query the random oracle on a superposition of all inputs, so he "sees" the output of the oracle on all points. Proving that it is still impossible to produce k input/output pairs is thus more complicated, and existing methods fail to prove that this problem is difficult. Therefore, it is not immediately clear that we have the same sharp threshold as before.

In Section 4.1 we use the rank method to bound the probability that any (even computationally unbounded) quantum adversary succeeds. Then in Section 4.2 we show that our bound is tight by giving an efficient algorithm for this problem that achieves the lower bound. In particular, for an oracle $H : \mathcal{X} \to \mathcal{Y}$ we consider two cases:

- Exponentially-large range $\mathcal{Y}$ and polynomial k, q. In this case, we will see that the success probability even when $k = q+1$ is negligible. That is, to produce even one additional input/output pair is hard. Thus, we get the same sharp threshold as in the classical case
- Constant size range $\mathcal{Y}$ and polynomial k, q. We show that even when q is a constant fraction of k we can still produce k input/output pairs with overwhelming probability using only q quantum queries. This is in contrast to the classical case, where the success probability for $q = ck$, $c < 1$, is negligible in k.

4.1 A Tight Upper Bound

Theorem 5. *Let A be a quantum algorithm making q queries to a random oracle $H : \mathcal{X} \to \mathcal{Y}$ whose range has size n, and produces $k > q$ pairs $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$. The probability that the x_i values are distinct and $y_i = H(x_i)$ for all $i \in [k]$ is at most $\frac{1}{n^k} C_{k,q,n}$.*

Proof. The complete proof is given in the full version [BZ13]. Here we prove the special case where k is equal to the size of the domain $\mathcal{X}$. In this case, any quantum algorithm that outputs k distinct input/output pairs must output *all* input/output pairs. Similar to the proof of Corollary 1, we will set $\mathcal{S} = \mathcal{X}$, and use Theorem 4 to bound the rank of A at $C_{k,q,n}$. Now, any algorithm making *zero* queries succeeds with probability at most $1/n^k$. Theorem 3 then bounds the success probability of any q query algorithm as $C_{k,q,n}/n^k$. $\square$

For this paper, we are interested in the case where $n = |\mathcal{Y}|$ is exponentially large, and we are only allowed a polynomial number of queries. Suppose $k = q + 1$, the easiest non-trivial case for the adversary. Then, the probability of success is

$$\frac{1}{n^{q+1}} \sum_{r=0}^{q} \binom{q+1}{r} (n-1)^r = 1 - \left(1 - \frac{1}{n}\right)^{q+1} \leq \frac{q+1}{n} \ . \tag{4.1}$$

Therefore, to produce even one extra input/output pair is impossible, except with exponentially small probability, just like in the classical case. This proves the first part of Theorem 1.

4.2 An Optimal Attack

In this section, we present a quantum algorithm for the problem of computing $H(x_i)$ for k different x_i values, given only $q < k$ queries:

Theorem 6. *Let $\mathcal{X}$ and $\mathcal{Y}$ be sets, and fix integers $q < k$, and k distinct values $x_1, ..., x_k \in \mathcal{X}$. There exists a quantum algorithm A that makes q non-adaptive quantum queries to any function $H : \mathcal{X} \to \mathcal{Y}$, and produces $H(x_1), ..., H(x_k)$ with probability $C_{k,q,n}/n^k$, where $n = |\mathcal{Y}|$.*

The proof appears in the full version [BZ13], and is similar to the algorithm of [vD98], though generalized to handle arbitrary range sizes. This algorithm has the same success probability as in Theorem 5, showing that both our attack and lower bound of Theorem 5 are optimal. This proves the second part of Theorem 1.

As we have already seen, for exponentially-large $\mathcal{Y}$, this attack has negligible advantage for any $k > q$. However, if $n = |\mathcal{Y}|$ is constant, we can do better. The error probability is

$$\sum_{r=q+1}^{k} \binom{k}{r} \left(1 - \frac{1}{n}\right)^r \left(\frac{1}{n}\right)^{k-r} = \sum_{s=0}^{k-q-1} \binom{k}{s} \left(\frac{1}{n}\right)^s \left(1 - \frac{1}{n}\right)^{k-s} \ .$$

This is the probability that k consecutive coin flips, where each coin is heads with probability $1/n$, yields fewer than $k - q$ heads. Using the Chernoff bound, if $q > k(1 - 1/n)$, this probability is at most

$$e^{-\frac{n}{2k}(q-k(1-1/n))^2} \ .$$

For a constant n, let c be any constant with $1 - 1/n < c < 1$. If we use $q = ck$ queries, the error probability is less than

$$e^{-\frac{n}{2k}(k(c+1/n-1))^2} = e^{-\frac{nk}{2}(c+1/n-1)^2} ,$$

which is exponentially small in k. Thus, for constant n, and any constant c with $1 - 1/n < c < 1$, using $q = ck$ quantum queries, we can determine k input/output pairs with overwhelming probability. This is in contrast to the classical case, where with any constant fraction of k queries, we can only produce k input/output pairs with negligible probability. As an example, if H outputs two bits, it is possible to produce k input/output pairs of of H using only $q = 0.8k$ quantum queries. However, with $0.8k$ classical queries, we can output k input/output pairs with probability at most $4^{-0.2k} < 0.76^k$.

5 Quantum-Accessible MACs

Using Theorem 5 we can now show that a quantum secure pseudorandom function [Zha12b] gives rise to the quantum-secure MAC, namely $S(k,m) = \mathsf{PRF}(k,m)$. We prove that this mac is secure.

Theorem 7. *If* $\mathsf{PRF} : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ *is a quantum-secure pseudorandom function and* $1/|\mathcal{Y}|$ *is negligible, then* $S(k,m) = \mathsf{PRF}(k,m)$ *is a EUF-qCMA-secure MAC.*

Proof. Let A be a polynomial time adversary that makes q quantum queries to $S(k,\cdot)$ and produces $q+1$ valid input/output pairs with probability ϵ. Let Game 0 be the standard quantum MAC attack game, where A makes q quantum queries to MAC_k. By definition, A's success probability in this game is ϵ.

Let Game 1 be the same as Game 0, except that $S(k,\cdot)$ is replaced with a truly random function $O : \mathcal{X} \to \mathcal{Y}$, and define A's success probability as the probability that A outputs $q+1$ input/output pairs of O. Since PRF is a quantum-secure PRF, A's advantage in distinguishing Game 0 from Game 1 is negligible.

Now, in Game 1, A makes q quantum queries to a random oracle, and tries to produce $q+1$ input/output pairs. However, by Theorem 5 and Eq. (4.1) we know that A's success probability is bounded by $(q+1)/|\mathcal{Y}|$ which is negligible. It now follows that ϵ is negligible and therefore, S is a EUF-qCMA-secure MAC. $\square$

5.1 Carter-Wegman MACs

In this section, we show how to modify the Carter-Wegman MAC so that it is secure in the quantum setting presented in Section 2.2. Recall that H is an XOR-universal family of hash functions from $\mathcal{X}$ into $\mathcal{Y}$ if for any two distinct points x and y, and any constant $c \in \mathcal{Y}$,

$$\Pr_{h \leftarrow H}[H(x) - H(y) = c] = 1/|\mathcal{Y}|$$

The Carter-Wegman construction uses a pseudorandom function family PRF with domain $\mathcal{X}$ and range $\mathcal{Y}$, and an XOR-universal family of hash functions $\mathcal{H}$ from $\mathcal{M}$ to $\mathcal{Y}$. The key is a pair (k,H), where k is a key for PRF and H is a function drawn from $\mathcal{H}$. To sign a message, pick a random $r \in \mathcal{X}$, and return $(r,\ \mathsf{PRF}(k,r) + H(m))$.

This MAC is not, in general, secure in the quantum setting presented in Section 2.2. The reason is that the same randomness is used in all slots of a quantum chosen message query, that is the signing oracle computes:

$$\sum_m \alpha_m |m\rangle \longrightarrow \sum_m \alpha_m |m, r, \mathsf{PRF}(k,r) + H(m)\rangle$$

where the same r is used for all classical states of the superposition. For example, suppose $\mathcal{H}$ is the set of functions $H(x) = ax + b$ for random a and b. With even a single quantum query, the adversary will be able to obtain a and

$\mathsf{PRF}(k, r) + b$ with high probability, using the algorithm from Theorem 10 in Section 6. Knowing both of these will allow the adversary to forge any message.

We show how to modify the standard Carter-Wegman MAC to make it secure in the quantum setting.

Construction 1 (Carter-Wegman). *The Quantum Carter-Wegman MAC (QCW-MAC) is built from a pseudorandom function* PRF*, an XOR-universal set of functions* $\mathcal{H}$*, and a pairwise independent set of functions* $\mathcal{R}$*.*

Keys: The secret key for QCW-MAC is a pair (k, H)*, where* k *is a key for* PRF *and* $H : \mathcal{M} \to \mathcal{Y}$ *is drawn from* $\mathcal{H}$

Signing: To sign a message m *choose a random* $R \in \mathcal{R}$ *and output the pair* $\big(R(m),\ \mathsf{PRF}(k, R(m)) + H(m)\ \big)$ *as the tag. When responding to a quantum chosen message query, the same* R *is used in all classical states of the superposition.*

Verification: To verify that (r, s) *is a valid tag for* m*, accept iff* $\mathsf{PRF}(k, r) + H(m) = s$*.*

Theorem 8. *The Quantum Carter-Wegman MAC is a EUF-qCMA secure MAC.*

The proof is given in the full version [BZ13].

6 q-time MACs

In this section, we develop quantum one-time MACs, MACs that are secure when the adversary can issue only *one* quantum chosen message query. More generally, we will study quantum q-time MACs.

Classically, any pairwise independent function is a one-time MAC. In the quantum setting, Corollary 1 shows that when the range is much larger than the domain, this still holds. However, such MACs are not useful since we want the tag to be short. We first show that when the range is not larger than the domain, pairwise independence is not enough to ensure security:

Theorem 9. *For any set* $\mathcal{Y}$ *of prime-power size, and any set* $\mathcal{X}$ *with* $|\mathcal{X}| \geq |\mathcal{Y}|$*, there exist* $(q + 1)$*-wise independent functions from* $\mathcal{X}$ *to* $\mathcal{Y}$ *that are not q-time MACs.*

To prove this theorem, we treat $\mathcal{Y}$ as a finite field, and assume $\mathcal{X} = \mathcal{Y}$, as our results are easy to generalize to larger domains. We use random degree q polynomials as our $(q + 1)$-wise independent family, and show in Theorem 10 below that such polynomials can be completely recovered using only q quantum queries. It follows that the derived MAC cannot be q-time secure since once the adversary has the polynomial it can easily forge tags on new messages. The proof of the following theorem appears in the full version [BZ13]:

Theorem 10. *For any prime power* n*, there is an efficient quantum algorithm that makes only* q *quantum queries to an oracle implementing a degree-*q *polynomial* $F : \mathbb{F}_n \to \mathbb{F}_n$*, and completely determines* F *with probability* $1 - O(qn^{-1})$*.*

The theorem shows that a $(q+1)$-wise independence family is not necessarily a secure quantum q-time MAC since after q quantum chosen message queries the adversary extracts the entire secret key. The case $q = 1$ is particularly interesting.

6.1 Sufficient Conditions for a One-Time Mac

We show that, while pairwise independence is not enough for a one-time MAC, 4-wise independence is. We first generalize a theorem of Zhandry [Zha12a]:

Lemma 1. *Let A be any quantum algorithm that makes c classical queries and q quantum queries to an oracle H. If H is drawn from a $(c+2q)$-wise independent function, then the output distribution of A is identical to the case where H is truly random.*

Proof. The complete proof is given in the full version [BZ13]. If $q = 0$, then this theorem is trivial, since the c classical outputs A sees are distributed randomly. If $c = 0$, then the theorem reduces to that of Zhandry [Zha12a]. By adapting the proof of the $c = 0$ case to the general case, we get the lemma. □

Using this lemma we show that $(3q+1)$-wise independence is sufficient for q-time MACs.

Theorem 11. *Any $(3q+1)$-wise independent family with domain $\mathcal{X}$ and range $\mathcal{Y}$ is a quantum q-time secure MAC provided $(q+1)/|\mathcal{Y}|$ is negligible.*

Proof. Let D be some $(3q+1)$-wise independent function. Suppose we have an adversary A that makes q quantum queries to an oracle H, and attempts to produces $q+1$ input/output pairs. Let ϵ_R be the probability of success when H is a random oracle, and let ϵ_D be the probability of success when H is drawn from D. We construct an algorithm B with access to H as follows: simulate A with oracle access to H. When A outputs $q+1$ input/output pairs, simply make $q+1$ queries to H to check that these are valid pairs. Output 1 if and only if all pairs are valid. Therefore, B makes q quantum queries and $c = q+1$ classical queries to H, and outputs 1 if and only if A succeeds: if H is random, B outputs 1 with probability ϵ_R, and if H is drawn from D, B outputs 1 with probability ϵ_D. Now, since D is $(3q+1)$-wise independent and $3q+1 = 2q+c$, Lemma 1 shows that the distributions of outputs when H is drawn from D is identical to that when H is random, meaning $\epsilon_D = \epsilon_R$.

Thus, when H is drawn from D, A's succeeds with the same probability that it would if H was random. But we already know that if H is truly random, A's success probability is less than $(q+1)/|\mathcal{Y}|$. Therefore, when H is drawn from D, A succeeds with probability less than $(q+1)/|\mathcal{Y}|$, which is negligible. Hence, if H is drawn from D, H is a q-time MAC. □

7 Conclusion

We introduced the rank method as a general technique for obtaining lower bounds on quantum oracle algorithms and used this method to bound the probability that a quantum algorithm can evaluate a random oracle $\mathcal{O} : \mathcal{X} \to \mathcal{Y}$

at k points using $q < k$ queries. When the range of $\mathcal{Y}$ is small, say $|\mathcal{Y}| = 8$, a quantum algorithm can recover k points of $\mathcal{O}$ from only $0.9k$ queries with high probability. However, we show that when the range $\mathcal{Y}$ is large, no algorithm can produce k input-output pairs of $\mathcal{O}$ using only $k-1$ queries, with non-negligible probability. We use these bounds to construct the first MACs secure against quantum chosen message attacks. We consider both PRF and Carter-Wegman constructions. For one-time MACs we showed that pair-wise independence does not ensure security, but four-way independence does.

These results suggest many directions for future work. First, can these bounds be generalized to signatures to obtain signatures secure against quantum chosen message attacks? Similarly, can we construct encryption systems secure against quantum chosen ciphertext attacks where decryption queries are superpositions of ciphertexts?

Acknowledgments. We thank Luca Trevisan and Amit Sahai for helpful conversations about this work. This work was supported by NSF, DARPA, IARPA, the Air Force Office of Scientific Research (AFO SR) under a MURI award, Samsung, and a Google Faculty Research Award. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA, IARPA, DoI/NBC, or the U.S. Government.

References

[Aar02] Aaronson, S.: Quantum lower bound for the collision problem. In: STOC, pp. 635–642 (2002)

[Amb00] Ambainis, A.: Quantum lower bounds by quantum arguments. In: STOC, pp. 636–643 (2000)

[Amb06] Ambainis, A.: Polynomial degree vs. quantum query complexity. J. Comput. Syst. Sci. 72(2), 220–238 (2006)

[ASdW09] Ambainis, A., Spalek, R., de Wolf, R.: A new quantum lower bound method, with applications to direct product theorems and time-space tradeoffs. Algorithmica 55(3), 422–461 (2009)

[BBC$^+$01] Beals, R., Buhrman, H., Cleve, R., Mosca, M., de Wolf, R.: Quantum Lower Bounds by Polynomials. Journal of the ACM (JACM) 48(4), 778–797 (2001)

[BCK96] Bellare, M., Canetti, R., Krawczyk, H.: Pseudorandom functions revisited: The cascade construction and its concrete security. In: FOCS, pp. 514–523 (1996)

[BDF$^+$11] Boneh, D., Dagdelen, Ö., Fischlin, M., Lehmann, A., Schaffner, C., Zhandry, M.: Random Oracles in a Quantum World. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 41–69. Springer, Heidelberg (2011), http://arxiv.org/abs/1008.0931

[BHK$^+$11] Brassard, G., Høyer, P., Kalach, K., Kaplan, M., Laplante, S., Salvail, L.: Merkle Puzzles in a Quantum World. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 391–410. Springer, Heidelberg (2011), http://www.springerlink.com/index/X744P0G21126R4K1.pdf

[BKR00] Bellare, M., Kilian, J., Rogaway, P.: The security of the cipher block chaining message authentication code. J. Comput. Syst. Sci. 61(3) (2000)

[BS08] Brassard, G., Salvail, L.: Quantum Merkle Puzzles. In: Second International Conference on Quantum, Nano and Micro Technologies (ICQNM 2008), pp. 76–79 (February 2008)

[BZ13] Boneh, D., Zhandry, M.: Quantum-secure message authentication codes. In: Johansson, T., Nguyen, P. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 593–609. Springer, Heidelberg (2013), `http://eccc.hpi-web.de/report/2012/136`

[Can01] Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: Proc. of FOCS. IEEE (2001)

[DFNS11] Damgård, I., Funder, J., Nielsen, J.B., Salvail, L.: Superposition attacks on cryptographic protocols. CoRR, abs/1108.6313 (2011)

[GGM86] Goldreich, O., Goldwasser, S., Micali, S.: How to Construct Random Functions. Journal of the ACM (JACM) 33(4), 792–807 (1986)

[HSS11] Hallgren, S., Smith, A., Song, F.: Classical cryptographic protocols in a quantum world. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 411–428. Springer, Heidelberg (2011)

[KdW03] Kerenidis, I., de Wolf, R.: Exponential lower bound for 2-query locally decodable codes via a quantum argument. In: Proceedings of the 35th Annual ACM Symposium on Theory of Computing (STOC), pp. 106–115 (2003)

[KK11] Kane, D.M., Kutin, S.A.: Quantum interpolation of polynomials. Quantum Information & Computation 11(1&2), 95–103 (2009) (first published in 2009)

[Sha79] Shamir, A.: How to share a secret. Commun. ACM 22(11), 612–613 (1979)

[Unr10] Unruh, D.: Universally Composable Quantum Multi-party Computation. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 486–505. Springer, Heidelberg (2010), `http://www.springerlink.com/index/L833767317P46473.pdf`

[vD98] van Dam, W.: Quantum oracle interrogation: Getting all information for almost half the price. In: FOCS, pp. 362–367 (1998)

[WC81] Wegman, M.N., Carter, L.: New hash functions and their use in authentication and set equality. J. Comput. Syst. Sci. 22(3), 265–279 (1981)

[Zha12a] Zhandry, M.: Secure Identity-Based Encryption in the Quantum Random Oracle Model (Full version available at the Cryptology ePrint Archives: `http://eprint.iacr.org/2012/076/`). In: Safavi-Naini, R. (ed.) CRYPTO 2012. LNCS, vol. 7417, pp. 758–775. Springer, Heidelberg (2012)

[Zha12b] Zhandry, M.: How to Construct Quantum Random Functions. In: Proceedings of FOCS (2012), Full version available at the Cryptology ePrint Archives: `http://eprint.iacr.org/2012/182/`

One-Sided Device-Independent QKD and Position-Based Cryptography from Monogamy Games

Marco Tomamichel[1], Serge Fehr[2], Jędrzej Kaniewski[1], and Stephanie Wehner[1]

[1] Centre for Quantum Technologies, National University of Singapore
`{cqtmarco,j.kaniewski}@nus.edu.sg`, `wehner@comp.nus.edu.sg`

[2] CWI Amsterdam, The Netherlands
`serge.fehr@cwi.nl`

Abstract. A serious concern with quantum key distribution (QKD) schemes is that, when under attack, the quantum devices in a real-life implementation may behave differently than modeled in the security proof. This can lead to real-life attacks against provably secure QKD schemes.

In this work, we show that the standard BB84 QKD scheme is *one-sided device-independent*. This means that security holds even if Bob's quantum device is arbitrarily malicious, as long as Alice's device behaves as it should. Thus, we can completely remove the trust into Bob's quantum device *for free*, without the need for changing the scheme, and without the need for hard-to-implement loophole-free violations of Bell inequality, as is required for fully (meaning two-sided) device-independent QKD.

For our analysis, we introduce a new quantum game, called a *monogamy-of-entanglement* game, and we show a strong parallel repetition theorem for this game. This new notion is likely to be of independent interest and to find additional applications. Indeed, besides the application to QKD, we also show a direct application to *position-based quantum cryptography*: we give the first security proof for a one-round position-verification scheme that requires only single-qubit operations.

1 Introduction

Background. Quantum key distribution (QKD) makes use of quantum mechanical effects to allow two parties, Alice and Bob, to exchange a secret key while being eavesdropped by an attacker Eve [5,11]. In principle, the security of QKD can be rigorously proven based solely on the laws of quantum mechanics [27,33,31]; in particular, the security does not rely on the assumed hardness of some computational problem. However, these security proofs typically make stringent assumptions about the devices used by Alice and Bob to prepare and measure the quantum states that are communicated. These assumptions are not necessarily satisfied by real-world devices, leaving the implementations of QKD schemes open to hacking attacks [25].

One way to counter this problem is by protecting the devices in an ad-hoc manner against known attacks. This is somewhat unsatisfactory in that the implementation may still be vulnerable to *unknown* attacks, and the fact that the scheme is in principle provably secure loses a lot of its significance.

Another approach is to try to remove the assumptions on the devices necessary for the security proof; this leads to the notion of *device-independent* (DI) QKD. This line of research can be traced back to Mayers and Yao [28] as well as [2,1]. After some limited results [26,13], the possibility of DI QKD has recently been shown in the most general case by Reichhardt *et al.* in [30]. In a typical DI QKD scheme, Alice and Bob check if the classical data obtained from the quantum communication violates a Bell inequality, which in turn ensures that there is some amount of fresh randomness in the data that cannot be known by Eve. This can then be transformed into a secret key using standard cryptographic techniques like information reconciliation and randomness extraction.

While this argument shows that DI QKD is theoretically possible, the disadvantage of such schemes is that they require a *loophole free* violation of a Bell inequality by Alice and Bob. This makes fully DI QKD schemes very hard to implement and very sensitive to any kind of noise and to inefficiencies of the physical devices: any deficiency will result in a lower observed (loophole free) Bell inequality violation, and currently conceivable experimental parameters are insufficient to provide provable security. Trying to find ways around this problem is an active line of research, see e.g. [12,24,7,23].

Our Result. Here, we follow a somewhat different approach, not relying on Bell tests, but making use of the *monogamy of entanglement*. Informally, the latter states that if Alice's state is fully entangled with Bob's, then it cannot be entangled with Eve's, and vice versa. As a consequence, if Alice measures a quantum system by randomly choosing one of two incompatible measurements, it is impossible for Bob and Eve to *both* have low entropy about Alice's measurement outcome. Thus, if one can verify that Bob has low entropy about Alice's measurement during the run of the scheme, it is guaranteed that Eve's entropy is high, and thus that a secret key can be distilled.

Based on this idea, we show that the standard BB84 QKD scheme [5] is *one-sided* DI. This means that only Alice's quantum device has to be trusted, but no assumption about Bob's measurement device has to be made in order to prove security. Beyond that it does not communicate the measurement outcome to Eve, Bob's measurement device may be arbitrarily malicious.

One-sided DI security of BB84 was first claimed in [38]. However, a close inspection of their proof sketch, which is based on an entropic uncertainty relation with quantum side information, reveals that their arguments are insufficient to prove full one-sided DI security (as confirmed by the authors). It needs to be assumed that Bob's measurement device is *memoryless.* The same holds for the follow up work [37,6] of [38].

One-sided DI security is obviously weaker than fully DI security (as e.g. achieved in [30]). Still, what is interesting is that there is no need for a new scheme — good old BB84 does it. In that sense, we obtain one-sided DI security *for free.* In particular, no hard-to-implement loophole-free Bell tests are needed.

Despite the practical motivation, our result is of theoretical nature. This is because, as in all contemporary fully DI schemes, our analysis (implicitly) assumes that every qubit sent by Alice is indeed received by Bob, or, more generally, whether it is received or not does not depend on the basis it is to be measured in; this is not necessarily satisfied in practical implementations — and some recent attacks on QKD take advantage of exactly this effect by blinding the detectors whenever a measurement in a basis not to Eve's liking is attempted [25].

Our analysis of BB84 QKD with one-sided DI security admits a noise level of up to 1.5%. This is significantly lower than the 11% tolerable for standard (i.e. not DI) security. We believe that this is not inherent to the scheme but an artifact of our analysis. Improving this bound by means of a better analysis is an open problem (it *can* be slightly improved by using a better scheme, e.g., the 6-state scheme). Nonetheless, one-sided DI QKD appears to be an attractive alternative to DI QKD in an asymmetric setting, when we can expect from one party, say, a server, to invest into a very carefully designed, constructed, and tested apparatus, but not the other party, the user, and/or in case of a star network with one designated link being connected with many other links.

Technique. In order to prove one-sided DI security of BB84, we introduce and study a new quantum game, which we call a *monogamy of entanglement* game (or simply a *monogamy* game). This is a game of a specific form, played by three parties, Alice, Bob and Charlie. Of central importance to us is the monogamy game $\mathsf{G}_{\text{BB84}}^{\times n}$, which is as follows.

Preparation Phase: Bob and Charlie agree on and prepare an arbitrary quantum state ρ_{ABC}, where ρ_A consists of n qubits. They pass ρ_A to Alice and hold on to ρ_B and ρ_C, respectively. After this phase, Bob and Charlie are no longer allowed to communicate.

Question Phase: Alice chooses $\theta \in \{0,1\}^n$ uniformly at random and announces θ to Bob and Charlie. Additionally, she measures every qubit ρ_{A_i} of ρ_A in the computational basis if $\theta_i = 0$, and in the Hadamard basis if $\theta_i = 1$. This results in a bit string $x \in \{0,1\}^n$.

Answer Phase: Bob and Charlie independently form a guess of x by performing measurements (which may depend on θ) on ρ_B and ρ_C, respectively.

Winning Condition: The game is won if *both* Bob and Charlie guess x correctly.

From the perspective of classical information processing, our game may appear somewhat trivial — after all, if Bob and Charlie were to provide some classical information k to Alice who would merely apply a randomly chosen function f_θ, they could predict the value of $x = f_\theta(k)$ perfectly from k and θ. In quantum mechanics, however, the outcome of a measurement is in general not deterministic, and the well-known uncertainty principle [15] places a limit on how well observers can predict the outcome of incompatible measurements. For instance, if Bob and Charlie were restricted to classical memory (i.e., ρ_B and ρ_C are "empty"), it is not too hard to see that the best strategy gives a winning probability of $(\frac{1}{2} + \frac{1}{2\sqrt{2}})^n \approx 0.85^n$.

In a fully quantum world, however, uncertainty is not quite the end of the story, as indeed Bob and Charlie are allowed to have *quantum* memory. To illustrate the power of such a memory, consider the same game played just between Alice and Bob. As Einstein, Podolsky and Rosen famously observed [10]: if ρ_{AB} is a maximally entangled state, then once Bob learns Alice's choice of measurement θ, he can perform an adequate measurement on his share of the state to obtain x himself. That is, there exists a strategy for Bob to guess x perfectly. Does this change when we add the extra player, Charlie? We can certainly be hopeful as it is known that quantum entanglement is "monogamous" [34] in the sense that the more entangled Bob is with Alice, the less entangled Charlie can be. In the extreme case where ρ_{AB} is maximally entangled, even if Bob can guess x perfectly every time, Charlie has to resort to making an uninformed random guess. As both of them have to be correct in order to win the game, this strategy turns out to be worse than optimal (see below).

An analysis of our game thus requires a tightrope walk between uncertainty on the one hand, and the monogamy of entanglement on the other. Writing $p_{\text{win}}(\mathsf{G}_{\text{BB84}}^{\times n})$ for the maximal winning probability, maximized over the choice of the initial state ρ_{ABC} and over the measurements performed by Bob and Charlie, we prove that

$$p_{\text{win}}(\mathsf{G}_{\text{BB84}}^{\times n}) = \left(\frac{1}{2} + \frac{1}{2\sqrt{2}}\right)^n . \tag{1}$$

We thus see that, interestingly, monogamy of entanglement wins out entirely, cancelling the power of Bob and Charlie's quantum memory — the optimal winning probability can be achieved without any entanglement at all. We also show a generalization of (1), which upper bounds $p_{\text{win}}(\mathsf{G}_{\text{BB84}}^{\times n})$ for a variant of the game $\mathsf{G}_{\text{BB84}}^{\times n}$ for which Bob and Charlie need to guess the string x only *approximately.*

Our result in particular implies that $p_{\text{win}}(\mathsf{G}_{\text{BB84}}^{\times n}) = p_{\text{win}}(\mathsf{G}_{\text{BB84}})^n$, i.e., *strong parallel repetition* holds. This means that one cannot play n parallel executions of the game $\mathsf{G}_{\text{BB84}} = \mathsf{G}_{\text{BB84}}^{\times 1}$ better than repeating the optimal strategy for one execution n times. Even classically, analyzing the n-fold parallel repetition of games or tasks is typically challenging. In many cases, only non-strong parallel repetition holds, meaning that $p_{\text{win}}(\mathsf{G}^{\times n}) \leq \varepsilon^n$ for some $\varepsilon < 1$, but with $\varepsilon > p_{\text{win}}(\mathsf{G})$. Furthermore, proving such (strong or not) parallel repetition theorems tends to be intriguingly difficult; examples include the parallel repetition of interactive proof systems (see e.g. [29]) or the analysis of communication complexity tasks (see e.g. [19]). In a quantum world, such an analysis is often exacerbated further by the presence of entanglement and the fact that quantum information cannot generally be copied. Famous examples include the analysis of the "parallel repetition" of channels in quantum information theory (where the problem is referred to as the additivity of capacities), see e.g. [14], entangled non-local games [16], or the question whether an eavesdropper's optimal strategy in QKD is to perform the optimal strategy for each round.

In this light, our proof of (1) is surprisingly simple. It is inspired by techniques due to Kittaneh [18] and uses merely tools from linear algebra. At the core of the proof is a newly derived operator norm inequality that bounds the norm $\| \sum_i A_i \|$ of the sum of positive semi-definite operators $A_1, \dots, A_N$ via the respective norms of the square root of pairwise products $A_i A_j$.

In the context of one-sided DI QKD, it turns out that the game $\mathsf{G}_{\text{BB84}}^{\times n}$ pretty much captures an execution of BB84, with Eve playing the role of Charlie, and considering a *gedankenexperiment* where Eve *measures* her quantum side information in order to try to guess the raw key x Alice obtains. Our bound on $p_{\text{win}}(\mathsf{G}_{\text{BB84}}^{\times n})$ then implies that no matter what measurement Bob's device performs, if the outcome of his measurement is strongly correlated to Alice's raw key x, then Eve has a hard time in guessing x. The latter holds for any measurement Eve may perform, and as such it follows that x has lower bounded min-entropy conditioned on Eve's quantum side information. As a consequence, a secret key can be extracted from x using standard techniques.

Further Application. We expect our notion of and our results on monogamy games to find other applications. Indeed, one additional direct application is to *position verification.* Here, we consider a 1-dimensional setting where a *prover* wants to convince two *verifiers* that he controls a certain position, *pos*. The verifiers are located at known positions around *pos*, and they are honest and connected by secure communication channels. Moreover, all parties are assumed to have synchronized clocks, and the message delivery time between any two parties is assumed to be proportional to the distance between them.

Position verification and variants thereof (like *distance bounding*) is a rather well-studied problem in the field of wireless security (see e.g. the references in [9]). It was shown in [9] that in the presence of colluding adversaries at different locations, position verification is impossible classically, even with computational hardness assumptions. That is, the prover can always trick the verifiers into believing that he controls a position. The fact that the classical attack requires the adversary to *copy* information, initially gave hope that we may circumvent the impossibility result using quantum communication. However, such schemes were subsequently broken [17,22] and indeed a general impossibility proof holds [8]: without any restriction on the adversaries, in particular on the amount of pre-shared entanglement they may hold, no quantum scheme for position verification can be secure. This impossibility proof was constructive but required the dishonest parties to share a number of EPR pairs that grows doubly-exponentially in the number of qubits the honest parties exchange. This was reduced by Beigi and König [3] to a single exponential amount. On the other hand, there are schemes for position verification that are provably secure against adversaries that have no pre-shared entanglement, or only hold a couple of entangled qubits [8,22,3].

However, all known schemes that are provably secure with a negligible soundness error (the maximal probability that a coalition of adversaries can pass the position verification test for position *pos* without actually controlling that specific position) against adversaries with no or with bounded pre-shared entanglement are either *multi-round* schemes, or require the honest participants to manipulate large quantum states.

In the full version [36], we present the first provably secure *one-round* position verification scheme with negligible soundness error in which the honest parties are only required to perform single qubit operations. We prove its security against adversaries with an amount of pre-shared entanglement that is *linear* in the number of qubits transmitted by the honest parties.

Outline. In Section 2, we introduce the terminology and notation used throughout this work, and we derive the operator norm inequality that is central to our main result. In Section 3, we discuss the monogamy game $\mathsf{G}_{\text{BB84}}^{\times n}$, prove a strong parallel repetition theorem, and discuss some generalizations. In Section 4, we then make use of these results to prove one-sided DI security of BB84. The application to position verification is given in the full version [36].

2 Technical Preliminaries

Basic Notation and Terminology. We assume the reader to be familiar with the basic concepts of quantum information theory; we merely fix some notation and terminology here.

Let $\mathcal{H}$ be an arbitrary, finite dimensional complex Hilbert space. $\mathcal{L}(\mathcal{H})$ and $\mathcal{P}(\mathcal{H})$ denote *linear* and *positive semi-definite* operators on $\mathcal{H}$, respectively. Note that an operator $A \in \mathcal{P}(\mathcal{H})$ is in particular *Hermitian*, meaning that $A^\dagger = A$. The set of *density operators* on $\mathcal{H}$, i.e., the set of operators in $\mathcal{P}(\mathcal{H})$ with unit trace, is denoted by $\mathcal{S}(\mathcal{H})$. For $A, B \in \mathcal{L}(\mathcal{H})$, we write $A \geq B$ to express that $A - B \in \mathcal{P}(\mathcal{H})$. When operators are compared with scalars, we implicitly assume that the scalars are multiplied by the identity operator, which we denote by $1_{\mathcal{H}}$, or 1 if $\mathcal{H}$ is clear from the context. A *projector* is an operator $P \in \mathcal{P}(\mathcal{H})$ that satisfies $P^2 = P$. A *POVM* (short for *positive operator valued measure*) is a set $\{N_x\}_x$ of operators $N_x \in \mathcal{P}(\mathcal{H})$ such that $\sum_x N_x = 1$, and a POVM is called *projective* if all its elements N_x are projectors. We use the *trace distance*

$$\Delta(\rho, \sigma) := \max_{0 \leq E \leq 1} \operatorname{tr}(E(\rho - \sigma)) = \frac{1}{2}\operatorname{tr}|\rho - \sigma|, \quad \text{where } |L| = \sqrt{L^\dagger L},$$

as a metric on density operators $\rho, \sigma \in \mathcal{S}(\mathcal{H})$.

The most prominent example of a Hilbert space is the qubit space, $\mathcal{H} \equiv \mathbb{C}^2$. The vectors $|0\rangle = \binom{1}{0}$ and $|1\rangle = \binom{0}{1}$ form the *computational* basis, and the vectors $H|0\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$ and $H|1\rangle = (|0\rangle - |1\rangle)/\sqrt{2}$ the *Hadamard* basis, where H denotes the Hadamard matrix. More generally, we often consider systems composed of n qubits, $\mathcal{H} \equiv \mathbb{C}^2 \otimes \cdots \otimes \mathbb{C}^2$. For $x, \theta \in \{0,1\}^n$, we write $|x^\theta\rangle$ as a shorthand for the state vector $H^{\theta_1}|x_1\rangle \otimes \cdots \otimes H^{\theta_n}|x_n\rangle \in \mathcal{H}$.

The Schatten ∞-Norm. For $L \in \mathcal{L}(\mathcal{H})$, we use the Schatten ∞-norm $\|L\| := \|L\|_\infty = s_1(L)$, which evaluates the largest singular value of L. It is easy to verify that this norm satisfies $\|L\|^2 = \|L^\dagger L\| = \|LL^\dagger\|$. Also, for $A, B \in \mathcal{P}(\mathcal{H})$, $\|A\|$ coincides with the largest eigenvalue of A, and $A \leq B$ implies $\|A\| \leq \|B\|$. Finally, for any block-diagonal operator $A \oplus B$ we have $\|A \oplus B\| = \max\{\|A\|, \|B\|\}$.

We need the following fact. Note that the statement does not hold in general if the projectors are replaced by general positive semi-definite operators.

Lemma 2.1. *Let $P, Q \in \mathcal{P}(\mathcal{H})$ be projectors with $P \leq Q$, and let $L \in \mathcal{L}(\mathcal{H})$. Then, it holds that $\|PL\| \leq \|QL\|$ and $\|LP\| \leq \|LQ\|$.*

Proof. $\|PL\|^2 = \|L^\dagger P^\dagger PL\| = \|L^\dagger PL\| \leq \|L^\dagger QL\| = \|L^\dagger Q^\dagger QL^\dagger\| = \|QL\|^2$, and the proof of the second statement follows analogously. □

Applying the lemma twice, we get $\|PQ\|^2 \leq \|P'Q\|^2 \leq \|P'Q'\|^2 = \|P'Q'P'\|$ for any two pairs of projectors satisfying $P \leq P'$ and $Q \leq Q'$.

One of our main tools is the following Lemma 2.2, which bounds the Schatten norm of the sum of n positive semi-definite operators by means of their pairwise products. We derive the bound using a construction due to Kittaneh [18], which was also used by Schaffner [32] to derive a similar, but less general, result.

We call two permutations $\pi : [N] \to [N]$ and $\pi' : [N] \to [N]$ of the set $[N] := \{1, \ldots, N\}$ *orthogonal* if $\pi(i) \neq \pi'(i)$ for all $i \in [N]$. The N cyclic shifts for instance form a set of N permutations of $[N]$ that are mutually orthogonal.

Lemma 2.2. *Let $A_1, A_2, \ldots, A_N \in \mathcal{P}(\mathcal{H})$, and let $\{\pi^k\}_{k\in[N]}$ be a set of N mutually orthogonal permutations of $[N]$. Then,*

$$\Big\| \sum_{i\in[N]} A_i \Big\| \leq \sum_{k\in[N]} \max_{i\in[N]} \Big\| \sqrt{A_i}\sqrt{A_{\pi^k(i)}} \Big\| .$$

Proof. We define $X = [X_{ij}]$ as the $N \times N$ block-matrix with blocks given by $X_{ij} = \delta_{j1}\sqrt{A_i}$. The two matrices $X^\dagger X$ and $XX^\dagger$ are easy to evaluate, namely $(X^\dagger X)_{ij} = \delta_{i1}\delta_{j1}\sum_i A_i$ and $(XX^\dagger)_{ij} = \sqrt{A_i}\sqrt{A_j}$, respectively. As such, we see that $\|\sum_i A_i\| = \|X^\dagger X\| = \|XX^\dagger\|$.

Next, we decompose $XX^\dagger$ into $XX^\dagger = D_1 + D_2 + \ldots D_N$, where the matrices D_k are defined by the permutations π^k, respectively, as $(D_k)_{ij} = \delta_{j,\pi^k(i)}\sqrt{A_i}$ $\sqrt{A_j}$. The requirement on the permutations ensures that $XX^\dagger = \sum_k D_k$. Moreover, since the matrices D_k are constructed such that they contain exactly one non-zero block in each row and column, they can be transformed into a block-diagonal matrix $D'_k = \bigoplus_i \sqrt{A_i}\sqrt{A_{\pi^k(i)}}$ by a unitary rotation. Hence, using triangle inequality and the unitary invariance of the norm, we get $\|\sum_k A_k\| = \|XX^\dagger\| \leq \sum_k \|D_k\| = \sum_k \|D'_k\| = \sum_k \max_i \|\sqrt{A_i}\sqrt{A_{\pi^k(i)}}\|$. □

CQ-States and Min-Entropy. A state $\rho_{XB} \in \mathcal{S}(\mathcal{H}_X \otimes \mathcal{H}_B)$ is called a *classical-quantum* (CQ) state with classical X over $\mathcal{X}$, if it is of the form

$$\rho_{XB} = \sum_{x\in\mathcal{X}} p_x |x\rangle\langle x|_X \otimes \rho_B^x ,$$

where $\{|x\rangle\}_{x\in\mathcal{X}}$ is a fixed basis of $\mathcal{H}_X$, $\{p_x\}_{x\in\mathcal{X}}$ is a probability distribution, and $\rho_B^x \in \mathcal{S}(\mathcal{H}_B)$. For such a state, X can be understood as a random variable that is correlated with (potentially quantum) side information B.

If $\lambda : \mathcal{X} \to \{0,1\}$ is a predicate on $\mathcal{X}$, then we denote by $\Pr_\rho[\lambda(X)]$ the probability of the *event* $\lambda(X)$ under ρ; formally, $\Pr_\rho[\lambda(X)] = \sum_x p_x\, \lambda(x)$.

We also define the state $\rho_{XB|\lambda(X)}$, which is the state of the X and B conditioned on the event $\lambda(X)$. Formally,

$$\rho_{XB|\lambda(X)} = \frac{1}{\Pr_\rho[\lambda(X)]} \sum_x p_x \lambda(x) |x\rangle\langle x|_X \otimes \rho_B^x \,.$$

For a CQ-state $\rho_{XB} \in \mathcal{S}(\mathcal{H}_X \otimes \mathcal{H}_B)$, the *min-entropy* of X conditioned on B [31] can be expressed in terms of the maximum probability that a measurement on B yields the correct value of X, i.e. the guessing probability. Formally, we define [20] $H_{\min}(X|B)_\rho := -\log p_{\text{guess}}(X|B)_\rho$, where

$$p_{\text{guess}}(X|B)_\rho := \max_{\{N_x\}_x} \sum_x p_x \operatorname{tr}(\rho_B^x N_x).$$

Here, the optimization is taken over all POVMs $\{N_x\}_x$ on B, and here and throughout this paper, log denotes the binary logarithm.

In case of a CQ-state $\rho_{XB\Theta}$ with classical X, and with additional classical side information Θ, we can write $\rho_{XB\Theta} = \sum_\theta p_\theta \, |\theta\rangle\langle\theta| \otimes \rho_{XB}^\theta$ for CQ states ρ_{XB}^θ. The min-entropy of X conditioned on B and Θ then evaluates to $H_{\min}(X|B\Theta)_\rho = -\log p_{\text{guess}}(X|B\Theta)_\rho$, where $p_{\text{guess}}(X|B\Theta)_\rho = \sum_\theta p_\theta \, p_{\text{guess}}(X|B)_{\rho^\theta}$. An intuitive explanation of the latter equality is that the optimal strategy to guess X simply chooses an optimal POVM on B depending on the value of Θ.

An overview of the min-entropy and its properties can be found in [35]. We merely point out the *chain rule* here: for a CQ-state $\rho_{XB\Theta}$ with classical X and Θ, where Θ is over $\{0,1\}^n$, it holds that $H_{\min}(X|B\Theta)_\rho \geq H_{\min}(X|B)_\rho - n$.

3 Parallel Repetition of Monogamy Games

In this section, we formalize the notion of a monogamy game, and we show strong parallel repetition for the game $\mathsf{G}_{\text{BB84}}^{\times n}$. Then, we generalize our analysis to arbitrary projective measurements for Alice, and to the case where Bob and Charlie are allowed to make some errors.

Definition 3.1. *A* monogamy-of-entanglement game G *consists of a finite dimensional Hilbert space $\mathcal{H}_A$ and a list of projective measurements $\mathcal{M}^\theta = \{F_x^\theta\}_{x\in\mathcal{X}}$ on a $\mathcal{H}_A$, indexed by $\theta \in \Theta$, where $\mathcal{X}$ and Θ are finite sets.*

We typically use less bulky terminology and simply call G a *monogamy game.* Note that for any positive integer n, the n-fold *parallel repetition* of G, denoted as $\mathsf{G}^{\times n}$ and naturally specified by $\mathcal{H}_A^{\otimes n}$ and $\{F_{x_1}^{\theta_1} \otimes \cdots \otimes F_{x_n}^{\theta_n}\}_{x_1,\ldots,x_n}$ for $\theta_1, \ldots, \theta_n \in \Theta$, is again a monogamy game.

Definition 3.2. *We define a* strategy $\mathcal{S}$ *for a monogamy game* G *as a list*

$$\mathcal{S} = \left\{\rho_{ABC}, \, P_x^\theta, \, Q_x^\theta\right\}_{\theta\in\Theta, x\in\mathcal{X}} , \tag{2}$$

where $\rho_{ABC} \in \mathcal{S}(\mathcal{H}_A \otimes \mathcal{H}_B \otimes \mathcal{H}_C)$, and $\mathcal{H}_B$ and $\mathcal{H}_C$ are arbitrary finite dimensional Hilbert spaces. Furthermore, for all $\theta \in \Theta$, $\{P_x^\theta\}_{x\in\mathcal{X}}$ and $\{Q_x^\theta\}_{x\in\mathcal{X}}$ are POVMs on $\mathcal{H}_B$ and $\mathcal{H}_C$, respectively. A strategy is called pure *if the state ρ_{ABC} is pure and all the POVMs are projective.*

If $\mathcal{S}$ is a strategy for game G, then the n-fold parallel repetition of $\mathcal{S}$, which is naturally given, is a particular strategy for the parallel repetition $\mathsf{G}^{\times n}$; however, it is important to realize that there exist strategies for $\mathsf{G}^{\times n}$ that are not of this form. In general, a strategy $\mathcal{S}_n$ for $\mathsf{G}^{\times n}$ is given by an arbitrary state $\rho_{A_1 \ldots A_n BC} \in \mathcal{S}(\mathcal{H}_A^{\otimes n} \otimes \mathcal{H}_B \otimes \mathcal{H}_C)$ (with arbitrary $\mathcal{H}_B$ and $\mathcal{H}_C$) and by arbitrary POVM elements on $\mathcal{H}_B$ and $\mathcal{H}_C$, respectively, not necessarily in product form.

The winning probability for a game G and a fixed strategy $\mathcal{S}$, denoted by $p_{\text{win}}(\mathsf{G}, \mathcal{S})$, is defined as the probability that the measurement outcomes of Alice, Bob and Charlie agree when Alice measures in the basis determined by a randomly chosen $\theta \in \Theta$ and Bob and Charlie apply their respective POVMs $\{P_x^\theta\}_x$ and $\{Q_x^\theta\}_x$. The optimal winning probability, $p_{\text{win}}(\mathsf{G})$, maximizes the winning probability over all strategies. The following makes this formal.

Definition 3.3. *The winning probability for a monogamy game* G *and a strategy* $\mathcal{S}$ *is defined as*

$$p_{\text{win}}(\mathsf{G}, \mathcal{S}) := \sum_{\theta \in \Theta} \frac{1}{|\Theta|} \operatorname{tr}\big(\Pi^\theta \rho_{ABC}\big), \quad \textit{where} \quad \Pi^\theta := \sum_{x \in \mathcal{X}} F_x^\theta \otimes P_x^\theta \otimes Q_x^\theta . \tag{3}$$

The optimal winning probability is $p_{\text{win}}(\mathsf{G}) := \sup_{\mathcal{S}} \, p_{\text{win}}(\mathsf{G}, \mathcal{S})$*, where the supremum is taken over all strategies* $\mathcal{S}$ *for* G.

In fact, due to a standard purification argument and Neumark's dilation theorem, we can restrict the supremum to pure strategies (cf. [36]).

Strong Parallel Repetition for G_{BB84}. We are particularly interested in the game G_{BB84} and its parallel repetition $\mathsf{G}_{\text{BB84}}^{\times n}$. The latter is given by $\mathcal{H}_A = (\mathbb{C}^2)^{\otimes n}$ and the projectors $F_x^\theta = |x^\theta\rangle\langle x^\theta| = H^{\theta_1}|x_1\rangle\langle x_1|H^{\theta_1} \otimes \cdots \otimes H^{\theta_n}|x_n\rangle\langle x_n|H^{\theta_n}$ for $\theta, x \in \{0,1\}^n$. The following shows the exact value of $p_{\text{win}}(\mathsf{G}_{\text{BB84}}^{\times n})$, and in particular it shows strong parallel repetition.

Theorem 3.4. *For any* $n \in \mathbb{N}$, $n \geq 1$, *we have*

$$p_{\text{win}}(\mathsf{G}_{\text{BB84}}^{\times n}) = \left(\frac{1}{2} + \frac{1}{2\sqrt{2}}\right)^n . \tag{4}$$

Proof. We first show that this probability can be achieved. For $n = 1$, consider the following strategy. Bob and Charlie prepare the state $|\phi\rangle := \cos\frac{\pi}{8}|0\rangle + \sin\frac{\pi}{8}|1\rangle$ and send it to Alice. Then, they guess that Alice measures outcome 0, independent of θ. Formally, this is the strategy $\mathcal{S}_1 = \big\{|\phi\rangle\langle\phi|, P_x^\theta = \delta_{x0}, Q_x^\theta = \delta_{x0}\big\}$. The optimal winning probability is bounded by the winning probability of this strategy,

$$p_{\text{win}}(\mathsf{G}_{\text{BB84}}) \geq \left(\cos\frac{\pi}{8}\right)^2 = \frac{1}{2} + \frac{1}{2\sqrt{2}} ,$$

and the lower bound in Eq. (4) follows by repeating this simple strategy n times.

To show that this simple strategy is optimal, let us now fix an arbitrary, pure strategy $\mathcal{S}_n = \{\rho_{A_1\ldots A_n BC}, P_x^\theta, Q_x^\theta\}$. From the definition of the norm, we have $\operatorname{tr}(M\rho_{ABC}) \leq \|M\|$ for any $M \geq 0$. Using this and Lemma 2.2, we find

$$p_{\text{win}}(\mathsf{G}_{\text{BB84}}^{\times n}, \mathcal{S}_n) \leq \frac{1}{2^n}\Big\| \sum_\theta \Pi^\theta \Big\| \leq \frac{1}{2^n} \sum_k \max_\theta \big\|\Pi^\theta \Pi^{\pi^k(\theta)}\big\|, \tag{5}$$

where the optimal permutations π^k are to be determined later. Hence, the problem is reduced to bounding the norms $\big\|\Pi^\theta \Pi^{\theta'}\big\|$, where $\theta' = \pi^k(\theta)$. The trivial upper bound on these norms, 1, leads to $p_{\text{win}}(\mathsf{G}_{\text{BB84}}^{\times n}, \mathcal{S}_n) \leq 1$. However, most of these norms are actually very small as we see below.

For fixed θ and k, we denote by $\mathcal{T}$ the set of indices where θ and θ' differ, by $\mathcal{T}^c$ its complement, and by t the Hamming distance between θ and θ' (i.e., $t = |\mathcal{T}|$). Consider the projectors

$$\bar{P} = \sum_x |x_\mathcal{T}^\theta\rangle\langle x_\mathcal{T}^\theta| \otimes 1_{\mathcal{T}^c} \otimes P_x^\theta \otimes 1_C \;\text{ and }\; \bar{Q} = \sum_x |x_\mathcal{T}^{\theta'}\rangle\langle x_\mathcal{T}^{\theta'}| \otimes 1_{\mathcal{T}^c} \otimes 1_B \otimes Q_x^{\theta'},$$

where $|x_\mathcal{T}^\theta\rangle$ is $|x^\theta\rangle$ restricted to the systems corresponding to rounds with index in $\mathcal{T}$, and $1_{\mathcal{T}^c}$ is the identity on the remaining systems.

Since $\Pi^\theta \leq \bar{P}$ and $\Pi^{\theta'} \leq \bar{Q}$, we can bound $\big\|\Pi^\theta \Pi^{\theta'}\big\|^2 \leq \big\|\bar{P}\bar{Q}\bar{P}\big\|$ using Lemma 2.1. Moreover,

$$\begin{aligned}
\bar{P}\bar{Q}\bar{P} &= \sum_{x,y,z} |x_\mathcal{T}^\theta\rangle\langle x_\mathcal{T}^\theta | y_\mathcal{T}^{\theta'}\rangle\langle y_\mathcal{T}^{\theta'} | z_\mathcal{T}^\theta\rangle\langle z_\mathcal{T}^\theta| \otimes 1_{\mathcal{T}^c} \otimes P_x^\theta P_z^\theta \otimes Q_y^{\theta'} \\
&= \sum_{x,y} |\langle x_\mathcal{T}^\theta | y_\mathcal{T}^{\theta'}\rangle|^2 \, |x_\mathcal{T}^\theta\rangle\langle x_\mathcal{T}^\theta| \otimes 1_{\mathcal{T}^c} \otimes P_x^\theta \otimes Q_y^{\theta'} \\
&= 2^{-t} \sum_x |x_\mathcal{T}^\theta\rangle\langle x_\mathcal{T}^\theta| \otimes 1_{\mathcal{T}^c} \otimes P_x^\theta \otimes 1_C,
\end{aligned}$$

where we used that $P_x^\theta P_z^\theta = \delta_{xz} P_x^\theta$ and $|\langle x_\mathcal{T}^\theta | y_\mathcal{T}^{\theta'}\rangle|^2 = 2^{-t}$. The latter relation follows from the fact that the two bases are diagonal to each other on each qubit with index in $\mathcal{T}$. From this follows directly that $\|\bar{P}\bar{Q}\bar{P}\| = 2^{-t}$. Hence, we find $\big\|\Pi^\theta \Pi^{\theta'}\big\| \leq \sqrt{2^{-t}}$. Note that this bound is independent of the strategy and only depends on the Hamming distance between θ and θ'.

To minimize the upper bound in (5), we should choose permutations π^k that produce tuples $(\theta, \theta' = \pi^k(\theta))$ with the same Hamming distance as this means that the maximization is over a uniform set of elements. A complete mutually orthogonal set of permutations with this property is given by the bitwise XOR, $\pi^k(\theta) = \theta \oplus k$, where we interpret k as an element of $\{0,1\}^n$. Using this construction, we get exactly $\binom{n}{t}$ permutations that create pairs with Hamming distance t, and the bound in Eq. (5) evaluates to

$$\frac{1}{2^n} \sum_k \max_\theta \big\|\Pi^\theta \Pi^{\pi^k(\theta)}\big\| \leq \frac{1}{2^n} \sum_{t=0}^n \binom{n}{t} \left(\frac{1}{\sqrt{2}}\right)^t = \left(\frac{1}{2} + \frac{1}{2\sqrt{2}}\right)^n.$$

As this bound applies to all pure strategies, we conclude the proof. $\square$

Arbitrary Games, and Imperfect Guessing. The above upper-bound techniques can be generalized to an arbitrary monogamy game, G, specified by an arbitrary finite dimensional Hilbert space $\mathcal{H}_A$ and arbitrary projective measurements $\{F^\theta_x\}_{x\in\mathcal{X}}$, indexed by $\theta \in \Theta$, and with arbitrary finite $\mathcal{X}$ and Θ. The only additional parameter relevant for the analysis is the *maximal overlap* of the measurements, $c(\mathsf{G}) := \max \|F^\theta_x F^{\theta'}_{x'}\|^2$, where the max is over all $\theta \neq \theta' \in \Theta$ and all x, $x' \in \mathcal{X}$. $c(\mathsf{G})$ satisfies $1/|\mathcal{X}| \leq c(\mathsf{G}) \leq 1$ and $c(\mathsf{G}^{\times n}) = c(\mathsf{G})^n$. This is in accordance with the definition of the overlap as it appears in entropic uncertainty relations, e.g. in [21]. Note also that in the case of G_{BB84}, we have $c(\mathsf{G}_{\text{BB84}}) = \frac{1}{2}$.

In addition to considering arbitrary monogamy games, we also generalize Theorem 3.4 to the case where Bob and Charlie are not required to guess *perfectly* but are allowed to make some errors. The maximal winning probability in this case is defined as follows, where we again restrict to pure strategies.

Definition 3.5. *Let $\mathcal{Q} = \{(\pi^q_B, \pi^q_C)\}_q$ be a set of pairs of permutations of $\mathcal{X}$, indexed by q, with the meaning that in order to win, Bob and Charlie's respective guesses for x must form a pair in $\{(\pi^q_B(x), \pi^q_C(x))\}_q$. Then, the optimal winning probability of G with respect to $\mathcal{Q}$ is*

$$p_{\text{win}}(\mathsf{G}; \mathcal{Q}) := \sup_{\mathcal{S}} \sum_{\theta\in\Theta} \frac{1}{|\Theta|} \operatorname{tr}(\Pi^\theta \rho_{ABC}) \;\; \textit{with} \;\; \Pi^\theta := \sum_{x\in\mathcal{X}} F^\theta_x \otimes \sum_q P^\theta_{\pi^q_B(x)} \otimes Q^\theta_{\pi^q_C(x)}$$

where the supremum is taken over all pure strategies $\mathcal{S}$ for G.

We find the following upper bound on the guessing probability, generalizing the upper bound on the optimal winning probability established in Theorem 3.4. The proof closely follows the proof of the upper bound in Theorem 3.4, and is deferred to the full version [36].

Theorem 3.6. *For any positive $n \in \mathbb{N}$, we have*

$$p_{\text{win}}(\mathsf{G}^{\times n}; \mathcal{Q}) \leq |\mathcal{Q}| \left(\frac{1}{|\Theta|} + \frac{|\Theta| - 1}{|\Theta|} \sqrt{c(\mathsf{G})} \right)^n .$$

Recall that in case of G_{BB84}, we have $|\mathcal{Q}| = 1$, $|\Theta| = 2$, and $c(\mathsf{G}_{\text{BB84}}) = \frac{1}{2}$, leading to the bound stated in Theorem 3.4.

One particularly interesting example of the above theorem considers binary measurements, i.e. $\mathcal{X} = \{0, 1\}$, where Alice will accept Bob's and Charlie's answers if and only if they get less than a certain fraction of bits wrong. More precisely, she accepts if $d(x, y) \leq \gamma n$ and $d(x, z) \leq \gamma' n$, where $d(\cdot, \cdot)$ denotes the Hamming distance and y, z are Bob's and Charlie's guesses, respectively. In this case, we let $\mathcal{Q}^n_{\gamma,\gamma'}$ consist of all pairs of permutations (π^q_B, π^q_C) on $\{0, 1\}^n$ of the form $\pi^q_B(x) = x \oplus k$, $\pi^q_C(x) = x \oplus k'$, where $q = \{k, k'\}$, and $k, k' \in \{0, 1\}^n$ have Hamming weight at most γ and γ', respectively. One can upper bound $|\mathcal{Q}^n_{\gamma,\gamma'}| \leq 2^{nh(\gamma)+nh(\gamma')}$, where $h(\cdot)$ denotes the binary entropy. We thus find

$$p_{\text{win}}(\mathsf{G}^{\times n}; \mathcal{Q}^n_{\gamma,\gamma'}) \leq \left(2^{h(\gamma)+h(\gamma')} \, \frac{1 + (|\Theta| - 1)\sqrt{c(\mathsf{G})}}{|\Theta|} \right)^n .$$

4 Application: One-Sided Device-Independent QKD

In the following, we assume some familiarity with quantum key distribution (QKD). For simplicity, we consider an entanglement-based [11] variant of the BB84 QKD scheme [5], where Bob waits with performing the measurement until Alice tells him the right bases. This protocol is impractical because it requires Bob to store qubits. However, it is well known that security of this impractical version implies security of the original, more practical BB84 QKD scheme [4]. It is straightforward to verify that this implication also holds in the one-sided device-independent setting we consider here.

The entanglement-based QKD scheme, **E-QKD**, is described in Figure 1. It is (implicitly) parameterized by positive integers $0 < t, s, \ell < n$ and a real number $0 \leq \gamma < \frac{1}{2}$. Here, n is the number of qubits exchanged between Alice and Bob, t is the size of the sample used for parameter estimation, s is the leakage (in bits) due to error correction, and ℓ is the length (in bits) of the final key. Finally, γ is the tolerated error in Bob's measurement results.

State Preparation: Alice prepares n EPR pairs $\frac{1}{\sqrt{2}}\big(|0\rangle \otimes |0\rangle + |1\rangle \otimes |1\rangle\big)$. Then, of each pair, she keeps one qubit and sends the other to Bob.

Confirmation: Bob confirms receipt of the n qubits. (After this point, there cannot be any communication between Bob's device and Eve.)

Measurement: Alice chooses random $\Theta \in \{0,1\}^n$ and sends it to Bob, and Alice and Bob measure the EPR pairs in basis Θ to obtain X and Y, respectively. (Remember: Bob's device may produce Y in an arbitrary way, using a POVM chosen depending on Θ acting on a state provided by Eve.)

Parameter Estimation: Alice chooses a random subset $T \subset \{1, \dots, n\}$ of size t, and sends T and X_T to Bob. If the relative Hamming distance, $d_{\text{rel}}(X_T, Y_T)$, exceeds γ then they abort the protocol and set $K = \perp$.

Error Correction: Alice sends a syndrome $S(X_{\bar{T}})$ of length s and a random universal$_2$ hash function $F : \{0,1\}^{n-t} \to \{0,1\}^\ell$ to Bob.

Privacy Amplification: Alice computes $K = F(X_{T^c})$ and Bob $\hat{K} = F(\hat{X}_{T^c})$, where $\hat{X}_{T^c}$ is the corrected version of Y_{T^c}.

Fig. 1. An entanglement-based QKD scheme **E-QKD**

A QKD protocol is called *perfectly secure* if it either aborts and outputs an empty key, $K = \perp$, or it produces a key that is uniformly random and independent of Eve's (quantum and classical) information E^+ gathered during the execution of the protocol. Formally, this means that the final state must be of the form $\rho_{KE^+} = \Pr_\rho[K \neq \perp] \cdot \mu_K \otimes \rho_{E^+|K\neq\perp} + \Pr_\rho[K = \perp] \cdot |\perp\rangle\langle\perp|_K \otimes \rho_{E^+|K=\perp}$, where μ_K is a 2^ℓ-dimensional completely mixed state, and $|\perp\rangle\langle\perp|_K$ is orthogonal to μ_K.

Relaxing this condition, a protocol is called *δ-secure* if ρ_{KE^+} is δ-close to the above form in trace distance, meaning that ρ_{KE^+} satisfies

$$\Pr_{\rho}[K \neq \perp] \cdot \Delta(\rho_{KE^+|K\neq\perp}, \mu_K \otimes \rho_{E^+|K\neq\perp}) \leq \delta\,. \tag{6}$$

It is well known and has been proven in various ways that **E-QKD** is δ-secure (with small δ) with a suitable choice of parameters, assuming that all quantum operations are correctly performed by Alice and Bob. We now show that the protocol remains secure even if Bob's measurement device behaves arbitrarily and possibly maliciously. The only assumption is that Bob's device does not communicate with Eve after it received Alice's quantum signals. This restriction is clearly necessary as there would otherwise not be any asymmetry between Bob and Eve's information about Alice's key. Note that the scheme is well known to satisfy *correctness* and *robustness*; hence, we do not argue these here.

Theorem 4.1. *Consider an execution of* **E-QKD***, with an arbitrary measurement device for Bob. Then, for any $\varepsilon > 0$, protocol* **E-QKD** *is δ-secure with*

$$\delta = 5e^{-2\varepsilon^2 t} + \ 2^{-\frac{1}{2}\left(\log(1/\beta_\circ)n - h(\gamma+\varepsilon)n - \ell - t - s + 2\right)} \quad \textit{where} \quad \beta_\circ = \frac{1}{2} + \frac{1}{2\sqrt{2}}.$$

Note that with an optimal error correcting code, the size of the syndrome for large n approaches the Shannon limit $s = nh(\gamma)$. The security error δ can then be made negligible in n with suitable choices of parameters if $\log(1/\beta_\circ) > 2h(\gamma)$, which roughly requires that $\gamma \leq 0.015$. Hence, the scheme can tolerate a noise level up to 1.5% asymptotically.[1]

The formal proof is given below. The idea is rather simple: We consider a *gedankenexperiment* where Eve *measures* her system, using an arbitrary POVM, with the goal to guess X. The execution of **E-QKD** then pretty much coincides with $\mathsf{G}_{\mathrm{BB84}}^{\times n}$, and we can conclude from our results that if Bob's measurement outcome Y is close to X, then Eve must have a hard time in guessing X. Since this holds for any measurement she may perform, this means her min-entropy on X is large and hence the extracted key K is secure.

Proof. Let $\rho_{\Theta TABE} = \rho_\Theta \otimes \rho_T \otimes |\psi_{ABE}\rangle\langle\psi_{ABE}|$ be the state before Alice and Bob perform the measurements on A and B, respectively, where system E is held by the adversary Eve. Here, the random variable Θ contains the choice of basis for the measurement, whereas the random variable T contains the choice of subset on which the strings are compared (see the protocol description in Fig. 1.) Moreover, let $\rho_{\Theta TXYE}$ be the state after Alice and Bob measured, where — for every possible value θ — Alice's measurement is given by the projectors $\{|x^\theta\rangle\langle x^\theta|\}_x$, and Bob's measurement by an arbitrary but fixed POVM $\{P_x^\theta\}_x$.

As a *gedankenexperiment*, we consider the scenario where Eve wants to guess the value of Alice's raw key, X. Eve wants to do this during the parameter estimation step of the protocol, exactly *after* Alice broadcast T but *before* she broadcasts X_T.[2] For this purpose, we consider an arbitrary measurement

[1] This can be improved slightly by instead considering a six-state protocol, where the measurement is randomly chosen among three mutually unbiased bases on the qubit.

[2] Note that the effect of Eve learning X_T is taken into account later, in Eq. (8).

strategy of Eve that aims to guess X. Such a strategy is given by — for every basis choice, θ, and every choice of sample, τ — a POVM $\{Q_x^{\theta,\tau}\}_x$. The values of θ and τ have been broadcast over a public channel, and are hence known to Eve at this point of the protocol. She will thus choose a POVM depending on these values to measure E and use the measurement outcome as her guess.

For our *gedankenexperiment*, we will use the state, $\rho_{\Theta TXYZ}$, which is the (purely classical) state that results after Eve applied her measurement on E. Let $\varepsilon > 0$ be an arbitrary constant. By our results from Section 3, it follows that for any choices of $\{P_x^\theta\}_x$ and $\{Q_x^{\theta,\tau}\}_x$, we have

$$\Pr_\rho[d_{\mathrm{rel}}(X,Y) \leq \gamma+\varepsilon \ \wedge \ Z=X] \leq p_{\mathrm{win}}(\mathsf{G}_{\mathrm{BB84}}^{\times n}; \mathcal{Q}_{\gamma+\varepsilon,0}^n) \leq \beta^n$$

with $\beta = 2^{h(\gamma+\varepsilon)} \cdot \beta_\circ$, where d_{rel} denotes the relative Hamming distance. This uses the fact that Alice's measurement outcome is independent of T, and T can in fact be seen as part of Eve's system for the purpose of the monogamy game.

We now construct a state $\tilde{\rho}_{\Theta TXYE}$ as follows.

$$\tilde{\rho}_{\Theta TXYE} = \Pr_\rho[\Omega] \cdot \rho_{\Theta TXYE|\Omega} + \big(1 - \Pr_\rho[\Omega]\big) \cdot \sigma_{\Theta TXYE},$$

where Ω denotes the event $\Omega = \{d_{\mathrm{rel}}(X,Y) \leq d_{\mathrm{rel}}(X_T,Y_T) + \varepsilon\}$, and we take $\sigma_{T\Theta XYE}$ to be an arbitrary state with classical Θ, T, X and Y for which $d_{\mathrm{rel}}(X,Y) = 1$, and hence $d_{\mathrm{rel}}(X_T,Y_T) = 1$. Informally, the event Ω indicates that the relative Hamming distance of the sample strings X_T and Y_T determined by T was representative of the relative Hamming distance between the whole strings, X and Y, and the state $\tilde{\rho}_{\Theta TXYE}$ is so that this is satisfied with certainty. By construction of $\tilde{\rho}_{\Theta TXYE}$, we have $\Delta(\rho_{\Theta TXYE}, \tilde{\rho}_{\Theta TXYE}) \leq 1 - \Pr_\rho[\Omega]$, and by Hoeffding's inequality,

$$1 - \Pr_\rho[\Omega] = \Pr_\rho[d_{\mathrm{rel}}(X,Y) > d_{\mathrm{rel}}(X_T,Y_T) + \varepsilon] \leq e^{-2\varepsilon^2 t}.$$

Moreover, note that the event $d_{\mathrm{rel}}(X_T,Y_T) \leq \gamma$ implies $d_{\mathrm{rel}}(X,Y) \leq \gamma + \varepsilon$ under $\tilde{\rho}_{\Theta TXYE}$. Thus, for every choice of strategy $\{Q_x^{\theta,\tau}\}_x$ by the eavesdropper, the resulting state $\tilde{\rho}_{\Theta TXYZ}$, obtained by applying $\{Q_x^{\theta,\tau}\}_x$ to E, satisfies

$$\begin{aligned}\Pr_{\tilde{\rho}}[d_{\mathrm{rel}}(X_T,Y_T) \leq \gamma \wedge Z=X] &\leq \Pr_{\tilde{\rho}}[d_{\mathrm{rel}}(X,Y) \leq \gamma+\varepsilon \wedge Z=X] \\ &\leq \Pr_\rho[d_{\mathrm{rel}}(X,Y) \leq \gamma+\varepsilon \wedge Z=X] \leq \beta^n.\end{aligned} \tag{7}$$

We now introduce the event $\Gamma = \{d_{\mathrm{rel}}(X_T,Y_T) \leq \gamma\}$, which corresponds to the event that Bob does not abort the protocol. Expanding the left hand side of (7) to $\Pr_{\tilde{\rho}}[\Gamma] \cdot \Pr_{\tilde{\rho}}[Z = X|\Gamma]$ and observing that $\Pr_{\tilde{\rho}}[\Gamma]$ does not depend on the strategy $\{Q_x^{\theta,\tau}\}_x$, we can conclude that

$$\forall \{Q_x^{\theta,\tau}\}_x : \ \Pr_{\tilde{\rho}}[Z=X|\Gamma] \leq \beta^{(1-\alpha)n}$$

where $\alpha \geq 0$ is determined by $\Pr_{\tilde{\rho}}[\Gamma] = \beta^{\alpha n}$. Therefore, by definition of the min-entropy, $H_{\min}(X|\Theta TE, \Gamma)_{\tilde{\rho}} \geq n(1-\alpha)\log(1/\beta)$. (This notation means that the min-entropy of X given Θ, T and E is evaluated for the state $\tilde{\rho}_{\Theta TXYE|\Gamma}$, conditioned on not aborting.) By the chain rule, it now follows that

$$\begin{aligned} H_{\min}(X|\Theta TX_TSE,\Gamma)_{\tilde{\rho}} &\geq H_{\min}(XX_TS|\Theta TE,\Gamma)_{\tilde{\rho}} - t - s \\ &\geq n(1-\alpha)\log(1/\beta) - t - s\,. \end{aligned} \tag{8}$$

Here, the min-entropy is evaluated for the state $\tilde{\rho}_{X\Theta TX_TSE}$ that is constructed from $\tilde{\rho}_{X\Theta TE}$ by calculating the error syndrome and copying X_T from X as done in the prescription of the protocol. In particular, $\Delta(\tilde{\rho}_{X\Theta TX_TSE}, \rho_{X\Theta TX_TSE}) \leq e^{-2\varepsilon^2 t}$. Finally, privacy amplification with universal$_2$ hashing applied to the state $\tilde{\rho}_{X\Theta TX_TSE}$ ensures that the key K satisfies [31]

$$\Delta(\tilde{\rho}_{KF\Theta TX_TSE|\Gamma},\, \mu_K \otimes \tilde{\rho}_{F\Theta TX_TE|\Gamma}) \leq \frac{1}{2}\sqrt{\beta^{(1-\alpha)n}\, 2^{\ell+t+s}}\,.$$

And, in particular, recalling that $\Pr_{\tilde{\rho}}[\Gamma] = \beta^{\alpha n}$, we have

$$\Pr_{\tilde{\rho}}[\Gamma]\cdot\Delta(\tilde{\rho}_{KF\Theta TX_TSE|\Gamma},\, \mu_K \otimes \tilde{\rho}_{F\Theta TX_TE|\Gamma}) \leq \frac{1}{2}\sqrt{\beta^{n}\, 2^{\ell+t+s}}\,.$$

Using $\beta = 2^{h(\gamma+\varepsilon)}\beta_\circ$ and applying Lemma 4.2 below concludes the proof. □

Lemma 4.2. *Let ρ_{XB}, $\tilde{\rho}_{XB} \in \mathcal{S}(\mathcal{H}_X \otimes \mathcal{H}_B)$ be two CQ states with X over $\mathcal{X}$. Also, let $\lambda : \mathcal{X} \to \{0,1\}$ be a predicate on $\mathcal{X}$ and $\Lambda = \lambda(X)$, and let $\tau_X \in \mathcal{S}(\mathcal{H}_X)$ be arbitrary. Then*

$$\Pr_{\rho}[\Lambda]\cdot\Delta(\rho_{XB|\Lambda},\, \tau_X \otimes \rho_{B|\Lambda}) \leq 5\Delta(\rho_{XB},\, \tilde{\rho}_{XB}) + \Pr_{\tilde{\rho}}[\Lambda]\cdot\Delta(\tilde{\rho}_{XB|\Lambda},\, \tau_X \otimes \tilde{\rho}_{B|\Lambda})\,.$$

Proof. We set $\delta := \Delta(\rho_{XB}, \tilde{\rho}_{XB})$. From $\Delta(\rho_{XB}, \tilde{\rho}_{XB}) = \delta$ it follows in particular that the two distributions P_X and $\tilde{P}_X$ are δ-close, and thus that the state

$$\sigma_{XB} := \Pr_{\rho}[\Lambda]\cdot\tilde{\rho}_{XB|\Lambda} + \Pr_{\rho}[\neg\Lambda]\cdot\tilde{\rho}_{XB|\neg\Lambda}$$

is δ-close to $\tilde{\rho}_{XB}$, and hence 2δ-close to ρ_{XB}, where $\neg\Lambda$ is the negation of the event Λ. Since Λ is determined by X, we can write

$$\Delta(\rho_{XB}, \sigma_{XB}) = \Pr_{\rho}[\Lambda]\cdot\Delta(\rho_{XB|\Lambda},\, \tilde{\rho}_{XB|\Lambda}) + \Pr_{\rho}[\neg\Lambda]\cdot\Delta(\rho_{XB|\neg\Lambda},\, \tilde{\rho}_{XB|\neg\Lambda})\,,$$

from which it follows that $\Pr_\rho[\Lambda]\cdot\Delta(\rho_{XB|\Lambda},\, \tilde{\rho}_{XB|\Lambda}) \leq 2\delta$, and, by tracing out X, also that $\Pr_\rho[\Lambda]\cdot\Delta(\rho_{B|\Lambda}, \tilde{\rho}_{B|\Lambda}) \leq 2\delta$. We can now conclude that

$$\begin{aligned} \Pr_{\rho}[\Lambda]\cdot\Delta(\rho_{XB|\Lambda},\, \tau_X \otimes \rho_{B|\Lambda}) &\leq 4\delta + \Pr_{\rho}[\Lambda]\cdot\Delta(\tilde{\rho}_{XB|\Lambda},\, \tau_X \otimes \tilde{\rho}_{B|\Lambda}) \\ &\leq 5\delta + \Pr_{\tilde{\rho}}[\Lambda]\cdot\Delta(\tilde{\rho}_{XB|\Lambda},\, \tau_X \otimes \tilde{\rho}_{B|\Lambda})\,, \end{aligned}$$

which proves the claim. □

References

1. Acín, A., Brunner, N., Gisin, N., Massar, S., Pironio, S., Scarani, V.: Device-Independent Security of Quantum Cryptography against Collective Attacks. Phys. Rev. Lett. 98(23) (2007)
2. Barrett, J., Hardy, L., Kent, A.: No Signaling and Quantum Key Distribution. Phys. Rev. Lett. 95(1) (June 2005)
3. Beigi, S., König, R.: Simplified Instantaneous Non-Local Quantum Computation with Applications to Position-Based Cryptography. New J. Phys. 13(9), 093036 (2011)
4. Bennett, C.H., Brassard, G., Mermin, N.: Quantum Cryptography Without Bell's Theorem. Phys. Rev. Lett. 68(5), 557–559 (1992)
5. Bennett, C.H., Brassard, G.: Quantum Cryptography: Public Key Distribution and Coin Tossing. In: Proc. IEEE Int. Conf. on Comp., Sys. and Signal Process., Bangalore, pp. 175–179. IEEE (1984)
6. Branciard, C., Cavalcanti, E.G., Walborn, S.P., Scarani, V., Wiseman, H.M.: One-sided device-independent quantum key distribution: Security, feasibility, and the connection with steering. Phys. Rev. A 85(1), 010301 (2012)
7. Braunstein, S., Pirandola, S.: Side-Channel-Free Quantum Key Distribution. Phys. Rev. Lett. 108(13), 130502 (2012)
8. Buhrman, H., Chandran, N., Fehr, S., Gelles, R., Goyal, V., Ostrovsky, R., Schaffner, C.: Position-Based Quantum Cryptography: Impossibility and Constructions. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 429–446. Springer, Heidelberg (2011)
9. Chandran, N., Goyal, V., Moriarty, R., Ostrovsky, R.: Position Based Cryptography. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 391–407. Springer, Heidelberg (2009)
10. Einstein, A., Podolsky, B., Rosen, N.: Can Quantum-Mechanical Description of Physical Reality Be Considered Complete? Phys. Rev. 47(10), 777–780 (1935)
11. Ekert, A.K.: Quantum cryptography based on Bell's theorem. Phys. Rev. Lett. 67(6), 661–663 (1991)
12. Gisin, N., Pironio, S., Sangouard, N.: Proposal for Implementing Device-Independent Quantum Key Distribution Based on a Heralded Qubit Amplifier. Phys. Rev. Lett. 105(7) (August 2010)
13. Hänggi, E., Renner, R.: Device-Independent Quantum Key Distribution with Commuting Measurements (September 2010), `http://arxiv.org/abs/1009.1833`
14. Hastings, M.: A Counterexample to Additivity of Minimum Output Entropy. Nature Physics 5, 255 (2009)
15. Heisenberg, W.: Über den anschaulichen Inhalt der quantentheoretischen Kinematik und Mechanik. Z. Phys. 43(3-4), 172–198 (1927)
16. Kempe, J., Vidick, T.: Parallel Repetition of Entangled Games. In: 43rd STOC, pp. 353–362. ACM (2011)
17. Kent, A., Munro, W.J., Spiller, T.P.: Quantum Tagging: Authenticating Location via Quantum Information and Relativistic Signalling Constraints (August 2010), `http://arxiv.org/abs/1008.2147`
18. Kittaneh, F.: Norm Inequalities for Certain Operator Sums. Journal of Functional Analysis 143(2), 337–348 (1997)
19. Klauck, H.: A Strong Direct Product Theorem for Disjointness. In: 42nd STOC, pp. 77–86. ACM (2010)

20. König, R., Renner, R., Schaffner, C.: The Operational Meaning of Min- and Max-Entropy. IEEE Trans. on Inf. Theory 55(9), 4337–4347 (2009)
21. Krishna, M., Parthasarathy, K.R.: An Entropic Uncertainty Principle for Quantum Measurements. Indian J. Stat. 64(3), 842–851 (2002)
22. Lau, H.-K., Lo, H.-K.: Insecurity of Position-based Quantum-Cryptography Protocols Against Entanglement Attacks. Phys. Rev. A 83(1), 1–12 (2011)
23. Lim, C.C.W., Portmann, C., Tomamichel, M., Renner, R., Gisin, N.: Device-Independent Quantum Key Distribution with Local Bell Test (July 2012), http://arxiv.org/abs/1208.0023
24. Lo, H.-K., Curty, M., Qi, B.: Measurement-Device-Independent Quantum Key Distribution. Phys. Rev. Lett. 108(13), 130503 (2012)
25. Lydersen, L., Wiechers, C., Wittmann, C., Elser, D., Skaar, J., Makarov, V.: Hacking Commercial Quantum Cryptography Systems by Tailored Bright Illumination. Nat. Photon. 4(10), 686–689 (2010)
26. Masanes, L., Pironio, S., Acín, A.: Secure Device-independent Quantum Key Distribution With Causally Independent Measurement Devices. Nat. Commun. 2, 238 (2011)
27. Mayers, D.: Quantum Key Distribution and String Oblivious Transfer in Noisy Channels. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 343–357. Springer, Heidelberg (1996)
28. Mayers, D., Yao, A.: Quantum Cryptography with Imperfect Apparatus. In: 39th FOCS, pp. 503–509. IEEE Computer Society (1998)
29. Raz, R.: A Parallel Repetition Theorem. SIAM Journal on Computing 27, 763–803 (1998)
30. Reichardt, B.W., Unger, F., Vazirani, U.: Classical Command of Quantum Systems via Rigidity of CHSH Games (September 2012), http://arxiv.org/abs/1209.0449
31. Renner, R.: Security of Quantum Key Distribution. PhD thesis, ETH Zurich (December 2005), http://arxiv.org/abs/quant-ph/0512258
32. Schaffner, C.: Cryptography in the Bounded-Quantum-Storage Model. PhD thesis, University of Aarhus (September 2007), http://arxiv.org/abs/0709.0289
33. Shor, P., Preskill, J.: Simple Proof of Security of the BB84 Quantum Key Distribution Protocol. Phys. Rev. Lett. 85(2), 441–444 (2000)
34. Terhal, B.: Is Entanglement Monogamous? IBM J. Reasearch and Development 48(1), 71–78 (2004)
35. Tomamichel, M.: A Framework for Non-Asymptotic Quantum Information Theory. PhD thesis, ETH Zurich (March 2012), http://arxiv.org/abs/1203.2142
36. Tomamichel, M., Fehr, S., Kaniewski, J., Wehner, S.: Strong Parallel Repetition for a Monogamy-of-Entanglement Game (October 2012), http://arxiv.org/abs/1210.4359
37. Tomamichel, M., Hayashi, M.: A Hierarchy of Information Quantities for Finite Block Length Analysis of Quantum Tasks (August 2012), http://arxiv.org/abs/1208.1478
38. Tomamichel, M., Renner, R.: Uncertainty Relation for Smooth Entropies. Phys. Rev. Lett. 106(11) (March 2011)

Quadratic Span Programs and Succinct NIZKs without PCPs

Rosario Gennaro[1,*], Craig Gentry[2],
Bryan Parno[3], and Mariana Raykova[4,**]

[1] City University of New York
rosario@ccny.cuny.edu
[2] IBM T.J.Watson Research Center
cbgentry@us.ibm.com
[3] Microsoft Research
parno@microsoft.com
[4] IBM T.J.Watson Research Center
mariana@cs.columbia.edu

Abstract. We introduce a new characterization of the NP complexity class, called *Quadratic Span Programs* (QSPs), which is a natural extension of *span programs* defined by Karchmer and Wigderson. Our main motivation is the quick construction of succinct, easily verified arguments for NP statements.

To achieve this goal, QSPs use a new approach to the well-known technique of *arithmetization* of Boolean circuits. Our new approach yields dramatic performance improvements. Using QSPs, we construct a NIZK argument – in the CRS model – for Circuit-SAT consisting of just *7 group elements*. The CRS size and prover computation are *quasi-linear*, making our scheme seemingly quite practical, a result supported by our implementation. Indeed, our NIZK argument attains the shortest proof, most efficient prover, and most efficient verifier of any known technique. We also present a variant of QSPs, called *Quadratic Arithmetic Programs* (QAPs), that "naturally" compute *arithmetic* circuits over large fields, along with succinct NIZK constructions that use QAPs.

Finally, we show how QSPs and QAPs can be used to efficiently and publicly verify outsourced computations, where a client asks a server to compute $F(x)$ for a given function F and must verify the result provided by the server in considerably less time than it would take to compute F from scratch. The resulting schemes are the most efficient, general-purpose publicly verifiable computation schemes.

[*] The research of this author was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

[**] Supported by NSF Grant No.1017660.

1 Introduction

Arithmetization of Boolean computations is a well known technique: it maps a Boolean circuit to a set of polynomial (e.g., quadratic) equations over a field. The celebrated result IP=PSPACE [35, 41] used arithmetization as a crucial tool and set the stage for the PCP theorem [2–4, 20], which provided *a new characterization of NP* that revolutionized the notion of "proof" – in particular, it shows that NP statements have probabilistically checkable proofs (PCPs) that can be verified in time polylogarithmic in the size of a classical proof.

Cryptographers quickly seized on the potential applicability of PCPs to secure computation. Kilian [32] showed how to use PCPs to construct interactive *arguments* (i.e., computationally sound proof systems [14]) for NP that are *succinct* – i.e., polylogarithmic in their communication complexity. Micali [36] showed how to make these arguments *non-interactive* in the random oracle model. Recent work [8, 19, 26] (see also [17]) has improved Micali's construction by removing the random oracle, which is known to be uninstantiable [15], and replacing it with an "extractable collision-resistant hash function" (ECRH), whose security relies on the plausible, but *non-falsifiable* [37], assumption that for any algorithm that computes an image of the ECRH, there is an extractor (that watches the algorithm) that computes a pre-image.[1] These recent constructions have been called succinct non-interactive arguments (SNARGs) of *knowledge* (SNARKs), since, under the knowledge assumption, the SNARG permits "knowledge" extraction of the entire hash preimage – i.e., the entire PCP.

PCPs are not the only arithmetization technique for creating SNARKs. Groth shows how to arithmetize a Boolean circuit so that a proof of its satisfiability can be written using only a constant number of group elements [27] (after a single pre-processing stage to establish a common reference string (CRS) [9, 10]).

Our work provides a brand new form of arithmetization which we call *Quadratic Span Programs* (QSPs), since it is a generalization of the notion of Span Programs proposed by Karchmer and Wigderson [31]. We show that our new arithmetization technique yields far more efficient SNARKs than either PCP-based or Groth-like proofs. Using QSPs, we construct a NIZK argument in the CRS model for circuit SAT consisting of just *7 group elements*. The CRS size and prover computation are *quasi-linear* in the circuit size, making our scheme quite practical, to the point where we have implemented and evaluated it (see Section 5). A variant of our technique works directly on *arithmetic* circuits over large fields, obtaining *Quadratic Arithmetic Programs* (QAPs) and avoiding the complexity of a Boolean description of an arithmetic computation (see Section 4).

[1] We know that the security of succinct non-interactive arguments cannot be based on falsifiable assumptions via black box reductions [1, 23]; hence non-falsifiable "knowledge" assumptions seem unavoidable in this context.

1.1 Quadratic Span Programs

QSPs are a natural extension of span programs (SPs), a linear-algebraic model of computation introduced by Karchmer and Wigderson [31].[2] An SP of size m over a field F consists of a set $\mathcal{V} = \{v_0(x), v_1(x), \dots, v_m(x)\}$ of polynomials of degree $d-1$, a partition of the indices $\mathcal{I} = \{1, \dots, m\}$ into two sets $\mathcal{I}_{labeled}$ and $\mathcal{I}_{free}$, and a further partition of $\mathcal{I}_{labeled}$ as $\cup_{i\in[n], j\in\{0,1\}}\mathcal{I}_{ij}$ meant to represent n Boolean inputs. The SP is said to "compute" a function f if the following is true for all input assignments $u \in \{0,1\}^n$: the polynomial $v_0(x)$ can be expressed as a linear combination of the polynomials that "belong" to the input assignment u – namely, the set of polynomials $\mathcal{V}_u$ with indices in $\mathcal{I}_u = \mathcal{I}_{free} \cup_i \mathcal{I}_{i,u_i}$ – iff $f(u) = 1$.

Functions with polynomial size SPs are in NC^2, since linear algebra is in NC^2. Consequently, it is widely believed that SPs *cannot* efficiently compute all functions in P (or verify all NP relations).

We define QSPs somewhat similarly to SPs.

Definition 1 (Quadratic Span Program). *A quadratic span program (QSP) Q over field F contains two sets of polynomials $\mathcal{V} = \{v_k(x) : k \in \{0, \dots, m\}\}$ and $\mathcal{W} = \{w_k(x) : k \in \{0, \dots, m\}\}$ and a divisor polynomial $D(x)$, all from $F[x]$. Q also contains a partition of the indices $\mathcal{I} = \{1, \dots, m\}$ into two sets $\mathcal{I}_{labeled}$ and $\mathcal{I}_{free}$, and a further partition of $\mathcal{I}_{labeled}$ as $\cup_{i\in[n], j\in\{0,1\}}\mathcal{I}_{ij}$.*

For input $u \in \{0,1\}^n$, let $\mathcal{I}_u = \mathcal{I}_{free} \cup_i \mathcal{I}_{i,u_i}$ be the set of indices that "belong" to input u. Q accepts *an input $u \in \{0,1\}^n$ iff there exist tuples $(a_1, \dots, a_m)$ and $(b_1, \dots, b_m)$ from F^m, with $a_k = 0 = b_k$ for all $k \notin \mathcal{I}_u$:*

$$D(x) \quad \text{divides} \quad \Big(v_0(x) + \sum_{k=1}^{m} a_k \cdot v_k(x)\Big) \cdot \Big(w_0(x) + \sum_{k=1}^{m} b_k \cdot w_k(x)\Big) . \qquad (1)$$

Q "computes" a Boolean function $f : \{0,1\}^n \to \{0,1\}$ if it accepts exactly those inputs u where $f(u) = 1$. Q has size m and degree $\deg(D(x))$.

QSPs are a natural extension of (linear) SPs. An SP accepts an input u if and only if the target polynomial can be written as an affine linear combination of polynomials that "belong" to u. A QSP accepts an input u if and only if the divisor polynomial divides a *product* of two affine linear combinations of polynomials that "belong" to u, where "product" is polynomial multiplication.

Unlike SPs, QSPs can efficiently compute any function in P, and the "canonical QSP" we build has performance parameters that yield faster SNARKs, as stated in the two theorems below.

Theorem 1. *(Informal) For any Boolean circuit C with s gates and any field F of size at least $d = O(s)$, there is a QSP of size and degree $O(s)$ (with small constants) over F that computes C.*

[2] SPs were first defined [31] in terms of vectors $\{\mathbf{v}_0, \mathbf{v}_1, \dots, \mathbf{v}_m\}$, rather than polynomials. The "target" vector $\mathbf{v}_0$ must be expressible as a linear combination of the vectors that "belong" to the input assignment u (as defined above). Our definition in terms of polynomials is equivalent [22]: just think of each vector as the evaluation of the corresponding polynomial on a fixed set of points.

Theorem 2. *(Informal) Given a circuit C with s gates, computing the polynomials $D(x)$, $\mathcal{V}$ and $\mathcal{W}$ of our "canonical" QSP, which computes C, takes $O(s)$ work ($O(s)$ F operations). Given $u \in \{0,1\}^n$ for which $C(u) = 1$, computing suitable tuples $(a_1, \ldots, a_m)$, $(b_1, \ldots, b_m) \in \{0,1\}^m$ that satisfy Equation 1 takes $O(s)$ work. Given $(a_1, \ldots, a_m)$, computing $v(x) = v_0(x) + \sum_{k=1}^{m} a_k \cdot v_k(x)$ takes $O(s)$ work. (Similarly for $w(x)$.) Computing the quotient $h(x) = v(x) \cdot w(x)/D(x)$ takes $\tilde{O}(s)$ work.*

We obtain such performance by exploiting the sparseness of the polynomials $v_k(x)$'s and $w_k(x)$'s in our canonical QSP. In particular, they behave similarly to Lagrange basis polynomials $\ell_j(x) = \prod_{i \neq j}(x - r_i)/(r_j - r_i)$ in that they each evaluate to 0 at almost all roots of $D(x)$, which is a product of linear terms. This makes it easy to compute $v(x)$ and $w(x)$ in linear time by representing them by their evaluation at these roots. Computing $h(x)$ in purely linear, versus quasi-linear, time remains an open problem.

1.2 From QSPs to SNARKs, NIZKs, and Verifiable Computing

We use QSPs to build SNARKs and NIZKs in the CRS model [9, 10].

SNARKs. Our SNARK for f uses a CRS in which the QSP polynomials (e.g., $\{v_k(x)\}$) are represented by terms $g^{v_k(\sigma)}$ (etc.), where g is a generator of a bilinear group [12], and $\sigma \in F$ is secret. The CRS size is linear in the circuit size of f. To oversimplify, to compute a SNARK, the prover uses its satisfying input to compute tuples $(a_1, \ldots, a_m)$ and $(b_1, \ldots, b_m)$, and then uses them and the CRS to compute $g_v = g^{v(\sigma)}$, $g_w = g^{w(\sigma)}$, $g_h = g^{h(\sigma)}$ for $v(x)$, $w(x)$, $h(x)$ as defined in Theorem 2. The verifier confirms that $e(g_v, g_w) = e(g_h, g^{t(\sigma)})$, where e is the bilinear map. (The actual scheme is more complicated – see Section 3.2.) For security, we require a non-falsifiable "knowledge" assumption which, as noted above, is necessary [1, 23].

NIZK. It is straightforward to randomize our public-verifier SNARK to make it statistical zero knowledge and obtain a non-interactive zero-knowledge (NIZK) argument [9, 10]. Details are in Section 3.3.

Verifiable Computation. In the full version [22], we use our QSP-based SNARK to achieve a very efficient scheme for public verifiable computation [21, 39].

Remark on Efficiency and Adaptivity. In the description above, the CRS (the QSP polynomials) depend on a particular language or relation. We can achieve an "adaptive" solution (where first the CRS is fixed, and then the language or relation is selected) by applying our QSP construction to the universal circuit, at the cost of expanding the circuit by a logarithmic factor, yielding quasi-linear complexity for CRS size and prover computation.

1.3 Comparisons to Other Work on Succinct Arguments

PCP-based Protocols. Ishai, Kushilevitz and Ostrovsky [30] were perhaps the first to seriously investigate how to tweak PCP-techniques to yield the best possible succinct arguments. However in their solution the prover's computation (and also the verifier's computation in a pre-processing step) is *quadratic* in the size of the classical proof.

Very recently, Ben-Sasson et al. present a new PCP scheme with quasilinear complexity for the prover and the CRS [7]. Our direct construction of QSPs yields better asymptotic performance, even before these PCPs are converted into SNARKs.

Groth-like Schemes. Groth et al. [28, 29] previously constructed NIZKs over bilinear groups with various attractive properties, but with size linear in the circuit. More recently, Groth essentially found a way to compress the proof into a constant number of group elements [27] (still higher than ours – 42 group elements versus 7 for ours). Security relies on a non-falsifiable "knowledge of exponent" assumption, similar to the one we use.

The main drawbacks of Groth's succinct NIZK are the prover complexity and the CRS size, which are both *quadratic* in the circuit size. Lipmaa [33] showed how to reduce the size of the CRS in Groth's construction from quadratic to quasi-linear in the circuit size, but prover complexity remains quadratic.

2 Quadratic Span Programs (QSPs)

Above, we defined Quadratic Span Programs (QSPs) in a manner that is superficially similar to that of span programs (SPs). The crucial difference is that QSPs can compute any efficiently computable function. We demonstrate this via an explicit construction of a QSP for any circuit C.[3] The construction uses two components: a *gate checker* and a *wire checker*.

2.1 A Gate Checker

While we do not know how to efficiently construct SPs for arbitrary functions $f \in$ P, we *can* always efficiently construct an SP for a function *related* to f, called the *gate checker function* for f, which ensures that a set of wire values is consistent with the gates in a circuit for f.

Definition 2 (Gate Checker Function). *Let $f : \{0,1\}^n \to \{0,1\}$ be a function whose Boolean circuit C has s gates. Let $N = n + s$ – the total number of wires in C (wires that fan out are considered one wire). Define $\phi : \{0,1\}^N \to \{0,1\}$ to be a function that outputs '1' iff the input is a valid assignment of C's wires with output wire set to '1'. We say that ϕ is the* gate checker function *for f.*

[3] The full version of the paper [22] gives a formal reduction from circuit SAT to a QSP satisfiability problem, hence proving that QSP SAT is NP complete.

An SP for the gate checker function ϕ does *not*, however, compute the function f; such an SP has labeled (non-free) polynomials even for the interior wires of C, whereas an SP for f is only permitted to have labeled polynomials for C's input wires. If we simply move the polynomials for the interior wires to the "free" set, then we might introduce additional valid linear combinations that do not satisfy C; in particular these linear combinations could use polynomials that correspond to conflicting assignments (both '0' and '1') for some interior wire in C.

What we prove however, is that these conflicting assignments are the only possible problem we introduce by moving the polynomials for the interior wires to the "free" set. In other words, if we restrict the linear combination to use polynomials associated with *at most* one value per wire, then the SP for ϕ can also be used to compute the function f. The following lemma formalizes the property.

Lemma 1. *Let $S = (\{v_0(x), \ldots, v_m(x)\}, \mathcal{I}_{free}, \mathcal{I}_{labeled} = \cup_{i \in [N], j \in \{0,1\}} \mathcal{I}_{ij})$ be an SP that computes the gate checker function ϕ of f. Then, for all $u \in \{0,1\}^n$, the following is true iff $f(u) = 1$: there exists a tuple $(a_1, \ldots, a_m)$ satisfying the following constraints:*

- ***Target in Span:*** *$v_0(x) = \sum_k a_k \cdot v_k(x)$.*
- ***Correct Inputs:*** *For all $k \in \cup_{i=1}^n I_{i\overline{u}_i}$, we have $a_k = 0$.*
- ***No Double Assignments:*** *For all $i \in \{n+1, \ldots, N\}$ and all $k_1 \in I_{i0}$ and $k_2 \in I_{i1}$, at most one of a_{k_1}, a_{k_2} is nonzero.*

In particular, if $f(u) \neq 1$, then a linear combination that satisfies the first and second constraints must violate the third – i.e., must make a "double assignment" of some wire $i \in \{n+1, \ldots, N\}$.

Proof. (Lemma 1) If $f(u) = 1$, then we can assign the wires of C validly with the output wire set to 1. Therefore, we can extend $u \in \{0,1\}^n$ to an input $u' \in \{0,1\}^N$ that satisfies ϕ. Since u' satisfies ϕ, there is a linear combination $(a_1, \ldots, a_m)$ such that $v_0(x) = \sum_k a_k \cdot v_k(x)$ and $a_k = 0$ for all $k \in \cup_{i=1}^n I_{i\overline{u'}_i}$, thus satisfying the constraints listed in the lemma.

Conversely, suppose that $(a_1, \ldots, a_m)$ satisfies the constraints. Then, since S computes ϕ, there is an extension $u' \in \{0,1\}^N$ of $u \in \{0,1\}^n$ such that $\phi(u') = 1$ and such that u' "agrees" with the tuple $(a_1, \ldots, a_m)$ in the sense that $a_k = 0$ for all $k \in I_{i\overline{u'}_i}$, $i \in [N]$. Since $\phi(u') = 1$ where u' is an extension of u, and since ϕ tests the satisfaction of f's Boolean circuit, we must have $f(u) = 1$. ∎

Looking ahead, our construction will use the span program for ϕ to obtain efficient proofs about the correct evaluation of f. The second component of our construction, the wire checker, will efficiently verify that the **No Double Assignments** property holds.

2.2 A Wire Checker

To prevent double wire assignments, we introduce some additional polynomials in the form of a *wire checker*, defined as follows.[4]

[4] While Definition 3 resembles a QSP, a wire checker is *not*, on its own, a QSP.

Definition 3 (Aggregate Wire Checker). *Let $\mathcal{I} = \cup_{i\in[N],j\in\{0,1\}}\mathcal{I}_{ij}$ be a partition of $[m]$. An aggregate wire checker for $\mathcal{I}$ consists of polynomials $D(x)$, $\mathcal{V} = \{v_k(x) : k \in \mathcal{I}\}$ and $\mathcal{W} = \{w_k(x) : k \in \mathcal{I}\}$ such that*

$$D(x) \quad \text{divides} \quad \Big(\sum_{k\in\mathcal{I}} a_k \cdot v_k(x)\Big) \cdot \Big(\sum_{k\in\mathcal{I}} b_k \cdot w_k(x)\Big) \tag{2}$$

if $\{a_k\}$ and $\{b_k\}$ indicate consistent bit assignments of all N bits (i.e., for each $i \in [N]$, for some bit B_i, $a_k = b_k = 0$ for all $k \in \mathcal{I}_{i\bar{B}_i}$), but not if $\{a_k\}$ and $\{b_k\}$ indicate inconsistent bit assignments of any of the N bits in the following sense: For some $i \in [N]$,

- *There exist $k_a \in \mathcal{I}_{i0}$ and $k'_a \in \mathcal{I}_{i1}$ and $k_b \in \mathcal{I}_{i0} \cup I_{i1}$ such that $a_{k_a} \neq 0$, $a_{k'_a} \neq 0$ and $b_{k_b} \neq 0$, or*
- *There exist $k_a \in \mathcal{I}_{i0} \cup I_{i1}$ and $k_b \in \mathcal{I}_{i0}$ and $k'_b \in \mathcal{I}_{i1}$ such that $a_{k_a} \neq 0$, $b_{k_b} \neq 0$ and $b_{k'_b} \neq 0$.*

The size of the wire checker is $|\mathcal{I}|$, and the degree is $\deg(D(x))$.

To construct an aggregate wire checker, we first construct a checker for a single wire. Let $\mathcal{I}_0 = \{1, \ldots, L_0\}$, $\mathcal{I}_1 = \{L_0 + 1, \ldots, L_0 + L_1\}$, and $\mathcal{I} = \mathcal{I}_0 \cup \mathcal{I}_1$ be the indices associated with the wire.

<u>Construction of a Wire Checker.</u>

1. Let $L_{max} = \max(L_0, L_1)$. For $L' = 3L_{max} - 2$, select distinct roots $\mathcal{R}^{(0)} = \{r_1^{(0)}, \ldots, r_{L'}^{(0)}\}$ and $\mathcal{R}^{(1)} = \{r_1^{(1)}, \ldots, r_{L'}^{(1)}\}$ from F. Set $\mathcal{R} = \mathcal{R}^{(0)} \cup \mathcal{R}^{(1)}$. Set $D(x) = \prod_{r\in\mathcal{R}}(x - r)$.
2. Interpolate the polynomials in $\{v_k(x)\}$ and $\{w_k(x)\}$ to have degree $(L' + L_0 - 1)$ if $k \in \mathcal{I}_0$ and $(L' + L_1 - 1)$ if $k \in \mathcal{I}_1$, and to satisfy:
   (a) For $k \in \mathcal{I}_0$, $v_k(r) = 0$ for all $r \in \mathcal{R}^{(0)} \cup \{r_1^{(1)}, \ldots, r_{L_0}^{(1)}\}$ except $v_k(r_k^{(1)}) = 1$, and $w_k(r) = 0$ for all $r \in \mathcal{R}^{(1)} \cup \{r_1^{(0)}, \ldots, r_{L_0}^{(0)}\}$ except $w_k(r_k^{(0)}) = 1$.
   (b) For $k \in \mathcal{I}_1$, $v_k(r) = 0$ for all $r \in \mathcal{R}^{(1)} \cup \{r_1^{(0)}, \ldots, r_{L_1}^{(0)}\}$ except $v_k(r_{k-L_0}^{(0)}) = 1$, and $w_k(r) = 0$ for all $r \in \mathcal{R}^{(0)} \cup \{r_1^{(1)}, \ldots, r_{L_1}^{(1)}\}$ except $w_k(r_{k-L_0}^{(1)}) = 1$.

Lemma 2. *The construction above is a wire checker.*

Proof. (Lemma 2) Clearly, $D(x)$ divides the product in Equation 2 – i.e., $(\sum_{k\in\mathcal{I}} a_k \cdot v_k(r)) \cdot (\sum_{k\in\mathcal{I}} b_k \cdot w_k(r)) = 0$ for all $r \in \mathcal{R}$ – if $\{a_k\}$, $\{b_k\}$ indicate consistent assignments.

If $\{a_k\}$ indicates a double assignment and $\{b_k\}$ is nonzero, then $\sum_{k\in\mathcal{I}_0} a_k \cdot v_k(x)$ has at most $L_0 - 1$ roots in $\mathcal{R}^{(1)}$, since it is nonzero of degree $L' + L_0 - 1$ and already has $\mathcal{R}^{(0)}$ as roots. A similar analysis shows that $\sum_{k\in\mathcal{I}_1} a_k \cdot v_k(x)$ has at most $L_1 - 1$ roots in $\mathcal{R}^{(0)}$. Note that $\sum_{k\in\mathcal{I}} a_k \cdot v_k(x)$ has exactly the same roots in $\mathcal{R}^{(1)}$ that $\sum_{k\in\mathcal{I}_0} a_k \cdot v_k(x)$ does, since the other part of the sum – namely, $\sum_{k\in\mathcal{I}_1} a_k \cdot v_k(x)$ – has everything in $\mathcal{R}^{(1)}$ as a root. Similarly, $\sum_{k\in\mathcal{I}} a_k \cdot v_k(x)$ has exactly the same roots in $\mathcal{R}^{(0)}$ that $\sum_{k\in\mathcal{I}_1} a_k \cdot v_k(x)$ does. So, $\sum_{k\in\mathcal{I}} a_k \cdot v_k(x)$ has at most $L_0 + L_1 - 2 \leq 2L_{max} - 2$ roots in $\mathcal{R}$. Since $\sum_{k\in\mathcal{I}} b_k \cdot w_k(x)$ is nonzero and degree-$(L' + L_{max} - 1)$, it has at most $L' + L_{max} - 1$ roots in $\mathcal{R}$. So, the overall product has at most $L' + 3L_{max} - 3 < 2L'$ roots, and is therefore not divisible by $D(x)$. ■

Using the Chinese Remainder Theorem, we compose the wire checkers for individual wires into an aggregate wire checker for the whole circuit.

Construction of an Aggregate Wire Checker.

1. *Generate all of the roots and the divisor polynomial.* For each wire $i \in [N]$, select distinct roots for $\mathcal{R}^{(i0)}$ and $\mathcal{R}^{(i1)}$ from F as in the single-wire checker. Note that the roots are distinct across the i's as well. Set $\mathcal{R} = \cup_i \mathcal{R}^{(i0)} \cup \mathcal{R}^{(i1)}$. Set the aggregate wire checker's divisor polynomial to $D(x) = \prod_i D_i(x) = \prod_{r \in \mathcal{R}} (x - r)$.
2. *Generate polynomials for the individual wire checkers.* For each wire $i \in [N]$, construct the sets of polynomials $\mathcal{V}^{(i)}$ and $\mathcal{W}^{(i)}$ as in the single-wire checker.
3. *Compose individual wire checkers via CRT.* For $i \in [N]$, for $k \in \mathcal{I}_{i0} \cup \mathcal{I}_{i1}$, interpolate $v_k(x)$ to be of degree at most $\deg(D(x)) - 1$ and satisfy $v_k(x) = v_k^{(i)}(x) \bmod D_i(x)$ and $v_k(x) = 0 \bmod D(x)/D_i(x)$. Analogously for $w_k(x)$. Set $\mathcal{V} = \{v_k(x)\}$ and $\mathcal{W} = \{w_k(x)\}$.

Lemma 3. *The above construction is an aggregate wire checker.*

Proof. (Lemma 3) If $\{a_k\}$, $\{b_k\}$ indicate consistent assignments, then they are consistent on the i-th bit for k restricted to $\mathcal{I}_{i0} \cup \mathcal{I}_{i1}$. Hence, $D_i(x)$ divides the product in Eqn.2 when the summations are restricted to $k \in \mathcal{I}_{i0} \cup \mathcal{I}_{i1}$. Since $v_k(x)$ and $w_k(x)$ are divisible by $D_i(x)$ for all $k \notin \mathcal{I}_{i0} \cup \mathcal{I}_{i1}$, the overall (unrestricted) product in Eqn.2 is divisible by $D_i(x)$. Since this holds for all i, the product is divisible by $D(x)$.

If, for some i, $\{a_k\}$ indicates a double assignment of the i-th bit and $\{b_k\}$ is nonzero over $k \in \mathcal{I}_{i0} \cup \mathcal{I}_{i1}$, then, by Lemma 2, $D_i(x)$ does not divide the product in Eqn.2 when the summations are restricted to $k \in \mathcal{I}_{i0} \cup \mathcal{I}_{i1}$. As above, $D_i(x)$ divides everything else, and thus the overall product in Eqn.2 is not divisible by $D_i(x)$, and thus not divisible by $D(x)$. ■

2.3 Conscientious Span Programs

Notice that the aggregate wire checker definition above enforces a slightly weaker condition than forbidding double assignments: it states that double assigning a wire with the $\{a_k\}$ (i.e., using non-zero a_k values from both $\mathcal{I}_{i0}$ *and* $\mathcal{I}_{i1}$) is forbidden, unless the $\{b_k\}$ indicate a *non*-assignment of that wire – i.e., all the corresponding $b_k = 0$ (and vice versa for a double assignment in the $\{b_k\}$).

To compensate for the weakness of the wire checker, we require the SP being checked to be *conscientious*, which guarantees that every satisfying linear combination uses *at least* one polynomial from the sets associated with its input. In our canonical QSP, we will use the wire checker above on *two* instances of a conscientious SP for ϕ. Conscientiousness guarantees that each instance includes a non-zero coefficient for each wire used in the satisfying assignment, and hence the wire checker will always catch double assignments in either instance.

More formally, we define a conscientious SP as follows:

Definition 4. *Let $S = (\{v_0(x), \ldots, v_m(x)\}, \mathcal{I}_{free}, \mathcal{I}_{labeled} = \cup_{i\in[n], j\in\{0,1\}} \mathcal{I}_{ij})$ be an SP. We say that S is a* conscientious *SP for $f : \{0,1\}^n \to \{0,1\}$ if, for any tuple $(a_1, \ldots, a_m)$ that satisfies the usual SP requirements that $f(u) = 1$ for $u \in \{0,1\}^n$ iff (1) $v_0(x) = \sum_k a_k \cdot v_k(x)$ and (2) for all $k \in \cup_{i=1}^n I_{i\bar{u}_i}$ we have $a_k = 0$, we also have the property that for all $i \in [n]$, there exists $k \in I_{iu_i}$ such that $a_k \neq 0$. Let m be the size of the SP and $\deg(v_0(x)) + 1$ be the degree of the SP.*

To construct a conscientious SP for ϕ, we first build a conscientious SP for a single NAND gate.

Lemma 4. *There is a degree-9 conscientious SP for NAND of size 12.*

Proof. (Lemma 4) Choose a set of 9 distinct roots in F to get $\mathcal{R} = (r_0, r_{l0}, r'_{l0}, r_{l1}, r'_{l1}, r_{r0}, r'_{r0}, r_{r1}, r'_{r1})$. Define 9 "linearly independent" polynomials $\{v_0(x), v_{l0}(x), v'_{l0}(x), v_{l1}(x), v'_{l1}(x), v_{r0}(x), v'_{r0}(x), v_{r1}(x), v'_{r1}(x)\}$ to be the corresponding Lagrange basis polynomials for $\mathcal{R}$; that is, they are the degree-8 polynomials obtained by interpolating such that $\forall r \in \mathcal{R}$, $v_0(r) = 0$, except that $v_0(r_0) = 1$; $v_{l0}(r) = 0$, except that $v_{l0}(r_{l0}) = 1$, and so on. We will use the convention that the pair of polynomials $\mathcal{V}_{l0} = (v_{l0}(x), v'_{l0}(x))$ belongs to the assignment of 0 to the left wire, etc.

Set $v_{o0}(x) = v_0(x) - v_{l1}(x) - v_{r1}(x)$ and $\mathcal{V}_{o0} = \{v_{o0}(x)\}$, so that one can express $v_0(x)$ as a linear combination of polynomials in $\mathcal{V}_{l1} \cup \mathcal{V}_{r1} \cup \mathcal{V}_{o0}$.

Set $v_{o1}(x) = v_0(x) - v_{l0}(x) - v_{r0}(x)$, $v'_{o1}(x) = v_0(x) - v'_{l0}(x) - v'_{r1}(x)$, and $v''_{o1}(x) = v_0(x) - v'_{l1}(x) - v'_{r0}(x)$, and $\mathcal{V}_{o1} = \{v_{o1}(x), v'_{o1}(x), v''_{o1}(x)\}$, so that one can express $v_0(x)$ as a linear combination of polynomials associated to the other satisfying gate assignments.

That the above polynomials define a conscientious SP for NAND of the claimed size and degree follows by inspection. The details are elaborated in the full version. ■

To obtain a conscientious SP for an entire circuit, we build a conscientious SP for each gate, using a distinct set of roots $\mathcal{R}_i$ for each SP, and then compose the gate SPs together using the Chinese Remainder Theorem, just as we did when building the aggregate wire checker.

Lemma 5. *Suppose a circuit C consists of s Boolean gates from some set Γ – e.g, $\Gamma = \{\text{NAND}\}$. Suppose that, for each gate $g \in \Gamma$, there is a conscientious SP of size m' and degree d' that computes whether its input is a satisfying assignment of g's input/output wires. Then there is a conscientious SP S of size $m = s \cdot m'$ and degree $d = s \cdot d'$ that computes the gate checker function ϕ for C. S is a straightforward composition of SPs $\{S_g\}$ for the individual gates g of C.*

Intuition. The proof is constructive. For each gate g, build an SP $S^{(g)}$ following Lemma 4, obtaining from each a unique set of roots $\mathcal{R}^{(g)}$ and polynomials $\{v_0^{(g)}(x)\} \cup \mathcal{V}^{(g)}$. Let $\mathcal{R} = \cup_g \mathcal{R}^{(g)}$. Let $v_0(x)$ be a polynomial such that

$v_0(r) = v_0^{(g)}(r)$ for all $r \in \mathcal{R}^{(g)}$ and all gates g in the circuit. For each gate g, extend g's polynomials such that for all $v(x) \in \mathcal{V}^{(g)}$, and $r \in \mathcal{R}/\mathcal{R}^{(g)}$, $v(r) = 0$. The aggregate SP's set of polynomials $\mathcal{V}$ will consist of $v_0(x)$ along with all of the extended polynomials. Since the roots used in each SP are unique across all SPs, this composition preserves all of the local linearity relationships created by Lemma 4; it also does not introduce any new relationships, since the unique roots prevent "interactions" across the gate SPs. See the full version of the paper [22] for the full proof.

2.4 The Canonical Quadratic Span Program

We now describe how to take any polynomial-time computable function f, and construct a polynomial-size QSP that computes f. The construction uses the Chinese Remainder Theorem (CRT) to merge the two components above, the gate checker and the wire checker, so that the quadratic test (Eq. 1) checks both at once. The wire checker's guarantee of no double assignments relies on the fact that the SP for the gate checker is conscientious, and hence must use at least one polynomial for each wire to arrive at a satisfying linear combination. Thus, we can conclude that the wire values are consistent with the circuit's gates, and that no wire is set to both 0 and 1.

More specifically, we build two copies of the conscientious SP for the gate checker, ensuring that all of the roots used are distinct. One copy will become the $\mathcal{V}$ polynomials in the QSP, while the other copy will become the $\mathcal{W}$ polynomials. We then construct the polynomials for the aggregate wire checker described above, using a third set of distinct roots. Since all of the divisor polynomials from the different components have different roots, they are relatively prime. Hence, we can use the CRT to define the final QSP polynomials so that they match the value of the constituent polynomials from each component.

The Canonical QSP: $Q_{can,f}$.

1. Take as input the Boolean circuit C for $f : \{0,1\}^n \rightarrow \{0,1\}$, which has s gates.
2. Using disjoint sets of roots $\mathcal{R}^{(\mathcal{V})}$ and $\mathcal{R}^{(\mathcal{W})}$, construct two instances of the conscientious gate checker SP for C – namely, $S^{(\mathcal{V})} = (\hat{\mathcal{V}} = \{\hat{v}_0(x), \ldots, \hat{v}_m(x)\}$, $\mathcal{I}_{free}, \mathcal{I}_{labeled})$ and $S^{(\mathcal{W})} = (\hat{\mathcal{W}} = \{\hat{w}_0(x), \ldots, \hat{w}_m(x)\}, \mathcal{I}_{free}, \mathcal{I}_{labeled})$.
3. Define $\hat{D}^{(\mathcal{V})}(x) = \prod_{r \in \mathcal{R}^{(\mathcal{V})}}(x - r)$ and $\hat{D}^{(\mathcal{W})}(x) = \prod_{r \in \mathcal{R}^{(\mathcal{W})}}(x - r)$. Note that because we use distinct roots for each incarnation, the resulting divisor polynomials $\hat{D}^{(\mathcal{V})}(x)$ and $\hat{D}^{(\mathcal{W})}(x)$ are relatively prime.
4. Using disjoint sets of roots $\mathcal{R} = \{\mathcal{R}^{(i0)}, \mathcal{R}^{(i1)} : i \in [N]\}$ and the partition of $\mathcal{I}_{labeled}$, construct the aggregate wire checker from Lemma 3, which consists of the following polynomials: $D'(x) = \prod_{r \in \mathcal{R}}(x - r)$, $\mathcal{V}' = \{v_1'(x), \ldots, v_m'(x)\}$ and $\mathcal{W}' = \{w_1'(x), \ldots, w_m'(x)\}$.
5. Define $D(x) = \hat{D}^{(\mathcal{V})}(x) \cdot \hat{D}^{(\mathcal{W})}(x) \cdot D'(x)$.
6. Finally, define $\mathcal{V} = \{v_0(x), \ldots, v_m(x)\}$ and $\mathcal{W} = \{w_0(x), \ldots, w_m(x)\}$ using the CRT to interpolate $v_k(x)$ and $w_k(x)$ as follows:

$$v_k(x) = \begin{cases} \hat{v}_k(x) \bmod \hat{D}^{(\mathcal{V})}(x) \\ v'_k(x) \bmod D'(x) \\ \;1 \quad \bmod \hat{D}^{(\mathcal{W})}(x) \text{ if } k = 0 \\ \;0 \quad \bmod \hat{D}^{(\mathcal{W})}(x) \text{ if } k \neq 0 \end{cases} \quad w_k(x) = \begin{cases} \hat{w}_k(x) \bmod \hat{D}^{(\mathcal{W})}(x) \\ w'_k(x) \bmod D'(x) \\ \;1 \quad \bmod \hat{D}^{(\mathcal{V})}(x) \text{ if } k = 0 \\ \;0 \quad \bmod \hat{D}^{(\mathcal{V})}(x) \text{ if } k \neq 0 \end{cases}$$

7. Output $Q_{can,f} = (\mathcal{V}, \mathcal{W}, D(x), \mathcal{I}_{free}, \mathcal{I}_{labeled} = \cup_{i \in [n], j \in \{0,1\}} \mathcal{I}_{ij})$, where the labeled indices $\cup_{i \in [n+1,N]} \mathcal{I}_{ij}$ from the gate checker SP for C have been moved to $\mathcal{I}_{free}$.

The proof of the following theorem is in the full version [22].

Theorem 3. *For any Boolean circuit C with n inputs, s gates, and $N = n + s$ total wire values, the canonical QSP computes C.*

2.5 Performance and Technical Issues

By Lemmas 4 and 5, given a function f whose Boolean circuit has s (NAND) gates, we have a conscientious SP of size $12s$ and degree $9s$ for f's gate-checker function. However, for performance and technical reasons, we use a larger conscientious SP of size $36s$ and degree $27s$.

The first reason we use a larger SP is that we transform f's Boolean circuit to one with fan-out two (except that one "dummy" input, set to '1', may feed into multiple gates). The resulting circuit may be larger by a constant factor. We reduce fan-out to two before applying the SP composition lemma (Lemma 5) because we want the evaluation vectors $\{(v_k(r_1), \ldots, v_k(r_d)), (w_k(r_1), \ldots, w_k(r_d)) : k \in [m], r \in \mathcal{R}\}$ of our QSP to be *sparse* – i.e., to have only constant nonzero support. Sparseness allows us, for example, to compute $v(x) = v_0(x) + \sum a_k \cdot v_k(x)$ very quickly in evaluation representation, in time linear in the degree of the QSP.

The second reason is that we obtain a *strong QSP*.

Definition 5 (Strong QSP). *A QSP $Q = (\mathcal{V}, \mathcal{W}, t(x), \mathcal{I}_{free}, \mathcal{I}_{labeled} = \cup_{i \in [n], j \in \{0,1\}} \mathcal{I}_{ij})$ is a strong QSP if $|\mathcal{I}_{ij}| = 1$ for all $i \in [n], j \in \{0,1\}$ and the QSP divisibility requirement (Eq.1) holds only if $\{a_k\}$, $\{b_k\}$ are "unequivocally" bound to some input $u \in \{0,1\}^n$ – in particular, $a_k = 1 = b_k$ for all $\{k = \mathcal{I}_{iu_i}\}$ and $a_k = 0 = b_k$ for all $\{k = \mathcal{I}_{i\bar{u}_i}\}$.*

In a strong QSP, the labeled sets are singletons, and the QSP can be satisfied only by applying an unequivocal 0/1 linear combination to the labeled vectors. Ultimately, this property helps improve the performance of our cryptographic constructions for NIZKs and verifiable computation, since a verifier who knows part of the circuit input (e.g., the statement u portion of the input to a relation) will be able to "predict" the portion of the QSP linear combination that corresponds to u (and therefore this portion does not need to be "sent" by the prover).

When it is applied to the partition $\mathcal{I}_{labeled} = \cup_{i \in [N], j \in \{0,1\}} \mathcal{I}_{ij}$ of the SP for the gate checker function, the size of the aggregate wire checker is $|\mathcal{I}_{labeled}| \leq 24s$

and the degree is $76s$. (See full version [22] for details.) Since the QSP has two SPs and one aggregate wire checker, and since composing the SPs with the wire checker does not increase the size, the QSP has size $36s$ and degree $130s$.

3 Overview of Cryptographic Constructions and Security

We build SNARKs and NIZKs in the common reference string (CRS) model [9, 10] for relations $R(u, w)$ with n'-bit statements and $(n - n')$-bit witnesses. We apply our QSPs for n-bit inputs to the circuit computing R.

Groth's construction [27] specifically targets the circuit SAT relation; in particular, he takes u to be a circuit that can be chosen adaptively and uses $R(u, w) = u(w)$. The CRS size and prover computation grow quadratically with $|u|$. The verifier computation is $O(|u|)$, but it can be reduced to $O(1)$ in an amortized sense with u-dependent pre-processing. To compare directly with Groth, we can handle u being an adaptively-chosen circuit by constructing R from a universal circuit. In this case, the size of the circuit computing R may be larger than $|u|$ by a logarithmic factor, which correspondingly increases the CRS size and prover computation to $\tilde{O}(|u|)$. The verifier computation is $O(|u|)$, but it can be reduced to $O(1)$ in an amortized sense just as in Groth. If u, or any part of u, can be chosen non-adaptively, our scheme becomes more efficient.

We present our constructions with their proof intuition, deferring the formal proofs to the full version [22].

3.1 Definitions

First, we define a SNARK for a Prover P who holds a witness w which he can use to convince a Verifier V of a statement u.

Definition 6 (SNARK). *We say that $\Pi = (\mathsf{Gen}, \mathsf{P}, \mathsf{V})$ is a succinct non-interactive argument of knowledge (*SNARK*) with security parameter κ for an* NP *language L with a corresponding* NP *relation R with n'-bit statements and $(n - n')$-bit witnesses , if it satisfies the following properties:*
Perfect Completeness: *For all $\mathcal{A}$,*

$$\Pr\left[\begin{array}{c} \mathsf{V}(\mathsf{priv}, u, \pi) = 1 \\ \textit{if}\,(u, w) \in R \end{array} \;\middle|\; \begin{array}{c} (\mathsf{crs}, \mathsf{priv}) \leftarrow \mathsf{Gen}(1^\kappa) \\ (u, w) \leftarrow \mathcal{A}(\mathsf{crs}) \\ \pi \leftarrow \mathsf{P}(\mathsf{crs}, u, w) \end{array}\right] = 1,$$

where $\mathsf{P}(\mathsf{crs}, u, w)$ runs in time $\mathrm{poly}(\kappa, n)$*.*
Soundness: *For all efficient $\mathcal{A}$,*

$$\Pr\left[\begin{array}{c} \mathsf{V}(\mathsf{priv}, u, \pi) = 1 \\ u \notin L \end{array} \;\middle|\; \begin{array}{c} (\mathsf{crs}, \mathsf{priv}) \leftarrow \mathsf{Gen}(1^\kappa) \\ (u, \pi) \leftarrow \mathcal{A}(1^\kappa, \mathsf{crs}) \end{array}\right] = \mathtt{negl}(\kappa).$$

Succinctness: *The proof length is $|\pi| = \mathrm{poly}(\kappa)$.*

Extraction: *For any poly-size prover* P^**, there exists a poly-size extractor* $\mathcal{E}_{P^*}$*, such that for any auxiliary information* $z \in \{0,1\}^\kappa$*, the following holds*

$$\Pr\left[\begin{array}{c} \mathsf{V}(\mathsf{priv}, u, \pi) = 1 \\ (u,w) \notin \mathcal{R} \end{array} \middle| \begin{array}{c} (\mathsf{crs}, \mathsf{priv}) \leftarrow \mathsf{Gen}(1^\kappa) \\ (u, \pi) \leftarrow \mathsf{P}^*(\mathsf{crs}, z) \\ w \leftarrow \mathcal{E}_{P^*}(\mathsf{crs}, z) \end{array}\right] = \mathtt{negl}(\kappa).$$

We omit the standard definition of NIZKs. Note that, to build a NIZK, it suffices to build (as we do) a SNARK that is statistical zero-knowledge.

3.2 Our SNARK Construction

We can create a SNARK for an NP relation $R = \{(u,w)\}$ with n'-bit statements and $(n-n')$-bit witnesses by building a canonical QSP for the function f such that $f(u,w) = 1$ iff $(u,w) \in R$. At a high-level, the prover uses his inputs to evaluate the circuit for f, hence obtaining linear combinations for the QSP that satisfy Eq.1. He uses these combinations to compute $v(x) = v_0(x) + \sum a_k \cdot v_k(x)$ (and similarly for $w(x)$), and convinces the verifier that the QSP's quadratic property holds (Eq.1), which implies $f(u,w) = 1$, by calculating $h(x)$ such that $h(x) \cdot D(x) = v(x)w(x)$.

To protect against malicious provers, all of the calculations described above are performed over *encoded* values. Specifically, the CRS holds an encoding of the evaluation of each polynomial (e.g., the $\{v_k(x)\}$) at a secret point σ. The encoding permits homomorphic operations, which allow the prover to calculate $v(\sigma)$, $w(\sigma)$, and $h(\sigma)$ inside the encoding. The encoding also permits a quadratic equality check so that the verifier can check that Equation 1 holds.

An encoding scheme $\mathcal{E}$ has two algorithms (Setup, E), where Setup takes the security parameter and generates parameters for the scheme, and E (possibly randomized) produces an encoding for an element. Our preferred encoding is exponentiation within a bilinear group: $E(v_k(\sigma)) = g^{v_k(\sigma)}$, in which case, the quadratic equality check is performed via a pairing. One may also use an additively homomorphic encryption scheme, e.g., Paillier[5]: $E(v_k(\sigma)) = \mathbf{Enc}_{pk}(v_k(\sigma))$. In this case, the verifier needs a secret key sk to remove the encoding and perform the quadratic check, and hence the SNARK is designated-verifier.

As a final note, to ensure the prover uses circuit inputs matching u, the verifier calculates the portion of $v(\sigma)$ that corresponds to u independently, leaving the portion of $v(\sigma)$ that corresponds to the witness to the prover.

To base the security of our scheme on an existing knowledge of exponent assumption [27], we add terms to the CRS of the form $E(\alpha\sigma^i)$, $E(\alpha v_k(\sigma))$, $E(\alpha w_k(\sigma))$, $E(\beta_v v_k(\sigma))$, $E(\beta_w w_k(\sigma))$, and extend the proof with relations between these terms and those in the basic proof (see Section 3.4).

[5] Technically, our constructions apply only where the encoding space is a field, and the plaintext space of Paillier is a ring, not a field. However, it would be easy to extend our results to Paillier, using the fact that one is unlikely to encounter encodings of nontrivial zero divisors in $\mathbb{Z}_N$ unless one is able to factor N.

CRS generation Gen**:** On input security parameter κ, construct a common random string $CRS = (\mathsf{crs}_P, \mathsf{crs}_V)$. Let f be the function checking the relation $R(u, w)$ and let $Q_f = (\mathcal{V}, \mathcal{W}, D(x), \mathcal{I}_{free}, \mathcal{I}_{labeled} = \cup_{i\in[n], j\in\{0,1\}}\mathcal{I}_{ij}, \mathcal{I} = \mathcal{I}_{free} \cup \mathcal{I}_{labeled})$ be a QSP of size m and degree d for the functionality f[6]. Let $\mathcal{I}_{in} = \cup_{i=1}^{n'}\mathcal{I}_{ij}$ and $\mathcal{I}_{mid} = \mathcal{I} \setminus \mathcal{I}_{in}$. Generate public and private parameters (pk, sk) for the encoding scheme E. Generate uniformly at random $\alpha, \sigma, \beta_v, \beta_w, \gamma \leftarrow F^*$ and set the output:

$$\begin{aligned}\mathsf{crs}_P = \big(&pk, Q_f, n', \{E(\sigma^i)\}_{i\in[0,d]}, \{E(\alpha\sigma^i)\}_{i\in[0,d]},\\ &\{E(v_k(\sigma))\}_{k\in\mathcal{I}_{mid}}, \{E(w_k(\sigma))\}_{k\in\mathcal{I}},\\ &\{E(\alpha v_k(\sigma))\}_{k\in\mathcal{I}_{mid}}, \{E(\alpha w_k(\sigma))\}_{k\in\mathcal{I}},\\ &\{E(\beta_v v_k(\sigma))\}_{k\in\mathcal{I}_{mid}}, \{E(\beta_w w_k(\sigma))\}_{k\in\mathcal{I}}\big)\\ \mathsf{crs}_V = \big(&pk, sk, E(1), E(\alpha), E(\gamma), E(\beta_v\gamma), E(\beta_w\gamma),\\ &\{E(v_k(\sigma))\}_{k\in\{0\}\cup\mathcal{I}_{in}}, E(w_0(\sigma)), E(D(\sigma))\big).\end{aligned}$$

Prove P**:** On input crs_P, statement $u \in \{0,1\}^{n'}$ and witness w, P evaluates Q_f to obtain $(a_1, \ldots, a_m)$ and $(b_1, \ldots, b_m)$ and polynomial $h(x)$ such that

$$h(x) \cdot D(x) = \Big(v_0(x) + \sum_{k=1}^{m} a_k \cdot v_k(x)\Big) \cdot \Big(w_0(x) + \sum_{k=1}^{m} b_k \cdot w_k(x)\Big) .$$

Let $v_{mid}(x) = \sum_{k\in\mathcal{I}_{mid}} a_k \cdot v_k(x)$ and $w(x) = \sum_{k\in\mathcal{I}} b_k \cdot w_k(x)$. Then, P uses the encoding's homomorphism to output the following proof:

$$\begin{aligned}\pi = \Big(&E(v_{mid}(\sigma)), E(w(\sigma)), E(h(\sigma)),\\ &E(\alpha v_{mid}(\sigma)), E(\alpha w(\sigma)), E(\alpha h(\sigma)), E(\beta_v v_{mid}(\sigma) + \beta_w w(\sigma))\Big).\end{aligned}$$

Verify V**:** On input crs_V, u, and $\pi = (\pi_{v_{mid}}, \pi_w, \pi_h, \pi_{v'_{mid}}, \pi_{w'}, \pi_{h'}, \pi_y)$, V confirms that the terms are in the support of validly encoded elements. Let V_{mid}, W, H, V'_{mid}, W', H', and Y be what is encoded. V computes an encoding $E(v_{in}(\sigma))$ of $v_{in}(\sigma) = \sum_{k\in\mathcal{I}_{in}} a_k \cdot v_k(\sigma)$. V confirms that the following equations hold:

$$H \cdot D(\sigma) = (v_0(\sigma) + v_{in}(\sigma) + V_{mid}) \cdot (w_0(\sigma) + W),$$
$$V'_{mid} = \alpha V_{mid}, W' = \alpha W, H' = \alpha H, \gamma Y = (\beta_v\gamma)V_{mid} + (\beta_w\gamma)W.$$

3.3 Making the SNARK Statistical Zero-Knowledge (NIZKs)

In our NIZK construction, the prover simply *randomizes* each of the terms $v_0(\sigma) + v_{in}(\sigma) + V_{mid}$ and $w_0(\sigma) + W$ so that their product is still divisible by $D(\sigma)$, but the terms reveal nothing more about the original values. We achieve this by adding random multiples of $D(\sigma)$ to both terms, which preserves the divisibility property for their product. We supplement crs_P with additional terms

[6] For example, with circuit SAT, f is a universal circuit.

to facilitate computation of the remainder of the randomized proof. Specifically, we include: $E(D(\sigma))$, $E(\alpha D(\sigma))$, $E(\beta_v D(\sigma))$, $E(\beta_w D(\sigma))$, $E(v_0(\sigma))$, $E(\alpha v_0(\sigma))$, $E(w_0(\sigma))$ and $E(\alpha w_0(\sigma))$.

After generating a proof π as above, the prover randomizes it as follows. He picks random $\delta_{v_{mid}}, \delta_w \leftarrow F$ and outputs the following proof:

$$\pi' = \big(E(v'_{mid}(\sigma)), E(w'(\sigma)), E(h'(\sigma)),\\ E(\alpha v'_{mid}(\sigma)), E(\alpha w'(\sigma)), E(\alpha h'(\sigma)), E(\beta_v v'_{mid}(\sigma) + \beta_w w'(\sigma))\big),$$

where $v'_{mid}(x) = v_{mid}(x) + \delta_{v_{mid}} D(x)$, $w'(x) = w(x) + \delta_w D(x)$, $h'(x) = (v_0(x) + v_{in}(x) + v'_{mid}(x)) \cdot (w_0(x) + w'(x))/D(x)$, and $v_0(x)$, $v_{in}(x)$, $v_{mid}(x)$ and $w(x)$ are the values computed in the SNARK construction from the previous section. The encodings in the new proof π' can be computed efficiently from the encodings in π and the augmented crs_P.

3.4 Security

We base security on two assumptions, the q-power Diffie-Hellman (q-PDH) assumption and the q-power knowledge of exponent (q-PKE) assumption. When we instantiate our construction and the q-PDH and q-PKE assumptions with an encoding scheme $E(a) = g^a$ over a bilinear group, the q-PDH and q-PKE assumptions are virtually identical to those used by Groth in his NIZK construction [27].[7] Also, the bilinear group version of our q-PDH assumption is very similar to, but weaker than, assumptions that were used to construct hierarchical identity-based encryption and broadcast encryption schemes with short ciphertexts [11, 13].

The q-PDH assumption is a "conventional" falsifiable assumption, though still somewhat unusual in its dependence on q, which is related to the size of the circuits for the functions computed by our SNARKs.

Assumption 1 (q-PDH)**.** *Let κ be a security parameter, and $q = \mathrm{poly}(\kappa)$. The q-power Diffie-Hellman (q-PDH) assumption holds for encoding $\mathcal{E}$ if for all non-uniform probabilistic polynomial time adversaries $\mathcal{A}$ we have*

$$\Pr\left[\begin{array}{c} pk \leftarrow \mathcal{E}.\mathsf{Setup}(1^\kappa)\ ;\ \sigma \leftarrow F^*\ ; \\ \tau \leftarrow (pk, E(1), E(\sigma), \ldots, E(\sigma^q), E(\sigma^{q+2}), \ldots, E(\sigma^{2q}))\ ; \\ y \leftarrow \mathcal{A}(\tau)\ :\ y = E(\sigma^{q+1}) \end{array}\right] = \mathtt{negl}(\kappa).$$

The q-PKE assumption is a non-falsifiable "knowledge" assumption, similar in spirit to (but more complicated than) early knowledge-of-exponent assumptions (KEAs) [6, 18].

Assumption 2 (q-PKE)**.** *Let κ be a security parameter, and $q = \mathrm{poly}(\kappa)$. The q-power knowledge of exponent (q-PKE) assumption holds for encoding $\mathcal{E}$ if for*

[7] Our q-PDH assumption is actually weaker than his q-CPDH assumption, and our q-PKE assumption is identical to Groth's [27] and Lipmaa's [33], except that we extend the assumption to handle auxiliary inputs.

every non-uniform probabilistic polynomial time adversary $\mathcal{A}$, there exists a non-uniform probabilistic polynomial time extractor $\chi_{\mathcal{A}}$ such that

$$\Pr\left[\begin{array}{c} pk \leftarrow \mathcal{E}.\mathsf{Setup}(1^\kappa)\ ;\ \alpha, \sigma \leftarrow F^*; \\ \tau \leftarrow (pk, E(1), E(\sigma), \ldots, E(\sigma^q), E(\alpha), E(\alpha\sigma), \ldots, E(\alpha\sigma^q)); \\ (E(c), E(\hat{c}); a_0, \ldots, a_q) \leftarrow (\mathcal{A}||\chi_{\mathcal{A}})(\tau, z) : \\ \hat{c} = \alpha c \wedge c \neq \sum_{k=0}^{q} a_k \sigma^k \end{array}\right] = \mathtt{negl}(\kappa)$$

for any auxiliary information $z \in \{0,1\}^{\text{poly}(\kappa)}$ that is independent of α.

Next we state our main security theorem.

Theorem 4. *If the q-PDH and d-PKE assumptions hold for some $q \geq \max\{2d-1, d+2\}$, then the NIZK scheme defined in Section 3.3, instantiated with a QSP of degree d, is secure under Definition 6.*

Here, we provide some intuition, using a simpler version of our scheme, which has the following 6 element proof:

$$\pi = (E(v_{mid}(\sigma)), E(w(\sigma)), E(h(\sigma)), E(\alpha_v v_{mid}(\sigma)), E(\alpha_w w(\sigma)), E(\alpha_h h(\sigma)))\,.$$

For the version above, the intuition is that it is hard for the prover, who knows the CRS but not α_w, to output any pair $(E(W), E(W'))$ with $W' = \alpha_w W$ unless he *knows* a representation $\{b_k : k \in \mathcal{I}\}$ of W such that $W = \sum b_k w_k(\sigma)$. Knowledge of exponent assumptions (KEAs)[8] formalize this intuition: they say that for any algorithm that outputs a pair of encoded elements with ratio α_w, there is an extractor that "watches" the algorithm's computation and outputs the representation (the linear combination). In the security proof, extractors for the v, w and h terms extract out polynomials $v_{mid}(x)$, $w(x)$, $h(x)$ that are in the spans of $\{v_k(x) : k \in \mathcal{I}_{mid}\}$, $\{w_k(x) : k \in \mathcal{I}\}$, $\{x^i : i \in [d]\}$. If the proof verifies, then $(v_0(\sigma) + v(\sigma)) \cdot (w_0(\sigma) + w(\sigma)) = h(\sigma) \cdot D(\sigma)$ for $v(x) = v_{mid}(x) + \sum_{k \in \mathcal{I}_{in}} v_k(x)$. If indeed $(v_0(x) + v(x)) \cdot (w_0(x) + w(x)) = h(x) \cdot D(x)$ as polynomials, then the soundness of our QSP implies that we have extracted a *true* proof. Otherwise, $(v_0(x) + v(x)) \cdot (w_0(x) + w(x)) - h(x) \cdot D(x)$ is a nonzero polynomial having σ as a root, which allows the simulator to solve a hard problem.

We modified this simpler scheme to the more complicated SNARK construction in order to base security on assumptions slightly weaker than Groth's [27]. With these assumptions, we can only extract representations of the encoded terms with respect to the power basis $\{x^i\}$ (as in [27]), not with respect to $\{v_k(x) : k \in \mathcal{I}_{mid}\}$. Thus, this extraction does not guarantee that $v_{mid}(x)$ and $w(x)$ are in their proper spans. We ensure this via the final term $E(\beta_v v_{mid}(\sigma) + \beta_w w(\sigma))$, from which the simulator can solve a hard problem if $v_{mid}(x)$ or $w(x)$ lies outside its proper span.

3.5 Efficiency

Next we state the complexities for our SNARK construction and refer the reader to the full version of the paper [22] for the proofs.

[8] KEAs [6, 18, 24] exist for Paillier/RSA [19, 24], bilinear groups [27, 33], and even lattices [34].

Prover's Work. The prover computation requires a number of group operations linear in the size of the QSP, aside from the computation of $h(x)$, which can be computed in $O(d \cdot \log^2(d))$ time, where d is the degree of the QSP, via multipoint evaluation and interpolation. When we construct a SNARK for circuit SAT, we use a QSP for a universal circuit, which has size $O(|C| \log |C|)$ where $|C|$ is the maximum size of the circuits in the satisfiability problem.

Verifier's Work. The verification of the SNARK is proportional to the statement size and independent of the size of the witness. We can further reduce the verification work [22] to a constant plus a hash function evaluation by applying an ordinary hash function to the statement and proving a new relation which takes the the hash output as the statement.

4 Quadratic Programs for Arithmetic Circuits

We also construct *Quadratic Arithmetic Programs* (QAPs), a natural extension of QSPs which "naturally" compute arithmetic circuits modulo the group order p. For some functions, arithmetic circuits are much smaller than their Boolean counterparts, suggesting that, in such cases, QAPs are a more attractive option. In fact, it turns out (see [38]) that QAPs are *more efficient* than QSPs, even for the Boolean case.

The full details of the QAP construction appear in the final version [22]; here we present the definition of QAPs and our main result about them.

Definition 7 (Quadratic Arithmetic Programs (QAP)). *A quadratic arithmetic program (QAP) Q over field F contains three sets of polynomials $\mathcal{V} = \{v_k(x) : k \in \{0, \ldots, m\}\}$, $\mathcal{W} = \{w_k(x) : k \in \{0, \ldots, m\}\}$, $\mathcal{Y} = \{y_k(x) : k \in \{0, \ldots, m\}\}$, and a divisor polynomial $D(x)$, all from $F[x]$.*

Let $f : F^n \longrightarrow F^{n'}$ be a function having input variables with labels $1, \ldots, n$ and output variables with labels $m - n' + 1, \ldots, m$. We say that Q is a QAP that computes f if the following is true: $a_1, \ldots, a_n, a_{m-n'+1}, \ldots, a_m \in F^{n+n'}$ is a valid assignment to the input/output variables of f iff there exist $(a_{n+1}, \ldots, a_{m-n'}) \in F^{m-n-n'}$ such that $D(x)$ divides:

$$\Big(v_0(x) + \sum_{k=1}^{m} a_k \cdot v_k(x)\Big) \cdot \Big(w_0(x) + \sum_{k=1}^{m} a_k \cdot w_k(x)\Big) - \Big(y_0(x) + \sum_{k=1}^{m} a_k \cdot y_k(x)\Big).$$

The size of Q is m. The degree of Q is $\deg(D(x))$.

We prove that we can build very efficient QAPs for arbitrary circuits.

Theorem 5. *Let C be an arithmetic circuit with input from F^n that has s multiplication gates, each with fan-in two, and whose output gates are all multiplication gates. There is a QAP with size $n + s$ and degree s that computes C.*

5 Concrete Performance

We developed a system called Pinocchio [38] that includes a compiler that transforms a subset of C into either a QSP or QAP, and a set of programs for generating the CRS, creating proofs, and verifying proofs. It supports NIZK proofs and VC proofs, with both designated and public verifiers. We use a pairing-based encoding, with a 256-bit BN-curve [5] that provides 128 bits of security.

We find that QAPs outperform QSPs, and that Pinocchio significantly outperforms state-of-the-art systems [16, 40] based on PCPs [2, 25, 30].[9] For example, we measured the time for NxN matrix multiplication using random 32-bit matrix entries. For $N = 25$ to 100, Pinocchio's verifier takes 8-13ms, making it 5-7 *orders of magnitude* faster than previous work, while the worker takes 8.9-776.4s, making it $19 - 60\times$ faster.

Acknowledgments. We thank Nir Bitansky, Jens Groth, Yuval Ishai, Seny Kamara, Helger Lipmaa, and the anonymous reviewers for all of their helpful suggestions.

References

[1] Abe, M., Fehr, S.: Perfect NIZK with adaptive soundness. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 118–136. Springer, Heidelberg (2007)
[2] Arora, S., Lund, C., Motwani, R., Sudan, M., Szegedy, M.: Proof verification and the hardness of approximation problems. J. ACM 45(3), 501–555 (1998)
[3] Arora, S., Safra, S.: Probabilistic checking of proofs: A new characterization of NP. J. ACM 45(1), 70–122 (1998)
[4] Babai, L., Fortnow, L., Levin, L.A., Szegedy, M.: Checking computations in polylogarithmic time. In: STOC, pp. 21–31 (1991)
[5] Barreto, P.S.L.M., Naehrig, M.: Pairing-friendly elliptic curves of prime order. In: Preneel, B., Tavares, S. (eds.) SAC 2005. LNCS, vol. 3897, pp. 319–331. Springer, Heidelberg (2006)
[6] Bellare, M., Palacio, A.: The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 273–289. Springer, Heidelberg (2004)
[7] Ben-Sasson, E., Chiesa, A., Genkin, D., Tromer, E.: On the concrete-efficiency threshold of probabilistically-checkable proofs. In: STOC (to appear 2013)
[8] Bitansky, N., Canetti, R., Chiesa, A., Tromer, E.: From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In: ITCS, pp. 326–349 (2012)
[9] Blum, M., Feldman, P., Micali, S.: Non-interactive zero-knowledge and its applications (extended abstract). In: STOC, pp. 103–112 (1988)
[10] Blum, M., De Santis, A., Micali, S., Persiano, G.: Noninteractive zero-knowledge. SIAM Journal on Computing 20(6), 1084–1118 (1991)
[11] Boneh, D., Boyen, X., Goh, E.-J.: Hierarchical identity based encryption with constant size ciphertext. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 440–456. Springer, Heidelberg (2005)

[9] We are not aware of any implementations based on other non-PCP-based approaches (e.g., Groth [27] or Lipmaa [33]), making direct comparisons difficult.

[12] Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
[13] Boneh, D., Gentry, C., Waters, B.: Collusion resistant broadcast encryption with short ciphertexts and private keys. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 258–275. Springer, Heidelberg (2005)
[14] Brassard, G., Chaum, D., Crépeau, C.: Minimum disclosure proofs of knowledge. J. Comput. Syst. Sci. 37(2), 156–189 (1988)
[15] Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited. J. ACM 51(4), 557–594 (2004)
[16] Cormode, G., Mitzenmacher, M., Thaler, J.: Practical verified computation with streaming interactive proofs. In: ITCS (2012)
[17] Di Crescenzo, G., Lipmaa, H.: Succinct NP proofs from an extractability assumption. In: Beckmann, A., Dimitracopoulos, C., Löwe, B. (eds.) CiE 2008. LNCS, vol. 5028, pp. 175–185. Springer, Heidelberg (2008)
[18] Damgård, I.B.: Towards practical public key systems secure against chosen ciphertext attacks. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 445–456. Springer, Heidelberg (1992)
[19] Damgård, I., Faust, S., Hazay, C.: Secure two-party computation with low communication. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 54–74. Springer, Heidelberg (2012)
[20] Feige, U., Goldwasser, S., Lovász, L., Safra, S., Szegedy, M.: Interactive proofs and the hardness of approximating cliques. J. ACM 43(2), 268–292 (1996)
[21] Gennaro, R., Gentry, C., Parno, B.: Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 465–482. Springer, Heidelberg (2010)
[22] Gennaro, R., Gentry, C., Parno, B., Raykova, M.: Quadratic span programs and succinct NIZKs without PCPs. Cryptology ePrint Archive, Report 2012/215 (2012)
[23] Gentry, C., Wichs, D.: Separating succinct non-interactive arguments from all falsifiable assumptions. In: STOC, pp. 99–108. ACM (2011)
[24] Gjøsteen, K.: Subgroup membership problems and public key cryptosystems. PhD thesis, Norwegian University of Science and Technology (2004)
[25] Goldwasser, S., Kalai, Y.T., Rothblum, G.N.: Delegating computation: Interactive proofs for muggles. In: STOC, pp. 113–122 (2008)
[26] Goldwasser, S., Lin, H., Rubinstein, A.: Delegation of computation without rejection problem from designated verifier CS-proofs. IACR Cryptology ePrint Archive, 2011: 456 (2011)
[27] Groth, J.: Short pairing-based non-interactive zero-knowledge arguments. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 321–340. Springer, Heidelberg (2010)
[28] Groth, J., Ostrovsky, R., Sahai, A.: New techniques for noninteractive zero-knowledge. Journal of the ACM 59(3), 11:1–11:35 (2012)
[29] Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. SIAM Journal on Computing 41(5), 1193–1232 (2012)
[30] Ishai, Y., Kushilevitz, E., Ostrovsky, R.: Efficient arguments without short PCPs. In: IEEE Conference on Computational Complexity (2007)
[31] Karchmer, M., Wigderson, A.: On span programs. In: Structure in Complexity Theory Conference, pp. 102–111 (1993)
[32] Kilian, J.: A note on efficient zero-knowledge proofs and arguments (extended abstract). In: STOC, pp. 723–732 (1992)

[33] Lipmaa, H.: Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 169–189. Springer, Heidelberg (2012)
[34] Loftus, J., May, A., Smart, N.P., Vercauteren, F.: On CCA-secure somewhat homomorphic encryption. In: Miri, A., Vaudenay, S. (eds.) SAC 2011. LNCS, vol. 7118, pp. 55–72. Springer, Heidelberg (2012)
[35] Lund, C., Fortnow, L., Karloff, H.J., Nisan, N.: Algebraic methods for interactive proof systems. J. ACM 39(4), 859–868 (1992)
[36] Micali, S.: Computationally sound proofs. SIAM J. Comput. 30(4), 1253–1298 (2000); Extended abstract in FOCS (1994)
[37] Naor, M.: On cryptographic assumptions and challenges. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 96–109. Springer, Heidelberg (2003)
[38] Parno, B., Gentry, C., Howell, J., Raykova, M.: Pinocchio: Nearly practical verifiable computation. In: Proceedings of the IEEE Symposium on Security and Privacy (May 2013)
[39] Parno, B., Raykova, M., Vaikuntanathan, V.: How to delegate and verify in public: Verifiable computation from attribute-based encryption. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 422–439. Springer, Heidelberg (2012)
[40] Setty, S., Vu, V., Panpalia, N., Braun, B., Blumberg, A.J., Walfish, M.: Taking proof-based verified computation a few steps closer to practicality. In: Proceedings of USENIX Security (August 2012)
[41] Shamir, A.: IP = PSPACE. J. ACM 39(4), 869–877 (1992)

Zero-Knowledge Argument for Polynomial Evaluation with Application to Blacklists

Stephanie Bayer and Jens Groth[*]

University College London
{s.bayer,j.groth}@cs.ucl.ac.uk

Abstract. Verification of a polynomial's evaluation in a secret committed value plays a role in cryptographic applications such as non-membership or membership proofs. We construct a novel special honest verifier zero-knowledge argument for correct polynomial evaluation. The argument has logarithmic communication cost in the degree of the polynomial, which is a significant improvement over the state of the art with cubic root complexity at best. The argument is relatively efficient to generate and very fast to verify compared to previous work. The argument has a simple public-coin 3-move structure and only relies on the discrete logarithm assumption.

The polynomial evaluation argument can be used as a building block to construct zero-knowledge membership and non-membership arguments with communication that is logarithmic in the size of the blacklist. Non-membership proofs can be used to design anonymous blacklisting schemes allowing online services to block misbehaving users without learning the identity of the user. They also allow the blocking of single users of anonymization networks without blocking the whole network.

Keywords: Zero-knowledge argument, discrete logarithm, polynomial evaluation, anonymous blacklisting, membership and non-membership proofs.

1 Introduction

In many cryptographic applications a party wants to prove possession of a secret value u that fulfills a certain property. Since polynomials are widely used a natural question is for instance given a polynomial $P(X)$ and a value v whether the secret value u satisfies $P(u) = v$ in a prime order field $\mathbb{Z}_p$.

We propose a special honest verifier zero-knowledge argument of knowledge for two committed values u, v satisfying $P(u) = v$ for a given polynomial $P(X)$ of degree D. The argument has the following properties:

- It is based on the discrete logarithm assumption in a prime order group
- It has a standard 3-round public coin structure and a common reference string with just a few group elements

[*] The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement n. 307937 and the Engineering and Physical Sciences Research Council grant EP/G013829/1.

- Communication grows logarithmically in the degree of the polynomial
- Both the prover and the verifier are computationally efficient
- We have a working implementation, which gives us real life performance data

As a concrete application of our polynomial evaluation argument we will look at blacklisting anonymous users, which is a notoriously difficult cryptographic problem. Anonymizing networks such as Tor [35] allow a user to hide her IP address and are used by a number of groups including undercover police agents, abuse victims and citizens living under dictatorships. During the Arab Spring for instance the Tor network experienced a spike in users from Libya and Egypt [13]. However, anonymous access to services can also lead to abuse. Wikipedia for instance allows anonymous postings, but blocks the IP address of misbehaving users. This crude solution means that if one user of Tor abuses Wikipedia, all users whose traffic comes out of the same Tor relay with this IP-address are blocked. So one misbehaving user causes many innocent users to be punished. Johnson et al. [20] suggested the Nymble system to deal with this problem. In this alternative solution IP-addresses are not blocked but instead each user anonymously proves that she has not been blacklisted. Using the polynomial evaluation argument we construct a non-membership proof, which enables a user to efficiently prove that she has not been blacklisted.

We can also use our polynomial evaluation argument to construct efficient membership proofs. Membership proofs are useful when operating a whitelist access control system, or in applications such as e-voting or e-auctions where users want to prove that their votes are valid or their bids belong to a set of approved values.

1.1 Our Contribution

Our main contribution is an efficient special honest verifier zero-knowledge argument of knowledge for two secret committed values $u, v \in \mathbb{Z}_p$ satisfying $P(u) = v$ for a given polynomial $P(X) \in \mathbb{Z}_p[X]$, where p is a prime. We work over an order p group $\mathbb{G}$ and use the Pedersen commitment scheme, i.e., a commitment to u is of the form $g^u h^r$ for some $r \in \mathbb{Z}_p$. Given the coefficients of the polynomial $P(X) = \sum_{i=0}^{D} a_i X^i$ and two Pedersen commitments our zero-knowledge argument can demonstrate knowledge of openings of the commitments to values u and v such that $P(u) = v$.

Our polynomial evaluation argument is highly efficient. The communication complexity is $O(\log D)$ group and field elements, which is very small compared to the statement size of D field elements. The prover computes $O(\log D)$ exponentiations and $O(D \log D)$ multiplications in $\mathbb{Z}_p$, and the verifier calculates $O(\log D)$ exponentiations and $O(D)$ multiplications in $\mathbb{Z}_p$. The constants in the expressions are small and the argument very efficient in practice as illustrated by a concrete implementation. We refer to Sections 3 and 5 for further details on the efficiency and a comparison with previous solutions.

The argument has a simple 3-move public coin structure: the prover sends a message, the verifier picks a challenge uniformly at random from $\mathbb{Z}_p$, and the prover answers the challenge. It has perfect completeness, perfect special honest verifier zero-knowledge[1], and computational soundness, which is based on the discrete logarithm assumption in $\mathbb{G}$.

The discrete logarithm assumption is one of the most fundamental and well-studied cryptographic assumptions. There are several types of prime order groups where the discrete logarithm assumption is believed to hold, for instance an order p subgroup of $\mathbb{Z}_q^*$ where q is a large prime, or a group of points on an elliptic curve. There are examples of elliptic curve groups where group elements are roughly $|p|$ bits and the best known attacks have a complexity of $\Omega(\sqrt{p})$ group operations. In such groups a communication complexity of $O(k \log k)$ bits suffices to get a security level of 2^{-k} when $D = \text{poly}(k)$ and $|p| = O(k)$.[2]

Based on the polynomial evaluation argument we then construct zero-knowledge arguments for membership and non-membership with logarithmic communication complexity. More precisely, given a Pedersen commitment c and a list of values $\mathcal{L} = \{\lambda_1, \dots, \lambda_D\}$ we give a zero-knowledge argument of knowledge that c is a commitment to $u \in \mathcal{L}$ in the case of a membership proof or $u \notin \mathcal{L}$ in the case of a non-membership proof. Following Brands et al. [4] we do this by computing the polynomial $P(X) = \prod_{i=1}^{D}(X - \lambda_i)$ and demonstrating $P(u) = 0$ in the case of a membership proof or $P(u) \neq 0$ in the case of a non-membership proof. With our polynomial evaluation argument this requires only $O(\log D)$ communication, which is much smaller than the size of the list.

1.2 Related Work

Polynomial Evaluation Arguments. Given two committed values u, v we give a zero-knowledge argument that $P(u) = v$ for a public polynomial $P(X)$ of degree D. Kilian [21] gave a communication efficient argument for circuit satisfiability and several other general purpose zero-knowledge arguments for NP-languages exist. However, since they are not tailored for the discrete logarithm setting using them would require a costly NP-reduction.

In the prime order group setting there are already several general zero-knowledge techniques for the satisfiability of arithmetic circuits that can demonstrate the correctness of a polynomial evaluation. Using Cramer and Damgård [11] we would get a linear communication complexity for this problem and using Groth [15] we would get a communication complexity of $O(\sqrt{D})$ group elements. Using stronger assumptions and a pairing-based argument by Groth [16] we could reduce this further to a communication complexity of $O(D^{\frac{1}{3}})$ group elements.

[1] Standard techniques can be used to make the argument fully zero-knowledge against malicious adversaries at a negligible cost of a few extra group elements in the common reference string as described in Section 2.2. There is therefore no loss of generality in considering just the special honest verifier zero-knowledge case.

[2] It is uncommon to have zero-knowledge arguments for large statements where an asymptotic communication complexity of $O(k \log k)$ suffices for a security level of 2^{-k}. Hash-trees and cut-and-choose techniques give a communication complexity of $\Omega(k^2)$ for instance and RSA-type assumptions require group elements to be $k^{3-o(1)}$ bits to guard against factorization with the general number field sieve.

Fujisaki and Okamoto [14] looked at the specific problem of polynomial evaluation in an RSA-based context but their zero-knowledge argument has linear complexity. The most efficient zero-knowledge argument for correct polynomial evaluation stems from Brands et al. [4], which has a communication complexity of $O(\sqrt{D})$ group elements and is based on the discrete logarithm assumption just like our argument.

Membership Proofs and Non-membership Proofs. Proofs for set membership and non-membership for a committed $u \in \mathcal{L}$ or $u \notin \mathcal{L}$ where $\mathcal{L} = \{\lambda_1, \ldots, \lambda_D\}$ have been studied in their own right. The most straightforward approach is to prove in a one by one manner the conjunction $\lambda_1 \neq u \wedge \ldots \wedge \lambda_D \neq u$ in the case of non-membership or the disjunction $\lambda_1 = u \vee \ldots \vee \lambda_D = u$ in the case of membership. In the context of revoking members of group signature schemes Bresson and Stern [5] proposed such a solution based on the strong RSA assumption. Peng and Bao [32] gave a general discrete logarithm based zero-knowledge arguments of non-membership with linear complexity. Brands et al. [4] improved the communication complexity to $O(\sqrt{D})$ group elements and later Peng [31] gave a solution for a non-membership proof with the same complexity using techniques similar to Brands et al. [4].

Accumulators [2,9,37,29,8,38] provide another mechanism for giving membership proofs. An accumulator is a succinct aggregate of a set of values where it is possible to issue membership proofs for each accumulated value. A party in possession of such a membership proof, typically a few group elements, can then demonstrate that the value is included in the set. Non-membership accumulators have also been proposed [22,39]. However, most accumulators rely on a trusted party to maintain the accumulator and if corrupt this trusted party can issue arbitrary membership proofs. Furthermore, accumulators rely on cryptographic assumptions that have been less studied than the discrete logarithm problem, for instance the strong RSA assumption or the pairing-based q-SDH assumption. These assumptions also mean that group elements have to be large and once this has been factored in the accumulator-based solutions end up having larger communication than our membership and non-membership proofs for groups over elliptic or hyper-elliptic curves. The construction of Camacho et al. [6] does not rely on a trusted party and only assumes the existence of hash functions; however proofs in their setting depend logarithmically on the number of accumulated elements.

In Song's non-membership proof [34] the prover publishes a constant number of elements and the verifier checks these elements against a blacklist by carrying out a few operations for each blacklist element; several systems along these lines have been proposed [1,3,27]. The operations consist either of exponentiations or pairings, so this scheme places a heavy computational burden on the verifier.

Camenisch et al. [7] gave a membership proof where the elements in the set are signed by a trusted third party. Now membership can be proven with a constant number of group elements by demonstrating that the value has been signed. Related ideas have recently been used by Libert et al. [23] in the context of revoking group signatures, where a trusted third party signs representatives of sets that cover the whitelist of non-revoked users and the user gives a zero-knowledge proof of belonging to this set [28].

All these solutions suffer from similar drawbacks that accumulator-based solutions have though. They require trust in a third party to be honest when blacklisting members or signing messages, and to get efficient proofs the signatures are built from strong assumptions such as the strong RSA assumption or pairing-based assumptions.

A different approach is taken by Nymble-like systems [36,19,18,26], which also rely on a trusted third party. The user obtains a pseudonym, a “nymble”, from the trusted third party which is only valid for a certain time frame with one server. The blacklist consists of nymbles by misbehaving users and in [36,19] the server simply checks if the nymble of a connecting user is contained in the blacklist. To weaken the trust in the trusted third party Lofgren and Hopper [26] use accumulators together with the Nymble setup, while Henry and Goldberg [18] rely on the techniques of Brands et al. [4] for the user to give a zero-knowledge argument for the non-membership of the blacklist.

2 Preliminaries

We write $y = A(x; r)$ when the algorithm A on input x and randomness r, outputs y. We write $y \leftarrow A(x)$ for the process of picking randomness r at random and setting $y = A(x; r)$. We also write $y \leftarrow S$ for sampling y uniformly at random from a set S. We say a function $f : \mathbb{N} \to [0, 1]$ is negligible if $f(k) = O(k^{-c})$ for every constant $c > 0$. We say $1 - f$ is overwhelming if f is negligible. We will give a security parameter k written in unary as input to all parties in our protocols. Intuitively, the higher the security parameter the more secure the protocol.

2.1 The Pedersen Commitment Scheme

The Pedersen commitment scheme [30] works as follows. First the key generation algorithm $\mathcal{G}$ on input 1^k chooses a cyclic group $\mathbb{G}$ of k-bit prime order p and random generators g, h. The commitment key is $ck = (\mathbb{G}, p, g, h)$. To commit to $a \in \mathbb{Z}_p$ the committer picks randomness $r \in \mathbb{Z}_p$ and computes

$$\mathrm{com}_{ck}(a; r) = g^a h^r.$$

The Pedersen commitment scheme is computationally binding under the discrete logarithm assumption, i.e., a non-uniform probabilistic polynomial time adversary has negligible probability of finding two different openings of the same commitment. The Pedersen commitment scheme is perfectly hiding since the commitment is uniformly distributed in $\mathbb{G}$ no matter what the value a is.

The Pedersen commitment scheme is homomorphic. For all $a, b \in \mathbb{Z}_p$ and $r, s \in \mathbb{Z}_p$

$$\mathrm{com}_{ck}(a; r) \cdot \mathrm{com}_{ck}(b; s) = g^a h^r \cdot g^b h^s = g^{a+b} h^{r+s} = \mathrm{com}_{ck}(a + b; r + s).$$

We use the Pedersen commitment scheme because of its elegance and its security resting on the discrete logarithm assumption. However, our protocols could also work with other homomorphic commitment schemes and we will describe our arguments in a way such that it would be easy to plug in another homomorphic commitment scheme.

2.2 Zero-Knowledge Arguments of Knowledge

In the arguments we consider a prover $\mathcal{P}$ and a verifier $\mathcal{V}$ both of which are probabilistic polynomial time interactive algorithms. All our protocols will be 3-move public-coin arguments; first the prover sends a message, then the verifier responds with a random challenge, and finally the prover sends an answer to the challenge.

We assume the existence of a probabilistic polynomial time setup algorithm $\mathcal{G}$ that when given a security parameter k returns a common reference string σ. In this paper the common reference string will always be a public key ck for the Pedersen commitment scheme.

Let R be a polynomial time decidable ternary relation, we call w a witness for a statement x if $(\sigma, x, w) \in R$. We define the CRS-dependent language

$$L_\sigma = \{x \mid \exists w : (\sigma, x, w) \in R\}$$

as the set of statements x that have a witness w in the relation R.

The public transcript produced by $\mathcal{P}$ and $\mathcal{V}$ when interacting on inputs s and t is denoted by $tr \leftarrow \langle \mathcal{P}(s), \mathcal{V}(t) \rangle$. The transcript consists of the initial message from the prover, the random challenge from the verifier, the answer from the prover and the decision to accept or reject from the verifier. We write $\langle \mathcal{P}(s), \mathcal{V}(t) \rangle = b$ depending on whether the verifier rejects, $b = 0$, or accepts, $b = 1$.

Definition 1 (Argument of knowledge). *The triple* $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ *is called an* argument of knowledge *for the relation R if we have completeness and witness-extended emulation as defined below.*

Definition 2 (Perfect completeness). $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ *has perfect completeness if for all non-uniform polynomial time adversaries* $\mathcal{A}$

$$\Pr[\sigma \leftarrow \mathcal{G}(1^k); (x, w) \leftarrow \mathcal{A}(\sigma) : (\sigma, x, w) \notin R \text{ or } \langle \mathcal{P}(\sigma, x, w), \mathcal{V}(\sigma, x) \rangle = 1] = 1.$$

To define an argument of knowledge we follow Groth and Ishai [17] that borrowed the term witness-extended emulation from Lindell [25]. Informally, their definition says that given an adversary that produces an acceptable argument with some probability, there exist an emulator that produces a similar argument with the same probability and at the same time provides a witness w. We give a strong black-box definition where the emulator just has black-box access to a prover and verifier's interaction, which it can rewind and run again with fresh randomness for the verifier.

Definition 3 (Computational witness-extended emulation). $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ *has* witness-extended emulation *if for all deterministic polynomial time* $\mathcal{P}^*$ *there exists an expected polynomial time emulator* $\mathcal{X}$ *such that for all non-uniform polynomial time interactive adversaries* $\mathcal{A}$

$$\begin{aligned} &\Pr[\sigma \leftarrow \mathcal{G}(1^k); (x, s) \leftarrow \mathcal{A}(\sigma); tr \leftarrow \langle \mathcal{P}^*(\sigma, x, s), \mathcal{V}(\sigma, x) \rangle : \mathcal{A}(tr) = 1] \\ \approx\ &\Pr[\sigma \leftarrow \mathcal{G}(1^k); (x, s) \leftarrow \mathcal{A}(\sigma); (tr, w) \leftarrow \mathcal{X}^{\langle \mathcal{P}^*(\sigma, x, s), \mathcal{V}(\sigma, x) \rangle}(\sigma, x) : \\ &\quad \mathcal{A}(tr) = 1 \text{ and if } tr \text{ is accepting then } (\sigma, x, w) \in R], \end{aligned}$$

where the verifier picks fresh public coin challenges in each oracle call by $\mathcal{X}^{\langle \mathcal{P}^*(\sigma, x, s), \mathcal{V}(\sigma, x) \rangle}$.

In the definition, s can be interpreted as the state of $\mathcal{P}^*$, including the randomness. So, whenever $\mathcal{P}^*$ is able to make a convincing argument when in state s, the emulator can extract a witness. This is why we call it an argument of knowledge.

Definition 4 (Public coin). *An argument $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ is called* public coin *if the verifier chooses his messages uniformly at random and independently of the messages sent by the prover, i.e., the challenges correspond to the verifier's randomness ρ.*

An argument is zero-knowledge if it does not leak information about the witness beyond what can be inferred from the truth of the statement. We will present arguments that have special honest verifier zero-knowledge in the sense that if the verifier's challenge is known in advance, then it is possible to simulate the entire argument without knowing the witness.

Definition 5 (Perfect special honest verifier zero-knowledge). *A public coin argument $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ is called a* perfect special honest verifier zero knowledge (SHVZK) *argument for R if there exists a probabilistic polynomial time simulator $\mathcal{S}$ such that for all interactive non-uniform polynomial time adversaries $\mathcal{A}$ we have*

$$\Pr[\sigma \leftarrow \mathcal{G}(1^k); (x, w, \rho) \leftarrow \mathcal{A}(\sigma); tr \leftarrow \langle \mathcal{P}(\sigma, x, w), \mathcal{V}(\sigma, x; \rho)\rangle : (\sigma, x, w) \in R \text{ and } \mathcal{A}(tr) = 1]$$
$$= \Pr[\sigma \leftarrow \mathcal{G}(1^k); (x, w, \rho) \leftarrow \mathcal{A}(\sigma); tr \leftarrow \mathcal{S}(\sigma, x, \rho) : (\sigma, x, w) \in R \text{ and } \mathcal{A}(tr) = 1],$$

where ρ is the public coin randomness used by the verifier as the challenge.

Full zero-knowledge. In real life applications special honest verifier zero-knowledge may not suffice since a malicious verifier may give non-random challenges. However, it is easy to convert an SHVZK argument into a full zero-knowledge argument secure against *arbitrary* verifiers in the common reference string model using standard techniques. The conversion can be very efficient and only costs a small additive overhead, so we will in the paper without loss of generality just focus on building efficient SHVZK arguments.

One example of such a conversion that would work in our case is the following: The common reference string is set up with an additional group element y. The prover will now use an OR-proof [12] to show that she knows a witness for the statement being true or she knows the discrete logarithm of y. Since she does not know the discrete logarithm of y this is a convincing argument of knowledge. On the other hand, we can set the simulator up such that it does know the discrete logarithm of y and now it is easy to simulate proofs. This conversion yields an argument of knowledge with perfect zero-knowledge at the price of an extra group element in the common reference string.

3 Polynomial Evaluation Argument

Given a polynomial $P(U) = \sum_{i=0}^{D} a_i U^i$ and two commitments c_0, c_v, we will describe an argument of knowledge of openings of the commitments to values u and v such that $P(u) = v$. (The notation c_0 for the commitment to $u = u^{2^0}$ matches other commitments c_j to u^{2^j} that the prover will construct in the argument.)

By padding with zero-coefficients we can without loss of generality assume $D = 2^{d+1} - 1$. It is useful to write i in binary, i.e., $i = i_d \dots i_0$ where $i_j \in \{0, 1\}$. We can then rewrite the term U^i as $U^i = U^{\sum_{j=0}^{d} i_j 2^j} = \prod_{j=0}^{d} (U^{2^j})^{i_j}$. Substituting this in the polynomial we get

$$P(U) = \sum_{i=0}^{D} a_i U^i = \sum_{i_0,\dots,i_d=0}^{1} a_{i_d \dots i_0} \prod_{j=0}^{d} (U^{2^j})^{i_j}.$$

The prover will make commitments $c_1, \dots, c_d$ to $u^{2^1}, u^{2^2}, \dots, u^{2^d}$, use standard techniques to prove they are well-formed, and prove that when inserted into the rewritten polynomial we have $\sum_{i_0,\dots,i_d=0}^{1} a_{i_d \dots i_0} \prod_{j=0}^{d} (u^{2^j})^{i_j} = v$. Since $d = \lfloor \log D \rfloor$ the prover only makes a logarithmic number of commitments, which will help keep communication low.

To show the committed powers of u in $c_0, c_1, \dots, c_d$ evaluate to the commited v the prover picks random values $f_0, \dots, f_d \leftarrow \mathbb{Z}_p$ and defines a new polynomial

$$Q(X) = \sum_{i_0,\dots,i_d=0}^{1} a_{i_d \dots i_0} \prod_{j=0}^{d} (X u^{2^j} + f_j)^{i_j} X^{1-i_j} = X^{d+1} P(u) + X^d \delta_d + \dots + X \delta_1 + \delta_0.$$

The idea behind this choice of $Q(X)$ is that for each i_j either an Xu^{2^j} factor is included or an X factor is included so the coefficient of X^{d+1} is $P(u)$. Each f_j on the other hand is not multiplied by X and will therefore only affect the lower degree coefficients $\delta_0, \dots, \delta_d$ of $Q(X)$.

The prover will now demonstrate that the coefficient of X^{d+1} in the secret $Q(X)$ is the same as v in a way that cancels out the $\delta_0, \dots, \delta_d$ coefficients. The prover sends the verifier commitments $c_{f_0}, \dots, c_{f_d}$ to $f_0, \dots, f_d$, and commitments $c_{\delta_0}, \dots, c_{\delta_d}$ to $\delta_0, \dots, \delta_d$. Afterwards, the verifier will pick a random challenge $x \leftarrow \mathbb{Z}_p$. The prover will now open suitable products of the commitments in a way such that the verifier can check that the committed values u, v satisfy $Q(x) = x^{d+1} v + x^d \delta_d + \dots + \delta_0$. More precisely, after receiving the challenge x the prover opens each product $c_j^x c_{f_j}$ to $\bar{f}_j = x u^{2^j} + f_j$. Furthermore, the prover opens $c_v^{x^{d+1}} \prod_{j=0}^{d} c_{\delta_j}^{x^j}$ to $\bar{\delta} = \sum_{i_0,\dots,i_d=0}^{1} a_{i_0 \dots i_d} \prod_{j=0}^{d} \bar{f}_j^{\,i_j} x^{1-i_j}$. Note that the verifier can calculate $\bar{\delta}$ himself and therefore only accepts the opening if

$$\sum_{i_0,\dots,i_d=0}^{1} a_{i_0 \dots i_d} \prod_{j=0}^{d} \bar{f}_j^{\,i_j} x^{1-i_j} = x^{d+1} v + x^d \delta_d + \dots + x \delta_1 + \delta_0.$$

This has negligible probability of being true unless $P(u) = v$.

Returning to the commitments $c_1, \dots, c_d$ to $u^{2^1}, \dots, u^{2^d}$ we said the prover could use standard techniques to show that they contain the correct powers of u. To do this the prover sends some commitments c_{fu_j} to $f_j u^{2^j}$ to the verifier and later opens the commitments $c_{u_{j+1}}^x c_{u_j}^{-\bar{f}_j} c_{fu_j}$ to

$$x u^{2^{j+1}} - (x u^{2^j} + f_j) u^{2^j} + f_j u^{2^j} = 0.$$

The full polynomial evaluation argument is given below.

Common reference string: $ck \leftarrow \mathcal{G}(1^k)$

Statement: $P(U) = \sum_{i=0}^{D} a_i U^i = \sum_{i_0,\ldots,i_d=0}^{1} a_{i_d\ldots i_0} \prod_{j=0}^{d} (U^{2^j})^{i_j} \in \mathbb{Z}_p[U]$ and $c_0, c_v \in \mathbb{G}$

Prover's witness: $u, v, r_0, t \in \mathbb{Z}_p$ such that $c_0 = \mathrm{com}_{ck}(u; r_0), c_v = \mathrm{com}_{ck}(v; t)$ and $P(u) = v$

Initial message: Compute

1. $c_1 = \mathrm{com}_{ck}(u^{2^1}; r_1), \ldots, c_d = \mathrm{com}_{ck}(u^{2^d}; r_d)$ where $r_1, \ldots, r_d \leftarrow \mathbb{Z}_p$
2. $c_{f_0} = \mathrm{com}_{ck}(f_0; s_0), \ldots, c_{f_d} = \mathrm{com}_{ck}(f_d; s_d)$ where $f_0, s_0, \ldots, f_d, s_d \leftarrow \mathbb{Z}_p$
3. $\delta_0, \ldots, \delta_d \in \mathbb{Z}_p$ such that

$$\sum_{i_0,\ldots,i_d=0}^{1} a_{i_d\ldots i_0} \prod_{j=0}^{d} (Xu^{2^j} + f_j)^{i_j} X^{1-i_j} = X^{d+1} v + \sum_{j=0}^{d} X^j \delta_j$$

4. $c_{\delta_0} = \mathrm{com}_{ck}(\delta_0; t_0), \ldots, c_{\delta_d} = \mathrm{com}_{ck}(\delta_d; t_d)$ where $t_0, \ldots, t_d \leftarrow \mathbb{Z}_p$
5. $c_{fu_0} = \mathrm{com}_{ck}(f_0 u^{2^0}; \xi_0), \ldots, c_{fu_{d-1}} = \mathrm{com}_{ck}(f_{d-1} u^{2^{d-1}}; \xi_{d-1})$ where $\xi_0, \ldots, \xi_{d-1} \leftarrow \mathbb{Z}_p$

Send: $c_1, \ldots, c_d, c_{f_0}, \ldots, c_{f_d}, c_{\delta_0}, \ldots, c_{\delta_d}, c_{fu_0}, \ldots, c_{fu_{d-1}}$

Challenge: $x \leftarrow \mathbb{Z}_p$

Answer: Compute for all j

$$\bar{f}_j = xu^{2^j} + f_j \qquad \bar{r}_j = xr_j + s_j \qquad \bar{t} = x^{d+1} t + \sum_{i=0}^{d} t_i x^i \qquad \bar{\xi}_j = xr_{j+1} - \bar{f}_j r_j + \xi_j$$

Send: $\bar{f}_0, \bar{r}_0, \ldots, \bar{f}_d, \bar{r}_d, \bar{t}, \bar{\xi}_0, \ldots, \bar{\xi}_{d-1}$

Verification: Accept if and only if for all j

$$c_j^x c_{f_j} = \mathrm{com}_{ck}(\bar{f}_j; \bar{r}_j) \qquad c_{j+1}^x c_j^{-\bar{f}_j} c_{fu_j} = \mathrm{com}_{ck}(0; \bar{\xi}_j)$$

and

$$c_v^{x^{d+1}} \prod_{i=0}^{d} c_{\delta_i}^{x^i} = \mathrm{com}_{ck}\left(\sum_{i_0,\ldots,i_d=0}^{1} a_{i_d\ldots i_0} \prod_{j=0}^{d} \bar{f}_j^{\,i_j} x^{1-i_j}; \bar{t} \right)$$

Example: Let $\mathbb{G} = \langle g = 3 \rangle \subset \mathbb{Z}_{467}^*$, which has prime order $p = 233$ and let $h = 266$. The common reference string describing $(\mathbb{G}, p, g, h)$ is $ck = \{467, 233, 3, 266\}$. The statement consists of the polynomial $P(X) = 93X^4 + 3X^2 + 115X + 51 \in \mathbb{Z}_{233}[X]$ and commitments $c_{u_0} = 90, c_v = 68$. We have $d = \lfloor \log 4 \rfloor = 2$.

The prover knows values $u = 5, v = P(u) = 110 \in \mathbb{Z}_{233}$ and $r_0 = 201, t = 189 \in \mathbb{Z}_{233}$ such that $c_{u_0} = 90 \in \mathbb{G}$ and $c_v = 68 \in \mathbb{G}$. To prove knowledge of the witness the prover first picks $r_1 = 23, r_2 = 63$ at random from $\mathbb{Z}_{233}$ and computes commitments $c_1 = 387$ and $c_2 = 4$ to $u^{2^1} = 25$ and $u^{2^2} = 159$. The prover also picks $f_0 = 161, f_1 = 220, f_2 = 15, s_0 = 10, s_1 = 37, s_2 = 149$ randomly from $\mathbb{Z}_{233}$ and sets $c_{f_0} = 48, c_{f_1} = 4, c_{f_2} = 324$.

Next she computes $\delta_0, \delta_1, \delta_2$. She calculates the five products $\prod_{j=0}^{d} (Xu^{2^j} + f_j)^{i_j} X^{1-i_j}$ for $i = i_2 i_1 i_0 \in \{0, 1, 2, 3, 4\}$ using a binary tree.

$$\begin{tikzcd}
 & & & X^3 \\
 & & X^2 \arrow[ru, dash] \arrow[rd, dash] & \\
 & & & x^3u+X^2f_0=5X^3+161X^2 \\
 & X \arrow[ruu, dash] \arrow[rdd, dash] & & \\
 & & & X^3u^2+X^2f_1=25X^3+220X^2 \\
1 \arrow[ruu, dash] \arrow[rdd, dash] & & X^2u^2+Xf_1 \arrow[ru, dash] \arrow[rd, dash] & \\
 & & & \begin{array}{l} X^3u^3+X^2(u^2f_0+uf_1)+Xf_0f_1 \\ =75X^3+232X^2+4X \end{array} \\
 & Xu^4+f_2 \arrow[r, dash] & X^2u^4+Xf_2 \arrow[r, dash] & X^3u^4+X^2f_2=159X^3+15X^2
\end{tikzcd}$$

The prover takes the a_i and multiplies them on the result of the binary tree, to get

$$\begin{aligned}
i=0: &\quad a_0 \cdot X^3 = 51X^3 \\
i=1: &\quad a_1 \cdot (5X^3+161X^2) = 109X^3+108X^2 \\
i=2: &\quad a_2 \cdot (25X^3+220X^2) = 75X^3+194X^2 \\
i=3: &\quad a_3 \cdot (75X^3+232X^2) = 0 \\
i=4: &\quad a_4 \cdot (159X^3+15X^2) = 108X^3+230X^2
\end{aligned}$$

Last, to extract the values δ_i she adds for $i=0,1,2$ the coefficients for each X^i to get $\delta_0=0, \delta_1=0, \delta_2=66 \mod 233$. Now the prover picks $t_0=33, t_1=201, t_2=205$ at random and commits to the δ_i's to get $c_{\delta_0}=438, c_{\delta_1}=329, c_{\delta_2}=467$.

Finally, the prover calculates $f_0u=106, f_1u^2=174$ and commits to these values. So, she picks $\xi_0=13, \xi_1=75$ and computes $c_{fu_0}=352, c_{fu_1}=141$. She sends all the commitments to the verifier.

The verifier returns a random challenge $x=123 \in \mathbb{Z}_{233}$, and the prover calculates answers $\bar{f}_0=77, \bar{f}_1=33, \bar{f}_2=0, \bar{r}_0=35, \bar{r}_1=70, \bar{r}_2=209, \bar{t}=189, \bar{\xi}_0=180, \bar{\xi}_1=75$, and sends all values to the verifier.

The verifier checks first if all commitments are in $\mathbb{G}$ and all answers valid numbers in $\mathbb{Z}_{233}$. Then he checks for $i=0,1,2$ if $c_{u_i}^x c_{f_i}=\mathrm{com}_{ck}(\bar{f}_i;\bar{r}_i)$:

$$c_{u_0}^x c_{f_0}=68=\mathrm{com}_{ck}(\bar{f}_0;\bar{r}_0) \quad c_{u_1}^x c_{f_1}=91=\mathrm{com}_{ck}(\bar{f}_1;\bar{r}_1) \quad c_{u_2}^x c_{f_2}=220=\mathrm{com}_{ck}(\bar{f}_2;\bar{r}_2).$$

Next, he checks $c_{u_{i+1}}^x c_{u_i}^{-\bar{f}_i} c_{fu_i}=\mathrm{com}_{ck}(0,\bar{\xi}_i)$ for $i=0,1$, i.e.,

$$c_{u_1}^x c_{u_0}^{-\bar{f}_0} c_{fu_0}=157=\mathrm{com}_{ck}(0;\bar{\xi}_0) \quad c_{u_2}^x c_{u_1}^{-\bar{f}_1} c_{fu_1}=250=\mathrm{com}_{ck}(0;\bar{\xi}_1).$$

Then the verifier calculates $\bar{\delta}=\sum_{i_0,\ldots,i_d=0}^{1} a_{i_0\ldots i_d}\prod_{j=0}^{d}\bar{f}_j^{\,i_j}x^{1-i_j}$ in a binary tree fashion. The output leaves in the binary tree are $x^3=129, x^2\bar{f}_0=166, x^2\bar{f}_2=171, x\bar{f}_0\bar{f}_1=90, x^2\bar{f}_2=0$. He multiplies the values by the a_i's and adds the results together, to get $\bar{\delta}=a_0129+a_1166+a_2171+a_390+a_40=86 \in \mathbb{Z}_{233}$. Finally, he checks the last verification equation $c_v^{x^3} c_{\delta_2}^{x^2} c_{\delta_1}^{x^1} c_{\delta_0}^{x^0}=395=\mathrm{com}_{ck}\left(\bar{\delta};\bar{t}\right)$.

Efficiency. The communication in the polynomial evaluation argument for a degree $D = 2^{d+1} - 1$ polynomial is roughly $4d$ group elements and $3d$ field elements.

The prover uses $8d$ exponentiations to compute the commitments. She also has to calculate $\delta_0, \dots, \delta_d$ that are defined to satisfy $\sum_{i_0,\dots,i_d=0}^{1} a_{i_d \dots i_0} \prod_{j=0}^{d} (Xu^{2^j} + f_j)^{i_j} X^{1-i_j} = X^{d+1}v + \sum_{j=0}^{d} X^j \delta_j$. The prover can calculate the D degree $d+1$ polynomials $\prod_{j=0}^{d}(Xu^{2^j} + f_j)^{i_j} X^{1-i_j}$ in a binary-tree fashion for all choices of $i_0 \dots, i_d \in \{0,1\}$ at a cost of dD multiplications in $\mathbb{Z}_p$. Multiplying with the $a_{i_d \dots i_0}$'s uses another dD multiplications. The total cost for the prover is therefore $8d$ exponentiations in $\mathbb{G}$ and $2dD$ multiplications in $\mathbb{Z}_p$.

The verifier can check the argument using $6d$ exponentiations in $\mathbb{G}$ since the exponent x is used twice in the verification equations. He also needs to compute the sum $\sum_{i_0,\dots,i_d=0}^{1} a_{i_d \dots i_0} \prod_{j=0}^{d} \bar{f}_j^{\,i_j} x^{1-i_j}$, which can be done in a binary tree fashion for all choices of $i_0, \dots, i_d \in \{0,1\}$ using $2D$ multiplications in $\mathbb{Z}_p$.

We have ignored small constants in the calculations above and just focused on the dominant terms. Using multi-exponentiation techniques, randomized verification and other tricks it is possible to reduce the computation even further for the prover and verifier, so the estimates are quite conservative.

Theorem 6. *Assuming the discrete logarithm assumption holds the polynomial evaluation argument is a public coin perfect SHVZK argument of knowledge of openings of* c_0 *and* c_v *to* u *and* v *such that* $P(u) = v$.

Proof. Perfect completeness follows by careful inspection.

We will now argue perfect SHVZK. Given a challenge $x \in \mathbb{Z}_p$ the simulator picks $c_1, \dots, c_d, c_{\delta_1}, \dots, c_{\delta_d} \leftarrow \mathbb{G}$ and $\bar{f}_0, \bar{r}_0, \dots, \bar{f}_d, \bar{r}_d, \bar{t}, \bar{\xi}_0, \dots, \bar{\xi}_{d-1} \leftarrow \mathbb{Z}_p$ and for all j he sets $c_{f_j} = \mathrm{com}_{ck}(\bar{f}_j; \bar{r}_j) c_j^{-x}$ $c_{fu_j} = \mathrm{com}_{ck}(0; \bar{\xi}_j) c_{j+1}^{-x} c_j^{\bar{f}_j}$ and $c_{\delta_0} = \mathrm{com}_{ck}\left(\sum_{i_0,\dots,i_d=0}^{1} a_{i_d \dots i_0} \prod_{j=0}^{d} \bar{f}_j^{\,i_j} x^{1-i_j}; \bar{t}\right) c_v^{-x^{d+1}} \prod_{i=1}^{d} c_{\delta_i}^{-x^i}$.

This is a perfect simulation. In a real argument $c_1, \dots, c_d, c_{\delta_1}, \dots, c_{\delta_d}$ are uniformly random perfectly hiding commitments as in the simulation. In a real argument $\bar{f}_0, \bar{r}_0, \dots, \bar{f}_d, \bar{r}_d, \bar{t}, \bar{\xi}_0, \dots, \bar{\xi}_{d-1} \in \mathbb{Z}_p$ are also uniformly random because of the random choice of $f_0, r_0, \dots, f_d, r_d, t_0, \xi_0, \dots, \xi_{d-1}$. Finally, both in the simulation and in the real argument given these choices the verification equations uniquely determine the values of $c_{f_0}, \dots, c_{f_d}, c_{\delta_0}$ and $c_{fu_0}, \dots, c_{fu_{d-1}}$. This means simulated and real arguments given a challenge x have identical probability distributions.

Finally, we will show that the argument has witness-extended emulation. The emulator $\mathcal{X}$ runs the argument with random challenge $x \leftarrow \mathbb{Z}_p$ and if the transcript tr is accepting it rewinds until it has $d+2$ accepting arguments. For a prover with ϵ chance of making a convincing argument we expect the emulator to rewind $\frac{d+2}{\epsilon}\epsilon = d+2$ times, so $\mathcal{X}$ runs in expected polynomial time.

There is negligible probability that the verifier will end up with two or more transcripts with the same challenge x, so we just need to be able to extract a witness when we have $d+2$ transcripts with different challenges. Given $\bar{f}_j^{(1)}, \bar{r}_j^{(1)}$ and $\bar{f}_j^{(2)}, \bar{r}_j^{(2)}$ in the first two answers to challenges x_1 and x_2 the emulator can take linear combinations of the verification equations to get openings of the commitments

c_j. More precisely, we have that the two answers satisfy $c_j^{x_1} c_{f_j} = \mathrm{com}_{ck}(\bar{f}_j^{(1)}; \bar{r}_j^{(1)})$ $c_j^{x_2} c_{f_j} = \mathrm{com}_{ck}(\bar{f}_j^{(2)}; \bar{r}_j^{(2)})$. Picking α_1, α_2 such that $\alpha_1 x_1 + \alpha_2 x_2 = 1$ and $\alpha_1 + \alpha_2 = 0$ gives us $c_j = c_j^{\alpha_1 x_1 + \alpha_2 x_2} c_{f_j}^{\alpha_1 + \alpha_2} = \mathrm{com}_{ck}(\alpha_1 \bar{f}_j^{(1)} + \alpha_2 \bar{f}_j^{(2)}; \alpha_1 \bar{r}_j^{(1)} + \alpha_2 \bar{r}_j^{(2)})$, which is an opening of c_j.

Other types of linear combinations of the verification equations give us openings of the other commitments $c_{f_j}, c_{f u_j}, c_v$ and c_{δ_i} the prover sends in the initial message. In the case of c_{δ_i} we find the linear combination as follows: Let

$$M = \begin{pmatrix} 1 & x_1 & \dots x_1^{d+1} \\ \vdots & & \vdots \\ 1 & x_{d+2} & \dots x_{d+2}^{d+1} \end{pmatrix}.$$

Since it is a Vandermonde matrix with different $x_1, \dots, x_{d+2}$ it is invertible. By taking linear combinations of the verification equations

$$c_v^{x^{d+1}} \prod_{i=0}^{d} c_{\delta_i}^{x^i} = \mathrm{com}_{ck}\left(\sum_{i_0,\dots,i_d=0}^{1} a_{i_d \dots i_0} \prod_{j=0}^{d} \bar{f}_j^{\,i_j} x^{1-i_j}; \bar{t} \right)$$

for different challenges $x_1, \dots, x_{d+2}$ we get that

$$\begin{pmatrix} \delta_0 & t_0 \\ \vdots & \vdots \\ \delta_d & t_d \\ v & t \end{pmatrix} = M^{-1} \begin{pmatrix} \sum_{i_0,\dots,i_d=0}^{1} a_{i_d \dots i_0} \prod_{j=0}^{d} (\bar{f}_j^{(1)})^{i_j} x_1^{1-i_j} & \bar{t}^{(1)} \\ \vdots & \vdots \\ \sum_{i_0,\dots,i_d=0}^{1} a_{i_d \dots i_0} \prod_{j=0}^{d} (\bar{f}_j^{(d+2)})^{i_j} x_{d+2}^{1-i_j} & \bar{t}^{(d+2)} \end{pmatrix}$$

which gives us openings of $c_{\delta_0}, \dots, c_{\delta_d}$ and c_v.

We now have openings to all the commitments. Because the commitments are binding, each answer must be computed as they are by an honest prover in the argument. Therefore, the verification equations $c_j^x c_{f_j} = \mathrm{com}_{ck}(\bar{f}_j, \bar{r}_j)$ give us $\bar{f}_j = x u_j + f_j$, where u_j is the extracted value in c_j and f_j is the extracted value in c_{f_j}.

The verification equations $c_{j+1}^x c_j^{-\bar{f}_j} c_{f u_j} = \mathrm{com}_{ck}(0; \bar{\xi}_j)$ give us that the committed values satisfy $x u_{j+1} - (x u_j + f_j) u_j + \phi_j = 0$ for $j = 0, \dots, d-1$ with ϕ_j being the value we extracted for $c_{f u_j}$. Since each of the polynomial equalities is of degree 1 in x and holds for $d+2$ different challenges x we see that $u_{j+1} = u_j u_j$. Since $u_0 = u$ this gives us $u_1 = u^{2^1}, u_2 = u^{2^2}, \dots, u_d = u^{2^d}$.

Turning to the verification equation

$$c_v^{x^{d+1}} \prod_{i=0}^{d} c_{\delta_i}^{x^i} = \mathrm{com}_{ck}\left(\sum_{i_0,\dots,i_d=0}^{1} a_{i_d \dots i_0} \prod_{j=0}^{d} \bar{f}_j^{\,i_j} x^{1-i_j}; \bar{t} \right)$$

we now have that this corresponds to the degree $d+1$ polynomial equation

$$X^{d+1} v + X^d \delta_d + \dots + X \delta_1 + \delta_0 = \sum_{i_0,\dots,i_d=0}^{1} a_{i_d \dots i_0} \prod_{j=0}^{d} (X u^{2^j} + f_j)^{i_j} X^{1-i_j}.$$

With $d + 2$ different values $x_1, \dots, x_{d+2}$ satisfying the equation, we conclude the two polynomials are identical. Looking at the coefficient for X^{d+1} we conclude that the extracted openings of c_0 and c_v satisfy $P(u) = v$. □

4 Membership and Non-membership Arguments

In this section we will construct membership and non-membership arguments for committed values being included in whitelists or excluded from blacklists. In both cases the whitelists or blacklists are given as a set $\mathcal{L} \subset \mathbb{Z}_p$, and the goal is to show that the committed value $u \in \mathcal{L}$ in the case of membership or $u \notin \mathcal{L}$ in the case of non-membership.

Let us first describe a non-membership argument for a committed value not belonging to a set $\mathcal{L} = \{\lambda_1, \dots, \lambda_D\}$ using ideas from Brands et al. [4]. We define a polynomial $P(X) = \prod_{i=1}^{D}(X - \lambda_i)$ with the elements in the set as roots. With this choice of polynomial we have $u \in \mathcal{L}$ if and only if $P(u) = 0$. The prover has a commitment c and will demonstrate that the committed value u does not belong to $\mathcal{L}$ by showing $P(u) \neq 0$.

The prover computes $v = P(u)$ and makes a commitment to v. She can now give an SHVZK argument that the new commitment contains $v = P(u)$ using the polynomial evaluation argument from Section 3. To prove non-membership she just needs to prove $v \neq 0$. To do this the prover commits to $w = v^{-1}$ and uses a multiplication argument to show $v \cdot w = 1$, which will convince the verifier that $v \neq 0$. The main cost in this argument is the polynomial evaluation argument; multiplication arguments are standard cryptographic tools [10] that only cost a couple of group elements in communication.

Common reference string: $ck \leftarrow \mathcal{G}(1^k)$
Statement: $\mathcal{L} = \{\lambda_1, \dots, \lambda_D\} \subset \mathbb{Z}_p, P(X) = \prod_{i=1}^{D}(X - \lambda_i) \in \mathbb{Z}_p[X], c \in \mathbb{G}$
Prover's witness: $u, r \in \mathbb{Z}_p$ such that $c = \mathrm{com}_{ck}(u; r)$ and $u \notin \mathcal{L}$
Argument: Pick $s, t \leftarrow \mathbb{Z}_p$, compute $v = P(u), w = v^{-1}$ and $c_v = \mathrm{com}_{ck}(v; s), c_w = \mathrm{com}_{ck}(w; t)$, and send c_v, c_w to the verifier. Engage in parallel in an SHVZK multiplication argument [10] to show $v \cdot w = 1$ and in the SHVZK polynomial evaluation argument from Section 3 to show $P(u) = v$.
Verification: The verifier accepts $u \notin \mathcal{L}$ if and only if $c_v, c_w \in \mathbb{G}$ and the two SHVZK arguments are valid.

Theorem 7. *If the discrete logarithm assumption holds, the above protocol is a public coin SHVZK argument of knowledge of an opening of c to $u \notin \mathcal{L}$.*

Proof. Perfect completeness follows from the perfect completeness of the two SHVZK arguments.

The SHVZK simulator picks $c_v, c_w \leftarrow \mathbb{G}$ at random and runs the SHVZK simulators for the two underlying SHVZK arguments. Since the commitment scheme is perfectly hiding and the underlying SHVZK arguments are perfect SHVZK this gives us perfect SHVZK.

The protocol has witness-extended emulation. The emulator $\mathcal{X}$ runs the witness-extended emulator for the two underlying SHVZK arguments to get openings u, v, w of the commitments satisfying $v \cdot w = 1$ and $P(u) = v$. The first equality tells us $v \neq 0$

and the second equality then tells us $P(u) \neq 0$. This means u is a not a root of the polynomial $P(X) = \prod_{i=1}^{D}(X - \lambda_i)$ and therefore $u \notin \mathcal{L}$. □

It is easy to modify the non-membership argument into a membership argument. If $u \in \mathcal{L}$ then $P(u) = 0$. We therefore get a membership argument by removing c_w and instead of a multiplication argument letting the prover give an SHVZK argument for c_v containing $v = 0$. Arguments of knowledge of an opening of a commitment to 0 are standard and only cost a couple of group elements in communication [33].

Efficiency. The coefficients of the polynomial $P(X) = \prod_{i}^{D}(X - \lambda_i)$ can be computed in a binary tree fashion with the linear functions $X - \lambda_i$ as leaves. Fast polynomial multiplication techniques that rely on the Fast Fourier Transform can if p is an FFT friendly prime multiply two degree n polynomials using $O(n \log n)$ multiplications in $\mathbb{Z}_p$. This means the prover and the verifier can compute the coefficients of $P(X)$ using $O(D \log^2 D)$ multiplications in $\mathbb{Z}_p$. If the list stays fixed, the computation of the polynomials coefficients is a one-time cost. Single element additions or deletions can be done using D multiplications. If multiple elements are added or deleted at the same time the per element cost can be reduced by using fast polynomial multiplication and division techniques.

Once the coefficients of $P(X)$ are given, the main cost of the membership and non-membership arguments are dominated by the underlying polynomial evaluation argument. For moderate D the computation is dominated by the logarithmic number of exponentiations involved in the polynomial evaluation argument. For large D the computational cost of computing the coefficients of the polynomial matters more as do the multiplications in the polynomial evaluation argument.

5 Comparison and Implementation

The first approaches to prove for committed $u, v \in \mathbb{Z}_p$ that $p(u) = v$ for a given polynomial $P(X)$ with order D split in two parts: first construct commitments $c_1, \ldots, c_D$ to values $u, u^2 \ldots, u^D$ and then use the homomorphic property of the commitment scheme to get $P(u)$ as a linear combination of $u, u^2, \ldots, u^D$. This requires D commitments and D multiplication arguments to show that the commitments $c_1, \ldots, c_D$ have been correctly constructed and contain the correct powers of u. The cost can reduced to $O(\sqrt{D})$ as suggested in Brands et al. [4] by splitting the polynomial in $\sqrt{D}$ polynomials of degree $\sqrt{D}$ each.

Our protocol also has a two part structure but only needs $\log D$ commitments $c_1, \ldots, c_d$ and $\log D$ multiplication arguments to prove they have been correctly formed and contain $u^2, u^4, u^8, u^{16}, \ldots, u^{2^d}$. By using a sophisticated combination of these values in combination with the homomorphic properties of the commitment scheme, we then get the desired argument for $v = p(u)$. This reduces our communication to $O(\log D)$ group elements.

Table 1 gives the asymptotic communication and computation costs of polynomial evaluation arguments based on the discrete logarithm assumption. The polynomial evaluation argument from Brands et al. [4] achieves the best complexity, so we will in the

Table 1. Comparison of our polynomial argument with former work

SHVZK argument	Rounds	Time $\mathcal{P}$ Expos	Time $\mathcal{V}$ Expos	Size Elements
Fujisaki and Okamoto [14]	3	$O(D)$	$O(D)$	$O(D)\ \mathbb{G} + O(D)\ \mathbb{Z}_q$
Brands et al. [4]	3	$O(\sqrt{D})$	$O(\sqrt{D})$	$O(\sqrt{D})\ \mathbb{G} + O(\sqrt{D})\ \mathbb{Z}_q$
Groth [15]	7	$O(D)$	$O(\sqrt{D})$	$\sqrt{D}\ \mathbb{G} + \sqrt{D}\ \mathbb{Z}_q$
This paper	3	$O(\log D)$	$O(\log D)$	$O(\log D)\ \mathbb{G} + O(\log D)\ \mathbb{Z}_q$

Table 2. Detailed comparison of our blacklist argument with Brands et al. [4] argument

SHVZK argument	Rounds	Time $\mathcal{P}$ Expos	Time $\mathcal{P}$ Multip.	Time $\mathcal{V}$ Expos	Time $\mathcal{V}$ Multip.	Size Elements
[4]	3	$8\sqrt{D}$	$2D + 8\sqrt{D}$	$7\sqrt{D}$	$D + 4\sqrt{D}$	$4\sqrt{D}\ \mathbb{G} + 3\sqrt{D}\ \mathbb{Z}_q$
This paper	3	$8\log D$	$2D\log D$	$7\log D$	$2D$	$4\log D\ \mathbb{G} + 3\log D\mathbb{Z}_q$

following give a more detailed theoretical and practical comparison. In Table 2 we give a more detailed theoretical analysis that also counts the number of multiplications.

Based on Table 2 we would expect our verifier to run faster than Brands et. al.'s as our asymptotic computation cost is much smaller and we expect our argument size to be much smaller. Just looking at the numbers of exponentiations needed by the prover can be a little deceptive though since in our polynomial evaluation argument we need $O(D\log D)$ multiplications in $\mathbb{Z}_p$ to compute the δ_j and for large D this cost becomes dominant. Our performance gain for the prover is therefore largest in the range, where D is large enough for $\log D$ to be significantly smaller than $\sqrt{D}$ yet not so large that the cost of $D\log D$ multiplications in $\mathbb{Z}_p$ becomes dominant.

We implemented our polynomial evaluation argument and Brands et al.'s argument in C++ with the NTL library to obtain experimental confirmation of our theoretical analysis and to get a real life comparison based on similar implementation techniques.

For the comparison of the polynomial evaluation arguments we have used a 256-bit subgroup modulo a 1536-bit prime, assumed that the polynomial $P(X)$ is pre-computed and obtained the running time for polynomials with degree between 10 and $500,000$ elements. The performance measurements have been obtained on a MacBook Pro with a 2.54 GHz Intel Core 2 Duo CPU, 4 GB RAM running Mac OS X 10.8.6; all code is single threaded and optimized code using the multi exponentiation techniques by [24]. The results can be found in Table 3.

As expected our verifier runs faster than Brands et al.'s verifier and we also see that our communication compares very favorably against Brands et al.'s communication. For moderate size D it is also the case that our prover is the most efficient, however, for very large D the cost to calculate the δ_js becomes dominant for our prover and here Brands et al.'s prover is faster. Other experiments we have conducted show that for larger security parameters the crossover happens for even larger D and for all reasonable degrees of the polynomial our argument is faster.

Table 3. Our polynomial evaluation argument compared to Brands et al. [4]. All experiments used a 256-bit subgroup modulo a 1536-bit prime and a MacBook Pro, 2.54 CPU, 4 GB RAM.

Elements in list D	Prover Brands et al.	Prover This paper	Verifier Brands et al.	Verifier This paper	Communication Brands et al.	Communication This paper
10	21 ms	13 ms	24 ms	17 ms	12 KB	8 KB
100	66 ms	24 ms	69 ms	30 ms	37 KB	15 KB
1000	227 ms	41 ms	234 ms	45 ms	128 KB	21 KB
10000	747 ms	182 ms	759 ms	81 ms	406 KB	29 KB
100000	2386 ms	1420ms	2402 ms	217 ms	1295 KB	35 KB
1000000	8650 ms	15512 ms	8165 ms	1315 ms	4161 KB	41 KB

The computation cost of the non-membership argument by Brands et al. is smaller than the cost of their polynomial argument. It still requires very large degree D also for the non-membership argument of Brands et al. to become better from a computational perspective though; for moderate size D we have a clear performance advantage.

References

1. Ateniese, G., Song, D., Tsudik, G.: Quasi-efficient revocation in group signatures. In: Blaze, M. (ed.) FC 2002. LNCS, vol. 2357, pp. 183–197. Springer, Heidelberg (2003)
2. Benaloh, J.C., de Mare, M.: One-way accumulators: A decentralized alternative to digital signatures. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 274–285. Springer, Heidelberg (1994)
3. Boneh, D., Shacham, H.: Group signatures with verifier-local revocation. In: ACM Conference on Computer and Communications Security, pp. 168–177 (2004)
4. Brands, S., Demuynck, L., De Decker, B.: A practical system for globally revoking the unlinkable pseudonyms of unknown users. In: Pieprzyk, J., Ghodosi, H., Dawson, E. (eds.) ACISP 2007. LNCS, vol. 4586, pp. 400–415. Springer, Heidelberg (2007)
5. Bresson, E., Stern, J.: Efficient revocation in group signatures. In: Kim, K.-C. (ed.) PKC 2001. LNCS, vol. 1992, pp. 190–206. Springer, Heidelberg (2001)
6. Camacho, P., Hevia, A., Kiwi, M., Opazo, R.: Strong accumulators from collision-resistant hashing. Int. J. Inf. Sec. 11(5), 349–363 (2012)
7. Camenisch, J.L., Chaabouni, R., Shelat, A.: Efficient protocols for set membership and range proofs. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 234–252. Springer, Heidelberg (2008)
8. Camenisch, J., Kohlweiss, M., Soriente, C.: An accumulator based on bilinear maps and efficient revocation for anonymous credentials. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 481–500. Springer, Heidelberg (2009)
9. Camenisch, J., Lysyanskaya, A.: Dynamic accumulators and application to efficient revocation of anonymous credentials. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 61–76. Springer, Heidelberg (2002)
10. Chaum, D., Pedersen, T.P.: Wallet databases with observers. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 89–105. Springer, Heidelberg (1993)
11. Cramer, R., Damgård, I.B.: Zero-knowledge proofs for finite field arithmetic or: Can zero-knowledge be for free? In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 424–441. Springer, Heidelberg (1998)

12. Cramer, R., Damgård, I.B., Schoenmakers, B.: Proof of partial knowledge and simplified design of witness hiding protocols. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 174–187. Springer, Heidelberg (1994)
13. Dingledine, R.: Tor and circumvention: Lessons learned. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 485–486. Springer, Heidelberg (2011)
14. Fujisaki, E., Okamoto, T.: Statistical zero knowledge protocols to prove modular polynomial relations. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 16–30. Springer, Heidelberg (1997)
15. Groth, J.: Linear algebra with sub-linear zero-knowledge arguments. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 192–208. Springer, Heidelberg (2009)
16. Groth, J.: Efficient zero-knowledge arguments from two-tiered homomorphic commitments. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 431–448. Springer, Heidelberg (2011)
17. Groth, J., Ishai, Y.: Sub-linear zero-knowledge argument for correctness of a shuffle. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 379–396. Springer, Heidelberg (2008)
18. Henry, R., Goldberg, I.: Extending nymble-like systems. In: IEEE Symposium on Security and Privacy, pp. 523–537 (2011)
19. Henry, R., Henry, K., Goldberg, I.: Making a nymbler nymble using verbs. In: Atallah, M.J., Hopper, N.J. (eds.) PETS 2010. LNCS, vol. 6205, pp. 111–129. Springer, Heidelberg (2010)
20. Johnson, P., Kapadia, A., Tsang, P., Smith, S.: Nymble: Anonymous ip-address blocking. In: Borisov, N., Golle, P. (eds.) PET 2007. LNCS, vol. 4776, pp. 113–133. Springer, Heidelberg (2007)
21. Kilian, J.: A note on efficient zero-knowledge proofs and arguments. In: STOC, pp. 723–732 (1992)
22. Li, J., Li, N., Xue, R.: Universal accumulators with efficient nonmembership proofs. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 253–269. Springer, Heidelberg (2007)
23. Libert, B., Peters, T., Yung, M.: Scalable group signatures with revocation. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 609–627. Springer, Heidelberg (2012)
24. Lim, C.: Efficient multi-exponentiation and application to batch verification of digital signatures (2000), `http://dasan.sejong.ac.kr/~chlim/pub/multiexp.ps`
25. Lindell, Y.: Parallel coin-tossing and constant-round secure two-party computation. J. Cryptology 16(3), 143–184 (2003)
26. Lofgren, P., Hopper, N.: Bnymble: More anonymous blacklisting at almost no cost (a short paper). In: Danezis, G. (ed.) FC 2011. LNCS, vol. 7035, pp. 268–275. Springer, Heidelberg (2012)
27. Nakanishi, T., Funabiki, N.: A short verifier-local revocation group signature scheme with backward unlinkability. In: Yoshiura, H., Sakurai, K., Rannenberg, K., Murayama, Y., Kawamura, S.-i. (eds.) IWSEC 2006. LNCS, vol. 4266, pp. 17–32. Springer, Heidelberg (2006)
28. Naor, D., Naor, M., Lotspiech, J.: Revocation and tracing schemes for stateless receivers. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 41–62. Springer, Heidelberg (2001)
29. Nguyen, L.: Accumulators from bilinear pairings and applications. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 275–292. Springer, Heidelberg (2005)
30. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992)
31. Peng, K.: A general, flexible and efficient proof of inclusion and exclusion. In: Kiayias, A. (ed.) CT-RSA 2011. LNCS, vol. 6558, pp. 33–48. Springer, Heidelberg (2011)
32. Peng, K., Bao, F.: Improving applicability, efficiency and security of non-membership proof. In: International Symposium on Data, Privacy, and E-Commerce, pp. 39–44 (2010)

33. Schnorr, C.: Efficient signature generation by smart cards. J. Cryptology 4(3), 161–174 (1991)
34. Song, D.: Practical forward secure group signature schemes. In: ACM Conference on Computer and Communications Security, pp. 225–234 (2001)
35. Tor. The tor project Inc., `https://www.torproject.org/`
36. Tsang, P., Kapadia, A., Cornelius, C., Smith, S.: Nymble: Blocking misbehaving users in anonymizing networks. IEEE Trans. Dependable Sec. Comput. 8(2), 256–269 (2011)
37. Tsudik, G., Xu, S.: Accumulating composites and improved group signing. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 269–286. Springer, Heidelberg (2003)
38. Wang, P., Wang, H., Pieprzyk, J.: A new dynamic accumulator for batch updates. In: Qing, S., Imai, H., Wang, G. (eds.) ICICS 2007. LNCS, vol. 4861, pp. 98–112. Springer, Heidelberg (2007)
39. Yu, K.Y., Yuen, T.H., Chow, S.S.M., Yiu, S.M., Hui, L.C.K.: Pe(ar)2: Privacy-enhanced anonymous authentication with reputation and revocation. In: Foresti, S., Yung, M., Martinelli, F. (eds.) ESORICS 2012. LNCS, vol. 7459, pp. 679–696. Springer, Heidelberg (2012)

Resource-Restricted Indifferentiability

Grégory Demay, Peter Gaži, Martin Hirt, and Ueli Maurer

Department of Computer Science, ETH Zurich, Switzerland
{gregory.demay,peter.gazi,hirt,maurer}@inf.ethz.ch

Abstract. A major general paradigm in cryptography is the following argument: Whatever an adversary could do in the real world, it could just as well do in the ideal world. The standard interpretation of "just as well" is that the translation from the real to the ideal world, usually called a simulator, is achieved by a probabilistic polynomial-time algorithm. This means that a polynomial blow-up of the adversary's time and memory requirements is considered acceptable.

In certain contexts this interpretation of "just as well" is inadequate, for example if the concrete amount of memory used by the adversary is relevant. The example of Ristenpart et al. (Eurocrypt 2011), for which the original indifferentiability notion introduced by Maurer et al. (Eurocrypt 2004) is shown to be insufficient, turns out to be exactly of this type. It requires a fine-grained statement about the adversary's memory capacity, calling for a generalized treatment of indifferentiability where specific resource requirements can be taken into account by modeling them explicitly.

We provide such treatment and employ the new indifferentiability notion to prove lower bounds on the memory required by any simulator in a domain extension construction of a public random function. In particular, for simulators without memory, even domain extension by a single bit turns out to be impossible. Moreover, for the construction of a random oracle from an ideal compression function, memory roughly linear in the length of the longest query is required. This also implies the impossibility of such domain extension in any multi-party setting with potential individual misbehavior by parties (i.e., no central adversary).

1 Introduction

1.1 Simulation-Based Security

The so-called "real world – ideal world" paradigm is underlying all current cryptographic frameworks aiming for composable security statements. Using the language of [MRH04, MR11], the purpose of a protocol is to construct an "ideal" resource (which is secure by definition) from "real" resources assumed to be available. The security of such a construction is then argued by showing that if some misbehaving entity (adversary) deviates from the prescribed protocol in the real world, it cannot achieve anything more than what would also be possible in the ideal world. Since the ideal resource is considered secure by definition, any such action is seen as harmless, thus implying the security of the protocol using the real resources.

The translation of the adversarial actions from the real world to the ideal world is described by exhibiting an algorithm performing it, called a *simulator*. The above argument for the real construction being as secure as the ideal resource is then valid as long as we assume that the adversary can, in addition to executing its attack, also translate it into the ideal-world setting by performing the job of the simulator itself. Simulators are typically modeled as probabilistic polynomial-time (PPT) Turing machines, which implies also polynomial memory (the range of the tape that can be accessed within this time) and randomness limitations. This potentially leads to a polynomial blow-up of the attack's resource requirements when translated from the real to the ideal world.

The implicit step of considering this overhead acceptable is hard-coded into most of the existing frameworks, such as the universal composability [Can01], indifferentiability [MRH04] and reactive simulatability [BPW04]. It is appropriate in most natural settings and hence the results in the above-mentioned frameworks have a wide scope of applicability. However, there are practical settings where this rough approach is not sufficient and a more fine-grained analysis is needed. One such scenario was recently exhibited in [RSS11] in the context of indifferentiability, considering the setting of auditable storage. Before we introduce our contributions, let us briefly review both the indifferentiability notion and the example from [RSS11].

1.2 The Case of Indifferentiability

Indifferentiability was introduced in [MRH04] as a generalization of indistinguishability for settings where some access to the internal state of the considered resources is available publicly, within reach of any potential attacker. The framework comes with a composition theorem loosely interpreted as saying that an ideal resource can be replaced by an indifferentiable construction in any context.

The indifferentiability framework found its most important application in the analysis of hash function constructions [CDMP05]. Many existing cryptographic constructions are proven secure in the random oracle model [BR93], but once we instantiate the random oracle (RO) by an existing cryptographic hash function H, such a proof can be seen at most as a heuristic argument towards the security of the construction [CGH98]. However, if one uses a hash function construction H^f that was proven indifferentiable from a RO when using an ideal compression function f, this excludes any possible attacks exploiting the structure of H and reduces the security of the construction to the security of the underlying compression function f, a more compact object that is simpler to analyze. As a consequence, an indifferentiability proof in the setting with an ideal compression function is generally considered an important argument towards the security of a practical hash function design and many of the SHA-3 candidates (including the winner Keccak [BDPVA08a]) enjoy such a proof (see e.g. [BDPVA08b, CN08, DRRS09, DRS09, AMP10]).

Storage-Auditing Scenario from [RSS11]. Re-examining the guarantees provided by indifferentiability, in [RSS11] the authors present an example of a two-party protocol for storage verification. Its goal is to allow the first party (the user) to verify that the second party (the server – e.g. a storage service) is properly storing a certain piece of data that the user has provided earlier. The protocol is using a hash function and as long as it is modeled as a RO, it is clearly impossible for a malicious server to pass the verification without actually storing the user's data. However, as observed in [RSS11] this is no longer true if the RO is replaced by a particular iterative construction with an underlying ideal compression function, even though this construction is known to be indifferentiable from a RO. This puts in question the meaning of an indifferentiability proof as a security argument relevant in all possible contexts.

The best way to understand this example is to consider in greater detail the memory requirements of the simulator used in the indifferentiability proof in question. The simulator is modeled as a PPT algorithm, guaranteeing that the real implementation is at least as good as the ideal RO as long as the attacker is capable of performing the tasks modeled by the simulator, in particular has polynomial amount of memory available. However, this is an unacceptable assumption if we want to investigate whether the server can pass the verification procedure *without* allocating all the memory required to store the user's message. As a side contribution, we give a more detailed explanation of this problem in the full version of our paper.

1.3 Contributions of This Paper

Our contributions are three-fold. First, we introduce a new formalism based on abstract cryptography (AC, [MR11]), allowing a fine-grained modelling of resource requirements, necessary to capture problems such as the one described above. Second, we apply this new formalism to the problem of domain extension of public random functions and prove lower bounds on the memory needed by any simulator in this type of constructions. And finally, we investigate the consequences of these bounds for settings with multiple parties that may potentially deviate from the prescribed behavior in an uncoordinated manner. We proceed by a detailed description of all three parts.

Memory-Aware Reducibility. In Section 3 we introduce the notion of *memory-aware reducibility* that is derived from reducibility[1] in the classical indifferentiability setting as given in [MRH04, MR11], but does not allow the memory requirements of the simulator to be "swept under the rug", requiring only that they are polynomial. In accordance with the spirit of the AC framework that is used to formalize it, our notion requires any memory necessary for the simulator to be explicitly modeled as a part of the ideal resource; with the intuitive meaning that the real construction is provably as good as the ideal resource as long

[1] The term "reducibility" is used in [MRH04] and, for consistency, also throughout this paper. It is to be understood in the same sense as the term "construction" used above, but the viewpoint is reversed. To construct S from R means the same as to reduce (the need for) S to (the need for) R.

as we assume that the adversary has the necessary amount of memory available. We also give a composition theorem for our new notion.

An independent approach to analyzing the complexity of the simulator in an indifferentiability statement appeared recently in [DRST12], where the authors focus on the number of queries the simulator issues per invocation. To the best of our knowledge, our work is the first one pointing out the importance of the simulator's memory requirements. However, we stress that the applicability of our approach goes beyond modeling memory, extending also to other resources such as computational power or randomness, would the investigated setting require it.

Simulator Memory for Domain Extension. In Section 4 we look at the most important application of indifferentiability: the question of *domain extension* for public random functions. More precisely, we consider constructions that can be used to obtain an arbitrary input-length RO $\mathbf{R}^{*,n}\colon \{0,1\}^* \to \{0,1\}^n$ from an ideal compression function $\mathbf{R}^{m,r}\colon \{0,1\}^m \to \{0,1\}^r$ in an indifferentiable way, such as the various variants of the Merkle-Damgård construction proposed in [CDMP05]. We also consider the question of finite domain extension, i.e., constructing $\mathbf{R}^{\ell,r}$ from $\mathbf{R}^{m,r}$ for $\ell > m$.

The formalism of memory-aware reducibility allows us to investigate the minimal necessary memory requirements of the simulator for *any* such domain-extension construction. We prove two lower bounds on the memory required by the simulator, with the following consequences (see Section 4 for the precise bounds):

1. With stateless simulators (i.e., without any memory) even domain extension by a single bit (i.e., $\ell = m + 1$) is impossible.
2. For the class of simulators issuing at most one query to the ideal resource per invocation, any simulator for a domain extension by d bits (i.e., $\ell - m = d$) requires at least d bits of memory.

These bounds hold for both the information-theoretic and the computational setting. They naturally imply analogous impossibility results for constructing an arbitrary input-length RO, with the obvious transition of ℓ denoting the length of the longest query issued to the RO. This answers negatively the open question of the existence of such a construction using no simulator memory asked in [RSS11]. As another consequence, we also obtain the irreducibility of the RO to the ideal cipher with respect to stateless simulators, in contrast to the equivalence of these two ideal primitives with respect to classical indifferentiability [CDMP05, CPS08, HKT11].

Random Oracles Used by Multiple Parties. The impossibility results described above have some intriguing consequences for the setting where a RO is being used in a protocol by multiple parties, if we consider that several of these parties might deviate from the prescribed protocol in a potentially non-coordinated way (for example due to conflicting goals). According to the AC framework, a security notion for such a situation has to involve local simulators for each of the parties that deviate from the protocol. Clearly, if a distinguisher is allowed to access two such simulators (for two of the parties) in the ideal world, these have to be essentially stateless as otherwise they would produce inconsistent results when brought to different states. On the other hand, our results described above imply that also for this setting, no stateless simulator can exist. Hence, roughly speaking, for settings where one cannot assume a central adversary coordinating all the actions of the misbehaving parties, no secure construction of a RO from an ideal compression function exists. This might be relevant in the contexts of *rational cryptography* [HT04], *incoercible computation* [CG96], *receipt-free voting* [BT94] or *collusion-free computation* [LMs05, AKL+09] and its recent composable variants [AKMZ12, CV12]. We formalize the above argument in Section 5 as an illustration of the impact of our results.

2 Preliminaries

Basic Notation. We denote sets by calligraphic letters or capital greek letters (e.g. $\mathcal{X}, \Sigma$) and their cardinalities by $|\mathcal{X}|, |\Sigma|$. For a superset clear from the context, we denote the complement of a set $\mathcal{X}$ by $\overline{\mathcal{X}}$. Throughout the paper all logarithms considered are to the base 2. The notation $\lceil\cdot\rceil$ corresponds to the usual ceiling function.

We denote random variables and concrete values they can take on by upper-case letters $X, Y, \ldots$ and lower-case letters $x, y, \ldots$, respectively. For random variables U and V with ranges $\mathcal{U}$ and $\mathcal{V}$, respectively, we let $\mathsf{P}_{U|V}$ be the corresponding conditional probability distribution, seen as a (partial) function $\mathcal{U} \times \mathcal{V} \to [0,1]$. Here the value $\mathsf{P}_{U|V}(u,v) = \mathsf{P}[U = u|V = v]$ is well defined for all $u \in \mathcal{U}$ and $v \in \mathcal{V}$ such that $\mathsf{P}_V(v) > 0$ and undefined otherwise. For a discrete random variable X with range $\mathcal{X}$ we denote by $H(X)$ the Shannon entropy of X, i.e., $H(X) = \sum_{x\in\mathcal{X}} -\mathsf{P}_X(x)\log\mathsf{P}_X(x)$ where $\mathsf{P}_X(x)$ denotes the probability that X takes on the value $x \in \mathcal{X}$. Moreover, we denote by $H(Y|X)$ the usual notion of conditional entropy of Y given X, satisfying the chain rule $H(Y|X) = H(XY) - H(X)$. For a probability $p \in [0,1]$ we also use the notion of binary entropy denoted $h(p)$ that is defined as the Shannon entropy of the binary random variable taking on the two possible values with probabilities p and $1-p$.

Resources, Converters and Distinguishers. To formulate our results we use the language of abstract systems [MR11, Mau11] to which we give here a self-contained introduction, partly following the exposition given in [MRT12]. At the highest level of abstraction, a system is an object with interfaces via which it interacts with its environment (consisting of other systems). Two systems can be composed by connecting one interface of each system, and the composed object is again a system. Also, every two different systems are mutually independent.

We consider three distinct types of systems: *resources*, *converters* and *distinguishers*. Resources[2] are denoted by upper-case boldface letters such as $\mathbf{S}, \mathbf{T}$.

[2] In this paper we sometimes also use the term “resources” in a more informal way to refer to computational power, memory, etc. This should cause no confusion, since these resources could also be formalized in the sense of the notion introduced above.

In this paper we mostly (but not always) consider resources with two interfaces, hence our exposition here will only cover this case. In the indifferentiability setting these interfaces are referred to as private and public (for reasons explained below). Examples of resources discussed below are a fixed input-length random oracle with input length m and output length r denoted $\mathbf{R}^{m,r}$; an arbitrary input-length random oracle with output length n denoted $\mathbf{R}^{*,n}$; and an ideal block cipher with key length k and block length n denoted $\mathbf{E}^{k,n}$. Unless indicated otherwise, we see these as 2-interface resources providing access to the same random function at each interface.

Converters are systems having one *inner* and one *outer* interface and are denoted by small Greek letters such as ϕ, π, σ. The set of all converters considered is denoted as Σ. A converter ϕ can be composed with a resource $\mathbf{S}$ by attaching the inner interface of ϕ to one of the interfaces of $\mathbf{S}$. For example, if ϕ is attached to the private interface of $\mathbf{S}$ this can be depicted as $-\textcircled{\phi}-\boxed{\mathbf{S}}-$. Note that the composed system is again a 2-interface resource that exposes the outer interface of ϕ instead of the interface of $\mathbf{S}$ to which ϕ was connected, together with the other interface of $\mathbf{S}$.

To describe the composition of resources and converters algebraically, we can take advantage of the restriction to 2-interface resources: We will understand the left and the right side of the symbol $\mathbf{S}$ as representing the private and the public interface of the system $\mathbf{S}$, respectively. Hence, attaching a converter π to the left (private) interface of a resource $\mathbf{S}$ results in a resource $\pi\mathbf{S}$ while attaching a converter σ to the right (public) interface of a resource $\mathbf{T}$ results in a resource $\mathbf{T}\sigma$. If two 2-interface resources $\mathbf{S}$ and $\mathbf{T}$ are used in parallel, this is denoted as $\mathbf{S}\|\mathbf{T}$ and is again a 2-interface resource; each of the interfaces of $\mathbf{S}\|\mathbf{T}$ allows to access the corresponding interface of both subsystems $\mathbf{S}$ and $\mathbf{T}$. Two converters ψ and ϕ can also be composed: either sequentially, obtaining a converter $\psi \circ \phi$ such that $(\psi \circ \phi)\mathbf{S} = \psi(\phi\mathbf{S})$; or in parallel, obtaining $\psi|\phi$ such that $(\psi|\phi)(\mathbf{S}\|\mathbf{T}) = (\psi\mathbf{S})\|(\phi\mathbf{T})$. The application of composed converters to the public interface works in an analogous way. The term id refers to the "identity converter" that forwards all inputs and outputs, we always assume $\mathsf{id} \in \Sigma$. For a 2-interface system $\mathbf{S}$ we sometimes denote by $[\mathbf{S}]_L(x)$ (resp. $[\mathbf{S}]_R(x)$) its response to a query x on its left (resp. right) interface.

We instantiate the general concept of abstract systems given above by considering (probabilistic) systems that communicate by passing messages from discrete sets and within discrete time steps. These can be formalized by the notion of random systems [Mau02], i.e., conditional distributions of the outputs of the system (as random variables) given all previous inputs and outputs, where each input or output is associated to a specific interface. Since being sufficient for our setting, we restrict our considerations to resources that only produce output in response to an input and on the same interface where the input was received. For a converter we assume that it is always invoked by a query at the outer interface, it then issues zero or more queries to the resource attached to its inner interface and finally produces an output at the outer interface. Under these assumptions, the behavior of composed systems is determined in the natural way, with the

Fig. 1. The real (left) and the ideal (right) setting considered for reducibility in the context of indifferentiability

parallel composition of two resources defined asynchronously: each input at an interface of $\mathbf{S}\|\mathbf{T}$ explicitly specifies one of the subsystems, and this subsystem is invoked with the input.

A *distinguisher* $\mathbf{D}$ is a system that connects to all interfaces of a resource $\mathbf{T}$ and outputs (at a separate interface) a single bit denoted B. The complete interaction of $\mathbf{D}$ and $\mathbf{T}$ defines a random experiment and the probability that the bit B is 1 in this experiment is written as $\mathsf{P}^{\mathbf{DT}}(B = 1)$. The *distinguishing advantage of* $\mathbf{D}$ *for the systems* $\mathbf{S}$ *and* $\mathbf{T}$ is then defined as $\Delta^{\mathbf{D}}(\mathbf{S},\mathbf{T}) := \left|\mathsf{P}^{\mathbf{DS}}(B = 1) - \mathsf{P}^{\mathbf{DT}}(B = 1)\right|$. We denote by $\mathcal{D}$ the set of all distinguishers considered and define $\Delta^{\mathcal{D}}(\mathbf{S},\mathbf{T}) := \sup_{\mathbf{D}\in\mathcal{D}} \Delta^{\mathbf{D}}(\mathbf{S},\mathbf{T})$.

Classical (weak) Indifferentiability. In the classical indifferentiability defined in [MRH04] one restricts only to resources having two interfaces. The first one, referred to as *private*, is meant to model the access to the resource by all honest parties. On the other hand, the second interface is called *public* and is present to model the adversarial access to the internal state of the resource.

Let $\mathbf{S}$ and $\mathbf{T}$ be such 2-interface resources. For given sets Σ and $\mathcal{D}$ of converters and distinguishers, respectively, we define $\mathbf{T}$ *being* ε*-reducible to* $\mathbf{S}$ *in the sense of weak indifferentiability* (denoted $\mathbf{S}\xrightarrow[\text{wi}]{\varepsilon}\mathbf{T}$) as

$$\mathbf{S}\xrightarrow[\text{wi}]{\varepsilon}\mathbf{T} \quad :\Leftrightarrow \quad (\exists\pi\in\Sigma)(\forall\mathbf{D}\in\mathcal{D})(\exists\sigma\in\Sigma) : \Delta^{\mathbf{D}}(\pi\mathbf{S},\mathbf{T}\sigma) \le \varepsilon$$

and refer to the converters π and σ as the protocol and the simulator, respectively. Usually we call $\mathbf{S}$ the real and $\mathbf{T}$ the ideal resource; hence also the random experiment of $\mathbf{D}$ interacting with $\pi\mathbf{S}$ (resp. $\mathbf{T}\sigma$) is called the real (resp. ideal) experiment. The two settings distinguished are depicted in Fig. 1.

Note that by choosing the sets Σ and $\mathcal{D}$, this definition covers both information-theoretic and computational indifferentiability; moreover, one could also easily derive an asymptotic definition. These remarks are also true for all other reducibility notions presented below.

Strong Indifferentiability. For given sets Σ and $\mathcal{D}$ we define $\mathbf{T}$ *being* ε*-reducible to* $\mathbf{S}$ *in the sense of strong indifferentiability* (denoted $\mathbf{S}\xrightarrow[\text{si}]{\varepsilon}\mathbf{T}$) as

$$\mathbf{S}\xrightarrow[\text{si}]{\varepsilon}\mathbf{T} \quad :\Leftrightarrow \quad (\exists\pi,\sigma\in\Sigma)(\forall\mathbf{D}\in\mathcal{D}) : \Delta^{\mathbf{D}}(\pi\mathbf{S},\mathbf{T}\sigma) \le \varepsilon.$$

Clearly reducibility under strong indifferentiability implies reducibility under the weak one and moreover, positive indifferentiability results (such as those in [CDMP05] showing security of MD-variants) typically prove this stronger type of statement by exhibiting a simulator that does not depend on the distinguisher. A detailed discussion of the relationship between these two forms of simulatability in various formalisms can be found in [HU05, Can01], here we only remark that both notions are composable in the spirit of Theorem 1 (see below).

Domain Extension for Hash Functions. Finally, we briefly introduce the domain extension construction chop-MD from [CDMP05] that will serve us as a useful example throughout the paper. Let $f\colon \{0,1\}^{r+d} \to \{0,1\}^r$ be a compression function. The function $\mathsf{chop\text{-}MD}^f\colon \{0,1\}^* \to \{0,1\}^{r/2}$ is as defined in the box above. The role of the function Pad is to append the length of the message and a padding in a decodable way to obtain m' with length being a multiple of d bits. It will not be relevant for our discussion.

```
function chop-MD^f(m)
    $m' \leftarrow \mathsf{Pad}(m)$
    parse $m'$ as $m_1\|\cdots\|m_b$ for $m_i \in \{0,1\}^d$
    $y_0 \leftarrow 0^r$ (or any fixed initialization vector)
    for $i = 1$ to $b$ do $y_i \leftarrow f(m_i\|y_{i-1})$
    return first $r/2$ bits of $y_b$
```

3 Memory-Aware Reducibility

Stateless simulators. To formally define memory-aware reducibility, we need to consider the class of stateless converters in the following sense. A stateless converter *uses no memory between answering outer queries*, i.e., its behavior for a particular query depends only on the query itself and the ongoing interaction at the inner interface, not on previous outer queries and the transcript of the interaction during their evaluation. However, it might of course be randomized, using fresh randomness at every invocation. This is captured by the following formal definition.

Definition 1. *A converter ϕ is* stateless *if there exists a sequence of conditional probability distributions $\mathsf{p}^{\phi}_{IX_{j+1}|X_1\ldots X_jY_1\ldots Y_jQ}$ for $j \geq 0$ such that whenever ϕ received a query q at the outer interface and has then issued the sequence of queries $x_1,\ldots,x_j$ to the inner interface, obtaining responses $y_1,\ldots,y_j$, then $\mathsf{p}^{\phi}_{IX_{j+1}|X_1\ldots X_jY_1\ldots Y_jQ}(i,x_{j+1},x_1,\ldots,x_j,y_1,\ldots,y_j,q)$ determines the probability that its next action will be to output the value x_{j+1} at interface $i \in \{\mathsf{inner},\mathsf{outer}\}$. For a set of converters Σ we denote by Σ_{sl} the set of all stateless converters from Σ.*

For example, the converter accessing an ideal compression function and realizing a Merkle-Damgård construction on top of it would be stateless according to the above definition.

Quantifying the Memory Requirements of the Simulator. Let $\mathbf{M}_s$ denote a resource that provides a dummy private interface and at the public (adversarial) interface, it provides the functionality of s-bit memory, i.e., allows efficient storage and retrieval of arbitrary information such that its size is in every point in time upper-bounded by s bits. To quantify the memory requirements of the simulator in a reducibility statement we shall require it to be stateless and only use the memory provided by the ideal resource, leading to the following formalism (broadly denoted as *memory-aware reducibility*).

Definition 2. *For given sets Σ and $\mathcal{D}$ of converters and distinguishers, respectively, we define* $\mathbf{T}$ being ε-reducible to $\mathbf{S}$ in the presence of s bits of adversarial memory *(denoted* $\mathbf{S}\xrightarrow[\mathrm{m}]{\varepsilon,s}\mathbf{T}$*) as*

$$\mathbf{S}\xrightarrow[\mathrm{m}]{\varepsilon,s}\mathbf{T} \quad :\Leftrightarrow \quad (\exists\pi\in\Sigma)(\forall\mathbf{D}\in\mathcal{D})(\exists\sigma\in\Sigma_{\mathsf{sl}}) : \Delta^{\mathbf{D}}(\pi\mathbf{S},[\mathbf{T}||\mathbf{M}_s]\sigma)\leq\varepsilon.$$

Informally speaking, the statement $\mathbf{S}\xrightarrow[\mathrm{m}]{\varepsilon,s}\mathbf{T}$ indicates that $\mathbf{T}$ can be constructed securely from $\mathbf{S}$ within error ε in an environment where the adversary has s bits of memory available. In other words, whatever the adversary can achieve in the real world he could also achieve in the ideal world, but it might need up to s more bits of memory to do so. Evaluating whether this is acceptable depends on the context in which we want to use $\mathbf{S}$ instead of $\mathbf{T}$.

As before, by specifying the sets of converters and distinguishers to be considered, this definition covers both computational and information-theoretic memory-aware reducibility; moreover, the transition to an asymptotic definition would be straightforward. Alongside the notion of reducibility, one could also explicitly define the underlying notion of *memory-aware indifferentiability* that would only consider the trivial protocol $\pi = \mathsf{id}$, leading to the same relationship between indifferentiability and the respective reducibility as in the classical case [MRH04]. Since our results make use of the reducibility notion, we omit this step.

In case of no memory (i.e., $s = 0$) the notion of memory-aware reducibility $\mathbf{S}\xrightarrow[\mathrm{m}]{\varepsilon,0}\mathbf{T}$ collapses to the notion of reducibility with stateless simulators. If we refer to this situation, we usually omit the 0 and simply write $\mathbf{S}\xrightarrow[\mathrm{m}]{\varepsilon}\mathbf{T}$. In this special case, the underlying indifferentiability notion is technically equivalent to the notion of reset indifferentiability introduced in [RSS11]: First, if the simulator is stateless, then it can be used also in the scenario with resets with the same outcome. On the other hand, any simulator that satisfies the requirements of reset indifferentiability must be able to simulate successfully even in presence of an adversary that resets it before every query, hence there also exists an equivalent stateless simulator. However, our motivation to introduce stateless simulators is completely different. We do not put them forward to define a security notion by themselves, but only as a tool for modeling the memory requirements of the simulator explicitly.

Composability. The formalism of memory-aware reducibility given above leads to statements that are composable under some natural closure assumptions on the sets Σ and $\mathcal{D}$ of converters and distinguishers considered. Here we only state the respective composition theorem informally.

Theorem 1 (Informal). *Let Σ be closed under both sequential composition $\circ$ and parallel composition $|$ and let $\mathcal{D}$ be closed under the emulation of any converter and of any resource. Let* $\mathbf{S}$, $\mathbf{T}$ *and* $\mathbf{V}$ *be resources such that* $\mathbf{S}\xrightarrow[\mathrm{m}]{\varepsilon_1,s_1}\mathbf{T}$ *and* $\mathbf{T}\xrightarrow[\mathrm{m}]{\varepsilon_2,s_2}\mathbf{V}$*. Then we have:*

1. *For any resource* $\mathbf{U}$, $\mathbf{S}||\mathbf{U}\xrightarrow[\mathrm{m}]{\varepsilon_1,s_1}\mathbf{T}||\mathbf{U}$ *and* $\mathbf{U}||\mathbf{S}\xrightarrow[\mathrm{m}]{\varepsilon_1,s_1}\mathbf{U}||\mathbf{T}$,
2. $\mathbf{S}\xrightarrow[\mathrm{m}]{\varepsilon_1+\varepsilon_2,s_1+s_2}\mathbf{V}$.

4 Lower Bounds on Simulator Memory for Any Domain-Extending Construction

We now investigate the amount of memory that we must assume to be available to the adversary in order to be able to conclude the security of classical domain extension constructions for hash functions.

4.1 Fixed Input-Length Random Oracles

The following theorem upper-bounds the achievable domain extension for fixed input-length random oracles, given a bound on the memory available to the simulator. In particular, it implies that without simulator memory, even domain extension by a single bit becomes impossible, thus solving an open problem introduced in [RSS11].

Theorem 2. *Assume that for any $\pi \in \Sigma$, the distinguisher* $\mathbf{D}$ *constructed from π according to Fig. 2 is present in $\mathcal{D}$. Then any reduction $\mathbf{R}^{m,r} \xrightarrow[\mathrm{m}]{\varepsilon,s} \mathbf{R}^{\ell,r}$ with*[3] *$r \geq 2$ and $\varepsilon \leq 0.04$ satisfies*[4]

$$\ell - m \leq s + \lceil \log(\min\{s,t\}) \rceil \tag{1}$$

where $t \geq 1$ denotes an upper bound on the number of queries the simulator issues to the ideal primitive $\mathbf{R}^{\ell,r}$ to answer a single query.

Proof. Recalling Def. 2, let us denote by π the protocol performing the reduction from the statement and let us consider a distinguisher $\mathbf{D}$ interacting with either $\pi\mathbf{R}^{m,r}$ or $[\mathbf{R}^{\ell,r}\|\mathbf{M}_s]\sigma$, where σ is the stateless simulator corresponding to $\mathbf{D}$.

Proof Overview. We only consider the trivial distinguisher $\mathbf{D}$ given in Fig. 2 that chooses a random input $X \in \{0,1\}^\ell$ and then evaluates the $\{0,1\}^\ell$-domain function on the input X in two different ways. First it queries the private (left) interface for the whole input X; second it simulates the protocol π on X on its own and uses the public (right) interface to answer the $\{0,1\}^m$-queries issued by π. Moreover, it never repeats a query to the right interface: in case the simulated protocol π would issue a repeated query, it is answered as before. We will refer to this modified (simulated) protocol π as π'; note that $\mathbf{D}$ is capable of this modification since it can keep the history of query-answer pairs in its state. Finally, $\mathbf{D}$ outputs 1 if and only if the two values obtained from these evaluations are equal. The distinguisher $\mathbf{D}$ participating in both the real and the ideal setting is depicted in Fig. 3. Note that it is natural to expect that this simple distinguisher $\mathbf{D}$ is present in any reasonable distinguisher class.

[3] The bound degrades gracefully for smaller r and bigger ε. In particular, for the same ε and $r = 1$ with no memory ($s = 0$) domain extension by a single bit is still impossible.

[4] To avoid handling the special case $s = 0$ separately we use the notational convention $\log 0 = 0$ throughout this section.

Distinguisher $\mathbf{D}(\mathbf{S})$: where $\mathbf{S} \in \{\pi\mathbf{R}^{m,r}, [\mathbf{R}^{\ell,r} \| \mathbf{M}_s]\sigma\}$

```
1: $X \xleftarrow{\$} \{0,1\}^\ell$
2: query $Y := [\mathbf{S}]_L(X)$
3: simulate $\pi$ to evaluate $\hat{Y} := \pi(X)$
      answer new inner queries by querying $[\mathbf{S}]_R$
      answer repeated inner queries consistently
4: if $Y = \hat{Y}$ then
5:     return 1
6: return 0
```

Fig. 2. The distinguisher $\mathbf{D}$ for the proof of Theorem 2

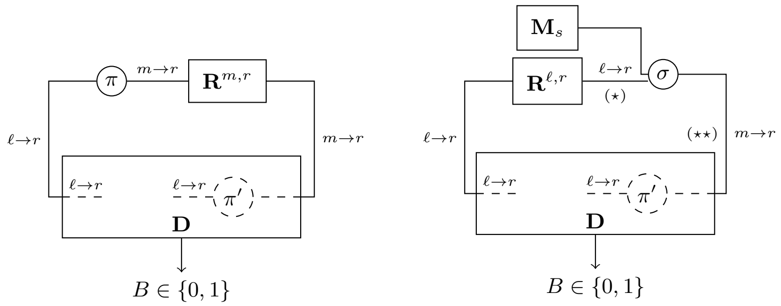


Fig. 3. The real and the ideal setting for the proof of Theorem 2. The notation $i \to o$ describes an interface that accepts queries from $\{0,1\}^i$ and responds with elements from $\{0,1\}^o$.

Clearly if $\mathbf{D}$ interacts with $\pi\mathbf{R}^{m,r}$ it always outputs 1. It remains to analyze the probability of $\mathbf{D}$ outputting 1 when interacting with $[\mathbf{R}^{\ell,r} \| \mathbf{M}_s]\sigma$. To this end, we consider the ideal setting depicted on the right-hand side of Fig. 3 and upper-bound the probability that the output of the protocol π' simulated by $\mathbf{D}$ will be the correct value $\mathbf{R}^{\ell,r}(X)$. Informally speaking, we do this by upper-bounding the amount of useful information that π' can obtain about the actual values of $\mathbf{R}^{\ell,r}$ and show that it is not enough to recover $\mathbf{R}^{\ell,r}(X)$ with sufficient probability.

We use two separate approaches to bound this amount, each proving the above claim for one of the values in the minimum term in (1). In the first approach, we upper-bound the number of distinct queries the simulator σ is able to issue to $\mathbf{R}^{\ell,r}$ in any of its possible configurations (determined by the query it is answering and the state of its memory), thus using the channel denoted $(\star)$ in Fig. 3 as the "bottle-neck" to be considered. On the other hand, in the second approach we upper-bound the information provided by σ to π', this time the channel $(\star\star)$ acting as the "bottle-neck". We now give the details of both approaches.

First approach: the channel $(\star)$. To capture the randomness involved in the ideal distinguishing experiment, we denote by R_w the (fresh, independent) internal randomness used by σ when it is answering an outer query $w \in \{0,1\}^m$ *for the first time*[5] and let $R_\sigma := \{R_w\}_{w\in\{0,1\}^m}$. Moreover, let $R_{\mathbf{R}}$ denote the overall randomness of the ideal resource $\mathbf{R}^{\ell,r}$, i.e., its function table. For a fixed randomness $R_\sigma = r_\sigma$ and $R_{\mathbf{R}} = r_{\mathbf{R}}$ where $r_\sigma = \{r_w\}_{w\in\{0,1\}^m}$, let us denote by $f(w,z,r_w,r_{\mathbf{R}}) \subseteq \{0,1\}^\ell$ the set of all queries that the (stateless) simulator σ issues to the random oracle $\mathbf{R}^{\ell,r}$ while evaluating an outer query w with the available memory $\mathbf{M}_s$ containing value $z \in \{0,1\}^s$, using randomness r_w while the responses from $\mathbf{R}^{\ell,r}$ are determined by $r_{\mathbf{R}}$. Since the random variables R_w and $R_{\mathbf{R}}$ represent the only sources of randomness in this evaluation, f is a well-defined deterministic mapping and by our assumption $|f(w,z,r_w,r_{\mathbf{R}})| \le t$ for all possible inputs. Let us define $\mathcal{S}_{r_\sigma,r_{\mathbf{R}}}$ to be the set of all possible queries under all inputs (w,z) for this fixed randomness $(r_\sigma, r_{\mathbf{R}})$, i.e.,

$$\mathcal{S}_{r_\sigma,r_{\mathbf{R}}} := \bigcup_{\substack{w\in\{0,1\}^m \\ z\in\{0,1\}^s}} f(w,z,r_w,r_{\mathbf{R}}),$$

then we have $|\mathcal{S}_{r_\sigma,r_{\mathbf{R}}}| \le 2^{m+s+\log t}$ for any $(r_\sigma, r_{\mathbf{R}})$. Since $X \in \{0,1\}^\ell$ was chosen at random and independently from $\mathcal{S}_{R_\sigma,R_{\mathbf{R}}}$, we obtain $\mathsf{P}(X \in \mathcal{S}_{R_\sigma,R_{\mathbf{R}}}) \le 2^{m+s+\log t}/2^\ell = 2^{m+s+\log t-\ell}$. Hence, if $\ell - m > s + \lceil \log t \rceil$ then $X \notin \mathcal{S}_{R_\sigma,R_{\mathbf{R}}}$ with probability at least 1/2. However, if $X \notin \mathcal{S}_{R_\sigma,R_{\mathbf{R}}}$ then π' has no information about $\mathbf{R}^{\ell,r}(X)$ and hence can only guess it successfully with negligible probability. Therefore, any proper simulation requires $\ell - m \le s + \lceil \log t \rceil$.

Second approach: the channel ($\star\star$). In this case, let us denote by $\sigma_z(w)$ the response of σ to a query $w \in \{0,1\}^m$ with the available memory set to the value $z \in \{0,1\}^s$ and let us denote by $Z'(w,z) \in \{0,1\}^s$ the new contents of the memory after this invocation of σ. Note that since σ is stateless, both $\sigma_z(w)$ and $Z'(w,z)$ are random variables fully determined by the function table of $\mathbf{R}^{\ell,r}$ and the internal randomness of σ used during this invocation. We can now define T to be the table containing a sample of $\sigma_z(w)$ and $Z'(w,z)$ for all possible w and z, formally $T := \{(\sigma_z(w), Z'(w,z))\}_{(w,z)\in\{0,1\}^m\times\{0,1\}^s}$. Then T can be seen as a random variable distributed over $\{0,1\}^{(r+s)\cdot 2^{m+s}}$ and is again determined by the function table of $\mathbf{R}^{\ell,r}$ and the randomness used by σ.

We now consider a different protocol ρ instead of π' which we allow to be stateful, but we only provide it with access to T, not σ (which we denote by ρ^T). We claim that the probability of the best such ρ in reconstructing $\mathbf{R}^{\ell,r}(X)$ given access to T is not smaller than the same probability for π' given access to the right interface of $[\mathbf{R}^{\ell,r}\|\mathbf{M}_s]\sigma$, i.e., we have

$$\max_\rho \mathsf{P}[\rho^T(X) = \mathbf{R}^{\ell,r}(X)] \ge \mathsf{P}\left[[[\mathbf{R}^{\ell,r}\|\mathbf{M}_s]\sigma\pi']_R(X) = \mathbf{R}^{\ell,r}(X)\right]. \qquad (2)$$

This is because one possible ρ to be considered on the left side of (2) is the following: it simulates π' and answers each of its queries to σ using the respective

[5] Formally, one can imagine σ being replaced by a stateful simulator that chooses all random variables R_w at the beginning and then uses it when the query w arrives for the first time. This view does not change the outcomes of the experiment.

value from T instead (recall that π' asks each query at most once). It also keeps track of the memory contents in its own state, updating it after each answered query according to the value given in T. This ρ clearly achieves equality in (2).

Now, since any ρ as described above only has access to T, we can use a corollary of the well-known Fano's inequality [Fan61] to upper-bound the probability of ρ successfully reconstructing $\mathbf{R}^{\ell,r}(X)$ based on T. To simplify the notation, we shall denote by F the whole function table of $\mathbf{R}^{\ell,r}$ seen as a random variable (uniformly distributed over $\{0,1\}^{r2^\ell}$). The value X is chosen independently at random, hence we can lower-bound the probability $\bar{p}_e$ of error in a randomly chosen bit of $\mathbf{R}^{\ell,r}(X)$ as follows:

$$\begin{aligned} h(\bar{p}_e) &\geq \frac{1}{r2^\ell} H(F|T) = \frac{1}{r2^\ell}\left(H(FT) - H(T)\right) \geq \frac{1}{r2^\ell}\left(H(F) - H(T)\right) \\ &\geq \frac{1}{r2^\ell}\left(r2^\ell - (r+s)2^{m+s}\right) = 1 - 2^{m+s-\ell} - \left(\frac{s}{r}\right)2^{m+s-\ell}. \end{aligned}$$

The first inequality follows from Fano's inequality, see e.g. [CK11, Corollary 3.8] or the full version of this paper. Now if $\ell - m > s + \lceil \log s \rceil$ then since m, s, ℓ are integers we get $2^{m+s-\ell} \leq 1/2$ and $(s/r) \cdot 2^{m+s-\ell} \leq 1/2r$, hence $h(\bar{p}_e) \geq 1/2 - 1/2r$, resulting in $\bar{p}_e \geq 0.04$ for $r \geq 2$. Therefore any simulator successful beyond 96% has to satisfy $\ell - m \leq s + \lceil \log s \rceil$ as desired. □

Before we apply our result also to other contexts, note that our argument above is completely information-theoretic and hence the bound applies to both information-theoretic *and* computational memory-aware reducibility.

4.2 Arbitrary Input-Length Random Oracles

Seen from a different perspective, the above theorem also imposes a lower bound on the required simulator memory for any reduction of an arbitrary input-length random oracle to a fixed input-length random oracle (i.e., an ideal compression function) as a function of the lengths of hashed messages.

In the statement below we shall again consider the distinguisher given in Fig. 2, this time for the setting of the reduction $\mathbf{R}^{m,r} \xrightarrow[\mathrm{m}]{\varepsilon,s} \mathbf{R}^{*,r}$. To emphasize that it chooses the value X from the set $\{0,1\}^\ell \subseteq \{0,1\}^*$, we shall denote it $\mathbf{D}_\ell$, note that it again implicitly depends on a protocol π. One could give a similar statement also for a distinguisher asking several private queries and using the public interface to evaluate the protocol π on the longest one.

Corollary 1. *If for every $\pi \in \Sigma$ the distinguisher $\mathbf{D}_\ell$ described above is present in $\mathcal{D}$ then any reduction $\mathbf{R}^{m,r} \xrightarrow[\mathrm{m}]{\varepsilon,s} \mathbf{R}^{*,r}$ with $r \geq 2$ and $\varepsilon \leq 0.04$ satisfies*

$$s \geq \ell - m - \lceil \log(\min\{s,t\}) \rceil$$

where $t \geq 1$ denotes an upper bound on the number of queries the simulator itself issues to the ideal primitive to answer a single query. For the more general case $\mathbf{R}^{m,r} \xrightarrow[\mathrm{m}]{\varepsilon,s} \mathbf{R}^{*,n}$ we still have $s \geq \ell - m - \lceil \log t \rceil$ under the same assumptions.

The proof is analogous to the proof of Theorem 2 and we omit it. To illustrate the meaning of the above statement, let us consider the domain extension construction chop-MD described in Section 2. The simulator presented in [CDMP05] to show its indifferentiability from a random oracle would use (without optimizations) roughly $(1 + r/m) \cdot \ell$ bits of memory to answer all queries of the distinguisher $\mathbf{D}_\ell$ considered in Corollary 1, while always asking at most one query to the ideal primitive to answer a single query itself. Our result implies that for any indifferentiable domain extension construction, if the respective simulator is of this single-query form then it needs at least $\ell - m$ bits of memory. Since typically $\ell \gg m$, this implies that the simulator given in [CDMP05] has essentially optimal memory requirements within this class (i.e., linear in ℓ).

4.3 Random Oracle vs. Ideal Cipher

Our proof of Theorem 2 relies on information-theoretic arguments that remain valid also after introducing additional permutation structure into the real resource. Hence, as a side result, we also obtain the impossibility of reducing an arbitrary input-length random oracle to an ideal cipher with respect to stateless simulators. This is in contrast to the results of [CDMP05] that demonstrate the possibility of such reduction with respect to stateful simulators. The proof of the following corollary uses the same arguments as part $(\star\star)$ in the proof of Theorem 2 and is hence omitted.

Corollary 2. *If for every $\pi \in \Sigma$ and for $\ell = k + n + \lceil \log(n/r) \rceil + 2$ the distinguisher $\mathbf{D}_\ell$ considered in Corollary 1 is present in $\mathcal{D}$, then any reduction $\mathbf{E}^{k,n} \underset{\mathrm{m}}{\overset{\varepsilon}{\rightarrow}} \mathbf{R}^{*,r}$ has to satisfy $\varepsilon \geq 0.1$.*

5 Domain Extension Is Impossible in a General Multi-party Setting

As a particular application of our results, in this section we present some interesting consequences of the lower bound on simulator memory for domain extension of public random functions established in the previous section.

The approach taken in any indifferentiability analysis is to model the system in question as having two interfaces: the private one and the public one, as described in Section 2. However, we often consider the constructed primitives to be used in an environment or protocol involving multiple parties. For example, a random oracle is typically understood to be available to all entities participating in a protocol (or possibly many concurrent protocols) that use it. The generic translation of an indifferentiability result into a security guarantee for such a setting is then tacitly assumed. Namely, we view *all* the honest parties as accessing identical copies of the private interface of the real primitive, each party running a local copy of the protocol π realizing the reduction. On the other hand, *all* the misbehaving parties are allowed to access the internals of the construction via identical copies of the public interface.

This implicit reasoning step imposes some requirements on the reduction used. For example, when constructing a random oracle from an ideal compression function, all honest parties should use the same protocol π and moreover, it should be stateless in the sense of Definition 1. This is intuitively easy to see, since an inherently stateful protocol could lead to inconsistent behavior observed by different honest parties. In the ideal world the resource (a random oracle) is stateful, with the state (its function table) accessible to all honest parties. If in the real world a part of this state was stored by the protocol, different parties running different instances of the protocol could obtain different function values for the same query. Naturally, typical protocols constructing a random oracle from an ideal compression function such as the variants of the Merkle-Damgård construction proposed in [CDMP05] are indeed designed to be stateless.

It turns out that for a generic transition from an indifferentiability statement to a security guarantee in a setting with multiple parties, using a stateless protocol is in general by itself *not* sufficient. However, before we can formally approach this question, we first have to describe how we formulate security requirements in the multi-party setting. For this task we use the approach of abstract cryptography (AC) of Maurer and Renner.

AC Reducibility. Here we only give a very brief introduction to the AC framework required for our exposition, further details and the justification of the framework are given in [MR11]. The framework introduces a strong notion of isomorphism given at a very abstract level that, when applied to the particular setting of abstract systems, gives rise to the security notion described below. Its main technical difference compared to other simulation-based security definitions (e.g. [Can01, BPW04]) relevant for our discussion is that it requires the existence of a *local* simulator for each of the parties.

From now on, we will be discussing more general resources having n interfaces labeled $1, \ldots, n$, hence we also have to extend our notation. If $\hat{\phi} = (\phi_1, \ldots, \phi_n)$ is an n-tuple of converters and $\mathbf{S}$ is an n-interface resource, we write $\hat{\phi}\mathbf{S}$ to denote the resource $\mathbf{S}$ with the converter ϕ_i applied to its i-th interface for all $i \in \{1, \ldots, n\}$. For a subset $\mathcal{P} \subseteq \{1, \ldots, n\}$ and an n-tuple of converters $\hat{\phi} = (\phi_1, \ldots, \phi_n)$ let us denote by $\hat{\phi}_{\mathcal{P}}$ the n-tuple of converters that is obtained from $\hat{\phi}$ by replacing all converters on positions *not* in $\mathcal{P}$ by the identity converter id. Hence, for two n-interface resources $\mathbf{S}$ and $\mathbf{T}$, the notation $\hat{\pi}_{\mathcal{P}}\mathbf{S}$ below denotes the system $\mathbf{S}$ with a protocol from $\hat{\pi}$ connected to every interface in $\mathcal{P}$ while $\hat{\sigma}_{\overline{\mathcal{P}}}\mathbf{T}$ denotes $\mathbf{T}$ with a simulator from $\hat{\sigma}$ connected to every interface *not* in $\mathcal{P}$.

Let $\mathbf{S}$ and $\mathbf{T}$ be n-interface resources. For some understood Σ and $\mathcal{D}$, we say that $\mathbf{T}$ is ε-reducible to $\mathbf{S}$ in the sense of AC (denoted $\mathbf{S} \xrightarrow[\text{AC}]{\varepsilon} \mathbf{T}$) if there exist two n-tuples of converters $\hat{\pi} = (\pi_1, \ldots, \pi_n)$ and $\hat{\sigma} = (\sigma_1, \ldots, \sigma_n)$ such that for every subset $\mathcal{P}$ of indices $\{1, \ldots, n\}$ and every distinguisher $\mathbf{D} \in \mathcal{D}$ we have $\Delta^{\mathbf{D}}(\hat{\pi}_{\mathcal{P}}\mathbf{S}, \hat{\sigma}_{\overline{\mathcal{P}}}\mathbf{T}) \leq \varepsilon$, i.e.:

$$\mathbf{S} \xrightarrow[\text{AC}]{\varepsilon} \mathbf{T} \quad :\Leftrightarrow \quad (\exists \hat{\pi}, \hat{\sigma} \in \Sigma^n)(\forall \mathcal{P} \subseteq \{1, \ldots, n\})(\forall \mathbf{D} \in \mathcal{D}) : \Delta^{\mathbf{D}}(\hat{\pi}_{\mathcal{P}}\mathbf{S}, \hat{\sigma}_{\overline{\mathcal{P}}}\mathbf{T}) \leq \varepsilon. \tag{3}$$

For a 1-interface resource $\mathbf{S}$, let us denote by $\hat{\mathbf{S}}_n$ the n-interface resource that provides access to the same internal copy of $\mathbf{S}$ on each of its interfaces (including the same randomness). For $\mathcal{P} \subseteq \{1, \ldots, n\}$ and a distinguisher $\mathbf{D}$ from the class $\mathcal{D}$ let $\mathsf{Proj}_{\mathcal{P}}(\mathbf{D})$ denote a new distinguisher for the 2-interface indifferentiability setting that works exactly as $\mathbf{D}$ does but asks all $\mathbf{D}$'s queries to interfaces in $\mathcal{P}$ at the private interface instead and all $\mathbf{D}$'s queries to interfaces in $\overline{\mathcal{P}}$ at the public interface instead. Moreover, let $\mathsf{Proj}_{\mathcal{P}}(\mathcal{D}) := \{\mathsf{Proj}_{\mathcal{P}}(\mathbf{D}) \mid \mathbf{D} \in \mathcal{D}\}$.

Generic Transition to n-Party Setting. Now we are ready to state a theorem that formalizes the above-mentioned generic transition from any indifferentiability statement to a more meaningful statement in the multi-party AC setting: it turns out that using stateless protocols *and* simulators is sufficient. Since the isomorphism notion introduced in the AC framework requires us to make statements where the simulators are chosen independently of the distinguisher (such as in (3)), to relate indifferentiability to AC we make use of its strong version described in Section 2. The proof of Theorem 3 is deferred to the full version of our paper.

Theorem 3. *Let* $\mathbf{S}$*,* $\mathbf{T}$ *be* 1*-interface resources and let* $n \in \mathbb{N}$*. If* $\hat{\mathbf{S}}_2 \xrightarrow[\text{si}]{\varepsilon} \hat{\mathbf{T}}_2$ *for a class of converters* Σ *and distinguishers* $\mathcal{D}$*, and both the protocol* π *and the simulator* σ *used in this reduction are stateless, then we have* $\hat{\mathbf{S}}_n \xrightarrow[\text{AC}]{\varepsilon} \hat{\mathbf{T}}_n$ *for the class of converters* Σ *and any class of distinguishers* $\mathcal{D}'$ *such that* $\mathsf{Proj}_{\mathcal{P}}(\mathcal{D}') \subseteq \mathcal{D}$ *for all* $\mathcal{P} \subseteq \{1, \ldots, n\}$*.*

Impossibility of Domain Extension. Let us now consider the specific case of the domain extension for random functions in the n-party case[6] (i.e., the reduction $\hat{\mathbf{R}}_n^{m,r} \xrightarrow[\text{AC}]{\varepsilon} \hat{\mathbf{R}}_n^{\ell,r}$ with $\ell > m$). In this case using inherently stateful simulators σ_i would also lead to inconsistencies, for the same reason as described for the protocols π_i. Note that we cannot claim that such a reduction cannot be achieved using a stateful simulator, since its stateful behavior might not manifest in the distinguishing experiment. However, any such stateful simulator could be replaced by a stateless one without significant impact, as formalized in Lemma 1 below (its proof is deferred to the full version). Later we observe that the simulators cannot be stateless (for the same reason as in the indifferentiability case), leading to the impossibility result.

For the statement of Lemma 1, we will assume that the set of distinguishers $\mathcal{D}$ satisfies a simple closure property. For any $\mathbf{D} \in \mathcal{D}$ asking queries only to interfaces 1 and 2, let $\mathbf{D}_{(i)}$ denote a distinguisher that proceeds the same way as $\mathbf{D}$ but when it asks its i-th query to interface 2, it asks the same query also to interface 3. At the end, $\mathbf{D}_{(i)}$ determines its output bit solely on whether the response to its i-th query to interface 2 was consistent with the response to the same query to interface 3. We assume $\mathbf{D}_{(i)} \in \mathcal{D}$ for all $\mathbf{D} \in \mathcal{D}$ and all $1 \leq i \leq q$ where q is an upper bound on the number of $\mathbf{D}$'s queries to interface 2.

[6] In the rest of the section we will use symbols such as $\mathbf{R}^{m,r}$ to refer to the single-interface resource and use the introduced notation to explicitly state the number of interfaces we want to consider (e.g., $\hat{\mathbf{R}}_n^{m,r}$).

Lemma 1. *Consider some fixed $n \geq 3$, $\ell > m$ and some fixed sets of converters Σ and distinguishers $\mathcal{D}$ satisfying the property given above. Assume that there exists a reduction $\hat{\mathbf{R}}_n^{m,r} \xrightarrow[\mathrm{AC}]{\varepsilon} \hat{\mathbf{R}}_n^{\ell,r}$ via a tuple of protocols $\hat{\pi} = (\pi_1, \ldots, \pi_n)$ and simulators $\hat{\sigma} = (\sigma_1, \ldots, \sigma_n)$. Then there also exists a tuple of simulators $\hat{\sigma}' = (\sigma_1, \sigma_2', \sigma_3, \ldots, \sigma_n)$ such that σ_2' is stateless and for every distinguisher $\mathbf{D} \in \mathcal{D}$ accessing only interfaces 1 and 2 we have $\Delta^{\mathbf{D}}(\hat{\pi}_{\{1\}}\hat{\mathbf{R}}_n^{m,r}, \hat{\sigma}'_{\overline{\{1\}}}\hat{\mathbf{R}}_n^{\ell,r}) \leq (q+1)\varepsilon$ where q is an upper bound on the number of its queries to interface 2.*

Let us now denote by $\hat{\mathbf{D}}$ the distinguisher given in Fig. 2 (implicitly parametrized by a converter $\pi \in \Sigma$) modified into the n-interface setting as follows: it uses interface 1 for all its (originally) private-interface queries, while using interface 2 for all public-interface queries. If $\hat{\mathbf{D}} \in \mathcal{D}$ then the upper bound given in Lemma 1 applies to $\Delta^{\hat{\mathbf{D}}}(\hat{\pi}_{\{1\}}\hat{\mathbf{R}}_n^{m,r}, \hat{\sigma}'_{\overline{\{1\}}}\hat{\mathbf{R}}_n^{\ell,r})$. On the other hand, since $\ell > m$ and σ_2' uses no memory, following the proof of Theorem 2 we also get that $\Delta^{\hat{\mathbf{D}}}(\hat{\pi}_{\{1\}}\hat{\mathbf{R}}_n^{m,r}, \hat{\sigma}'_{\overline{\{1\}}}\hat{\mathbf{R}}_n^{\ell,r}) > 0.04$. Combining these observations we get the following corollary.

Corollary 3. *Consider some fixed $n \geq 3$, $r \geq 2$, $\ell > m$ and sets of converters Σ and distinguishers $\mathcal{D}$ satisfying the properties required in Lemma 1 and additionally such that for each $\pi \in \Sigma$ the respective $\hat{\mathbf{D}}$ is in $\mathcal{D}$. If there exists a reduction $\hat{\mathbf{R}}_n^{m,r} \xrightarrow[\mathrm{AC}]{\varepsilon} \hat{\mathbf{R}}_n^{\ell,r}$ via a tuple of protocols $\hat{\pi} = (\pi_1, \ldots, \pi_n)$ then $\varepsilon > 0.04/(p+1)$ where p is an upper bound on the number of $\{0,1\}^m$-queries the protocol π_1 used for this reduction needs to evaluate on one $\{0,1\}^\ell$-input.*

Hence by the above result it is impossible to extend the domain of a public random function even by a single bit in a multi-party environment where the parties must be modeled as possibly having conflicting goals or deviating from the protocol in an uncoordinated manner (or, technically speaking, in any scenario where a proper modeling requires the use of local simulators). The above result extends trivally also to the case of infinite domain extension, i.e., the construction of a public random oracle from an ideal compression function. This is in contrast to the two-party indifferentiability setting (with several constructions that achieve this transformation) where one implicitly makes the assumption that all dishonest parties are coordinated by a hypothetical central adversary. This seems to be a very strong assumption in particular for random oracles that are typically thought of as being used by many different parties in many different applications. Of course, a particular use of a construction proven secure in the 2-party scenario within a multi-party setting as discussed above might still be secure under some additional assumptions, however our result indicates that such use should always be explicitly justified.

6 Conclusions

We have introduced a general way of treating simulation-based security in situations where a more fine-grained quantification of a certain resource is necessary.

Focusing on indifferentiability as the security notion in question and memory as the resource, this also allowed us to explain from a different perspective the unexpected security failure of the protocol given in [RSS11] when used with the construction chop-MD.

We proceeded by giving lower bounds on the required simulator memory for any reduction of a public random oracle to a public random function, showing that memory roughly linear in the length of the longest query is necessary, and that with no memory even domain extension by a single bit becomes impossible.

Finally, we applied our result to the setting where the random oracle is used by multiple parties with no central adversary to coordinate potential misbehavior. We showed that special care must be taken in such settings when replacing the random oracle by a construction using an ideal compression function, since no construction secure in *every* such setting exists.

Acknowledgements. We would like to thank Thomas Holenstein and Robin Künzler for valuable input and fruitful discussions at the early stages of this work. In particular, they independently gave a proof of the impossibility of single-bit domain extension for the case of simulators without memory. We would also like to thank the anonymous reviewers for pointing out a bug in an earlier version of our proof.

Our research was supported by the grants SNF 200020-132794, UK/23/2012 and by the Zurich Information Security and Privacy Center (ZISC).

References

[AKL$^+$09] Alwen, J., Katz, J., Lindell, Y., Persiano, G., Shelat, A., Visconti, I.: Collusion-free multiparty computation in the mediated model. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 524–540. Springer, Heidelberg (2009)

[AKMZ12] Alwen, J., Katz, J., Maurer, U., Zikas, V.: Collusion-preserving computation. In: Safavi-Naini, R. (ed.) CRYPTO 2012. LNCS, vol. 7417, pp. 124–143. Springer, Heidelberg (2012)

[AMP10] Andreeva, E., Mennink, B., Preneel, B.: On the Indifferentiability of the Grøstl Hash Function. In: Garay, J.A., De Prisco, R. (eds.) SCN 2010. LNCS, vol. 6280, pp. 88–105. Springer, Heidelberg (2010)

[BDPVA08a] Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Keccak specifications. In: Submission to NIST (Round 1) (2008)

[BDPVA08b] Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: On the indifferentiability of the sponge construction. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 181–197. Springer, Heidelberg (2008)

[BPW04] Backes, M., Pfitzmann, B., Waidner, M.: A general composition theorem for secure reactive systems. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 336–354. Springer, Heidelberg (2004)

[BR93] Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: ACM Conference on Computer and Communications Security, pp. 62–73 (1993)

[BT94] Benaloh, J., Tuinstra, D.: Receipt-free secret-ballot elections. In: STOC 1994: Proceedings of the 26th Annual ACM Symposium on Theory of Computing, pp. 544–553. ACM, New York (1994)

[Can01] Canetti, R.: Universally composable security: a new paradigm for cryptographic protocols. In: FOCS 2001: Proceedings of the 42nd IEEE Annual Symposium on Foundations of Computer Science, pp. 136–145. IEEE Computer Society Press (October 2001), Full version at `http://eprint.iacr.org/2000/067`

[CDMP05] Coron, J.-S., Dodis, Y., Malinaud, C., Puniya, P.: Merkle-Damgård Revisited: How to Construct a Hash Function. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 430–448. Springer, Heidelberg (2005)

[CG96] Canetti, R., Gennaro, R.: Incoercible multiparty computation. In: FOCS 1996: Proceedings of the 37th IEEE Annual Symposium on Foundations of Computer Science, pp. 504–513. IEEE Computer Society (1996)

[CGH98] Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited. In: STOC 1998: Proceedings of the 30th Annual ACM Symposium on Theory of Computing, pp. 209–218. ACM (1998)

[CK11] Csiszár, I., Körner, J.: Information theory: coding theorems for discrete memoryless systems, 2nd edn. Cambridge University Press (2011)

[CN08] Chang, D., Nandi, M.: Improved Indifferentiability Security Analysis of chopMD Hash Function. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 429–443. Springer, Heidelberg (2008)

[CPS08] Coron, J.-S., Patarin, J., Seurin, Y.: The random oracle model and the ideal cipher model are equivalent. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 1–20. Springer, Heidelberg (2008)

[CV12] Canetti, R., Vald, M.: Universally composable security with local adversaries. In: Visconti, I., De Prisco, R. (eds.) SCN 2012. LNCS, vol. 7485, pp. 281–301. Springer, Heidelberg (2012)

[DRRS09] Dodis, Y., Reyzin, L., Rivest, R., Shen, E.: Indifferentiability of Permutation-Based Compression Functions and Tree-Based Modes of Operation, with Applications to MD6. In: Dunkelman, O. (ed.) FSE 2009. LNCS, vol. 5665, pp. 104–121. Springer, Heidelberg (2009)

[DRS09] Dodis, Y., Ristenpart, T., Shrimpton, T.: Salvaging Merkle-Damgård for Practical Applications. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 371–388. Springer, Heidelberg (2009)

[DRST12] Dodis, Y., Ristenpart, T., Steinberger, J., Tessaro, S.: To Hash or Not to Hash Again (In)Differentiability Results for H 2 and HMAC. In: Safavi-Naini, R. (ed.) CRYPTO 2012. LNCS, vol. 7417, pp. 348–366. Springer, Heidelberg (2012)

[Fan61] Fano, R.: Transmission of Information: A Statistical Theory of Communications. The MIT Press, Cambridge (1961)

[HKT11] Holenstein, T., Künzler, R., Tessaro, S.: The equivalence of the random oracle model and the ideal cipher model, revisited. In: Proceedings of the 43rd Annual ACM Symposium on Theory of Computing, pp. 89–98. ACM, New York (2011)

[HT04] Halpern, J., Teague, V.: Rational secret sharing and multiparty computation. In: STOC 2004: Proceedings of the 36th Annual ACM Symposium on Theory of Computing, pp. 623–632. ACM, New York (2004)

[HU05] Hofheinz, D., Unruh, D.: Comparing two notions of simulatability. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 86–103. Springer, Heidelberg (2005)

[LMs05] Lepinksi, M., Micali, S., Shelat, A.: Collusion-free protocols. In: STOC 2005: Proceedings of the 37th Annual ACM Symposium on Theory of Computing, pp. 543–552. ACM, New York (2005)

[Mau02] Maurer, U.: Indistinguishability of random systems. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 110–132. Springer, Heidelberg (2002)

[Mau11] Maurer, U.: Constructive cryptography – a new paradigm for security definitions and proofs. In: Mödersheim, S., Palamidessi, C. (eds.) TOSCA 2011. LNCS, vol. 6993, pp. 33–56. Springer, Heidelberg (2012)

[MR11] Maurer, U., Renner, R.: Abstract cryptography. In: Chazelle, B. (ed.) The Second Symposium on Innovations in Computer Science, ICS 2011, pp. 1–21. Tsinghua University Press (January 2011)

[MRH04] Maurer, U., Renner, R., Holenstein, C.: Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 21–39. Springer, Heidelberg (2004)

[MRT12] Maurer, U., Rüedlinger, A., Tackmann, B.: Confidentiality and integrity: A constructive perspective. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 209–229. Springer, Heidelberg (2012)

[RSS11] Ristenpart, T., Shacham, H., Shrimpton, T.: Careful with composition: Limitations of the indifferentiability framework. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 487–506. Springer, Heidelberg (2011)

On Concurrently Secure Computation in the Multiple Ideal Query Model

Vipul Goyal[1] and Abhishek Jain[2,⋆]

[1] Microsoft Research, India
vipul@microsoft.com
[2] MIT and Boston University
abhishek@csail.mit.edu

Abstract. The multiple ideal query (MIQ) model was introduced by Goyal, Jain and Ostrovsky [Crypto'10] as a relaxed notion of security which allows one to construct concurrently secure protocols in the plain model. The main question relevant to the MIQ model is how many queries must we allow to the ideal world adversary? The importance of the above question stems from the fact that if the answer is positive, then it would enable meaningful security guarantees in many application scenarios, as well as, lead to resolution of long standing open questions such as fully concurrent password based key exchange in the plain model.

In this work, we continue the study of the MIQ model and prove severe lower bounds on the number of ideal queries per session. Following are our main results:

1. There exists a two-party functionality that cannot be securely realized in the MIQ model with only a constant number of ideal queries per session.
2. There exists a two-party functionality that cannot be securely realized in the MIQ model by any constant round protocol, with *any polynomial number of ideal queries* per session.

Both of these results are unconditional and even rule out protocols proven secure using a non-black-box simulator. We in fact prove a more general theorem which allows for trade-off between round complexity and the number of ideal queries per session. We obtain our negative results in the following two steps:

1. We first prove our results with respect to *black-box* simulation, i.e., we only rule out simulators that make black-box use of the adversary.
2. Next, we give a technique to "compile" our negative results w.r.t. black-box simulation into full impossibility results (ruling out *non-black-box* simulation as well) in the MIQ model. Interestingly, our compiler uses ideas from the work on obfuscation using tamper-proof hardware, even though our setting does not involve any hardware.

⋆ This material is based on research sponsored by NSF grant #1218461 and DARPA under agreement number FA8750-11-2-0225. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA or the U.S. Government. The author also thanks RISCS (Reliable Information Systems and Cyber Security) institute.

1 Introduction

The notion of secure computation is central to cryptography. Introduced in the seminal works of [43,18], secure multi-party computation allows a group of (mutually) distrustful parties $P_1, \ldots, P_n$, with private inputs $x_1, \ldots, x_n$, to jointly compute any functionality f in such a manner that the honest parties obtain correct outputs and no group of malicious parties learn anything beyond their inputs and prescribed outputs.

The classical results for secure computation are only in the *stand-alone* setting where security holds only if a single protocol session is executed in isolation. Unfortunately, as it has become increasingly evident over the last two decades, stand-alone security does not suffice in real-world scenarios where several protocol sessions may be executed *concurrently* – a typical example being protocols executed over modern networked environments such as the Internet.

Background: Concurrently Secure Computation. Towards that end, the last decade has a seen a significant effort by the cryptographic community towards obtaining protocols that are *concurrently composable*, i.e., protocols that remain secure even when executed concurrently over an insecure network. For example, we could require security under *concurrent self-composition* (which is the focus of this work): a protocol should remain secure even when there are multiple copies executing concurrently. The framework of universal composability (UC) [7] was introduced to capture the setting of *concurrent general composition*, where a protocol may be executed concurrently not only with several copies of itself but also with other arbitrary protocols.

General positive results for UC secure computation are known based on various trusted setup assumptions, such as a common random string [7,8,12,3,13,25,29]. However, a driving goal in cryptographic research is to eliminate the need to trust other entities. As such, positive results for concurrently-secure computation in the *plain model* (which is the main focus of this work) are highly desirable, both from a theoretical and practical viewpoint.

Negative Results for Concurrent Composition. Unfortunately, in the plain model, by and large, most of the results have been negative. UC secure protocols for most functionalities of interest were ruled out in [8,10,40,27]. These impossibility results were extended to the setting of general composition by Lindell [30]. Later, Lindell [31] established broad negative results even for the setting of concurrent self-composition by showing equivalence of concurrent self-composition and general composition for functionalities that allow each party to "communicate" to the other via the output. Following the work of Barak et al. [5] and Goyal [20], recently Agrawal et al. [1] and Garg et al. [16] ruled out essentially all non-trivial two-party functionalities for concurrent self-composition, even in the setting where the inputs of honest parties are fixed in advance for all the protocol sessions.

On the positive side, it is known how to realize zero-knowledge and related functionalities, with security in similar models (e.g., [14,42,28,39,5,32]). The recent work of Goyal [20] obtains positive results for a broader class of functionalities; however, (in keeping with the negative results mentioned above) these results are only relevant to the restricted setting where an honest party uses the *same*, static input in all of the sessions.

The Search for Relaxed Security Notions. While the above discussion paints a rather bleak picture of state of the art on concurrent security, fortunately, there is a brighter side. Indeed, several prior works have studied relaxations of the standard definition of secure computation that bypass the above negative results, yet provide strong and meaningful security guarantees in the concurrent setting. A well studied notion is that of security w.r.t. super-polynomial simulation [37,41,6,29,11,15] which intuitively guarantees that a real-world adversary does not learn any more information than what can be computed in super-polynomial time in the ideal world. Another notion is that of input-indistinguishable computation [34,15] which intuitively guarantees that an adversary cannot decide which input (out of possibly many inputs leading to the same output) is used by the honest party in the protocol.

Recently, Goyal, Jain and Ostrovsky [23] introduced the *multiple ideal query* (MIQ) model for concurrent self-composition where the ideal world adversary is allowed to make more than one output query per session to the ideal functionality. The exact number of queries allowed is a priori fixed by a parameter λ. In our view, the main advantage of this notion over the previously discussed notions is that it provides an (arguably) intuitive, easy to understand, security guarantee. In particular, in this model, one can precisely measure the amount of "extra" information that the adversary can potentially learn. The security guarantee is provided with standard polynomial time simulation (and adversary) and follows the ideal/real world security formalization. Furthermore, for functionalities such as password-based key exchange, the MIQ definition in fact implies the previous standard definition [19] when $\lambda = \mathsf{O}(1)$.

The MIQ model has also proven relevant in the related setting of resettability. Goyal and Sahai [24] introduced the notion of resettable secure computation and construct such protocols in the model where the ideal adversary can "reset" the trusted party at any point. This gives the ideal adversary the power to query the trusted party multiple times (per session in the real world). This allows them to get a general positive result for all PPT computable functionalities in the plain model in the resettable ideal world setting.

The multiple ideal query model has also proven relevant as technical tool. In particular, the recent positive results of Goyal [20] can be seen as obtained using the following two step paradigm. First, very roughly, Goyal constructs a protocol secure in the multiple ideal query model. Then, the additional queries made to the ideal trusted party are eliminated by constructing an "output predictor".

We believe the study of the MIQ model is well motivated: both because the guarantee provided in the concurrent setting is interesting and non-trivial on its own, as well as the connection it has to constructing secure protocols in other related settings.

In this work, we continue the study of the MIQ model.

Our Question: How many Queries? The main question relevant to the MIQ model is how many queries must we allow to the ideal world adversary? Note that if we allow a large number of queries, then the security guarantee may quickly degrade and become meaningless; in particular, the adversary may be able to learn the input of the honest party in the worse case. On the other hand, if the number of allowed queries is very small, say $1 + \epsilon$ per session, then the security guarantee is very close to that of the standard definition.

To exemplify this further, consider the oblivious polynomial evaluation functionality [35,36] where two parties wish to jointly evaluate a polynomial over a point. The input of party P_1 is a polynomial Q, while the input of P_2 is a point α. At the end of the protocol, the party P_2 gets $Q(\alpha)$ as the output. This is a natural functionality with applications to list intersection, mutual authentication, metering on the web, etc (see [36] for more details on these).

Now, note that if we only allow, say, 2 queries to a malicious P_2 in the ideal world (per real world session), then as long as Q is a high-degree polynomial, the security guarantee for P_1 is still quite meaningful. Instead of a single point, now a malicious adversary may learn the output on two points of its choice (from an exponential domain of points). The adversary still does not learn any information about what the polynomial evaluates to on rest of the (exponential) domain. On the other hand, if we allow too many queries (exceeding the degree of the polynomial), then the ideal world adversary may be able to learn the entire polynomial Q thus rendering the security guarantee meaningless.

The only known positive results in the MIQ model are due to [23,21]. Goyal et. al. [23] provide a construction where the *average* number of queries in the ideal world per real world session is a constant (with the constant depending upon the adversary). This was further improved in a recent result [21] which provides a construction where the *average* number of ideal queries in any session are $(1 + \frac{1}{\text{poly}(n)})$.

If the guarantee on the number of queries per session is only in expectation, this means that in some sessions, the ideal adversary may still be able to make a large number of queries (while keeping the number of queries low in other sessions). Considering the oblivious polynomial evaluation example above, this means that the security in some sessions may be *completely* compromised. Furthermore, consider the problem of concurrent password based key exchange [26,17,9,2,23]. An interesting question that has remained open so far is designing a concurrent password based key exchange in the plain model where different pair of parties share different (but correlated) passwords. Indeed, this is the most natural setting for password based key exchange (PAKE) and all currently known construction either do not provide concurrent security or are not in the plain model. We note that a positive result in the MIQ model where the number of queries per sessions is a *strict constant* would directly imply a positive result for this problem.[1] This raises the following natural question:

"*Do there exist concurrent secure protocols in the MIQ model where the number of ideal queries per session is a strict constant?*"

1.1 Our Results

In this work, we continue the study of the MIQ model and prove severe unconditional lower bounds on the number of ideal queries per session. Following are our main results:

1. There exists a two-party functionality that cannot be securely realized in the MIQ model with only a constant number of ideal queries per session.
2. There exists a two-party functionality that cannot be securely realized in the MIQ model by any constant round protocol, with *any polynomial number of ideal queries* per session.

Both of these results are unconditional and even rule out protocols proven secure using a non-black-box simulator. We in fact prove a more general theorem that provides a trade-off between the round-complexity and the number of ideal queries per session.

Let $\mathsf{ceil}_y(x)$ be recursively defined as $\mathsf{ceil}_y(x) = \lceil x \cdot \mathsf{ceil}_{y-1}(x) \rceil$. Our main result is stated as follows:

Theorem 1. *There exists a two-party functionality f such that for any $d = d(k)$ and $n = n(k)$ that satisfy $n^d = \text{poly}(k)$, no n-round protocol Π securely realizes f in the MIQ model with at most $\lambda = \mathsf{ceil}_d(1 + \frac{1}{n})$ number of queries per session.*

Application to Concurrent Precise Zero-Knowledge. While our main results concern with the MIQ model, interestingly, they also find applications in the setting of *precise simulation* [33]. Recall that in the setting of precise simulation, we wish to ensure that the resource utilization of the simulator is "close" to the resource utilization of the adversary in the real world interaction. The resource being studied is typically the running time, however, previous works have also considered a more general setting where the resource in question can be, e.g., memory. One can consider a general setting, where instead of focusing only on a particular resource (such as time), we consider many resources at the same time, such as memory, cache, power, etc. A general question we may ask is whether it is possible to perform simulation that achieves precision for each of these resources simultaneously.

To be more concrete, say we have a concurrent adversary interacting with the prover in many sessions and making use of k different resources (each resource

[1] The first positive result for concurrent PAKE in the plain model provided by Goyal et. al. [23] was for the *single* password setting where there is a single global correct password which every party is required to use for authentication. This restriction stems from their solution requiring a constant number of ideal queries per session on an *average*.

may be utilized by the adversary at any arbitrary point). A natural question is: can one obtain a zero-knowledge simulator such that its utilization of *each* resource is only within a constant factor of the adversary? Our negative results directly imply a negative answer for this question as well where the number of resources is equal to the number of sessions. In other words, viewing the ideal functionality query in session i as a utilization of the i-th resource, we directly have that the simulator will end up going over a constant factor for at least one of the resources. Similarly, our results also imply a severe lower bound for constant round protocols: there exists a resource whose utilization will, in fact, not be within any polynomial factor of the adversary's utilization.

Independent of our work, Pass [38] has been able to obtain a *positive* result in the *stand-alone* setting for the above problem. In particular, [38] gives a construction where the simulator is able to be precise in multiple resources simultaneously in the standalone setting.

1.2 Our Techniques

In this section, we give an overview of our techniques. We obtain our negative results in the following two steps: we first prove our results with respect to *black-box* simulation, i.e., we only rule out simulators that make black-box use of the adversary. Next, we give a technique to "compile" our negative results w.r.t. black-box simulation into full impossibility results (ruling out *non-black-box* simulation as well) in the MIQ model. Below, we discuss each of these steps separately.

Impossibility for Black-box Simulation. Recall that in order to prove security of a two-party computation protocol, we need to demonstrate that for every real world adversary $\mathcal{A}$ that controls one of the parties, there exists an ideal world adversary (or simulator $\mathcal{S}$) who can simulate the view of $\mathcal{A}$. Typically, the simulator $\mathcal{S}$ works by extracting the input a used by $\mathcal{A}$ and then querying the ideal functionality with a to receive the correct output; this output is then used to complete the simulation of $\mathcal{A}$'s view. Now, further recall that the only advantage that a black-box simulator has over the real adversary is the ability to *rewind*. In other words, a black-box simulator extracts the input of $\mathcal{A}$ by rewinding. However, in the concurrent setting, extracting the input of $\mathcal{A}$ in each session is a non-trivial task. In particular, given an adversarial scheduling, it may happen that in order to extract the input of $\mathcal{A}$ in a given session s, the simulator $\mathcal{S}$ rewinds past the beginning of another session s' (that is interleaved inside the protocol messages of session s). When this happens, $\mathcal{A}$ may change its input in session s'. Thus, the simulator $\mathcal{S}$ would be forced to query the ideal functionality more than once for the session s'.

We now briefly explain how the above intuition can be further extended to achieve a black-box impossibility result even when the simulator is allowed to make multiple queries to the ideal functionality. For concreteness, let us consider the (simplified) case where the simulator is allowed a fixed *constant* C number of queries per session. Note that in order to obtain the desired negative result, we need to construct a concurrent adversary $\mathcal{A}$ that can force any black-box simulator $\mathcal{S}$ to make more than C queries for at least one session. We now briefly discuss how to construct such an adversary. Let n be the round complexity of the protocol, where n is any polynomial in the security parameter.

Consider the following *static* adversarial scheduling of messages. Consider an "outer" session (say) s. We will call it a session at level 0. Now, between every round of messages in the outer session, place a new complete protocol session. We will call these n sessions to be at level 1. Next, we again place a new complete protocol session between every round of messages in each session at level 1. Note that this creates n^2 sessions at level 2. Repeat this process recursively until we reach level C, where there are exactly n^C sessions. Thus, in total, we have $m = \frac{n^{C+1}-1}{n-1}$ sessions, which is polynomial in the security parameter.

Now, as discussed earlier, a black-box simulator $\mathcal{S}$ must perform at least one rewinding in order to extract the input of $\mathcal{A}$ in the "outer" session s. Suppose that $\mathcal{S}$ rewinds the i^{th} round of session s. Then, this immediately implies that the i^{th} session at level 1 is executed at least twice, which in turn means that $\mathcal{S}$ will be forced to query the ideal functionality twice for that session. Now, note that $\mathcal{S}$ will need to extract $\mathcal{A}$'s input in each of these two executions in order to complete them successfully. Thus, assuming that (even if) $\mathcal{S}$ rewinds different rounds in each of these two executions, we have that there exist two sessions at level 2 that are each executed three times. Continuing this argument inductively, we can show that for every level i, there exists at least one session that is executed $i + 1$ times. As a result, we obtain that there exists a session s^* at level C that is executed $C + 1$ times. If the adversary chooses a different input in each of the $C + 1$ executions, we have that $\mathcal{S}$ must query the ideal functionality $C + 1$ times for session s^*. Thus, we conclude that the black-box simulator $\mathcal{S}$, who is only allowed C queries, must fail.[2]

From Black-Box to Non-Black-Box. We now discuss a compilation technique to transform our negative results for black-box simulation into full impossibility results that rule out non-black-box simulation as well.

Recall that the main advantage that a non-black-box simulator has over a black-box simulator is that the former can make use of the adversary's code. Then, our high-level approach is to "nullify" this advantage by making use of secure program obfuscation. We note, however, that general program obfuscation is known to be impossible [4]. Towards this end, our key idea is to use positive results on program obfuscation using stateless tamper-proof hardware tokens by Goyal et. al. [22]. In the obfuscation with hardware model, one can take the given program and convert it into an obfuscated program using an obfuscation key k. The obfuscated program, for execution, would require oracle access to a hardware token having the obfuscation key k. We denote the functionality of the token (required to run the obfuscated program) by f_{token} (parameterized by the key k).

[2] We remark that in order to prove our general result, a more tight analysis is necessary; see the technical sections.

Very roughly, our idea is to implement the "token functionality" f_{token} of [22] using two-party computation. An important point is that f_{token} is "robust" to any polynomial number of queries; thus, it is particularly suited to the MIQ model. In more detail, we obtain our negative result in the following three steps:

Toy Experiment: Let Π be any protocol for the f_{token} functionality and let $\mathcal{A}$ be any concurrent adversary for Π that rules out black-box simulators that make at most λ queries per session. We first consider a toy experiment involving three parties, namely, Alice, Bob and David. In this experiment, Alice and Bob interact in multiple ideal world executions of the token functionality f_{token}. At the same time, Bob and David are involved in concurrent real-world executions of Π where Bob and David follow the same scheduling of messages as defined by adversary $\mathcal{A}$. Furthermore, the adversary Bob is allowed to *reset* David at any point during their interaction.
David, who has a secret input secret is instructed to reveal secret to Bob if all the executions of Π are completed "successfully". The goal of Bob is to successfully complete its interaction with David and learn the value secret. Then, the main idea in this experiment is that, by relying on our black-box impossibility result, we show that no adversarial Bob can succeed in learning secret, except with negligible probability.

Ideal World: In the second step, we eliminate David from the above experiment by obfuscating his next-message function in the f_{token}-hybrid model and give it as an auxiliary input to Bob (while the corresponding obfuscation "key" is given to Alice). This results in a scenario where Alice and Bob are the only parties, who interact in multiple ideal world executions of f_{token}. From the security of obfuscation, we can argue that this experiment can be reduced to the toy experiment; as such, no adversarial Bob can learn secret, except with negligible probability.

Real World Experiment: We finally consider the real world experiment, which is the same as previous step, except that all the ideal world invocations of f_{token} are now replaced with real-world executions of protocol Π. It is not difficult to see that in this experiment, an adversary Bob can simply play a "man-in-the-middle" between Alice and David (since Bob has David's obfuscated code); as a result, Bob can learn secret with probability 1.

From above, it follows that the adversary Bob in the real world experiment is a PPT concurrent adversary for Π whose view cannot be simulated by *any* simulator that makes at most λ queries per session, thus yielding us the desired result. We refer the reader to the technical sections for more details.

2 Preliminaries

2.1 Our Model

In this section, we present our security model. Throughout this paper, we denote the security parameter by k.

Concurrently Secure Computation in the MIQ Model. We define our security model by extending the standard real/ideal paradigm for secure computation. We consider a malicious, static adversary. We do not require fairness and only consider security with abort. Finally, we only consider *computational* security. We now proceed to describe the ideal and real world experiments and then give our security definition.

Ideal model. We first define the ideal world experiment, where there is a trusted party for computing the desired two-party functionality f. Let there be two parties P_1 and P_2 that are involved in multiple sessions, say $m = m(k)$. An adversary may corrupt either of the two parties. As in the standard ideal world experiment for concurrently secure computation, the parties send their inputs to the trusted party and receive the output of f evaluated on their inputs. The main difference from the standard ideal world experiment is that the ideal adversary is allowed to make λ (as opposed to one) output queries (with possibly different inputs of its choice) in each session. The ideal world execution proceeds as follows.

Inputs: P_1 and P_2 obtain a vector of m inputs, denoted $\boldsymbol{x}$ and $\boldsymbol{y}$ respectively. The adversary is given auxiliary input z, and chooses a party to corrupt. Without loss of generality, we assume that the adversary corrupts P_2 (when the adversary controls P_1, the roles are simply reversed). The adversary receives the input vector $\boldsymbol{y}$ of the corrupted party.

Session initiation: The adversary initiates a new session by sending a start message to the trusted party. The trusted party then sends (start, i) to P_1, where i is the index of the session.

Honest parties send inputs to trusted party: Upon receiving (start, i) from the trusted party, honest party P_1 sends (i, x_i) to the trusted party, where x_i denotes P_1's input for session i.

Adversary sends input to trusted party: Whenever the adversary wishes, it may send a message $(i, \ell, y'_{i,\ell})$ to the trusted party for any $y'_{i,\ell}$ of its choice. Upon sending this pair, it receives back $(i, \ell, f(x_i, y'_{i,\ell}))$ where x_i is the input value that P_1 previously sent to the trusted party for session i. The only limitation is that for any i, the trusted party accepts at most λ tuples indexed by i from the adversary.

Adversary instructs trusted party to answer honest party: When the adversary sends a message of the type $(\mathsf{output}, i, \ell)$ to the trusted party, the trusted party sends $(i, f(x_i, y'_{i,\ell}))$ to P_1, where x_i and $y'_{i,\ell}$ denote the respective inputs sent by P_1 and adversary for session i.

Outputs: The honest party P_1 always outputs the values $f(x_i, y'_{i,\ell})$ that it obtained from the trusted party. The adversary may output an arbitrary (probabilistic polynomial-time computable) function of its auxiliary input z, input vector $\boldsymbol{y}$ and the outputs obtained from the trusted party.

The $\mathsf{ideal\ execution}$ of a function f with security parameter sec, input vectors $\boldsymbol{x}$, $\boldsymbol{y}$ and auxiliary input z to $\mathcal{S}$, denoted $\textsc{ideal}_{f,\mathcal{S}}(k, \boldsymbol{x}, \boldsymbol{y}, z)$, is defined as the output pair of the honest party and $\mathcal{S}$ from the above ideal execution.

Definition 1 (λ-Ideal Query Simulator). *Let $\mathcal{S}$ be a non-uniform probabilistic (expected)* PPT *machine representing the ideal-model adversary. We say that $\mathcal{S}$ is a λ-ideal query simulator if it makes at most λ output queries per session in the above ideal experiment.*

Real model. We now consider the real model in which a real two-party protocol is executed (and there exists no trusted third party). Let f be as above and let Π be a two-party protocol for computing f. Let $\mathcal{A}$ denote a non-uniform probabilistic polynomial-time adversary that controls either P_1 or P_2. The parties run concurrent executions of the protocol Π, where the honest party follows the instructions of Π in all executions. The honest party initiates a new session i with input x_i whenever it receives a start-session message from $\mathcal{A}$. The scheduling of all messages throughout the executions is controlled by the adversary. That is, the execution proceeds as follows: the adversary sends a message of the form (i, msg) to the honest party. The honest party then adds msg to its view of session i and replies according to the instructions of Π and this view. At the conclusion of the protocol, an honest party computes its output as prescribed by the protocol. Without loss of generality, we assume the adversary outputs exactly its entire view of the execution of the protocol.

The real concurrent execution of Π with security parameter k, input vectors $\boldsymbol{x}$, $\boldsymbol{y}$ and auxiliary input z to $\mathcal{A}$, denoted $\text{REAL}_{\Pi,\mathcal{A}}(k, \boldsymbol{x}, \boldsymbol{y}, z)$, is defined as the output pair of the honest party and $\mathcal{A}$, resulting from the above real-world process.

Definition 2 (λ-Secure Concurrent Computation in the MIQ Model). *A protocol Π is said to* λ-securely realize *a functionality f under concurrent self composition in the MIQ model if for every real model non-uniform* PPT *adversary $\mathcal{A}$, there exists a non-uniform (expected)* PPT *λ-ideal query simulator $\mathcal{S}$ such that for all polynomials $m = m(k)$, every pair of input vectors $\boldsymbol{x} \in X^m$, $\boldsymbol{y} \in Y^m$, every $z \in \{0,1\}^*s$, $\{\text{IDEAL}_{f,\mathcal{S}}(k, \boldsymbol{x}, \boldsymbol{y}, z)\}_{k\in\mathbb{N}} \overset{c}{\equiv} \{\text{REAL}_{\Pi,\mathcal{A}}(k, \boldsymbol{x}, \boldsymbol{y}, z)\}_{k\in\mathbb{N}}$.*

2.2 Obfuscation with Tamper-Proof Hardware Tokens

In this work, we use ideas from the area of obfuscation using tamper-proof hardware tokens. In particular, we use the positive result of Goyal et al. [22] on secure program obfuscation using *stateless* tamper-proof hardware tokens. Below, we give an abstract overview of the scheme of [22] that we will use in our negative results. We remark that the discussion below hides most of the internal details of the scheme of [22]. We refer the reader to [22] for the details of the scheme.

Obfuscation in Token Hybrid Model. In order to obfuscate a circuit C, the sender executes the following steps:

– Sample a token instance $T \leftarrow \mathsf{SampleT}$. The token instance has some secret values hardwired inside it. We will denote the secret values as a key K (that is drawn from some distribution $\mathcal{K}$). In other words, sampling of the token instance just involves to sampling the key K from the distribution $\mathcal{K}$.
– Compute the obfuscated program $\mathcal{O}(C) \leftarrow \mathsf{Obfuscate}(C, K)$

To compute $C(x)$ on any input x, the obfuscated program $\mathcal{O}(C)$ makes t_C queries of various types to the token T, where each q is drawn from some distribution $\mathcal{Q}$. Here, t_C, denoted as the *query parameter* of the obfuscation scheme, is an integer that depends on the size of the circuit C that is obfuscated.

Obfuscating Stateful Programs. The above description is only relevant to obfuscating *stateless* circuits C. We note that it is also possible to obfuscate the programs of *stateful* (or reactive) machines M in the above scheme by using standard techniques. The basic idea is that obfuscated code $\mathcal{O}(M)$ is computed in such a manner that its output on any given input x consists of both $M(x)$ and an *authenticated encryption* of its resultant *state* after the computation of $M(x)$.

We now recall the following security lemma from [22] (informally stated):

Lemma 1 ([22]). *Assuming the existence of one-way functions,* ($\mathsf{SampleT}$, $\mathsf{Obfuscate}$) *is a (stateful) program obfuscation scheme in the token-hybrid model with the following properties. For any adversary having the obfuscated program $\mathcal{O}(C)$ along with oracle access to the token, there exists an ideal world simulator having only black-box access to the circuit C, such that, the output distribution of the simulator is computationally indistinguishable from that of the adversary.*

The Token Functionality. A key idea that is used in our negative results is to implement the working of the hardware token T via a two-party secure computation protocol. To this end, we abstract the working of the token T as the following two-party functionality f_{token}, that we will refer to as the "token functionality".

Denote by f_{token} the token functionality with the key K hardwired inside the description of its circuit. The input and output interface of the functionality f_{token} is described as follows:

Inputs: Party P_1 gets a token key $K \leftarrow \mathcal{K}$ as input, while P_2 gets a query $q \leftarrow \mathcal{Q}$ as input.
Outputs: P_1 gets no output, while P_2 gets $f_{\mathsf{token}}(K; q)$.

Note that f_{token} is a deterministic functionality. We now state a lemma regarding the unpredictability of outputs of f_{token}. We note that this lemma is implicit in [22].

Lemma 2 (Unpredictability of Output of f_{token} [22]). *There exists a distribution $\mathcal{Q}$ (with super-logarithmic min-entropy) from which a query q can be sampled with the following properties. Any adversary $\mathcal{A}$, given oracle access to the functionality $f_{\mathsf{token}}(K;\cdot)$ (where K is sampled at random from $\mathcal{K}$) with the restriction that it is allowed to query $f_{\mathsf{token}}(K;\cdot)$ on any string except q, can output $f_{\mathsf{token}}(K;q)$ with only negligible probability.*

3 Black-Box Impossibility in the MIQ Model

In this section, we prove impossibility results for concurrently secure computation in the MIQ model with respect to black-box simulation. Due to lack of space, we only state our result statements here and refer the reader to the full version for their formal proofs.

Let $\mathsf{ceil}_y(x)$ be recursively defined as $\mathsf{ceil}_y(x) = \lceil x \cdot \mathsf{ceil}_{y-1}(x) \rceil$. Our general result, stated below, shows a trade-off between the query parameter λ and the round-complexity n of the protocol:

Theorem 2. *There exists a functionality f such that for any $d = d(k)$ and $n = n(k)$ that satisfy $n^d = \mathrm{poly}(k)$, no n-round protocol Π $\lambda = \mathsf{ceil}_d(1 + \frac{1}{n})$-securely realizes f in the MIQ model with respect to black-box simulation.*

Note that $\mathsf{ceil}_d(1 + \frac{1}{n}) \geq d + 1$. Thus, we obtain the following general corollary when substituting d with λ:

Corollary 1. *There exists a functionality f such that for any $\lambda = \lambda(k)$ and $n = n(k)$ that satisfy $n^\lambda = \mathrm{poly}(k)$, no n-round protocol Π λ-securely realizes f in the MIQ model with respect to black-box simulation.*

By plugging in $n = \mathrm{poly}(k)$ and $\lambda = \mathsf{O}(1)$ above, we get the following as a sub-corollary, ruling out general positive results in the MIQ model when a (black-box) simulator is allowed only a *constant number of ideal queries* per session:

Corollary 2. *There exists a functionality f that cannot be $\mathsf{O}(1)$-securely realized in the MIQ model with respect to black-box simulation.*

Finally, by plugging in $n = O(1)$ and $d = \log(k)$ in Theorem 2, we obtain the following corollary ruling out *constant-round* protocols in the MIQ model:

Corollary 3. *There exists a functionality f that cannot be securely realized in the MIQ model by any $O(1)$-round protocol w.r.t. black-box simulation, even if a (black-box) simulator is allowed any (fixed) $\mathrm{poly}(k)$ ideal queries per session.*

In fact, we will prove something stronger, as stated below. We first give the following definition:

Definition 3 (λ-special black-box adversary). *Let $\lambda = \lambda(k)$ and $\Pi = (P_1, P_2)$ be a protocol for functionality f. A* PPT *concurrent adversary $\mathcal{A}$ for Π that corrupts party P_2 is said to be* λ-special adversary *if the following holds:*

- *$\mathcal{A}$ outputs `accept` with probability 1 in the real-world execution with P_1.*
- *Except with negligible probability, no λ-ideal query black-box simulator can send a query to $\mathcal{A}$ such that it outputs `accept`.*

Now, it is easy to see that Theorem 2 is implied by the following theorem:

Theorem 3. *For every $d = d(k)$ and $n = n(k)$-round protocol Π for the f_{token} functionality, if $n^d = \mathrm{poly}(k)$, then there exists a* λ-special adversary *for Π, where $\lambda = \mathsf{ceil}_d(1 + \frac{1}{n})$.*

4 Full Impossibility in the MIQ Model

We now present full impossibility results for concurrently secure computation in the multiple ideal query model, ruling out non-black-box simulation as well. More concretely, we present a technique to "compile" our impossibility results w.r.t. black-box simulation into full impossibility results. We now state our main theorem of this section:

Theorem 4. *Let $\lambda = \lambda(k)$ and Π be any protocol for the f_{token} functionality. If there exists a* PPT *λ-special black-box adversary $\mathcal{A}$ for Π, then there exists a* PPT *concurrent adversary $\mathcal{B}$ for Π whose view cannot be simulated by any (potentially non-black-box)* PPT *λ-ideal query simulator.*

Combining Theorem 3 with the above theorem, we immediately obtain our main result stated in Theorem 1. We also derive the analogous corollaries of Corollary 1, 2, 3, ruling out non-black-box simulation as well. We skip the formal statements here due to lack of space.

4.1 Proof of Theorem 4

Let $\lambda = \lambda(k)$ and Π be any protocol for the f_{token} functionality. Then, given any λ-special black-box adversary $\mathcal{A}$ for Π, we now show how to construct a λ-special adversary $\mathcal{B}$ for Π. We use ideas from obfuscation using stateless tamper-proof hardware tokens [22] in order to show this transformation.

We recommend the reader to review the outline of the three main steps in our proof as described in Section 1.2. We now proceed to describe each of the steps in details. We first setup some notation.

Notation. Let m be the number of sessions that adversary $\mathcal{A}$ schedules for protocol Π. Let $K_1, \ldots, K_m$ denote a set of token keys, where each K_i is drawn at random from the distribution $\mathcal{K}$. Let $q_1, \ldots, q_m$ be a set of query strings for the token functionality, where each q_i is drawn at random from the distribution D (as defined in Lemma 2). Finally, let secret be a random string in $\{0,1\}^k$.

(Toy) Experiment I: Restating the Black-Box Impossibility Result. Consider the following scenario involving three parties, namely, Alice, Bob and David. Alice is given an input vector $(K_1, \ldots, K_m)$, while David is given an input vector $((K_1, q_1), \ldots, (K_m, q_m); \mathsf{secret})$. Here, the input values are chosen in the manner as described above. We describe the interaction between Alice and Bob, and Bob and David, separately.

Interaction between Alice and Bob. Alice and Bob are interacting in m ideal world executions of the functionality f_{token}, where Alice plays the role of P_1 using input vector $(K_1, \ldots, K_m)$ and Bob plays the role of P_2 using any inputs of its choice. In each of these m ideal world executions, the adversary Bob is allowed to query the token functionality λ times using any inputs of its choice.

Interaction between Bob and David. At the same time, Bob and David are interacting in m real-world concurrent executions of protocol Π, where Bob plays the the role of P_1 with any inputs of its choice and David plays the role of P_2 by simply running the code of the λ-special adversary $\mathcal{A}$.[3]

Bob and David are instructed to assume the identities of Alice and Bob, respectively, in these sessions. The messages of the m sessions follow the same schedule as defined by the adversary $\mathcal{A}$. Furthermore, the adversary Bob is allowed to *reset* David at any point during their interaction. That is, at any point during their interaction, Bob can choose to "rewind" David to an earlier state and create new threads of execution.

In each session $i \in [m]$, David verifies whether his output $y_i = f_{\mathsf{token}}(K_i, q_i)$ (where q_i is the input of David in that session, chosen in the same manner as $\mathcal{A}$ would); if this is not the case, then David aborts the interaction with Bob and outputs $\perp$ (i.e., David does not continue any remaining sessions with Bob). If *all* of the m sessions are completed successfully, then David sends secret to Bob (as the only additional message outside of the m concurrent executions of Π).

Lemma 3. *Bob outputs* secret *in Experiment I with negligible probability.*

Experiment II: Ideal World. We now "eliminate" the party David from the above toy experiment. Note that eliminating David results only in interactions between Alice and Bob, which is indeed our desired setting of concurrent self-composition. Very roughly, in order to eliminate David, we would like to *obfuscate* David's next message function and give it as auxiliary input to Bob. Since general program obfuscation is impossible in the plain model [4], our key idea is to use positive results on secure program obfuscation using tamper-proof hardware tokens, and adapt them to our setting. In particular, we will use the positive results of Goyal et al. [22] on secure program obfuscation using a stateless hardware token that implements the f_{token} functionality. We now give more details. Some of the notation used below is as defined in Section 2.2.

Eliminating David. Consider the next message function NMF of party David as described in Experiment I. Sample a key $K \leftarrow \mathcal{K}$. Compute the obfuscation of NMF in the f_{token}-hybrid model, where f_{token} has the key K hardwired. The resulting program, denoted as $\mathcal{O}(\mathsf{NMF}) \leftarrow \mathsf{Obfuscate}(K, \mathsf{NMF})$, is given as auxiliary input to Bob. The key K, on the other hand, is given as an additional input to Alice.

Ideal World Experiment. Experiment II, or in other words, the ideal world experiment is defined in the same manner as Experiment I, except that we eliminate the party David in the manner as described above. Note that in order to evaluate the program $\mathcal{O}(\mathsf{NMF})$ on any input, Bob would need to answer the queries q of $\mathcal{O}(\mathsf{NMF})$ to the token functionality $f_{\mathsf{token}}(K, \cdot)$. In particular, recall from Section 2.2 that the obfuscated code $\mathcal{O}(C)$ for a program C makes t_C queries to the token functionality, where t_C (referred to as the query parameter) depends on the size of the circuit C. Then, in the ideal world experiment, we have Alice and Bob engage in $t_{\mathsf{NMF}} \cdot m \cdot n$ additional ideal executions of f_{token}, where t_{NMF} is the

[3] In particular, if $\mathcal{A}$ chooses to ignore the inputs $q_1, \ldots, q_m$ and choose fresh inputs "on-the-fly", then David follows the same strategy.

query parameter for the obfuscation scheme of [22] as determined by the circuit description of NMF, m is the number of sessions that adversary $\mathcal{A}$ schedules, and n is the round complexity of protocol Π. In each of these $t_{\mathsf{NMF}} \cdot m \cdot n$ executions, Alice uses the key K as input, while Bob is allowed to use any inputs of its choice. Further, in each of these $t_{\mathsf{NMF}} \cdot m \cdot n$ ideal executions, Bob is allowed to query the token functionality λ times using any inputs of its choice.

Thus, overall the ideal world experiment between Alice and Bob consists of the following:

- m "main" ideal executions of f_{token}, where in each session $i \in [m]$, Alice uses key K_i as input, while Bob is allowed to use any inputs of its choice.
- $t_{\mathsf{NMF}} \cdot m \cdot n$ "auxiliary" ideal executions of f_{token}, where in each session, Alice uses key K as input, while Bob is allowed to use any inputs of its choice.

Further, as explained above, in *each* of the $m + t_{\mathsf{NMF}} \cdot m \cdot n$ ideal executions of f_{token}, Bob is allowed to query the ideal functionality λ times using any inputs of its choice.

Lemma 4. *Bob outputs* secret *in Experiment II with negligible probability.*

Experiment III: Real World. We now describe the final experiment, which is the real world experiment between Alice and Bob. This experiment is essentially the same as Experiment II, except that each ideal world execution of f_{token} is now replaced with a real world execution of protocol Π between Alice and Bob. In more detail, the interaction between Alice and Bob consists of the following:

- m "main" executions of protocol Π, where the messages of the m executions are scheduled in the same manner as defined by the adversary $\mathcal{A}$. In each $i \in [m]$ main session, Alice uses the keys K_i as her input.
- $t_{\mathsf{NMF}} \cdot m \cdot n$ "auxiliary" executions of protocol Π that are scheduled by Bob in the manner, as described below. In each of these auxiliary sessions, Alice uses key K as her input.

Alice's program: Alice is given input the keys $K_1, \ldots, K_n$ for the m "main" sessions, and key K for the $t_{\mathsf{NMF}} \cdot m \cdot n$ "auxiliary" sessions.

Alice behaves honestly according to the protocol Π and responds honestly to all protocol invocations made by Bob by using the code of P_1.

Bob's program: Bob is given as auxiliary input the obfuscated program $\mathcal{O}(\mathsf{NMF})$, where NMF is the next-message function of David (as described above). Let secret be the secret value hardwired in $\mathcal{O}(\mathsf{NMF})$.

For $i = 1, \ldots, m \cdot n$, do:

1. Upon receiving the i^{th} message from Alice in m main sessions, say a_i, suspend (temporarily) the ongoing session. Run the code $\mathcal{O}(\mathsf{NMF})$ on input a_i.[4]

[4] Note that the input to $\mathcal{O}(\mathsf{NMF})$ would also include the authenticated state information that $\mathcal{O}(\mathsf{NMF})$ would have output earlier. We exclude this from the description for simplicity of notation.

Whenever $\mathcal{O}(\mathsf{NMF})$ makes a query q, start a new auxiliary session of Π with Alice, and run the code of P_2 honestly using input q. Since $\mathcal{O}(\mathsf{NMF})$ makes t_{NMF} different queries, in total, t_{NMF} auxiliary sessions of Π are executed sequentially by Bob.

2. If $i < m \cdot n$, then on receiving the output d_i from $\mathcal{O}(\mathsf{NMF})$ outputs, resume the suspended "main" session and send d_i to Alice as the response to a_i. Otherwise, output the value $d_i = \mathsf{secret}$.

Lemma 5. *Bob outputs the value* secret *in Experiment III with probability* 1.

Completing the Proof of Theorem 4. The adversary $\mathcal{B}$ is simply the adversary Bob in Experiment III as described above. The proof of Theorem 4 then follows immediately from Lemma 4 and 5. We refer the reader to the full version for the missing proofs of the lemmas.

References

1. Agrawal, S., Goyal, V., Jain, A., Prabhakaran, M., Sahai, A.: New impossibility results for concurrent composition and a non-interactive completeness theorem for secure computation. In: Safavi-Naini, R. (ed.) CRYPTO 2012. LNCS, vol. 7417, pp. 443–460. Springer, Heidelberg (2012)
2. Barak, B., Canetti, R., Lindell, Y., Pass, R., Rabin, T.: Secure computation without authentication. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 361–377. Springer, Heidelberg (2005)
3. Barak, B., Canetti, R., Nielsen, J.B., Pass, R.: Universally composable protocols with relaxed set-up assumptions. In: FOCS, pp. 186–195 (2004)
4. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S.P., Yang, K.: On the (im)possibility of obfuscating programs. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 1–18. Springer, Heidelberg (2001)
5. Barak, B., Prabhakaran, M., Sahai, A.: Concurrent non-malleable zero knowledge. In: FOCS, pp. 345–354 (2006)
6. Barak, B., Sahai, A.: How to play almost any mental game over the net - concurrent composition via super-polynomial simulation. In: FOCS, pp. 543–552 (2005)
7. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: FOCS, pp. 136–147 (2001)
8. Canetti, R., Fischlin, M.: Universally composable commitments. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 19–40. Springer, Heidelberg (2001)
9. Canetti, R., Halevi, S., Katz, J., Lindell, Y., MacKenzie, P.: Universally composable password-based key exchange. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 404–421. Springer, Heidelberg (2005)
10. Canetti, R., Kushilevitz, E., Lindell, Y.: On the limitations of universally composable two-party computation without set-up assumptions. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 68–86. Springer, Heidelberg (2003)
11. Canetti, R., Lin, H., Pass, R.: Adaptive hardness and composable security in the plain model from standard assumptions. In: FOCS, pp. 541–550 (2010)
12. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: STOC, pp. 494–503 (2002)

13. Canetti, R., Pass, R., Shelat, A.: Cryptography from sunspots: How to use an imperfect reference string. In: FOCS, pp. 249–259 (2007)
14. Dwork, C., Naor, M., Sahai, A.: Concurrent zero-knowledge. In: STOC (1998)
15. Garg, S., Goyal, V., Jain, A., Sahai, A.: Concurrently secure computation in constant rounds. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 99–116. Springer, Heidelberg (2012)
16. Garg, S., Kumarasubramanian, A., Ostrovsky, R., Visconti, I.: Impossibility results for static input secure computation. In: Safavi-Naini, R. (ed.) CRYPTO 2012. LNCS, vol. 7417, pp. 424–442. Springer, Heidelberg (2012)
17. Gennaro, R., Lindell, Y.: A framework for password-based authenticated key exchange. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 524–543. Springer, Heidelberg (2003)
18. Goldreich, O., Micali, S., Wigderson, A.: How to play ANY mental game. In: STOC, pp. 218–229 (1987)
19. Goldreich, O., Lindell, Y.: Session-key generation using human passwords only. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 408–432. Springer, Heidelberg (2001)
20. Goyal, V.: Positive results for concurrently secure computation in the plain model. In: FOCS (2012)
21. Goyal, V., Gupta, D., Jain, A.: What information is leaked under concurrent composition? Manuscript (2013)
22. Goyal, V., Ishai, Y., Sahai, A., Venkatesan, R., Wadia, A.: Founding cryptography on tamper-proof hardware tokens. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 308–326. Springer, Heidelberg (2010)
23. Goyal, V., Jain, A., Ostrovsky, R.: Password-authenticated session-key generation on the internet in the plain model. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 277–294. Springer, Heidelberg (2010)
24. Goyal, V., Sahai, A.: Resettably secure computation. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 54–71. Springer, Heidelberg (2009)
25. Katz, J.: Universally composable multi-party computation using tamper-proof hardware. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 115–128. Springer, Heidelberg (2007)
26. Katz, J., Ostrovsky, R., Yung, M.: Efficient password-authenticated key exchange using human-memorable passwords. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 475–494. Springer, Heidelberg (2001)
27. Kidron, D., Lindell, Y.: Impossibility results for universal composability in public-key models and with fixed inputs. J. Cryptology 24(3), 517–544 (2011)
28. Kilian, J., Petrank, E.: Concurrent and resettable zero-knowledge in poly-logarithm rounds. In: STOC (2001)
29. Lin, H., Pass, R., Venkitasubramaniam, M.: A unified framework for concurrent security: universal composability from stand-alone non-malleability. In: STOC, pp. 179–188. ACM (2009)
30. Lindell, Y.: General composition and universal composability in secure multi-party computation. In: FOCS, pp. 394–403 (2003)
31. Lindell, Y.: Lower bounds for concurrent self composition. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 203–222. Springer, Heidelberg (2004)
32. Lindell, Y.: Lower bounds and impossibility results for concurrent self composition. J. Cryptology 21(2), 200–249 (2008)
33. Micali, S., Pass, R.: Local zero knowledge. In: STOC, pp. 306–315 (2006)
34. Micali, S., Pass, R., Rosen, A.: Input-indistinguishable computation. In: FOCS, pp. 367–378 (2006)

35. Naor, M., Pinkas, B.: Oblivious transfer and polynomial evaluation. In: STOC (1999)
36. Naor, M., Pinkas, B.: Oblivious polynomial evaluation. SIAM J. Comput. (2006)
37. Pass, R.: Simulation in quasi-polynomial time, and its application to protocol composition. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 160–176. Springer, Heidelberg (2003)
38. Pass, R.: Personal communication (2012)
39. Prabhakaran, M., Rosen, A., Sahai, A.: Concurrent zero knowledge with logarithmic round-complexity. In: FOCS, pp. 366–375 (2002)
40. Prabhakaran, M., Rosulek, M.: Cryptographic complexity of multi-party computation problems: Classifications and separations. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 262–279. Springer, Heidelberg (2008)
41. Prabhakaran, M., Sahai, A.: New notions of security: achieving universal composability without trusted setup. In: STOC, pp. 242–251 (2004)
42. Richardson, R., Kilian, J.: On the concurrent composition of zero-knowledge proofs. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 415–431. Springer, Heidelberg (1999)
43. Chi-Chih Yao, A.: How to generate and exchange secrets. In: FOCS, pp. 162–167 (1986)

Universally Composable Secure Computation with (Malicious) Physically Uncloneable Functions

Rafail Ostrovsky[1,2], Alessandra Scafuro[1], Ivan Visconti[3], and Akshay Wadia[1]

[1] Department of Computer Science, UCLA, USA
[2] Department of Mathematics, UCLA, USA
{rafail,scafuro,awadia}@cs.ucla.edu
[3] Dipartimento di Informatica, University of Salerno, Italy
visconti@dia.unisa.it

Abstract. Physically Uncloneable Functions (PUFs) [28] are noisy physical sources of randomness. As such, they are naturally appealing for cryptographic applications, and have caught the interest of both theoreticians and practitioners. A major step towards understanding and securely using PUFs was recently taken in [Crypto 2011] where Brzuska, Fischlin, Schröder and Katzenbeisser model PUFs in the Universal Composition (UC) framework of Canetti [FOCS 2001]. A salient feature of their model is that it considers *trusted* PUFs only; that is, PUFs which have been produced via the prescribed manufacturing process and are guaranteed to be free of any adversarial influence. However, this does not accurately reflect real-life scenarios, where an adversary could be able to create and use malicious PUFs.

The goal of this work is to extend the model proposed in [Crypto 2011] in order to capture such a real-world attack. The main contribution of this work is the study of the Malicious PUFs model. To this end, we first formalize the notion of "malicious" PUFs, and extend the UC formulation of Brzuska et al. to allow the adversary to create PUFs with *arbitrary* adversarial behaviour. Then, we provide positive results in this, more realistic, model. We show that, under computational assumptions, it is possible to UC-securely realize any functionality.

1 Introduction

The impossibility of secure computation in the universal composability framework was proved first by Canetti and Fischlin [9], and then strengthened by Canetti et al. in [10]. Impossibility of even weaker notions has been proved in [1, 5, 16].

As a consequence, several setup assumptions, and relaxations of the UC framework have been proposed to achieve UC security [4, 11, 19, 29].

In recent years, researchers have started exploring the use of secure hardware in protocol design. The idea is to achieve protocols with strong security guarantees (like UC) by allowing parties to use hardware boxes that have certain security properties. An example of the kind of security required from such a hardware box is that of *tamper-proofness*; i.e., the receiver of the box can only observe the input/output behaviour of the functionality that the box implements. This property was formalized by Katz in [20], and it was shown that UC security is possible by relying on the existence of tamper-proof programmable hardware tokens, and computational assumptions. Smart cards are well understood examples of such tokens, since they have been used in practice in the last decades. Several improvements and variations of Katz's model have been then proposed in follow up papers (e.g., [17]).

Spurred by technological advances in manufacturing, recently a new hardware component has gained a lot of attention: Physically Uncloneable Functions (PUFs) [27,28]. A PUF is a hardware device generated through a special physical process that implements a "random" function[1] that depends upon the physical parameters of the process. These parameters can not be "controlled", and producing a clone of the device is considered infeasible[2] . Once a PUF has been constructed, there is a physical procedure to query it, and to measure its answers. The answer of a PUF depends on the physical behavior of the PUF itself, and is assumed to be unpredictable, or to have high min-entropy. Namely, even after obtaining many challenge-response pairs, it is infeasible to predict the response to a new challenge.

Since their introduction by Pappu in 2001, PUFs have gained a lot of attention for cryptographic applications like anti-counterfeiting mechanisms, secure storage, RFID applications, identification and authentication protocols [14, 15, 18, 18, 22, 33, 34]. More recently PUFs have been used for designing more advanced cryptographic primitives. In [31] Rührmair shows the first construction of Oblivious Transfer, the security proof of which is later provided in [32]. In [3], Armknecht et al. deploy PUFs for the construction of memory leakage-resilient encryption schemes. In [23] Maes et al. provide construction and implementation of PUFKY, a design for PUF-based cryptographic key generators. There exist several implementations of PUFs, often exhibiting different properties. The work of Armknecht et al. [2] formalizes the security features of physical functions in accordance to existing literature on PUFs and proposes a general security framework for physical functions. A survey on PUF implementations is given in [24]. Very recently in [21] Katzenbeisser et al. presented the first large scale evaluation of the security properties of some popular PUFs implementations (i.e., intrinsic electronic PUFs).

Modeling PUFs in the UC Framework. Only very recently, Brzuska et al. [7] suggested a model for using PUFs in the UC setting that aims at abstracting real-world implementations. The unpredictability and uncloneability properties are modeled through an ideal functionality. Such functionality allows only the

[1] Technically, a PUF does not implement a function in the mathematical sense, as the same input might produce different responses.

[2] SRAM PUFs (memory-based PUFs) might be cloneable according to recent finding [6].

creation of trusted PUFs. In [7] PUFs are thought as non-PPT setup assumptions. As such, a PPT simulator cannot simulate a PUF, that is, PUFs are non-programmable. Although non-programmable, PUFs are not modeled as global setup [8]. [7] shows how to achieve unconditional UC secure Oblivious Transfer, Bit Commitment and Key Agreement with trusted PUFs.

1.1 Our Contribution

We observe that the UC formulation of PUFs proposed by Brzuska et al. makes the following crucial assumption: the model considers *trusted* PUFs only, that is, adversaries are assumed to be unable to produce fake/malicious PUFs. As we argue below, we feel that assuming that an adversary cannot misbehave by creating fake/malicious PUFs, might be unrealistic in the real world. Given that the study of PUFs is still in its infancy, it is risky to rely on assumptions on the impossibility of the adversaries in generating and accessing PUFs adversarially. The main contribution of this work is to study security models that capture such plausible real-world attacks, and provide protocols that are secure in the presence of such adversaries.

Modeling malicious PUFs. We augment the UC framework so to enable the adversary to create untrusted (malicious) PUFs. But what exactly are malicious PUFs? In real life, an adversary could tamper with a PUF in such a way that the PUF loses any of its security properties. Or the adversary may introduce new behaviours; for example, the PUF may start logging its queries. To keep the treatment of malicious behaviour as general as possible, we allow the adversary to send as PUF any hardware token that meets the syntactical requirements of a PUF. Thus, an adversary is assumed to be able to even produce fake PUFs that might be stateful and programmed with malicious code. We assume that a malicious PUF however cannot interact with its creator once is sent away to another party. If this was not the case, then we are back in the standard model (see the Introduction in the full version [26]).

UC secure computation with malicious PUFs. The natural question is whether UC security can be achieved in such a much more hostile setting. We give a positive answer to this question by constructing a *computational* UC commitment scheme in the malicious PUFs model. Our commitment scheme needs two PUFs that are transferred only once (PUFs do not go back-and-forth), at the beginning of the protocol and it requires computational assumptions. We avoid that PUFs go back-and-forth by employing a technique that requires OT. The results of Canetti, et al. [11] shows how to achieve general UC computation from computational UC commitments. Whether *unconditional* UC secure computation is possible in the malicious PUF model, is still an open problem.

Hardness assumptions with PUFs. Notice that as correctly observed in [7], since PUFs are not PPT machines, it is not clear if standard complexity-theoretic assumptions still hold in presence of PUFs. We agree with this observation. However the critical point is that even though there can exist a PUF that helps to break in polynomial time a standard complexity-theoretic assumptions, it is still unlikely that a PPT adversary can find such a PUF. Indeed a PPT machine can only generate a polynomial number of PUFs, therefore obtaining the one that allows to break complexity assumptions is an event that happens with negligible probability and thus it does not effect the concrete security of the protocols.

In light of the above discussion, only one of the following two cases is possible. 1) Standard complexity-theoretic assumptions still hold in presence of PPT adversaries that generate PUFs; in this case our construction is secure. 2) There exists a PPT adversary that can generate a PUF that breaks standard assumptions; in this case our construction is not secure, but the whole foundations of complexity-theoretic cryptography would fall down (which is quite unlikely to happen) with respect to real-world adversaries. We elaborate on this issue in Section 3.1.

Additional results. We now mention additional results that can be found in the full version of this paper [26] but have been omitted from the present conference version due to lack of space. Firstly, we further investigate the feasibility of achieving *unconditional security* in the malicious PUF model. We leave the important question of unconditional UC open, but provide a construction of an unconditional commitment scheme in the malicious PUF model. Secondly, we propose and study another modification to the original model of Brzuska et al. In the new model which we call "oblivious-query model", all parties (and the adversary) use trusted PUFs, but in the security proofs, the simulator is *not* allowed to observe the adversary's queries to its PUF. The main motivation for studying this modification is that the ability of the simulator to observe adversary's queries stems from the assumption that there is only a single, prescribed procedure for evaluating a PUF. As we discuss in detail in the full version, this assumption is not well-justified in the real world. Our main contribution in the oblivious-query model is the construction of an unconditional UC protocol for OT. Lastly, we show that if both adversarial modes discussed above are combined, viz., adversaries can create malicious PUFs and may query honest PUFs via non-prescribed processes, then UC security is impossible.

Independent work. Very recently and independently of us, van Dijk and Rührmair [36] also study the use of PUFs in cryptographic protocols. Among other things, they consider the "bad" PUF model where PUFs can be augmented with malicious behaviour like keeping a log of queries, etc. They show that unconditional OT is impossible using bad PUFs, but their setting is very different from ours. For a detailed discussion about the work of van Dijk and Rührmair, see the Introduction of the full version [26].

2 Definitions

Notation. We let n be the security parameter and PPT be the class of probabilistic polynomial time Turing machines. We use $v \xleftarrow{\$} A()$ when the algorithm A is randomized. We denote by $\mathsf{dis}_{\mathsf{ham}}(a, b)$ the Hamming distance of a and b.

Physically uncloneable functions. We follow definitions given in [7]. A PUF is a noisy physical source of randomness. The randomness property comes from an uncontrollable manufacturing process. A PUF is evaluated with a physical stimulus, called the *challenge*, and its physical output, called the *response*, is measured. Because the processes involved are physical, the function implemented by a PUF can not (necessarily) be modeled as a mathematical function, neither can be considered computable in PPT. Moreover, the output of a PUF is noisy, namely, querying a PUF twice with the same challenge, could yield to different outputs. The mathematical formalization of a PUF due to [7] is the following.

A PUF-family $\mathcal{P}$ is a pair of (not necessarily efficient) algorithms Sample and Eval, and is parameterized by the bound on the noise of PUF's response d_{noise} and the range of the PUF's output rg. Algorithm Sample abstracts the PUF fabrication process and works as follows. On input the security parameter, it outputs a PUF-index id from the PUF-family satisfying the security property (that we define soon) according to the security parameter. Algorithm Eval abstracts the PUF-evaluation process. On input a challenge q, it evaluates the PUF on q and outputs the response a of length rg. The output is guaranteed to have bounded noise d_{noise}, meaning that, when running $\mathsf{Eval}(1^n, \mathsf{id}, q)$ twice, the Hamming distance of any two responses a_1, a_2 is smaller than $d_{\mathsf{noise}}(n)$. Wlog, we assume that the challenge space of a PUF is a full set of strings of a certain length.

Definition 1 (Physically Uncloneable Functions). *Let rg denote the size of the range of the PUF responses of a PUF-family and d_{noise} denote a bound of the PUF's noise. $\mathcal{P} = (\mathsf{Sample}, \mathsf{Eval})$ is a family of (rg, d_{noise})-PUF if it satisfies the following properties.*

Index Sampling. *Let $\mathcal{I}_n$ be an index set. On input the security parameter n, the sampling algorithm Sample outputs an index $\mathsf{id} \in \mathcal{I}_n$ following a not necessarily efficient procedure. Each $\mathsf{id} \in \mathcal{I}_n$ corresponds to a set of distributions $\mathcal{D}_{\mathsf{id}}$. For each challenge $q \in \{0, 1\}^n$, $\mathcal{D}_{\mathsf{id}}$ contains a distribution $\mathcal{D}_{\mathsf{id}}(q)$ on $\{0, 1\}^{rg(n)}$. $\mathcal{D}_{\mathsf{id}}$ is not necessarily an efficiently sampleable distribution.*

Evaluation. *On input the tuple $(1^n, \mathsf{id}, q)$, where $q \in \{0, 1\}^n$, the evaluation algorithm Eval outputs a response $a \in \{0, 1\}^{rg(n)}$ according to distribution $\mathcal{D}_{\mathsf{id}}(q)$. It is not required that Eval is a PPT algorithm.*

Bounded Noise. *For all indexes $\mathsf{id} \in \mathcal{I}_n$, for all challenges $q \in \{0, 1\}^n$, when running $\mathsf{Eval}(1^n, \mathsf{id}, q)$ twice, the Hamming distance of any two responses a_1, a_2 is smaller than $d_{\mathsf{noise}}(n)$.*

In the paper we use $\mathsf{PUF}_{\mathsf{id}}(q)$ to denote $\mathcal{D}_{\mathsf{id}}(q)$. When not misleading, we omit id from $\mathsf{PUF}_{\mathsf{id}}$, using only the notation PUF.

Security of PUFs. We assume that PUFs enjoy the properties of *uncloneability* and *unpredictability.* Unpredictability is modeled via an entropy condition on the PUF distribution. Namely, given that a PUF has been measured on a polynomial number of challenges, the response of the PUF evaluated on a new challenge has still a significant amount of entropy. Formally,

Definition 2 (Unpredictability). *A (rg, d_{noise})-PUF family $\mathcal{P} = (\mathsf{Sample}, \mathsf{Eval})$ for security parameter n is $(d_{\mathsf{min}}(n), m(n))$-unpredictable if for any $q \in \{0,1\}^n$ and challenge list $\mathcal{Q} = (q_1, \ldots, q_{\mathtt{poly}(n)})$, one has that, if for all $1 \leq k \leq \mathtt{poly}(n)$ the Hamming distance satisfies $\mathsf{dis}_{\mathsf{ham}}(q, q_k) \geq d_{\mathsf{min}}(n)$, then the average min-entropy satisfies $\tilde{H}_\infty(\mathsf{PUF}(q)|\mathsf{PUF}(\mathcal{Q})) \geq m(n)$, where $\mathsf{PUF}(\mathcal{Q})$ denotes a sequence of random variables $\mathsf{PUF}(q_1), \ldots, \mathsf{PUF}(q_{\mathtt{poly}(n)})$ each corresponding to an evaluation of the PUF on challenge q_k. Such a PUF-family is called a $(rg, d_{\mathsf{noise}}, d_{\mathsf{min}}, m)$-PUF family.*

Fuzzy extractors. Fuzzy extractors of Dodis et al. [13] are applied to the outputs of the PUF, to convert such noisy, high-entropy measurements into *reproducible* randomness. Very informally, a fuzzy extractor is a pair of efficient randomized algorithms $(\mathsf{FuzGen}, \mathsf{FuzRep})$. FuzGen takes as input an ℓ-bit string, that is the PUF's response a, and outputs a pair (p, st), where st is a uniformly distributed string, and p is a public helper data string. FuzRep takes as input the PUF's noisy response a' and the helper data p and generates the very same string st obtained with the original measurement a. The security property of fuzzy extractors guarantees that, if the min-entropy of the PUF's responses are greater than a certain parameter m, knowledge of the public data p only, without the measurement a, does not give any information on the secret value st. The correctness property, guarantees that, all pairs of responses a, a' that are close enough, i.e., their hamming distance is less then a certain parameter t, will be recovered by FuzRep to the same value st generated by FuzGen. In order to apply fuzzy extractors to PUF's answers, it is sufficient to pick an extractor whose parameters match with the parameter of the PUF being used.

3 UC Security with Malicious PUFs

In Section 1 we have motivated the need of a different formulation of UC security with PUFs that allows the adversary to generate malicious PUFs. In this section we first show how to model malicious PUFs in the UC framework, and then show that as long as standard computational assumptions still hold when PPT adversaries can generate (even malicious) PUFs, there exist protocols for UC realizing any functionality with (malicious) PUFs.

3.1 Modeling Malicious PUFs

We allow our adversaries to send malicious PUFs to honest parties[3]. As discussed before, the motivation for malicious PUFs is that the adversary may have some

[3] Throughout this section, we assume the reader is familiar with the original UC PUF formulation of Brzuska et al. [7] (Section 4.2).

control over the manufacturing process and may be able to produce errors in the process that break the PUF's security properties. Thus, we would like parties to rely only on the PUFs that they themselves manufacture (or obtain from a source that they trust), and not on the ones they receive from other (possibly adversarial) parties.

Malicious PUFs families. In the real world, an adversary may create a malicious PUF in a number of ways. For example, it can tamper with the manufacturing process for an honestly-generated PUF to compromise its security properties (unpredictability, for instance). It may also introduce additional behaviour into the PUF token, like logging of queries. Taking inspiration from the literature on modeling tamper-proof hardware tokens, one might be tempted to model malicious PUFs analogously in the following way: to create a malicious PUF, the adversary simply specifies to the ideal functionality, the (malicious) code it wants to be executed instead of an honest PUF. Allowing the adversary to specify the malicious code enables the simulator to "rewind" the malicious PUF, which is used crucially in security proofs in the hardware token model. However, modeling malicious PUFs in this way would disallow the adversary from modifying honest PUFs (or more precisely, the honest PUF manufacturing process). To keep our treatment as general as possible, we do not place any restriction on a malicious PUF, except that it should have the same syntax as that of an honest PUF family, as specified in Definition 1. In particular, the adversary is *not* required to know the code of malicious PUFs it creates, and thus our simulator can not rely on rewinding in the security proofs. Formally, we allow the adversary to specify a "malicious PUF family", that the ideal functionality uses. Of course, in the protocol, we also want the honest parties to be able to obtain and send honestly generated PUFs. Thus our ideal functionality for PUFs, $\mathcal{F}_{\mathsf{PUF}}$ (Fig. 1) is parameterized by two PUF families: the normal (or honest) family $(\mathsf{Sample}_{\mathtt{normal}}, \mathsf{Eval}_{\mathtt{normal}})$ and the possibly malicious family $(\mathsf{Sample}_{\mathtt{mal}}, \mathsf{Eval}_{\mathtt{mal}})$. When a party P_i wants to initialize a PUF, it sends a $\mathsf{init}_{\mathsf{PUF}}$ message to $\mathcal{F}_{\mathsf{PUF}}$ in which it specifies the $\mathtt{mode} \in \{\mathtt{normal}, \mathtt{mal}\}$, and the ideal functionality uses the corresponding family for initializing the PUF. For each initialized PUF, the ideal functionality $\mathcal{F}_{\mathsf{PUF}}$ also stores a tag representing the family (i.e., $\mathtt{mal}$ or $\mathtt{normal}$) from which it was initialized. Thus, when the PUF needs to be evaluated, $\mathcal{F}_{\mathsf{PUF}}$ runs the evaluation algorithm corresponding to the tag.

As in the original formulation of Brzuska et al., the ideal functionality $\mathcal{F}_{\mathsf{PUF}}$ keeps a list $\mathcal{L}$ of tuples $(\mathsf{sid}, \mathsf{id}, \mathtt{mode}, \hat{P}, \tau)$. Here, sid is the session identifier of the protocol and id is the PUF identifier output by the $\mathsf{Sample}_{\mathtt{mode}}$ algorithm. As discussed above $\mathtt{mode} \in \{\mathtt{normal}, \mathtt{mal}\}$ indicates the mode of the PUF, and $\hat{P}$ identifies the party that currently holds the PUF. The final argument τ specifies transition of PUFs: $\tau = \mathsf{notrans}$ indicates the PUF is not in transition, while $\tau = \mathsf{trans}(P_j)$ indicates that the PUF is in transition to party P_j. Only the adversary may query the PUF during the transition period. Thus, when a party P_i hands over a PUF to party P_j, the corresponding τ value for that PUF is changed from $\mathsf{notrans}$ to $\mathsf{trans}(P_j)$, and the adversary is allowed to send evaluation queries to this PUF. When the adversary is done with querying the PUF, it sends a $\mathsf{ready}_{\mathsf{PUF}}$

message to the ideal functionality, which hands over the PUF to P_j and changes the PUFs transit flag back to $\mathsf{notrans}$. The party P_j may now query the PUF. The ideal functionality now waits for a $\mathsf{received_{PUF}}$ message from the adversary, at which point it sends a $\mathsf{received_{PUF}}$ message to P_i informing it that the hand over is complete. The ideal functionality is described formally in Fig. 1.

Allowing adversary to create PUFs. We deviate from the original formulation of $\mathcal{F}_{\mathsf{PUF}}$ of Brzuska et al. [7] in one crucial way: we allow the ideal-world adversary $\mathcal{S}$ to create new PUFs. That is, $\mathcal{S}$ can send a $\mathsf{init_{PUF}}$ message to $\mathcal{F}_{\mathsf{PUF}}$. In the original formulation of Brzuska et al., $\mathcal{S}$ could not create its own PUFs, and this has serious implications for the composition theorem. We thank Margarita Vald [35] for pointing out this issue. We elaborate on this in Appendix H in the full version [26]. Also, it should be noted that the PUF set-up is non-programmable, but not *global* [8]. The environment must go via the adversary to query PUFs, and may only query PUFs in transit or held by the adversary at that time.

We remark that the OT protocol of [7] for honest PUFs, fails in the presence of malicious PUFs. Consider the OT protocol in Fig. 3 in [7]. The security crucially relies on the fact that the receiver P_j *can not* query the PUF after receiving sender's first message, i.e., the pair (x_0, x_1). If it could do so, then it would query the PUF on both $x_0 \oplus v$ and $x_1 \oplus v$ and learn both s_0 and s_1. In the malicious PUF model however, as there is no guarantee that the receiver can not learn query/answer pairs when a malicious PUF that he created is not in its hands, the protocol no longer remains secure.

PUFs and computational assumptions. The protocol we present in the next section will use computational hardness assumptions. These assumptions hold against probabilistic polynomial-time adversaries. However, PUFs use physical components and are not modeled as PPT machines, and thus, the computational assumptions must additionally be secure against PPT adversaries that have access to PUFs. We remark that this is a reasonable assumption to make, as if this is not the case, then PUFs can be used to invert one-way functions, to find collisions in CRHFs and so on, therefore not only our protocol, but any computational-complexity based protocol would be insecure. Note that PUFs are physical devices that actually exist in the real world, and thus all real-world adversaries could use them.

To formalize this, we define the notion of "admissible" PUF families. A PUF family (regardless of whether it is honest or malicious) is called *admissible* with respect to a hardness assumption if that assumption holds even when the adversary has access to PUFs from this family. We will prove that our protocol is secure when the $\mathcal{F}_{\mathsf{PUF}}$ ideal functionality is instantiated with admissible PUF families. In particular, all the cryptographic tools that we use to construct our protocol can be based on the DDH assumption. From this point on in this paper, a "PUF family" would be taken to mean a PUF family which is admissible with respect to DDH.

$\mathcal{F}_{\mathsf{PUF}}$ uses PUF families $\mathcal{P}_1 = (\mathsf{Sample}_{\mathtt{normal}}, \mathsf{Eval}_{\mathtt{normal}})$ with parameters $(rg, d_{\mathsf{noise}}, d_{\mathsf{min}}, m)$, and $\mathcal{P}_2 = (\mathsf{Sample}_{\mathtt{mal}}, \mathsf{Eval}_{\mathtt{mal}})$.
It runs on input the security parameter 1^n, with parties $\mathbb{P} = \{P_1, \ldots, P_n\}$ and adversary $\mathcal{S}$.

- When a party $\hat{P} \in \mathbb{P} \cup \{\mathcal{S}\}$ writes $(\mathsf{init}_{\mathsf{PUF}}, \mathsf{sid}, \mathtt{mode}, \hat{P})$ on the input tape of $\mathcal{F}_{\mathsf{PUF}}$, where $\mathtt{mode} \in \{\mathtt{normal}, \mathtt{mal}\}$, then $\mathcal{F}_{\mathsf{PUF}}$ checks whether $\mathcal{L}$ already contains a tuple $(\mathsf{sid}, *, *, *, *)$:
  - If this is the case, then turn into the waiting state.
  - Else, draw $\mathsf{id} \leftarrow \mathsf{Sample}_{\mathtt{mode}}(1^n)$ from the PUF family. Put $(\mathsf{sid}, \mathsf{id}, \mathtt{mode}, \hat{P}, \mathsf{notrans})$ in $\mathcal{L}$ and write $(\mathsf{initialized}_{\mathsf{PUF}}, \mathsf{sid})$ on the communication tape of $\hat{P}$.
- When party $P_i \in \mathbb{P}$ writes $(\mathsf{eval}_{\mathsf{PUF}}, \mathsf{sid}, P_i, q)$ on $\mathcal{F}_{\mathsf{PUF}}$'s input tape, check if there exists a tuple $(\mathsf{sid}, \mathsf{id}, \mathtt{mode}, P_i, \mathsf{notrans})$ in $\mathcal{L}$.
  - If not, then turn into waiting state.
  - Else, run $a \leftarrow \mathsf{Eval}_{\mathtt{mode}}(1^n, \mathsf{id}, q)$. Write $(\mathsf{response}_{\mathsf{PUF}}, \mathsf{sid}, q, a)$ on P_i's communication input tape.
- When a party P_i sends $(\mathsf{handover}_{\mathsf{PUF}}, \mathsf{sid}, P_i, P_j)$ to $\mathcal{F}_{\mathsf{PUF}}$, check if there exists a tuple $(\mathsf{sid}, *, *, P_i, \mathsf{notrans})$ in $\mathcal{L}$.
  - If not, then turn into waiting state.
  - Else, modify the tuple $(\mathsf{sid}, \mathsf{id}, \mathtt{mode}, P_i, \mathsf{notrans})$ to the updated tuple $(\mathsf{sid}, \mathsf{id}, \mathtt{mode}, \perp, \mathsf{trans}(P_j))$. Write $(\mathsf{invoke}_{\mathsf{PUF}}, \mathsf{sid}, P_i, P_j)$ on $\mathcal{S}$'s communication input tape.
- When the adversary sends $(\mathsf{eval}_{\mathsf{PUF}}, \mathsf{sid}, \mathcal{S}, q)$ to $\mathcal{F}_{\mathsf{PUF}}$, check if $\mathcal{L}$ contains a tuple $(\mathsf{sid}, \mathsf{id}, \mathtt{mode}, \perp, \mathsf{trans}(*))$ or $(\mathsf{sid}, \mathsf{id}, \mathtt{mode}, \mathcal{S}, \mathsf{notrans})$.
  - If not, then turn into waiting state.
  - Else, run $a \leftarrow \mathsf{Eval}_{\mathtt{mode}}(1^n, \mathsf{id}, q)$ and return $(\mathsf{response}_{\mathsf{PUF}}, \mathsf{sid}, q, a)$ to $\mathcal{S}$.
- When $\mathcal{S}$ sends $(\mathsf{ready}_{\mathsf{PUF}}, \mathsf{sid}, \mathcal{S})$ to $\mathcal{F}_{\mathsf{PUF}}$, check if $\mathcal{L}$ contains the tuple $(\mathsf{sid}, \mathsf{id}, \mathtt{mode}, \perp, \mathsf{trans}(P_j))$.
  - If not found, turn into the waiting state.
  - Else, change the tuple $(\mathsf{sid}, \mathsf{id}, \mathtt{mode}, \perp, \mathsf{trans}(P_j))$ to $(\mathsf{sid}, \mathsf{id}, \mathtt{mode}, P_j, \mathsf{notrans})$ and write $(\mathsf{handover}_{\mathsf{PUF}}, \mathsf{sid}, P_i)$ on P_j's communication input tape and store the tuple $(\mathsf{received}_{\mathsf{PUF}}, \mathsf{sid}, P_i)$.
- When the adversary sends $(\mathsf{received}_{\mathsf{PUF}}, \mathsf{sid}, P_i)$ to $\mathcal{F}_{\mathsf{PUF}}$, check if the tuple $(\mathsf{received}_{\mathsf{PUF}}, \mathsf{sid}, P_i)$ has been stored. If not, return to the waiting state. Else, write this tuple to the input tape of P_i.

Fig. 1. The ideal functionality $\mathcal{F}_{\mathsf{PUF}}$ for malicious PUFs

3.2 Constructions for UC Security in the Malicious PUFs Model

In this section we present a construction for UC-secure commitment scheme in the malicious PUFs model, which yields UC-security for any PPT functionality via the [11] compiler.

We first recall some of the peculiarities of the PUFs model. A major difficulty when using PUFs, in contrast to say tamper-proof tokens, is that PUFs are *not programmable*. That is, the simulator can not simulate the answer of a PUF, and must honestly forward the queries to the $\mathcal{F}_{\mathsf{PUF}}$ functionality. The only power of the simulator is to intercept the queries made by the adversary to honest PUFs. Thus, in designing the protocol, we shall force parties to query the PUFs with the critical private information related to the protocol, therefore allowing the simulator to extract such information in straight-line. In the malicious PUFs model the behaviour of a PUF created and sent by an adversary is entirely in the adversary's control. A malicious PUF can answer (or even abort) adaptively on the query according to some pre-shared strategy with the malicious creator. Finally, a side effect of the unpredictability of PUFs, is that the creator of a honest PUF is not able to check the authenticity of the answer generated by its own PUF, without having the PUF in its hands (or having queried the PUF previously on the very same value).

Techniques and proof intuition. Showing UC security for commitments requires obtaining straight-line extraction against a malicious sender and straight-line equivocality against a malicious receiver. Our starting point is the equivocal commitment scheme of [12] which builds upon Naor's scheme [25]. Naor's scheme consists of two messages, where the first message is a randomly chosen string r that the receiver sends to the sender. The second message is the commitment of the bit b, computed using r. More precisely, to commit to bit b, the second message is $G(s) \oplus (r \wedge b^{|r|})$, where $G()$ is a PRG, and s a randomly chosen seed. The scheme has the property that if the string r is crafted appropriately, then the commitment is equivocal. [12] shows how this can be achieved by adding a coin-tossing phase before the commitment. The coin tossing of [12] proceeds as follows: the receiver commits to a random string α (using a statistically hiding commitment scheme), the sender sends a string β, and then the receiver opens the commitment. Naor's parameter r is then set as $\alpha \oplus \beta$.

Observe that if the simulator can choose β after knowing α, then it can control the output of the coin-tossing phase, and therefore equivocate the commitment. Thus, to achieve equivocality against a malicious receiver, the simulator must be able to extract α from the commitment. Similarly, when playing against a malicious sender, the simulator should be able to extract the value committed in the second message of Naor's commitment.

Therefore, to construct a UC-secure commitment, we need to design an extractable commitment scheme for both directions. The extractable commitment of α that we construct for the receiver, must be statistically-hiding (this is necessary to prove binding). We denote such commitment as $\mathsf{Com}_{\mathsf{sh}} = (\mathsf{S}_{\mathsf{sh}}, \mathsf{R}_{\mathsf{sh}})$. On the other hand, the commitment sent by the sender, must be extractable and allow for equivocation. We denote such commitment as $\mathsf{Com}_{\mathsf{eq}} = (\mathsf{S}_{\mathsf{eq}}, \mathsf{R}_{\mathsf{eq}})$. As we shall see soon, the two schemes require different techniques as they aim to different properties. However, they both share the following structure to achieve extractability.

The receiver creates a PUF and queries it with two randomly chosen challenges (q_0, q_1), obtaining the respective answers (a_0, a_1). The PUF is then sent to the sender. To commit to a bit b, the sender first needs to obtain the value q_b. This is done by running an OT protocol with the receiver. Then the sender queries the PUF with q_b and commits to the response a_b. Note that the sender does not commit to the bit directly, but to the *answer* of the PUF. This ensures extractability. To decommit to b, the sender simply opens the commitment of the PUF-answer sent before. Note that the receiver can check the authenticity of the PUF-answer without having its own PUF back. The simulator can extract the bit by intercepting the queries sent to the PUF and taking the one that is close enough, in Hamming distance, to either q_0 or q_1. Due to the security of OT, the sender can not get both queries (thus confusing the simulator), neither can the receiver detect which query has been transferred. Due to the binding property of the commitment scheme used to commit q_b, a malicious sender cannot postpone querying the PUF to the decommitment phase (thus preventing the simulator to extract already in the commitment phase). Due to the unpredictability of PUFs, the sender cannot avoid to query the PUF to obtain the correct response.

This protocol achieves extractability. To additionally achieve statistically hiding and equivocality, protocol $\mathsf{Com}_{\mathsf{sh}}$ and $\mathsf{Com}_{\mathsf{eq}}$ develop on this basic structure in different ways accordingly to the different properties that they achieve. The main difference is in the commitment of the answer a_b.

In Protocol $\mathsf{Com}_{\mathsf{sh}}$, S_{sh} commits to the PUF-response a_b using a statistically hiding commitment scheme. Additionally, S_{sh} provides a statistical zero-knowledge argument of knowledge of the message committed. This turns out to be necessary to argue about binding (that is only computational). Finally, the OT protocol executed to exchange q_0, q_1 must be statistically secure for the OT receiver. The formal description of protocol $\mathsf{Com}_{\mathsf{sh}}$ is provided in Fig. 2.

In Protocol $\mathsf{Com}_{\mathsf{eq}}$ the answer a_b is committed following Naor's commitment scheme. The input of S_{eq} is the Naor's parameter decided in the coin-flipping phase, and is the vector $\bar{r}$ of strings $r_1, \ldots, r_l$ (a_b is a l-bit string, where l is the range of the PUF). Earlier we said that the simulator can properly craft $\bar{r}$, so that it will be able equivocate the commitment of a_b. However, due to the structure of the extractable commitment shown above, being able to equivocate the commitment of a_b is not enough anymore. Indeed, in the protocol above, due to the OT protocol, the simulator will be able to obtain only one of the PUF-queries among (q_0, q_1), and it must choose the query q_b already in the commitment phase (when the secret bit b is not known to the simulator). Thus, even though the simulator has the power to equivocate the commitment to any string, it might not know the correct PUF-answer to open to. We solve this problem by asking the receiver to reveal both values (q_0, q_1) played in the OT protocol (along with the randomness used in the OT protocol), obviously only after S_{eq} has committed to the PUF-answer. Now, the simulator can: play the OT protocol with a random bit, commit to a random string (without querying the PUF), and then obtain both queries (q_0, q_1). In the decommitment phase, the simulator gets the actual bit b. Hence, it can query the PUF with input

Sender's Input: $b \in \{0,1\}$.

Commitment Phase

$\mathsf{R_{sh}}$: (Initialization and PUF exchange)

1. Create $\mathsf{PUF_R}$; obtain $a_0 \leftarrow \mathsf{PUF_R}(q_0)$, $a_1 \leftarrow \mathsf{PUF_R}(q_1)$, for $(q_0, q_1) \overset{\$}{\leftarrow} \{0,1\}^n$.
2. $(st_0, p_0) \leftarrow \mathsf{FuzGen}(a_0)$, $(st_1, p_1) \leftarrow \mathsf{FuzGen}(a_1)$.
3. Handover $\mathsf{PUF_R}$ to $\mathsf{S_{sh}}$.

$\mathsf{R_{sh}} \Leftrightarrow \mathsf{S_{sh}}$: (Statistical OT phase)
$\mathsf{R_{sh}}$ runs as the OT Sender with input (q_0, q_1), and $\mathsf{S_{sh}}$ runs as the OT Receiver with input b. Let q'_b be the local output of $\mathsf{S_{sh}}$.

$\mathsf{S_{sh}} \Leftrightarrow \mathsf{R_{sh}}$: (Statistically Hiding Commitment)
$\mathsf{S_{sh}}$ queries $\mathsf{PUF_R}$ on input q'_b and obtains a'_b. If $\mathsf{PUF_R}$ aborts, set $a'_b \overset{\$}{\leftarrow} \{0,1\}^l$. $\mathsf{S_{sh}}$ commits to a'_b using a statistically-hiding commitment scheme. Let c be the transcript of the commitment phase.

$\mathsf{S_{sh}} \Leftrightarrow \mathsf{R_{sh}}$: (Proof of knowledge of the Decommitment)
$\mathsf{S_{sh}}$ proves that he knows the decommitment of c running a statistical ZK Argument of Knowledge protocol. If the proof is not accepting, $\mathsf{R_{sh}}$ aborts.

Decommitment Phase

$\mathsf{S_{sh}}$: if $\mathsf{PUF_R}$ did not abort, send opening to a'_b to $\mathsf{R_{sh}}$.
$\mathsf{R_{sh}}$: if the opening for a'_b is accepting and $\mathsf{FuzRep}(a'_b, p_b) = st_b$ then accept b. Otherwise reject.

Fig. 2. Statistically Hiding Straight-Line Extractable Bit Commitment $\mathsf{Com_{sh}}$

q_b, obtain the PUF-answer, and equivocate the commitment so to open to such PUF-answer. There is a subtle issue here and is the possibility of *selective abort* of a malicious PUF. If the PUF aborts when queried with a particular string, then we have that the sender would abort already in the commitment phase, while the simulator aborts only in the decommitment phase. We avoid such problem by requiring that the sender continues the commitment phase by committing to a random string in case the PUF aborts. The above protocol is statistically binding (we are using Naor's commitment), straight-line extractable, and assuming that Naor's parameter was previously ad-hoc crafted, it is also straight-line equivocal. To commit to a bit we are committing to the l-bit PUF-answer, thus the size of Naor's parameter $\bar{r}$, is $N = (3n)l$. Protocol $\mathsf{Com_{eq}}$ is formally described in Fig. 3.

The final UC-secure commitment scheme $\mathsf{Com_{uc}} = (\mathsf{S_{uc}}, \mathsf{R_{uc}})$ consists of the coin-flipping phase, and the (equivocal) commitment phase. In the coin flipping, the receiver commits to α using the statistically hiding straight-line extractable commitment scheme $\mathsf{Com_{sh}}$. The output of the coin-flipping is the Naor's parameter $\bar{r}{=}\alpha \oplus \beta$ used as common input for the extractable/equivocal commitment scheme $\mathsf{Com_{eq}}$. Protocol $\mathsf{Com_{uc}} = (\mathsf{S_{uc}}, \mathsf{R_{uc}})$ is formally described in Fig. 4.

Both protocol $\mathsf{Com_{sh}}, \mathsf{Com_{eq}}$ require one PUF sent from the receiver to the sender. We remark that PUFs are transferred only *once at the beginning of the protocol.* We finally stress that we do not assume authenticated PUF delivery.

Sender's Input: Bit $b \in \{0,1\}$.
Common Input: $\bar{r} = (r_1, \ldots, r_l)$.

Commitment Phase

$\mathsf{R_{eq}}$: (Initialization and PUF exchange)

1. Create $\mathsf{PUF_R}$. Obtain $a_0 \leftarrow \mathsf{PUF_R}(q_0)$, $a_1 \leftarrow \mathsf{PUF_R}(q_1)$, for $(q_0, q_1) \overset{\$}{\leftarrow} \{0,1\}^n$.
2. $(st_0, p_0) \leftarrow \mathsf{FuzGen}(a_0)$, $(st_1, p_1) \leftarrow \mathsf{FuzGen}(a_1)$.
3. Handover $\mathsf{PUF_R}$ to $\mathsf{S_{eq}}$;
4. Choose random tape $\mathsf{ran_{OT}} \overset{\$}{\leftarrow} \{0,1\}^*$.

$\mathsf{R_{eq}} \Leftrightarrow \mathsf{S_{eq}}$: (Statistical OT phase)
$\mathsf{R_{eq}}$ runs as the OT Sender with input (q_0, q_1) using randomness $\mathsf{ran_{OT}}$, and $\mathsf{S_{eq}}$ runs as the OT Receiver with input b. Let q'_b be the local output of $\mathsf{S_{eq}}$. Let τ_{OT} be the transcript of the execution of the OT protocol.

$\mathsf{S_{eq}}$: (Statistically Binding Commitment)

1. $a'_b \leftarrow \mathsf{PUF_R}(q'_b)$. If $\mathsf{PUF_R}$ aborts, $a'_b \overset{\$}{\leftarrow} \{0,1\}^l$.
2. for $1 \leq i \leq l$, pick $s_i \overset{\$}{\leftarrow} \{0,1\}^n$, $c_i = G(s_i) \oplus (r_i \wedge a'_b[i])$. [a]
3. send $c_1, \ldots, c_l$ to $\mathsf{R_{eq}}$.

$\mathsf{R_{eq}}$: upon receiving $c_1, \ldots, c_l$, send $\mathsf{ran_{OT}}, q_0, q_1$ to $\mathsf{S_{eq}}$.
$\mathsf{S_{eq}}$: check if transcript τ_{OT} is consistent with $(\mathsf{ran_{OT}}, q_0, q_1, b)$. If the check fails abort.

Decommitment Phase

$\mathsf{S_{eq}}$: if $\mathsf{PUF_R}$ did not abort, send $((s_1, \ldots, s_l), a'_b), b$ to $\mathsf{R_{sh}}$.
$\mathsf{R_{eq}}$: if for all i, it holds that $(c_i = G(s_i) \oplus (r_i \wedge a'_b[i])) \wedge (\mathsf{FuzRep}(a'_b, p_b) = st_b)$ then accept. Else reject.

[a] where $(r_i \wedge a'_b[i])[j] = r_i[j] \wedge a'_b[i]$.

Fig. 3. Statistically Binding Straight-line Extractable/Equivocal Commitment $\mathsf{Com_{eq}}$

Namely, the privacy of the honest party is preserved even if the adversary interferes with the delivery process of the honest PUFs (e.g., by replacing the honest PUF).

Theorem 1. *If* $\mathsf{Com_{sh}} = (\mathsf{S_{sh}}, \mathsf{R_{sh}})$ *is a statistically hiding straight-line extractable commitment scheme in the malicious PUFs model, and* $\mathsf{Com_{eq}} = (\mathsf{S_{eq}}, \mathsf{R_{eq}})$ *is a statistically binding straight-line extractable and equivocal commitment scheme in the malicious PUFs model, then* $\mathsf{Com_{uc}} = (\mathsf{S_{uc}}, \mathsf{R_{uc}})$ *in Fig. 4, is a UC-secure commitment scheme in the malicious PUFs model.*

The above protocol can be used to implement the multiple commitment functionality $\mathcal{F}_{\mathsf{mcom}}$ by using independent PUFs for each commitment. Note that in our construction we can not reuse the *same* PUF when multiple commitments are executed *concurrently*[4]. The reason is that, in both sub-protocols $\mathsf{Com_{sh}}, \mathsf{Com_{eq}}$,

[4] Note that however our protocol enjoys parallel composition and reuse of the same PUF, i.e., one can commit to a string reusing the same PUF.

Sender's Input: Bit $b \in \{0,1\}$.

<u>Commitment Phase</u>

$\mathsf{R_{uc}} \Leftrightarrow \mathsf{S_{uc}}$: (Coin Flipping)

1. $\mathsf{R_{uc}}$ picks $\alpha \overset{\$}{\leftarrow} \{0,1\}^N$; commit to α by running as sender in protocol $\mathsf{Com_{sh}}$ while $\mathsf{S_{uc}}$ runs as receiver.
2. $\mathsf{S_{uc}}$ sends $\beta \overset{\$}{\leftarrow} \{0,1\}^N$ to $\mathsf{R_{uc}}$.
3. $\mathsf{R_{uc}}$ sends decommitment for α to $\mathsf{S_{uc}}$.
4. $\mathsf{S_{uc}}$: if the decommitment is not accepting, abort. Else $\mathsf{R_{uc}}$ and $\mathsf{S_{uc}}$ set $\bar{r} = \alpha \oplus \beta$.

$\mathsf{S_{uc}} \Leftrightarrow \mathsf{R_{uc}}$: (Equivocal Commitment)

$\mathsf{S_{uc}}$ commit to b by running as sender in $\mathsf{Com_{eq}}$ where $\mathsf{R_{uc}}$ runs as receiver, and the common input is $\bar{r}$.

<u>Decommitment Phase</u>

$\mathsf{S_{uc}}$ runs the decommitment phase of $\mathsf{Com_{eq}}$ to open b.
$\mathsf{R_{uc}}$ accepts if the decommitment of b is accepting.

Fig. 4. Computational UC Commitment Scheme $(\mathsf{S_{uc}}, \mathsf{R_{uc}})$

in the opening phase the sender forwards the answer obtained by querying the receiver's PUF. The answer of a malicious PUF can then convey information about the value committed in concurrent sessions that have not been opened yet.

When implementing $\mathcal{F}_{\mathsf{mcom}}$ one should also deal with malleability issues. In particular, one should handle the case in which the man-in-the-middle adversary forwards honest PUFs to another party. However such attack can be ruled out by exploiting the unpredictability of honest PUFs as follows. Let P_i be the creator of PUF_i, running an execution of the protocol with P_j. Before delivering its own PUF, P_i queries it with the identity of P_j concatenated with a random *nonce*. Then, at some point during the protocol execution with P_j it will ask P_j to evaluate PUF_i on such *nonce* (and the identity). Due to the unpredictability of PUFs, and the fact that *nonce* is a randomly chosen value, P_j is able to answer to such a query only if it *possesses* the PUF. The final step to obtain UC security for any functionality consists in using the compiler of [11], which only needs a UC secure implementation of the $\mathcal{F}_{\mathsf{mcom}}$ functionality.

4 Conclusion

We introduce the Malicious PUF model which models the very realistic attack of an adversary replacing a proper PUF with a "PUF-looking" device that implements an arbitrary malicious functionality. We show that in this model is possible to achieve UC-security relying on complexity-theoretic assumptions, by providing an implementation of UC-secure commitment scheme.

Acknowledgments. Research supported in part by NSF grants CCF-0916574; IIS-1065276; CCF-1016540; CNS-1118126; CNS-1136174; US-Israel BSF grant 2008411, OKAWA Foundation Research Award, IBM Faculty Research Award, Xerox Faculty Research Award, B. John Garrick Foundation Award, Teradata Research Award, Lockheed-Martin Corporation Research Award and the European Commission through the FP7 programme under contract 216676 ECRYPT II. This material is also based upon work supported by the Defense Advanced Research Projects Agency through the U.S. Office of Naval Research under Contract N00014-11-1-0392. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

The authors thank Margarita Vald for pointing out the problem with the original formulation of the [7] PUF ideal functionality where the adversary is not allowed to create PUFs. The authors also thank Marten van Dijk and Ulrich Rührmair for extensive discussions on their and our paper.

References

1. Agrawal, S., Goyal, V., Jain, A., Prabhakaran, M., Sahai, A.: New impossibility results for concurrent composition and a non-interactive completeness theorem for secure computation. In: Safavi-Naini, R. (ed.) CRYPTO 2012. LNCS, vol. 7417, pp. 443–460. Springer, Heidelberg (2012)
2. Armknecht, F., Maes, R., Sadeghi, A.-R., Standaert, F.-X., Wachsmann, C.: A formalization of the security features of physical functions. In: IEEE Symposium on Security and Privacy, pp. 397–412. IEEE Computer Society (2011)
3. Armknecht, F., Maes, R., Sadeghi, A.-R., Sunar, B., Tuyls, P.: Memory leakage-resilient encryption based on physically unclonable functions. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 685–702. Springer, Heidelberg (2009)
4. Barak, B., Canetti, R., Nielsen, J.B., Pass, R.: Universally composable protocols with relaxed set-up assumptions. In: Foundations of Computer Science (FOCS 2004), pp. 394–403 (2004)
5. Barak, B., Prabhakaran, M., Sahai, A.: Concurrent non-malleable zero knowledge. In: FOCS, pp. 345–354 (2006)
6. Boit, C., Helfmeier, C., Nedospasaov, D., Seifert, J.-P.: Private Communication (2013)
7. Brzuska, C., Fischlin, M., Schröder, H., Katzenbeisser, S.: Physically uncloneable functions in the universal composition framework. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 51–70. Springer, Heidelberg (2011)
8. Canetti, R., Dodis, Y., Pass, R., Walfish, S.: Universally composable security with global setup. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 61–85. Springer, Heidelberg (2007)
9. Canetti, R., Fischlin, M.: Universally composable commitments. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 19–40. Springer, Heidelberg (2001)
10. Canetti, R., Kushilevitz, E., Lindell, Y.: On the limitations of universally composable two-party computation without set-up assumptions. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 68–86. Springer, Heidelberg (2003)

11. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: 34th Annual ACM Symposium on Theory of Computing, Montréal, Québec, Canada, May 19-21. LNCS, pp. 494–503. ACM Press (2002)
12. Di Crescenzo, G., Ostrovsky, R.: On concurrent zero-knowledge with pre-processing (Extended abstract). In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 485–502. Springer, Heidelberg (1999)
13. Dodis, Y., Ostrovsky, R., Reyzin, L., Smith, A.: Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. SIAM J. Comput. 38(1), 97–139 (2008)
14. Eichhorn, I., Koeberl, P., van der Leest, V.: Logically reconfigurable pufs: memory-based secure key storage. In: Proceedings of the Sixth ACM Workshop on Scalable Trusted Computing, STC 2011, pp. 59–64. ACM, New York (2011)
15. Fischlin, M., Pinkas, B., Sadeghi, A.-R., Schneider, T., Visconti, I.: Secure set intersection with untrusted hardware tokens. In: Kiayias, A. (ed.) CT-RSA 2011. LNCS, vol. 6558, pp. 1–16. Springer, Heidelberg (2011)
16. Garg, S., Kumarasubramanian, A., Ostrovsky, R., Visconti, I.: Impossibility results for static input secure computation. In: Safavi-Naini, R. (ed.) CRYPTO 2012. LNCS, vol. 7417, pp. 424–442. Springer, Heidelberg (2012)
17. Goyal, V., Ishai, Y., Sahai, A., Venkatesan, R., Wadia, A.: Founding cryptography on tamper-proof hardware tokens. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 308–326. Springer, Heidelberg (2010)
18. Guajardo, J., Kumar, S.S., Schrijen, G.-J., Tuyls, P.: Fpga intrinsic pufs and their use for ip protection. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 63–80. Springer, Heidelberg (2007)
19. Kalai, Y.T., Lindell, Y., Prabhakaran, M.: Concurrent general composition of secure protocols in the timing model. In: 37th Annual ACM Symposium on Theory of Computing, pp. 644–653 (2005)
20. Katz, J.: Universally composable multi-party computation using tamper-proof hardware. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 115–128. Springer, Heidelberg (2007)
21. Katzenbeisser, S., Koccabas, Ü., Rozic, V., Sadeghi, A.-R., Verbauwhede, I., Wachsmann, C.: Pufs: Myth, fact or busted? a security evaluation of physically unclonable functions (pufs) cast in silicon. In: Prouff and Schaumont [30], pp. 283–301
22. Koccabas, Ü., Sadeghi, A.-R., Wachsmann, C., Schulz, S.: Poster: practical embedded remote attestation using physically unclonable functions. In: Chen, Y., Danezis, G., Shmatikov, V. (eds.) ACM Conference on Computer and Communications Security, pp. 797–800. ACM (2011)
23. Maes, R., Van Herrewege, A., Verbauwhede, I.: Pufky: A fully functional puf-based cryptographic key generator. In: Prouff and Schaumont [30], pp. 302–319
24. Maes, R., Verbauwhede, I.: Physically unclonable functions: A study on the state of the art and future research directions. In: Sadeghi, A.-R., Naccache, D. (eds.) Towards Hardware-Intrinsic Security, Information Security and Cryptography, pp. 3–37. Springer, Heidelberg (2010)
25. Naor, M.: Bit commitment using pseudo-randomness. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 128–136. Springer, Heidelberg (1990)
26. Ostrovsky, R., Scafuro, A., Visconti, I., Wadia, A.: Universally composable secure computation with (malicious) physically uncloneable functions. Cryptology ePrint Archive, Report 2012/143 (2012), `http://eprint.iacr.org/2012/143/`

27. Pappu, R.S., Recht, B., Taylor, J., Gershenfeld, N.: Physical one-way functions. Science 297, 2026–2030 (2002)
28. Pappu, R.S.: Physical One-Way Functions. PhD thesis, MIT (2001)
29. Prabhakaran, M., Sahai, A.: New notions of security: achieving universal composability without trusted setup. In: 36th Annual ACM Symposium on Theory of Computing, pp. 242–251 (2004)
30. Prouff, E., Schaumont, P.: CHES 2012. LNCS, vol. 7428. Springer, Heidelberg (2012)
31. Rührmair, U.: Oblivious transfer based on physical unclonable functions. In: Acquisti, A., Smith, S.W., Sadeghi, A.-R. (eds.) TRUST 2010. LNCS, vol. 6101, pp. 430–440. Springer, Heidelberg (2010)
32. Rührmair, U., Katzenbeisser, S., Busch, H.: Strong pufs: Models, constructions and security proofs. In: Sadeghi, A., Tuyls, P. (eds.) Towards Hardware Intrinsic Security: Foundations and Practice, pp. 79–96. Springer (2010)
33. Sadeghi, A.-R., Visconti, I., Wachsmann, C.: Enhancing rfid security and privacy by physically unclonable functions. In: Sadeghi, A.-R., Naccache, D. (eds.) Towards Hardware-Intrinsic Security, Information Security and Cryptography, pp. 281–305. Springer, Heidelberg (2010)
34. Tuyls, P., Batina, L.: Rfid-tags for anti-counterfeiting. In: Pointcheval, D. (ed.) CT-RSA 2006. LNCS, vol. 3860, pp. 115–131. Springer, Heidelberg (2006)
35. Vald, M.: Private Communication (2012)
36. van Dijk, M., Rührmair, U.: Physical unclonable functions in cryptographic protocols: Security proofs and impossibility results. IACR Cryptology ePrint Archive, 2012: 228 (2012)

How to Garble RAM Programs?[⋆]

Steve Lu[1] and Rafail Ostrovsky[2,⋆⋆]

[1] Stealth Software Technologies, Inc.
steve@stealthsoftwareinc.com
[2] Department of Computer Science and Department of Mathematics, UCLA
rafail@cs.ucla.edu

Abstract. Assuming solely the existence of one-way functions, we show how to construct Garbled RAM Programs (GRAM) where its size only depends on fixed polynomial in the security parameter times the program running time. We stress that we avoid converting the RAM programs into circuits. As an example, our techniques implies the first garbled *binary search* program (searching over sorted encrypted data stored in a cloud) which is poly-logarithmic in the data size instead of linear. Our result requires the existence of one-way function and enjoys the same non-interactive properties as Yao's original garbled circuits.

Keywords: Secure Computation, Oblivious RAM, Garbled Circuits.

1 Introduction

Often times, such as in cloud computation, one party wants to store some data remotely and then have the remote server perform computations on that data. If the client does not wish to reveal this data or the nature of the computation and the results of the computation to the remote server, then one must resort to using secure computation methods in order to process this remotely stored data. In other words, suppose two parties want to compute some program π on their private inputs without revealing to each other (or just one party) anything but the output. The earliest research in secure two-party computation modeled π as a circuit and was accomplished under Yao's Garbled Circuits [33] or the Goldreich-Micali-Wigderson [10] paradigm. Both of these approaches require the program π to be converted to a circuit. Even the recent work of performing secure computation via fully homomorphic encryption requires representing the program π as a circuit. However, many algorithms are more naturally and compactly represented as RAM programs, and converting these into circuits may lead to a huge blowup in program size and its running time.

Of course, there are known polynomial transformations between time-bounded RAM programs, time-bounded Turing Machines and circuits [8,27]: Given a T-time RAM program, [8] shows how one can transform it into a $O(T^3)$-time TM, and [27] shows how to transform a T-time TM into circuits of size $O(T \log T)$, which results in a $O(T^3 \log T)$ blowup. Our work aims at *circumventing* these transformation costs and executing RAM programs directly in a private manner, while retaining the same noninteractive properties as Yao's Garbled circuits. This goal is especially important for the

[⋆] Patent Pending.
[⋆⋆] Work done while consulting for Stealth Software Technologies, Inc.

case of complex real-world RAM programs with running time that is much larger than the input size. Unrolling these complicated RAM programs with multiple execution paths, recursion, multiple loops, etc. into a circuit makes the circuit size polynomially larger and often prohibitive.

It should be noted that our work is also important in practical applications where the sizes of the inputs are vastly different, such as database search, or where multiple queries against the same large data-set must be executed. When compiling a RAM program into a circuit, the compiled circuit must inherently be able to compute all execution paths of the RAM program. Thus, the circuit itself must be at least be as large as the input size, which in some applications may be is exponentially larger than execution path of the insecure solution (e.g. consider a binary search). One can argue that even if the circuit is large, we can "charge" the large circuit cost to the large input size, but in many cases this is unacceptable: consider the case where a large data is encrypted and uploaded *off-line*, such as a large database, and multiple encrypted queries are made *on-line*, where the insecure execution path is, for example, poly-logarithmic in the database size and we do not want to "pay" an on-line cost of circuit size which is linear in the database size.

An alternative approach for secure conversion of RAM programs into circuits is dynamic evaluation: even if the resulting circuit is large and the total size of the is resulting circuit is prohibitive, one can execute and even compile the large circuit dynamically and intelligently evaluate only parts of the circuit so as to "prune off" dead paths (e.g. short-circuiting techniques) to make the evaluation efficient, even in the case of large inputs. However, until now it was not known how to convert RAM programs into circuits which result in an efficient secure non-interactive execution in a way that does not reveal the execution path of the compiled RAM program. Naturally, using interaction, one can use the Goldreich-Micali-Wigderson [10] paradigm along with revealing bits along the way to help prune and determine execution path – however our ultimate goal is to explore the non-interactive garbling solutions for RAM programs without revealing the execution path.

Another alternative method for computing RAM programs without first converting them to circuits was proposed by Ostrovsky and Shoup [25] which used Oblivious RAM [11] as a building block. The Ostrovsky-Shoup compiler allows parties to execute Oblivious RAM programs directly, i.e., without first unrolling it into a circuit, which provided an alternative approach to secure RAM computation. The method was further improved by Gordon et al. [16] in order to perform sublinear amortized database search. Lu and Ostrovsky [21] considered two-server Oblivious RAM inside the Ostrovsky-Shoup compiler, which led to logarithmic overhead in both the computation and the communication complexity. Note that these three works allow secure RAM evaluation without having to unroll the program into a circuit and represent a different way to perform secure computation that reveals only the program running time. Among these, [21] is the best result for programs (instead of circuits) in terms of *computation complexity* and *communication complexity*. However, in terms of *round complexity*, these papers leave much to be desired: they all require at least one round for *each* CPU computation step, even using the so-called non-interactive RAM solution of [31], which reduces each read/write to one round between the client and the server. Since the running time of CPU is at least t steps for programs that run in time t, this leads to $\Omega(t)$ round complexity. In contrast, in this paper, we show how to retain poly-log overhead in communication and computation, and make the entire computation non-interactive in the OT-hybrid model, just like Yao.

1.1 The Blueprint for RAM Program Garbling

We describe our approach at a high level: we start with an ORAM compiler (with certain properties which we will describe later) that takes a program and converts it into an oblivious one. We call this new program the "ORAM CPU" because it can be thought of as a client running a CPU that performs a local computation followed by reading or writing something on the remote server. As a conceptual segue, consider the following change: instead of the ORAM CPU locally performing its computation, it creates a garbled circuit representing that computation, and also garbles all the inputs for that computation (the inputs are just the client state and the last fetched item, possibly with some randomness) and sends it to the server who then evaluates the circuit. The output of this computation is just the next state and the next read/write query, and the server preforms the read/write query locally, and sends back the result of the read/write query along with the state to the ORAM CPU. We emphasize that this is just a conceptual intermediate step, since this step does not actually give us any savings and possibly interferes with the security of the ORAM CPU by having its state revealed to the server.

Next, we change where the ORAM CPU state is stored: instead of letting the client hold it, it is stored on the server in garbled format. That is to say, the garbled circuit that the client sends to the server now outputs a garbled state instead of a regular state, which can then be used as input for the next ORAM CPU step. As long as the garbled circuit for the next CPU step uses the same input encoding as the one generated by our current CPU step, then the client does not need to interact with the server. However, the garbled CPU also performs read/write operations into ORAM memory that need to be carefully interleaved with our computations. We need to describe how this is done next.

Let us suppose that the ORAM compiler had the property that the ORAM CPU knows exactly when the contents of a memory location that it wants to read next was last written to (which is the case for many ORAM schemes). We attempt to perform the same strategy as we did with garbling the state: whenever the ORAM CPU wants to write something to memory. We store memory bits as Yao's garbled keys, based on the actual location, and the time last written. Thus, the bit stored in some particular location has one of the two garbled keys. However, this does not immediately work, because if each memory location uses a different encoding, the CPU circuit does not know which encoding to use when reading at some future time.

In order to resolve this, we construct a circuit that assists with this transition: the circuit takes as input a time step and memory location computes (in a garbled form) two possible encodings for 0/1 encoded in this location and outputs a garbled circuit encoded for that time step to "translate" keys stored in memory to keys needed by the CPU. Since this circuit does not require the knowledge of the memory location ahead of time, the client can generate as many of these as needed at the *start* of the computation. Indeed, if the ORAM program runs in t steps, the client can generate t of these circuits, garble them, and send them all to the server, non-interactively.

Note that we need Oblivious RAM with poly-log overhead where the client *size* is at most some fixed polynomial in the security parameter times some poly-log factor in n. This is because for every ORAM fetch operation, we also need to emulate the client's internal computation of the Oblivious RAM using garbled circuit, which incurs a multiplicative overhead in the size and the running time of the client. Thus, the smaller the client of Oblivious RAM, the more efficient our solution is: in order to achieve poly-log overhead, all Oblivious RAM schemes where client memory is larger than poly-logarithmic (e.g. [9,6]) is not useful for our purposes. We expand on the intuition in Section 3.1. In Section 3 we give the main construction for garbled RAM programs. When combined with oblivious transfer, this gives a one-round secure two-party RAM program computation in the semi-honest model (which can be extended to multi-party using the Beaver-Micali-Rogaway paradigm[2]), which we discuss in Section 4. In the full version [20], we also show how to construct a single-round ORAM.

1.2 Related Work on Secure RAM Computation

Oblivious RAM was introduced in the context of software protection by Goldreich and Ostrovsky [11]. In the original work by Goldreich [9], a solution was given with $O(\sqrt{n})$ and communication overhead where lookups could be done in a single round and $O(2^{\sqrt{\log n \log\log n}})$ communication overhead for a recursive solution. Subsequently, Ostrovsky [23,24] gave a solution with only poly-log overhead and constant client memory (the so-called "hierarchical solution").

Subsequent to Goldreich and Ostrovsky [23,24,9,11], works on Oblivious RAM (e.g. [31,32,26,13,14,28,15,18,29]) looked at improving the concrete and asymptotic parameters of Oblivious RAM. The notion of *Private Information Storage* introduced by Ostrovsky and Shoup [25] allows for private storage and retrieval of data, and was primarily concentrated in the information theoretic setting. This model differs from Oblivious RAM in the sense that, while the communication complexity of the scheme is sub-linear, the server performs a *linear* amount of work on the database. The work of Ostrovsky and Shoup [25] gives a multi-server solution to this problem in both the computational and the information-theoretic setting and introduces the Ostrovsky-Shoup compiler of transforming Oblivious RAM into secure RAM computation. The notion of single-server "PIR Writing" was subsequently formalized in Boneh, Kushilevitz, Ostrovsky and Skeith [5] where they provide a single-server solution. The case of amortized "PIR Writing" of multiple reads and writes was considered in [7].

With regard to secure computation for RAM programs, the implications of the Ostrovsky-Shoup compiler was explored in the work of Naor and Nissim [22] which shows how to convert RAM programs into so-called circuits with "lookup tables" (LUT). The Ostrovsky-Shoup compiler was further explored in the work of Gordon et al. [16] in the case of amortized programs. Namely, consider a client that holds a small input x, and a server that holds a large database D, and the client wishes to repeatedly perform private queries $f(x, D)$. In this model, an expensive initialization (depending only on D) is first performed. Afterwards, if f can be computed in time T with space S with a RAM machine, then there is a secure two-party protocol computing f in time $O(T) \cdot \mathsf{polylog}(S)$ with the client using $O(\log S)$ space and the server using $O(S \cdot \mathsf{polylog}(S))$ space. The secure RAM computation solution of Lu and Ostrovsky [21] can be viewed as a generalization of the [25] model where servers must also perform sublinear work.

1.3 Our Results

In this paper, we show how to garble any Random Access Machine (RAM) Program π_t that runs in time upper bounded by t while keeping all the non-interactive advantages of the Yao's Garbled Circuit approach. More specifically, we present a program garbling method which consists of a triple of polynomial-time algorithms (G, GI, GE). G takes as input any RAM program π_i that includes an upper bound t on its running time and a pseudorandom function (PRF) family F and a seed s for PRF of size k (a security parameter) and outputs a garbled program $\Pi_t = G(\pi_t, t, F, s)$, where all inputs are polynomial in the security parameter. Just like gabled circuits, we provide a way to garble any input x for π_t into Garbled Input $X = GI(x, s)$, and an algorithm to evaluate a garbled program on garbled inputs $GE(\Pi_t, t, X)$. The correctness requirement is that for any x, π_t, F, s it holds that $\pi_t(x) = GE(G(\pi_t, t, F, s), GI(x, s))$ with the security guarantee that nothing about x is revealed except its running time t, expressed in terms of computational indistinguishability ($\approx$) between the simulator Sim and garbled outputs. So far, the above description matches Yao's garbled circuit description. The difference is both in the running time and the size of garbled program for our new garbling method.

Main Theorem. *Assume one-way functions exist, and let the security parameter be k and let F be a PRF family based on the one-way function. Then, there exists a Program Garbling triple of poly-time algorithms G, GI, GE such that for any t any π_t and any input x of length n we have the following.*

Correctness: $\forall x, \pi_t, F, s$: $\pi_t(x) = GE\left[G(\pi_t, t, F, s), GI(x, s)\right]$.
Security: $\exists$ *poly-time simulator Sim, such that $\forall \pi, t, x, s$, where $|s| = k$,*

$[G(\pi_t, t, F, s), GI(x, s)] \approx Sim\left[1^k, t, |x|, \pi_t(x)\right]$.

Garbled Program Size: *The size of the garbled program*

$|G(\pi_t, t, F, s)| = O\left((|\pi| + t) \cdot\ k^{O(1)} \cdot \mathit{polylog}(n)\right)$.

Garbled Input Size: *Let $|x| = n$ and $|s| = k$. $\forall x, s$ the garbled input size*

$|GI(x, s)| = O\left(n \cdot k^{O(1)} \cdot \mathit{polylog}(n)\right)$.

Our main construction is a garbled program based on any one-way function (or a block-cipher), and is time-compact in the sense that if the original program runs in t time and has size n, our garbled RAM runs in $O(t \cdot poly(k, \log n))$.

1.4 Remarks

– **Making programs and outputs private.** We note that similar to Yao, we can make π_t to be a time-bounded **universal program** u_t, (i.e., an interpreter) and $x = (\pi'_t, y)$ include both time-bounded program π'_t and input y, so that $u_t(x) = \pi'_t(y)$. Part of the specification of π'_t may also include masking its output – i.e. to have output blinded (XORed) with a random string. That allows, just like Yao, to keep both the program and the output hidden from a machine that evaluates the garbled program. Such a modification has been utilized in the literature (see, e.g. [1]).

- **Reactive functionalities.** Our result shows that we can first garble a large input x, $|x| = n$ with garbled input size equal to $O(|x| \cdot k^{O(1)} \cdot \mathsf{polylog}(n))$ so that later, given private programs $\pi^1_{t_1}, \ldots, \pi^j_{t_j}, \ldots$ for polynomially many programs where program π^j runs in time t_j and potentially modifies x, (e.g., database updates) we can garble and execute all of these programs just revealing running times t_i, and nothing else. The size of each garbled program remains $O\left((|\pi^i| + t_i) \cdot k^{O(1)} \cdot \mathsf{polylog}(n)\right)$. It is also easy to handle the case where the length of x changes, provided that an upper bound by how much each program changes the length of x is known prior to garbling of next program.
- **Cloud computing.** As an example of the power of our result we outline secure cloud computation/delegation. In this simple application one party has an input and wants to store it remotely and then repeatedly run different private programs on this data. Reactive functionalities allow us to do this with one important restriction: we do not give the server a choice in adaptively selecting the inputs: but this is not an issue as the server itself has no inputs to the program. The other possible problem is if the programs themselves are contrived and circularly reference the code for the garbling algorithm. Such programs would be highly unnatural to run on data and so we disallow them in our setting.
- **Two-party computation.** Note that just like in Yao's garbled circuits, in order to transmit the garbled inputs corresponding to input bits held by a different party for the sake of secure two-party computation, one relies on Oblivious Transfer (OT) that can be done non-interactively in the OT-hybrid model. Here, we insist that the OT-selected inputs to our garbled program are committed to prior to receiving the RAM garbled program, i.e. non-adaptively [3].
- **Optimizations.** We remark that step two of our blueprint is applicable to almost all ORAM schemes with small CPU as follows: instead of collapsing in the hierarchical Oblivious RAMs multiple rounds of a single read/write to a single round, we can implement our step 2 directly for each round of each read/write (e.g. even inside a single read/write simulation of Oblivious RAM that requires multiple rounds) of the underlying Oblivious RAM: by implementing an oracle call for each Oblivious RAM CPU read/write using our method of compiling memory fetch "on the fly" into garbled circuits. Any Oblivious RAM where the CPU can tell precisely when any memory location was overwritten last can be complied using our approach. (We call such Oblivious RAMs "predictive memory" RAMs and explore this further in the full version.) For example, this property holds for [18] ORAM. It also allows a generic method to "collapse" all multi-round predictive memory Oblivious RAM with small CPU into a single round. Observe that the overall complexity for garbling programs depends both on the CPU complexity and the ORAM read/write complexity.
- **Tighter Input Compactness.** Using an ORAM scheme that has small input encoding and small size CPU (such as [18]) we can also make Input Compactness in our main theorem tighter: for all programs we can make garbled inputs to be $O(nk)$, where recall that n is the input size and k is the security parameter. We remark that if we wish to garble only "large" programs that run time at least $\Omega(n \cdot \log n \cdot k^{O(1)})$,

we can make Input Compactness even better under the assumption that one can encode inputs to garbled circuits to be of size $O(n + k)$ and have the garbled program "unpack" the inputs to the full $O(nk)$ size. Such packing techniques for have been recently developed for garbling the inputs of garbled circuits by Ishai and Kushilevitz [17].

– **Stronger Adversarial models.** As already mentioned we describe the scheme in the honest-but-curious model based on honest-but-curious Yao, and only in the non-adaptively secure setting (see [3] for further discussion of adaptivity.) There is a plethora of works that convert Yao's garbled circuits from honest-but-curious to malicious setting, as well strengthening its security in various settings. Since our machinery is build on top of Yao's garbled circuits (and Obvious RAMs that work in the fully adaptive setting), many of these techniques for stronger guarantees for Yao's garbled circuit apply in a straightforward manner to our setting as well. We postpone description of malicious models to the full version.

2 Preliminaries

2.1 Oblivious RAM

We work in the RAM model with stored programs, where there is a CPU that can run a program that performs a sequence of reads or writes to locations stored on a large memory. This machine, which we will refer to as the CPU or the client, can be viewed as a stateful[1] processor with only a few special data registers that store program counters, query counters, and cryptographic keys (primarily a seed for a PRF) and that CPU can run small programs which model a single CPU step. Given the CPU state Σ and the most recently read element x, $CPU(\Sigma, x)$ does simple operations such as addition, multiplication, updating program counter, or executing PRF followed by producing the next read/write command as well as updating to the next state Σ'.

Because we wish to hide the type of access performed by the client, we unify both types of accesses into a operation known as a *query*. A sequence of n queries can be viewed as a list of (memory location, data) pairs $(v_1, x_1), \ldots, (v_n, x_n)$, along with a sequence of operations $op_1, \ldots, op_n$, where op_i is a READ or WRITE operation. In the case of READ operations, the corresponding x value is ignored. The sequence of queries, including both the memory location and the data, performed by a client is known as the *access pattern*.

In our model, we wish to obliviously simulate the RAM machine with a client, which can be viewed as having limited storage, that has access to a server. However, the server is untrusted and assumed to malicious. An oblivious RAM is *secure* if the view of a any malicious server can be simulated in poly-time in a way that is indistinguishable from the view of the server during a real execution. For concreteness, we focus on sequence of buffers $B_k, B_{k+1}, \ldots, B_L$ of geometrically increasing sizes. Typically $k = O(1)$ (the first buffer is of constant size) and $L = \log n$ (the last buffer may contain all n elements), where n is the total number of memory locations. These buffers are

[1] We can consider a *stateless* version where all registers are stored in memory. For ease of exposition, we let the client hold local state.

standard bucketed hash tables, with buckets of size b. We refer the reader to [11] for more information.

2.2 Yao's Garbled Circuits

Garbled circuits were introduced by Yao [33]. A series of works looked at proving the security and formalizing the notions of garbled circuits, including Lindell and Pinkas [19], and recently, the work of Bellare et al. [4]. We refer the reader to the latter work for more details, and we briefly summarize the key properties.

A circuit garbling scheme we view as a triple of algorithms (G, GI, GE) where $G(1^k, C)$ takes as input a security parameter k and circuit C and outputs some garbled circuit Γ and garbling key gsk. $GI(x, gsk)$ converts an input x and a gsk into a garbled input X, and $GE(\Gamma, X)$ evaluates a garbled circuit on an garbled input.

We first make an observation that the labels (keys) on a given wire used in a garbled circuit can be re-used in additional newly generated gates, as long as the value does not change between the uses and it is not revealed whether this label represents 0 or 1. (For example, assume that garbled circuit evaluator is given a label on some input wire, which is a key representing a 0 or a 1. We claim that the same key can be used as input key for other garbled circuits that are generated later.) This observation allows us to execute garbled circuits in "parallel" or "sequentially" where some labels are re-used. Indeed, this observation is implicitly used in classic garbled circuits in gates where the fan-out is greater than 1: all outgoing wires share the same labels (see e.g. Footnote 8 in Lindell-Pinkas [19]).

Lemma 1. *Suppose $\mathcal{C}$ and $\mathcal{C}'$ are two circuits and suppose there is some input x for which we want to compute $\mathcal{C}(x)$ and $\mathcal{C}'(x)$ (resp. $\mathcal{C}(\mathcal{C}'(x))$). Suppose the wires $w_0, \ldots, w_n$ in $\mathcal{C}$ represent the input wires for x and similarly define $w'_0, \ldots, w'_n$ represent the input wires of x in $\mathcal{C}'$ (resp. $v'_0, \ldots, v'_n$ be the output wires of $\mathcal{C}'$). Let $k^b_{w_i}$ represent the label indicating wire $w_i = b$, and let C and C' be randomly garbled into $GC(\mathcal{C})$ and $GC(\mathcal{C}')$ under the restriction that $k^b_{w_i} = k^b_{w'_i}$ (resp. $k^b_{w_i} = k^b_{v'_i}$). Then the tuple $(GC(\mathcal{C}), GC(\mathcal{C}'), \{k^{x_i}_{w_i}\}^n_{i=0})$ can be computationally simulated.*

Proof. Consider the composite circuit $D = \mathcal{C}||\mathcal{C}'$ (resp. $E = \mathcal{C} \circ \mathcal{C}'$) which is just a copy of $\mathcal{C}$ and a copy of $\mathcal{C}'$ in parallel (resp. sequence). Then every garbling of D induces a garbling of $\mathcal{C}$ and $\mathcal{C}'$ with the restriction exactly as above. By the security of garbled circuits, there exists a simulator that can simulate $(GC(D), \{k^{x_i}_{w_i}\}^n_{i=0})$. We can construct a simulator for our lemma by simply taking this simulator and taking the output and separate out $GC(\mathcal{C})$ and $GC(\mathcal{C}')$, as the lemma requires.

Remark: If the data is encrypted bit by bit using Yao's keys, Lemma 1 allows us to run arbitrary garbled circuits on this data, akin to general purpose "function evaluation" on encrypted data. This observation itself has a number of applications, we describe these in the full version of the paper.

3 Non-interactive Garbled RAM Programs

3.1 Informal Description of Main Ideas

We consider the RAM model of computation as in the works of [11,23,24] where a RAM program along with data is stored in memory, and a small, stateful CPU with a $O(1)$ instruction set that can store $O(1)$ words that can be of size $\mathsf{polylog}(n) = poly(k)$ where k is the security parameter. Our starting point is a ORAM model that can tolerate fully malicious *tampering* adversary (see [24,11]). Each step of the CPU is simply a read/write call to main memory followed by executing its next CPU instruction. We now summarize our ideas for building Garbled RAM programs from an Oblivious RAM program.

In order to garble a RAM program π_t, we consider the two fundamental operations separately and show how to mesh them together: 1) Read/Write (v, x) from/to memory. 2) Execute an instruction step to update state and produce next read/write query: $\Sigma', \textsc{Read/Write}(v', x') \leftarrow CPU(\Sigma, x)$. Updating the state can include updating local registers, incrementing program counters and query counters, and updating cryptographic keys.

Our goal is to transform this into a *non-interactive* process by letting the client send the server enough garbled information to evaluate the program up to t steps, where t upper bounds the RAM program running time. We give some intuition as to how to construct a circuit for each step, and then how to garble them. The first part will be modeled as the circuit $\mathcal{C}_{ORAM}$, and the second part will be modeled as the circuit $\mathcal{C}_{CPU}$. The circuits satisfy a novel property: the *plain circuit* $\mathcal{C}_{ORAM}$ emulates a query for the ORAM client and outputs a bit representation of a garbled circuit GC_{ORAM}. This GC_{ORAM} has output encodings that will be compatible with the *garbled circuit* $GC(\mathcal{C}_{CPU})$ to evaluate a garbled the CPU's next step. We remark that GC_{ORAM} actually contains several sub-circuits, but is written as a single object for ease of exposition. If we generate t of these garbled circuits, then a party can evaluate a t-time garbled RAM program by consuming one garbled $\mathcal{C}_{ORAM}$ and one garbled $\mathcal{C}_{CPU}$ per time step.

We first consider the circuit $\mathcal{C}_{CPU}$, which is straightforward to describe. This circuit takes as input Σ representing the internal state of the CPU, and x the last memory contents read. Recall that the CPU performs a step $CPU(\Sigma, x)$ and updates the state to Σ' and gives the next read/write query to memory location v' and contents x'. In order to turn this into a circuit, we can sacrifice some efficiency and have a "universal" instruction in which we run *every* atomic instruction (from its constant sized instruction set) and simply multiplex the actual results using the instruction opcode. This universal instruction is modeled as a circuit which is of size $k^{O(1)}$. We remark that although this circuit is simple, the complexity arises from when we want to garble this circuit: the garbling must be done in a way so that the garbled inputs and outputs are compatible with GC_{ORAM}.

The circuit $\mathcal{C}_{ORAM}$ must emulate the client in Oblivious RAM (we can think of it as being a non-interactive client either by breaking out each individual step as a separate circuit, or using a non-interactive ORAM). The input of the circuit is just an ORAM read/write query[2], and the output of the circuit is **a bit representation that describes a set of garbled circuits, equivalent to what would have been produced via the ORAM client** which we call GC_{ORAM}. We give full details on the construction in Section 3.2. It is important that we argue that the result of this fetch can be combined with the evaluation of the CPU step. Observe that since the labels in our ORAM are generated as pseudo-random time-labeled encodings, so we know ahead of time only the encoding of the output (but know neither the input nor output) of the i-th invocation of the ORAM. Thus when garbling $\mathcal{C}_{CPU}$, the input encodings use exactly the output encodings from the respective outputs of the ORAM. Recall in our ORAM protocol the server sends back the encoded output to the client; here, we *do not* send it back, and instead keep the result and use it as input in the next CPU step (which is secure and correct via Lemma 1).

Then, putting it all together, to garble a RAM program π_t that runs in time t, the program garbling algorithm G generates t garbled $\mathcal{C}_{ORAM}$ and $\mathcal{C}_{CPU}$ circuits, and also encodes the initial state Σ_0 of the CPU with the program initialized, counters set to zero, and with fresh cryptographic keys. The full construction of G is given the next section, Section 3.2.

3.2 Main Construction of Garbled Programs

We first describe how to construct the algorithms G, GI, GE. Given a program π_t running in time t, we describe the algorithm G that converts it into a garbled program Π_t. In order to do so, we follow the two steps outlined above and we consider the construction of a circuit that performs an ORAM query $\mathcal{C}_{ORAM}$ and a circuit that runs one CPU step $\mathcal{C}_{CPU}$.

Our garbling algorithm G will provide enough garbled circuits to execute t steps of a program π_t. Each step is a garbled RAM query (done obliviously via ORAM) followed by a garbled CPU computation. It starts with a garbled encoding of the initial state Σ_0 of the CPU with the program π_t initialized, counters set to zero, and with fresh cryptographic keys. For each of the t time steps, it creates a garbled $GC(\mathcal{C}_{ORAM})$ for a read/write of that time step, then a garbled $GC(\mathcal{C}_{CPU})$ to perform a CPU step. We show how to construct $\mathcal{C}_{ORAM}$ and $\mathcal{C}_{CPU}$ such that they can be garbled and interleaved. We will show that this garbling is independent of the actual program path, regardless of what memory locations have been fetched, and is correct and secure.

First, we describe $\mathcal{C}_{ORAM}$ to mimic an oblivious read/write access to main memory. For this, it can just perform the steps in our Oblivious RAM, with one difference: G does not know ahead of time which memory location will be used. Hence, in order to overcome this, the circuit $\mathcal{C}_{ORAM}$ must take a memory location *as input* and internally formulate what the ORAM client computes. $\mathcal{C}_{ORAM}$ outputs what the "virtual" ORAM client would have sent to the server: a garbled circuit GC_{ORAM} representing a read/write query. The novelty in this construction is that when we feed a memory

[2] Since the ORAM client uses randomness as well as time-labeled encodings (which are outputs of the PRF), we will allow these to be *inputs* to $\mathcal{C}_{ORAM}$, so that they may be pre-computed "for free" rather than computed via the circuit. The circuit consumes these inputs in order to generate the output garbled circuit *without* having to evaluate these itself.

location v into $\mathcal{C}_{ORAM}$, the output precisely is a garbled ORAM read/write query relative to that memory location. In order to hide v, both $\mathcal{C}_{ORAM}$ and v are garbled into $GC(\mathcal{C}_{ORAM})$ and V respectively, and by the correctness of garbled evaluation, the output is still GC_{ORAM}. By the security of the underlying ORAM, this output GC_{ORAM} can actually be simulated.

Although it is a circuit that outputs another circuit, there is no circularity in this construction: given a query location and some fixed randomness, the behavior of the ORAM client is completely deterministic, straight-line, and takes $k^{O(1)} \cdot \mathsf{polylog}(n)$ steps, so the output can be represented by a circuit also of that size. This ORAM client is independent of the main program CPU which only uses ORAM as an "oracle". We emphasize this again, because G will most likely be ran by a client, G does not play the role of the ORAM client but rather *emulates* the ORAM client via $\mathcal{C}_{ORAM}$, so this is

Inputs: An ORAM query to read/write (v, x) and a query number ℓ. This circuit interprets the client performing the ℓ-th ORAM query, which uses randomness and time-labeled encodings based on ℓ. As such, this circuit also takes these randomness bits and pre-computed encodings as inputs.
Output: A garbled circuit GC_{ORAM} representing a read/write ORAM query.
Circuit Description: We describe the functionality of the circuit $\mathcal{C}_{ORAM}$. We recall our algorithm for a ORAM query. Using time-labeled encodings via PRFs, it generates a set of $|B_1| + 2L - 2$ garbled $GC(\mathcal{C}_{match})$ which has hard-coded location information built into it, with corresponding garbled $GC(\mathcal{C}_{next})$ circuits, and one final $GC(\mathcal{C}_{write})$ garbled circuit for writing the element back to the top level (and possibly an update circuit). Although the ORAM client evaluates these PRFs internally, we *do not* encode this as part of our circuit $\mathcal{C}_{ORAM}$, but rather we "consume" them as input. Similarly, the ORAM client must use randomness, which we also consume from the input of $\mathcal{C}_{ORAM}$.

1. For the top level, B_1, for each bucket, $\mathcal{C}_{ORAM}$ creates a time-labeled garbled circuit $GC(\mathcal{C}_{match})$ consuming the input encodings to be used as garbled labels.
2. For subsequent levels $i = 2 \dots L$:
   (a) The circuit $\mathcal{C}_{ORAM}$ computes $q_i^0 = h_i(v)$ and consumes q_i^1 from the input (the input itself is uniformly random)
   (b) Consume two secret keys for encryption sk_i^0 and sk_i^1 from the input and create a garbled circuit $GC(\mathcal{C}_{next})$
   (c) Create two time-labeled garbled circuits $GC(\mathcal{C}_{match})$, one that searches for w in bucket q_i^0 encrypted under sk_i^0, and one that searches for w in bucket q_i^1 encrypted under sk_i^1, again consuming the encoding from the input to $\mathcal{C}_{ORAM}$.
3. $\mathcal{C}_{ORAM}$ also creates a garbled $GC(\mathcal{C}_{write})$ that writes the result back to the first empty position the top level buffer B_k.
4. If ℓ is a multiple of $|B_1|$, then a reshuffle step is performed using the time-labeled garbled update circuit $GC(\mathcal{C}_{update})$.
5. The combined set of garbled circuits is referred to as GC_{ORAM}.

We point out that throughout this entire process, every time a query circuit is created, G increments ℓ in order to keep track of the time-labeled encodings required by the $\mathcal{C}_{ORAM}$ circuits.

Fig. 1. The ORAM Client Circuit $\mathcal{C}_{ORAM}$

not a client attempting to capture its own logic in a circuit. We provide a pseudocode description of $\mathcal{C}_{ORAM}$ in Figure 1.

Looking ahead, G will garble this circuit and ensure that the output of an ORAM query has the same encoding as that used to garble $\mathcal{C}_{CPU}$. The algorithm G can then garble both $\mathcal{C}_{CPU}$ and $\mathcal{C}_{ORAM}$ ahead of time, without having to know the memory location.

Next, we consider building the circuit which performs a single CPU step in the RAM program, $\mathcal{C}_{CPU}$ that is supposed to perform Σ', READ/WRITE$(v', x') \leftarrow CPU(\Sigma, x)$. In order to hide which instruction is being executed, we build the circuit to take an instruction opcode and we run every single-step instruction *from its constant sized instruction set (not all possible program paths)* of the CPU. The circuit multiplexes the actual results using the instruction opcode. This universal instruction is modeled as a circuit which is of size $k^{O(1)}$ and is independent of the ORAM circuit, independent of the queried locations, and independent of the current running time.

One may ask the question: How can this circuit be interleaved with the $\mathcal{C}_{ORAM}$ circuit if it is independent of it?

The answer is that when G garbles $\mathcal{C}_{CPU}$, the encoding will depend on the output of $\mathcal{C}_{ORAM}$ in the previous time-step. Note that this construction is not circular as each garbling only depends on the previous one, leading up to a total of t time steps. This can be done because G knows the encoding of the *output encoding* (but not the output) of the Oblivious RAM query, which *does not depend* on the location queried. This output encoding is then used for the input parameter encoding for $GC(\mathcal{C}_{CPU})$. We provide a pseudocode description of G in Figure 2.

Inputs: A program π_t with an upper bound on running time t, and a pseudo-random function family F along with a key s.
Algorithm Description: The algorithm G is performed as follows. It creates an encoding of the initial state of the CPU, Σ_0 with the program π_t initialized. It also encodes an initial program counter and cryptographic keys. We show how to construct $\mathcal{C}_{ORAM}$ and $\mathcal{C}_{CPU}$ such that they can be garbled and interleaved across t time steps. We must argue that this garbling is independent of the actual program path, regardless of what memory locations have been fetched, and is correct and secure.
For each time step $i = 1 \ldots t$, G creates:

1. A garbled read/write query circuit $GC(\mathcal{C}_{ORAM})$ for performing query number i on some (unknown variable) garbled location V_i (and X_i in the case of a write). G pre-computes randomness and PRF evaluations and hardwires them. Although G does not know the eventual output, it knows the *encoding* of it, which is *independent of the queried location.* It uses this encoding for the following:
2. A garbled instruction circuit $GC(\mathcal{C}_{CPU})$ with input wires of X_i using the encoding from above, and the input wires of Σ_i using the output encoding from the previous CPU step. The output is a garbled location V_{i+1} (and X_{i+1} in the case of a write) to be used in the next read/write query and an garbled updated state Σ_{i+1}.

Fig. 2. Program Garbling Algorithm G

Inputs: A garbled program Π_t with garbled input X.
Algorithm Description: The algorithm GE is performed as follows. It first stores the initial encoded program state and inputs into memory. Then, for each time step $i = 1 \ldots t$, GE performs:

1. Evaluate the garbled query circuit $GC(\mathcal{C}_{ORAM})$ on a garbled memory location V_i. The output is GC_{ORAM} which itself is a garbled circuit that represents a read/write query in our ORAM protocol. Execute the query playing the role of the server to obtain some garbled output X_i which is kept locally instead of sent to the client.
2. Evaluate the garbled instruction circuit $GC(\mathcal{C}_{CPU})$ on garbled inputs X_i and Σ_i. Obtain a new read/write query V_{i+1}.

After t steps, output the final value X_{t+1}.

Fig. 3. Garbled Program Evaluation Algorithm GE

The algorithm GI for garbling an input of size n is just the time-labeled encodings starting from wherever the RAM program expects the inputs to be located.

The algorithm GE used to evaluate a garbled program Π_t on garbled inputs evaluates the garbled circuit $GC(\mathcal{C}_{ORAM})$, then executing the garbled instruction $GC(\mathcal{C}_{CPU})$ one at a time, up to t times. The process is precisely performing the same steps as G except evaluating garbled circuits instead of generating them. In addition, once it gets the garbled ORAM query, it must also execute it as well. We provide a pseudocode description of G in Figure 3.

3.3 Main Result

We now state our main result:

Theorem 1. *Assume one-way functions exist, and let the security parameter be k and let F be a PRF family based on the one-way function. Then, there exists an efficient Program Garbling triple of algorithms G, GI, GE such that for any π_t any t and any input x of length n, we have the following.*

Correctness: $\forall x, \pi_t, F, s$:
$\pi_t(x) = GE\left[G(\pi_t, t, F, s), GI(x, s)\right]$.
Security: $\exists$ *poly-time simulator Sim, such that* $\forall \pi, t, x, s$, *where*
$|s| = k \left[G(\pi_t, t, F, s), GI(x, s)\right] \approx Sim\left[1^k, t, |x|, \pi_t(x)\right]$.
Program Size: *The size of the garbled program*
$|G(\pi_t, t, F, s)| = O\left((|\pi| + t) \cdot k^{O(1)} \cdot \mathsf{polylog}(n)\right)$.
Input Size: *Let* $|x| = n$ *and* $|s| = k$. $\forall x, s$ *the garbled input size*
$|GI(x, s)| = O\left(n \cdot k^{O(1)} \cdot \mathsf{polylog}(n)\right)$.

Proof. We give an outline of the proof of security, and refer the reader to the full version [20] for the full proofs.

Security. We design the simulator Sim as follows. We know that the server performs the following:

1. Evaluate the garbled query circuit $GC(\mathcal{C}_{ORAM})$ on a garbled memory location V_i. The output is GC_{ORAM} which itself is a garbled circuit that represents a read/write query in the underlying ORAM.
2. Execute the garbled ORAM query GC_{ORAM} playing the role of the server to obtain some garbled output X_i which is kept locally instead of sent to the client.
3. Evaluate the garbled instruction circuit $GC(\mathcal{C}_{CPU})$ on garbled inputs X_i and Σ_i. Obtain a new read/write query V_{i+1}.

The underlying Oblivious RAM is secure and uses time-labeled garbled circuits and encodings and can be simulated by some Sim_{ORAM}. Furthermore, the underlying Yao's garbled circuits are secure, and can be simulated by some Sim_{Yao}. Thus, the access pattern of the ORAM can be simulated even for tampering adversary, and we need only show that the garbled circuit emulating the ORAM client $GC(\mathcal{C}_{ORAM})$ and garbled instructions $GC(\mathcal{C}_{CPU})$ can also be simulated. The garbled circuits can be interleaved securely due to Lemma 1, and the time-labeled encodings themselves are just outputs of a PRF. By the security of Yao's garbled circuits and the underlying PRF, these can be simulated securely.

4 Application to Secure RAM Computation

We give an example application in which only one party has input and wants to repeatedly run programs on this data. Such is the case of secure cloud computing, where someone stores data in the cloud and then later runs computations against that data. We emphasize that in this setting, there is no issue of adaptivity because the server has no inputs. In the typical setting of two-party secure computation, we deal with this by making the server first perform OTs to retrieve its inputs *before* the client sends the garbled program. In the multi-party setting, the technique can be utilized in the Beaver-Micali-Rogaway paradigm [2] to achieve constant-round MPC with the same approach as in [2] but with garbled RAM programs. That is to say, in this application, a client wishes to store some data x on a remote server and then run various RAM programs on x without the server learning the results of the programs or x itself. Of course, the client could always ignore the server altogether and run all the programs on x locally, so we are envisioning a scenario in which the client does not want to carry around all of its data locally and wants to only store a few cryptographic keys or counters. To apply Garbled RAM programs to this application, the client first garbles the input x to get $X = GI(x)$ and sends it to the server. Then for each program the client wants to run, it recalls the encoding of the previous output and creates a garbled program using the labels of the previous output as inputs for the current program.

5 Conclusions and Open Problems

Recently, Goldwasser at. al. [12] have shown how to construct a *reusable* Garbled Yao. It is tempting to plug it into our construction to achieve reusable GRAM with compactness proportional to program size and independent of its running time. The idea is to compute poly-many iterations of the CPU computation using reusable Yao (instead of sending fresh garbled circuit for each CPU step) where CPU computes its own garbled keys for each step. This is possible only if there exists poly-time reusable circular-secure Garbled Yao with input encoding of size independent of the circuit size. Constructing such a gadget is an interesting open problem even under non-standard assumptions.

Acknowledgements. We thank Oded Goldreich and Daniel Wichs for very helpful discussions and the anonymous reviewers for their comments.

References

1. Applebaum, B., Ishai, Y., Kushilevitz, E.: From secrecy to soundness: Efficient verification via secure computation. In: Abramsky, S., Gavoille, C., Kirchner, C., Meyer auf der Heide, F., Spirakis, P.G. (eds.) ICALP 2010. LNCS, vol. 6198, pp. 152–163. Springer, Heidelberg (2010)
2. Beaver, D., Micali, S., Rogaway, P.: The round complexity of secure protocols (extended abstract). In: STOC, pp. 503–513 (1990)
3. Bellare, M., Hoang, V.T., Rogaway, P.: Adaptively secure garbling with applications to one-time programs and secure outsourcing. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 134–153. Springer, Heidelberg (2012)
4. Bellare, M., Hoang, V.T., Rogaway, P.: Foundations of garbled circuits. In: ACM Conference on Computer and Communications Security, pp. 784–796 (2012)
5. Boneh, D., Kushilevitz, E., Ostrovsky, R., Skeith III, W.E.: Public key encryption that allows PIR queries. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 50–67. Springer, Heidelberg (2007)
6. Boneh, D., Mazieres, D., Popa, R.A.: Remote oblivious storage: Making oblivious RAM practical. CSAIL Technical Report, MIT-CSAIL-TR-2011-018 (2011)
7. Chandran, N., Ostrovsky, R., Skeith III, W.E.: Public-key encryption with efficient amortized updates. In: Garay, J.A., De Prisco, R. (eds.) SCN 2010. LNCS, vol. 6280, pp. 17–35. Springer, Heidelberg (2010)
8. Stephen, A.: Cook and Robert A. Reckhow. Time bounded random access machines. Journal of Computer and System Sciences 7(4), 354–375 (1973)
9. Goldreich, O.: Towards a theory of software protection and simulation by oblivious RAMs. In: STOC, pp. 182–194 (1987)
10. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or a completeness theorem for protocols with honest majority. In: STOC, pp. 218–229 (1987)
11. Goldreich, O., Ostrovsky, R.: Software protection and simulation on oblivious RAMs. J. ACM 43(3), 431–473 (1996)
12. Goldwasser, S., Kalai, Y., Popa, R.A., Vaikuntanathan, V., Zeldovich, N.: Succinct functional encryption and applications: Reusable garbled circuits and beyond. Cryptology ePrint Archive, Report 2012/733 (2012)
13. Michael, T.: Goodrich and Michael Mitzenmacher. Privacy-preserving access of outsourced data via oblivious RAM simulation. In: Aceto, L., Henzinger, M., Sgall, J. (eds.) ICALP 2011, Part II. LNCS, vol. 6756, pp. 576–587. Springer, Heidelberg (2011)
14. Goodrich, M.T., Mitzenmacher, M., Ohrimenko, O., Tamassia, R.: Oblivious RAM simulation with efficient worst-case access overhead. In: CCSW, pp. 95–100 (2011)
15. Goodrich, M.T., Mitzenmacher, M., Ohrimenko, O., Tamassia, R.: Privacy-preserving group data access via stateless oblivious ram simulation. In: SODA, pp. 157–167 (2012)

16. Gordon, S.D., Katz, J., Kolesnikov, V., Krell, F., Malkin, T., Raykova, M., Vahlis, Y.: Secure two-party computation in sublinear (amortized) time. In: ACM Conference on Computer and Communications Security, pp. 513–524 (2012)
17. Ishai, Y., Kushilevitz, E.: Personal communication (2012)
18. Kushilevitz, E., Lu, S., Ostrovsky, R.: On the (in)security of hash-based oblivious RAM and a new balancing scheme. In: SODA, pp. 143–156 (2012)
19. Lindell, Y., Pinkas, B.: A proof of security of yao's protocol for two-party computation. J. Cryptology 22(2), 161–188 (2009)
20. Lu, S., Ostrovsky, R.: How to garble RAM programs. Cryptology ePrint Archive, Report 2012/601 (2012)
21. Lu, S., Ostrovsky, R.: Distributed oblivious RAM for secure two-party computation. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 377–396. Springer, Heidelberg (2013)
22. Naor, M., Nissim, K.: Communication preserving protocols for secure function evaluation. In: STOC, pp. 590–599 (2001)
23. Ostrovsky, R.: Efficient computation on oblivious RAMs. In: STOC, pp. 514–523 (1990)
24. Ostrovsky, R.: Software Protection and Simulation On Oblivious RAMs. PhD thesis, Massachusetts Institute of Technology, Dept. of Electrical Engineering and Computer Science (June 1992)
25. Ostrovsky, R., Shoup, V.: Private information storage (extended abstract). In: STOC, pp. 294–303 (1997)
26. Pinkas, B., Reinman, T.: Oblivious RAM revisited. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 502–519. Springer, Heidelberg (2010)
27. Pippenger, N., Fischer, M.J.: Relations among complexity measures. J. ACM 26(2), 361–381 (1979)
28. Shi, E., Chan, T.-H.H., Stefanov, E., Li, M.: Oblivious RAM with $O((\log N)^3)$ Worst-case Cost. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 197–214. Springer, Heidelberg (2011)
29. Stefanov, E., Shi, E., Song, D.: Towards practical oblivious RAM. In: NDSS (2012)
30. Wichs, D.: Personal Communication (March 2013)
31. Williams, P., Sion, R.: Single Round Access Privacy on Outsourced Storage. In: ACM CCS, pp. 293–304 (2012)
32. Williams, P., Sion, R., Carbunar, B.: Building castles out of mud: practical access pattern privacy and correctness on untrusted storage. In: ACM Conference on Computer and Communications Security, pp. 139–148 (2008)
33. Yao, A.C.-C.: Protocols for secure computations (extended abstract). In: FOCS, pp. 160–164 (1982)

Author Index