Masayuki Abe (Ed.)

LNCS 6477

Advances in Cryptology – ASIACRYPT 2010

16th International Conference on the Theory and Application of Cryptology and Information Security Singapore, December 2010, Proceedings

Springer

Lecture Notes in Computer Science 6477

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Masayuki Abe (Ed.)

Advances in Cryptology - ASIACRYPT 2010

16th International Conference on the Theory
and Application of Cryptology and Information Security
Singapore, December 5-9, 2010
Proceedings

Volume Editor

Masayuki Abe
3-9-11 Midori-cho, Musashino-shi, Tokyo 180-8585, Japan
E-mail: abe.masayuki@lab.ntt.co.jp

Library of Congress Control Number: 2010939472

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

LNCS Sublibrary: SL 4 – Security and Cryptology

ISSN 0302-9743
ISBN-10 3-642-17372-1 Springer Berlin Heidelberg New York
ISBN-13 978-3-642-17372-1 Springer Berlin Heidelberg New York

Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper 06/3180

Preface

ASIACRYPT 2010 was held in the Swissôtel Merchant Court in Singapore, during December 5–9, 2010. The conference was sponsored by the International Association for Cryptologic Research (IACR) in cooperation with the Coding and Cryptography Research Group of Nanyang Technological University. It was also supported by the Singapore Tourism Board, and co-sponsored by the National Research Foundation of Singapore, Lee Foundation, IBM Singapore Ltd., O'Connor's Singapore Ltd., Puffersoft Ltd., Privylink Ltd., Hewlett-Packard Singapore Ltd., Jardine OneSolution Ltd., and Singapore Mathematical Society. San Ling chaired the conference and I served as the Program Chair.

There were 216 valid submissions. The Program Committee aided by 221 external reviewers spent 83 days on reviews and discussions. They spared no effort to increase the quality of their reviews. Every paper received at least three independent reviews, and papers from the committee members received five reviews. In total, there were more than 730 reviews followed by intensive discussion. This long and tough process, wrapped up with an intensive face-to-face meeting by the committee members convened at UC Santa Barbara, yielded 35 accepted papers. I regret not being able to select more of such high-quality papers due to space limitations. The proceedings include the revised versions of the accepted papers. The authors are fully responsible for their contents.

The best paper award was given to "Rotational Rebound Attacks on Reduced Skein" by Dmitry Khovratovich, Ivica Nikolić, and Christian Rechberger. There were a further two best papers, "Improved Single-Key Attacks on 8-Round AES-192 and AES-256" by Orr Dunkelman, Nathan Keller, and Adi Shamir, and "Efficient Public-Key Cryptography in the Presence of Key Leakage" by Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs, that were solicited for full version submission to the *Journal of Cryptology*. The conference program included two invited talks: "Cryptography, from Theory to Practice: A Personal Perspective" by Hugo Krawczyk, and "Cryptographic Hash Functions and the SHA-3 Competition" by Bart Preneel.

There are many people I would like to acknowledge but only a few can be listed here. First I would like to thank all the authors of the submitted papers. I am deeply grateful to all the members of the Program Committee for their expertise and enthusiasm that brought success to a difficult project. I also want to express appreciation to the external reviewers listed in the following pages. Special thanks to Shai Halevi for providing and setting up the splendid review software, and Huaxiong Wang and his staff at Nanyang Technological University, who helped me to manage the review process in many ways. Finally, I am indebted to Kaoru Kurosawa, Mitsuru Matsui, Nigel Smart, and Tatsuaki Okamoto, who gave me invaluable advice as Chairs of past IACR conferences.

September 2010 Masayuki Abe

ASIACRYPT 2010

The 16th Annual International Conference on the Theory and Application of Cryptology and Information Security

December 5–9, 2010, Singapore

Sponsored by
the International Association for Cryptologic Research (IACR)

in cooperation with
the Coding and Cryptography Research Group of
Nanyang Technological University

General Chair

San Ling	Nanyang Technological University, Singapore

Program Chair

Masayuki Abe	NTT Information Sharing Platform Laboratories, Japan

Program Committee

Claude Carlet	University of Paris 8, France
Jean-Sébastien Coron	University of Luxembourg, Luxembourg
Yevgeniy Dodis	New York University, USA
Marc Fischlin	Darmstadt University of Technology, Germany
Henri Gilbert	Orange Labs, France
Dennis Hofheinz	Karlsruhe Institute of Technology, Germany
Thomas Johansson	Lund University, Sweden
Antoine Joux	DGA and Université de Versailles, UVSQ PRISM, France
Jonathan Katz	University of Maryland, USA
Lars R. Knudsen	Technical University of Denmark, Denmark
Kaoru Kurosawa	Ibaraki University, Japan
Xuejia Lai	Shanghai Jiao Tong University, China
Dong Hoon Lee	Korea University, Korea
Anna Lysyanskaya	Brown University, USA
Vadim Lyubashevsky	Tel Aviv University, Israel
Mitsuru Matsui	Mitsubishi Electric, Japan

Payman Mohassel	University of Calgary, Canada
Phong Q. Nguyen	INRIA and ENS, France
Jesper Buus Nielsen	Aarhus University, Denmark
Kaisa Nyberg	Helsinki University of Technology, Finland
Elisabeth Oswald	University of Bristol, UK
Renato Renner	ETH Zürich, Switzerland
Vincent Rijmen	K. U. Leuven, Belgium and TU Graz, Austria
Thomas Shrimpton	Portland State University, USA
Nigel P. Smart	University of Bristol, UK
François-Xavier Standaert	UCL, Belgium
Ron Steinfeld	Macquarie University, Australia
Willy Susilo	University of Wollongong, Australia
Vinod Vaikuntanathan	Microsoft Research, USA
Serge Vaudenay	EPFL, Switzerland
Hoeteck Wee	Queens College, CUNY, USA
Hongjun Wu	I2R and NTU, Singapore
Kan Yasuda	NTT Corporation, Japan
Hong-Sheng Zhou	University of Connecticut, USA

External Reviewers

Michel Abdalla	David Cash	Dejan Dukaric
Johan Aberg	Pierre-Louis Cayrel	Frederic Dupuis
Shweta Agarwal	Rafik Chaabouni	Nico Döttling
Martin Agren	Nishanth Chandran	Xiwen Fang
Hadi Ahmadi	Jung Hee Cheon	Sebastian Faust
Amy Alford	Joo Yeon Cho	Serge Fehr
Joel Alwen	Kwantae Cho	Matthieu Finiasz
Elena Andreeva	Kyu Young Choi	Dario Fiore
Frederik Armknecht	Sherman Chow	Matthias Fitzi
Nuttapong Attrapadung	Cheng-Kang Chu	Manuel Forster
Man Ho Au	Ji Young Chun	David Mandell Freeman
Paul Baecher	Iwen Coisel	Eiichiro Fujisaki
Joonsang Baek	Cas Cremers	Jun Furukawa
Kfir Barhum	Yang Cui	Martin Gagne
Aurélie Bauer	Dana Dachman-Soled	Sebastian Gajek
Almut Beige	Özgür Dagdelen	Steven Galbraith
Andrey Bogdanov	Oscar Dahlsten	David Galindo
Sasha Boldyreva	Christophe DeCanniere	Viktor Galliard
Julia Borghoff	Alexander W. Dent	Sanjam Garg
Charles Bouillaguet	Cunsheng Ding	Praveen Gauravaram
Billy Brumley	Ning Ding	Valerie Umana Gauthier
Christina Brzuska	Christophe Doche	Craig Gentry
Sébastien Canard	Ming Duan	Mark Gondree
David Canright	Leo Ducas	Zheng Gong

Dov Gordon
Aline Gouget
Jens Groth
Sylvain Guilley
Fuchun Guo
Jian Guo
Risto Hakala
Goichiro Hanaoka
Kristiyan Haralambiev
Carmit Hazay
Mathias Herrmann
Fumitaka Hoshino
Jialin Huang
Qiong Huang
Xinyi Huang
Jung Yeon Hwang
Sebastiaan Indesteege
Tetsu Iwata
Ragesh Jaiswal
Marc Joye
Kimmo Järvinen
Eike Kiltz
Kitak Kim
Thorsten Kleinjung
Kazukuni Kobara
Tetsutaro Kobayashi
François Koeune
Vladimir Kolesnikov
Woo Kwon Koo
Takeshi Koshiba
Daniel Kraschewski
Hugo Krawczyk
Noboru Kunihiro
Minoru Kuribayashi
Mario Lamberger
Gregor Leander
Ji-Seon Lee
Kwangsu Lee
Gaëtan Leurent
Allison Lewko
Chao Li
Benoît Libert
Tingting Lin
Georg Lippold
Joseph K. Liu
Adriana Lopez-Alt
Yiyuan Luo
Avradip Mandal
Mark Manulis
Xianping Mao
Atefeh Mashatan
Willi Meier
Florian Mendel
Giacomo de Meulenaer
Tomislav Nad
Jorge Nakahara Jr
Kris Narayan
Gregory Neven
Takashi Nishide
Ryo Nishimaki
Geon Tae Noh
Ryo Nojima
Peter Sebastian Nordholt
Adam O'Neil
Wakaha Ogata
Maria Cristina Onete
Claudio Orlandi
Khaled Ouafi
Jong Hwan Park
Rafael Pass
Kenneth G. Paterson
Arpita Patra
Serdar Pehlivanoglu
Chris Peikert
Olivier Pereira
Ludovic Perret
Christophe Petit
Duong Hieu Phan
Le Trieu Phong
Krzysztof Pietrzak
Gilles Piret
Emmanuel Prouff
Elizabeth Quaglia
Nik Raub
Francesco Regazzoni
Renato Renner
Hyun Sook Rhee
Matthieu Rivain
Andrea Röck
Minoru Saeki
Amit Sahai
Yasuyuki Sakai
Yu Sasaki
Martin Schläffer
Dominique Schröder
Gil Segev
Gautham Sekar
Pouyan Sepehrdad
Yannick Seurin
Hovav Shacham
Siamak Fayyaz Shahandashti
Emily Shen
Barhum Kfir Shlomo
Adam Smith
Boyeon Song
Paul Stankovski
Damien Stehlé
John Steinberger
Xiaorui Sun
Daisuke Suzuki
Petr Sušil
Bjoern Tackmann
Qiang Tang
Aris Tentes
Stefano Tessaro
Søren S. Thomsen
Mehdi Tibouchi
Elmar Tischhauser
Tomas Toft
Marco Tomamichel
Deniz Toz
Wei-Lung Dustin Tseng
Toyohiro Tsurumaru
Antonino Tumeo
Berkant Ustaoglu
Yevgeniy Vahlis
Kerem Varıcı
Frederik Vercauteren
Panagiotis Voulgaris
Martin Vuagnoux
Huaxiong Wang
Yongtao Wang
Bogdan Warinschi
Jian Weng

Steve Williams
Severin Winkler
Stefan Wolf
Qianhong Wu
Go Yamamoto
Scott Yilek
Kazuki Yoneyama
Yu Yu
Tsz Hon Yuen
Erik Zenner
Zhifang Zhang
Jinmin Zhong
Bo Zhu

Table of Contents

Hash Attacks

Symmetric-Key Cryptosystems

Block and Stream Ciphers

Protocols

Key Exchange

Foundation

Zero-Knowledge

Lattice-Based Cryptography

Secure Communication and Computation

Models, Notions, and Assumptions

Public-Key Encryption

Rotational Rebound Attacks on Reduced Skein

Dmitry Khovratovich[1,2], Ivica Nikolić[1], and Christian Rechberger[3]

[1] University of Luxembourg
[2] Microsoft Research Redmond, USA
[3] Katholieke Universiteit Leuven, ESAT/COSIC, and IBBT, Belgium
dkhovrat@microsoft.com, ivica.nikolic@uni.lu,
christian.rechberger@esat.kuleuven.be

Abstract. In this paper we combine a recent rotational cryptanalysis with the rebound attack, which results in the best cryptanalysis of Skein, a candidate for the SHA-3 competition. The rebound attack approach was so far only applied to AES-like constructions. For the first time, we show that this approach can also be applied to very different constructions. In more detail, we develop a number of techniques that extend the reach of both the inbound and the outbound phase, leading to cryptanalytic results on an estimated 53/57 out of the 72 rounds of the Skein-256/512 compression function and the Threefish cipher.

The new techniques include an analytical search for optimal input values in the rotational cryptanalysis, which allows to extend the outbound phase of the attack with a precomputation phase, an approach never used in any rebound-style attack before. Further we show how to combine multiple inside-out computations and neutral bits in the inbound phase of the rebound attack, and give well-defined rotational distinguishers as certificates of weaknesses for the compression functions and block ciphers.

Keywords: Skein, hash function, rotational cryptanalysis, rebound attack, distinguisher.

1 Introduction

Rotational cryptanalysis and the rebound attack proved to be very effective in the analysis of SHA-3 candidates and related primitives. Rotational cryptanalysis succeeded in the analysis of Addition-Rotation-XOR primitives (ARX), particularly in reduced variants of Threefish [10], Shabal [1], BMW [18]. Rebound attack, first presented in [16], is mostly aimed at byte-oriented primitives with a SPN structure. It gives the best attacks so far on reduced variants of the SHA-3 candidates Grøstl and ECHO [16,15], LANE [14], Cheetah [22] and the hash function Whirlpool [12], among others.

In this paper we introduce the combination of these two attacks with the application to the Skein compression function. We start with a number of preliminaries in Section 2. Our attacks will be based on methods to show non-random properties. For this we need definitions and bounds for distinguishers, which we give in

M. Abe (Ed.): ASIACRYPT 2010, LNCS 6477, pp. 1–19, 2010.

Section 3. There we introduce the rotational collision set property for n-bit compression functions and ideal ciphers, and demonstrate a lower bound about $q \cdot 2^n$ for the complexity of finding such set of size q in the black-box approach.

Then we proceed to the analysis of Skein and Threefish. We provide a much more careful and precise estimation of rotational probabilities compared to [10]. We represent the propagation of the rotational property analytically, and derive necessary conditions on the key bits to enlarge the rotational probability. We also correct [10] in terms of the independence assumptions, and find the best values of key bits with optimized search. Although we attack the tweaked version of Threefish [8], we stress that our attack is well applicable to the first version, and even benefits from more from the better diffusion the tweaked rotation constants provide.

This analysis gives us a simple rotational distinguisher for Threefish on up to 44 rounds. We advance even further and show how to put the rotational property into the outbound phase of the recent powerful *rebound attack*. The inner part of the rebound attack, the inbound phase, is accelerated with the method of the auxiliary path [9] and neutral bits [3]. In contrast to the first attacks on Skein, where auxiliary paths were used in the differential attacks, we show how to involve them into the rotational attack. As a result, we get a rotational distinguisher for the reduced Skein compression function. We attack 53 rounds of Skein-256 and 57 rounds of Skein-512 (Section 4).

Our results demonstrate substantial weaknesses both in the reduced Threefish cipher and the Skein compression function. The designers of Skein do not directly address the security of these primitives in the model that we consider, although the security of Threefish against all "standard attacks" is claimed. Also, our attacks show that the reduced Threefish does not behave as an ideal cipher, which is essential for the Skein security proofs. Had Skein have the reduced Threefish inside, the indifferentiability from the random oracle property of the Skein hash would be violated.

2 Preliminaries

2.1 Description of Skein

Skein is a family of hash functions, based on the block cipher Threefish of which the following versions are relevant for the SHA-3 proposal: Threefish-256 — 256-bit block cipher with 256-bit key and Threefish-512 — 512-bit block and key. Both the internal state I and the key K consist of N_w ($N_w = 4, 8$ for Threefish-256,-512, respectively) 64-bit words. The N_w words of the s-th subkey K^s are defined as follows:

$$\begin{aligned} K^s_j &= K_{(s+j) \bmod (N_w+1)}, \quad 0 \le j \le N_w - 4; \\ K^s_{N_w-3} &= K_{(s+N_w-3) \bmod (N_w+1)} + t_{s \bmod 3}; \\ K^s_{N_w-2} &= K_{(s+N_w-2) \bmod (N_w+1)} + t_{(s+1) \bmod 3}; \\ K^s_{N_w-1} &= K_{(s+N_w-1) \bmod (N_w+1)} + s, \end{aligned}$$

where s is a round counter, t_0 and t_1 are tweak words, and

Table 1. Summary of the attacks on Skein and Threefish

Rounds	Attack	Method	Reference
Skein/Threefish-256 (72 rounds)			
24*	Key recovery	Related-key differential	[7]
39	Key recovery	Related-key rotational	[10]
53	Distinguisher	Rotational rebound	Section 4
Skein/Threefish-512 (72 rounds)			
25*	Key recovery	Related-key differential	[7]
33*	Key recovery	Related-key boomerang	[5]
35*	Key recovery	Known-related-key distinguisher	[2]
42	Distinguisher	Related-key rotational	[10]
57	Distinguisher	Rotational rebound	Section 4

* — the attack was designed for the untweaked version.

$$t_2 = t_0 + t_1, \quad K_{N_w} = \lfloor 2^{64}/3 \rfloor \oplus \bigoplus_{j=0}^{N_w-1} K_j.$$

The formal description of internal rounds is as follows. Let N_r be the number of rounds ($N_r = 72$ for Threefish-256,-512). Then for every $1 \le d \le N_r$

- If $d \bmod 4 = 1$ add a subkey by setting $I_j \leftarrow I_j + K_j^{d/4}$;
- For $0 \le j < N_w/2$ set $(I_{2j}, I_{2j+1}) \leftarrow \mathrm{MIX}((I_{2j}, I_{2j+1}))$;
- Apply the permutation π on the state words.

At the end, a subkey $K^{N_r/4}$ is added. The operation MIX has two inputs x_0, x_1 and produces two outputs y_0, y_1 with the following transformation:

$$y_0 = x_0 + x_1$$
$$y_1 = (x_1 \lll_{R_{(d \bmod 8)+1,j}}) \oplus y_0$$

The exact values of the rotation constants $R_{i,j}$ as well the permutations π (which are different for each version of Threefish) can be found in [7]. We note that the rotation constants were changed in the Skein tweak [8], and we attack the new version although a similar analysis is applicable to the old version as well.

The compression function $F(H_{i-1}, M_i)$ of Skein is defined as:

$$F(H_{i-1}, M_i) = E_{H_{i-1},T_i}(M_i) \oplus M_i,$$

where $E_{K,T}(P)$ is the Threefish cipher, H_{i-1} is the previous chaining value, T_i is the tweak, and M_i is the message block.

The best known analysis of Skein is rotational distinguishers on the underlying Threefish cipher [10], which attack 39 rounds of Skein-256 and 42 rounds of Skein-512 (see Table 1).

2.2 Rotational Cryptanalysis

The main idea of the rotational cryptanalysis is to consider a pair of words where one is a rotation of the other. The $(X, \overleftarrow{X})$ is called a *rotational pair* [with a rotation amount r], where $\overleftarrow{X}$ the rotation of X by r bits to the left. A rotational pair is preserved by any bitwise transformation, particularly by the bitwise XOR and by any rotation. The probability that the rotational pair comes out of the addition is given by the following formula[6]

$$\mathbf{P}(\overleftarrow{x+y} = \overleftarrow{x} + \overleftarrow{y}) = \frac{1}{4}(1 + 2^{r-n} + 2^{-r} + 2^{-n}).$$

For large n and small r we get the following table:

r	p_r	$\log_2(p_r)$
1	0.375	−1.415
2	0.313	−1.676
3	0.281	−1.831

For $r = n/2$ the probability is close to 1/4. The same holds for rotations to the right. When an addition of rotational inputs does not produce rotational outputs then we say that the addition produced a rotational error.

The use of constants can violate the rotational property. Yet, if the constants are rotational as well, then the property is preserved, i.e. if $C = \overleftarrow{C}$ then $\overleftarrow{X} \oplus C = \overleftarrow{X \oplus C}$.

Rotational analysis deals with constants by introducing rotational corrections in pairs of inputs:

$$(X, \overleftarrow{X}_{\text{modified}}).$$

Then the rotational path is constructed so that the pre-fixed corrections and the errors from the failed modular addition compensate the errors from the use of constants.

We stress that in order to apply the rotational attack for the full scheme, all its inputs must be rotational pairs [with corrections].

2.3 Rebound Attack

The rebound attack [13,16] was described as a variant of differential cryptanalysis optimized to the cryptanalysis of hash functions, and at the same time can be seen as a high-level model for hash function cryptanalysis. So far it was mainly applied to AES-like constructions because of the simple way useful truncated differential characteristics can be found in them for a number of rounds.

The rebound attack is aimed to construct solutions for the most expensive part of a truncated differential trail. In the *inbound phase*, which covers only a few rounds, we construct solutions that connect low-weight input and output differences. In the *outbound phase* these solutions are propagated through the other rounds in both directions.

3 Rotational Distinguishers

In order to convincingly argue that a particular attack algorithm indeed shows non-random behavior of a hash function or a compression function, we need to argue that an attacker with only a black-box access to an ideal primitive of the same domain and range is not able to produce the same behavior with the same or better effort and probability.

Next in this section, we define a basic rotational distinguisher with corrections and give bounds on complexity of the resulting problems. Any shortcut algorithm will have to beat those bounds in order to make a convincing case for an attack. To do this, we adapt two known distinguisher concepts. The q-multicollision distinguisher of [4] will be the basis for a rotational distinguisher with corrections fixed by the attacker.

3.1 Rotational Distinguishers with Fixed Corrections

Due to the presence of counters, the rotational input pairs in Skein never convert to rotational output pairs. However, low-weight corrections applied to the input pairs, admit such a conversion:

$$\text{Skein}(\overleftarrow{X} \oplus e) \overset{\mathbb{P}}{=} \overleftarrow{\text{Skein}(X)},$$

where Skein is the compression function F, with reasonably high probability. We say that X is a *rotational collision* for function f, if

$$f(\overleftarrow{X}) = \overleftarrow{f}\,(X \oplus e).$$

When the rotational correction is not fixed, the rotational collision search complexity is given by an equivalent of the birthday paradox and is about $2^{n/2}$.

However, we provide a stronger distinguisher for the Skein compression function F, which asks for a set of rotational collisions with the same correction e:

$$\begin{cases} \overleftarrow{F(X_1)} = F(\overleftarrow{X_1} \oplus e); \\ \overleftarrow{F(X_2)} = F(\overleftarrow{X_2} \oplus e); \\ \dots \\ \overleftarrow{F(X_q)} = F(\overleftarrow{X_q} \oplus e). \end{cases}$$

Since the value of e is defined from the first equation, each new rotational collision costs about 2^n for a random function, and less for the Skein compression function as we show in the further text.

However, we prove the advantage of our distinguisher in a more strong setting by taking into account the fact that the Skein compression function is built on a block cipher $E_K(P)$:

$$F(IV, M) = E_{IV}(M) \oplus M.$$

We admit corrections only in the IV, so a rotational collision is formulated as

$$\overleftarrow{F(IV,M)} = F(\overleftarrow{IV} \oplus e, \overleftarrow{M}) \iff$$
$$\iff \overleftarrow{E_{IV}(M)} \oplus \overleftarrow{M} = E_{\overleftarrow{IV}\oplus e}(\overleftarrow{M}) \oplus \overleftarrow{M} \Leftrightarrow \overleftarrow{E_{IV}(M)} = E_{\overleftarrow{IV}\oplus e}(\overleftarrow{M}).$$

Thus the appropriate definition is as follows.

Definition 1. *A set*

$$\{e;\ (P_1,K_1),\ (P_2,K_2),\ldots,(P_q,K_q))\}$$

is called a rotational q-collision set *for a cipher* $E_K(\cdot)$ *if*

$$\begin{cases} \overleftarrow{E_{K_1}(P_1)} = E_{\overleftarrow{K_1}\oplus e}(\overleftarrow{P_1}); \\ \overleftarrow{E_{K_2}(P_2)} = E_{\overleftarrow{K_2}\oplus e}(\overleftarrow{P_2}); \\ \ldots \\ \overleftarrow{E_{K_q}(P_q)} = E_{\overleftarrow{K_q}\oplus e}(\overleftarrow{P_q}). \end{cases}$$

We follow the line of the first attack on the full AES [4] and compare the problem of finding a rotational collision set for an ideal cipher with that for reduced Threefish. Our results demonstrate that the versions of Threefish that we consider do not behave like an ideal cipher, and, thus, does not provide required security level for the Skein mode of operation (i.e., violate the random oracle property).

The complexity of the generic attack in measured in the number of queries to the encryption and decryption oracles of an ideal cipher.

Lemma 1. *To construct a rotational q-collision set for an ideal cipher with an n-bit block an adversary needs at least $O(q \cdot 2^{\frac{q-2}{q+2}n})$ queries on the average.*

Proof. The proof is similar to the proof of the multicollision lemma in [4]. We provide only a sketch of it.

First, we show that a rotational collision set is uniquely determined by $q+1$ query parameters. Then for any such set we compute the probability that it gives a collision set. The exact formula depends on the total number L of queries and their configuration, but the lower bound is

$$L \geq O(q \cdot 2^{\frac{q-2}{q+2}n})$$

4 Rotational Rebound Attack on Skein

4.1 Overview

Our attack consists of three parts: an inbound phase, an acceleration phase, and an outbound phase. In the inbound phase we prepare enough rotational pairs of

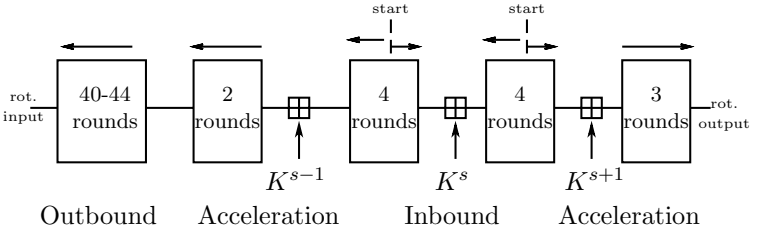


Fig. 1. The complete rotational rebound attack on Skein-256, -512

Table 2. Structure of the rebound attack on Skein

Outbound		Acceleration I	Inbound	Acceleration II
Rounds	Probability	Rounds	Rounds	Rounds
Skein-256 (53 rounds)				
3-42	2^{-244}	43-44	45-52	53-55
Skein-512 (57 rounds)				
3-46	2^{-495}	47-48	49-56	57-59

states for the outbound phase. The acceleration phase speeds up the outbound phase. An illustration of the attack proposal is given Fig. 1), while also given in Table 2.

The probability of the outbound phase depends on the values of particular key bits (see details in Section 4.4). As a result, we put global conditions on the keys, which are given in Tables 3 and 4.

For the distinguisher, we produce many M and K, such that

$$E_{\overleftarrow{K}\oplus e}(\overleftarrow{M}) = \overleftarrow{E_K(M)},$$

where E is the Threefish-256 reduced to rounds 2-54 (2-58 for the 512-bit version). For the Skein compression function, we produce many M, IV, and T such that

$$F(\overleftarrow{IV||T} \oplus e, \overleftarrow{M}) = \overleftarrow{F(IV||T, M)}$$

for the same e. The total complexity is about 2^{244} per pair in Skein-256, and 2^{495} per pair in Skein-512. Therefore, we are able to construct a set of rotational collisions for the Skein compression function with complexity lower than for a random function. Also, we can construct a rotational q-collision set for the cipher Threefish with complexity lower than for an ideal cipher. This proves the relevance of our attack.

Table 3. Pre-fixed values of key bits in Skein-256. The middle 58 bits of k_i coincide (with regard to the rotation) in related keys.

	K_0	K_1	K_2	K_3	K_4
K	0111..10	0100..11	0011..10	0000..11	0101..01
$\overleftarrow{K} \oplus e$	11..0011	00..1010	11..0110	00..1001	01..0011

Table 4. Pre-fixed values of key bits in Skein-512

	K_0	K_1	K_2	K_3	K_4	K_5	K_6	K_7	K_8
K	0111..01	0100..01	0011..01	0000..01	0111..10	0000..01	0011..01	0000..01	0001..10
$\overleftarrow{K} \oplus e$	11..0011	00..0010	11..0010	00..0001	11..0011	00..0010	11..0010	00..0001	01..0101

4.2 Inbound Phase

The inbound phase can be seen as the inner loop of the attack algorithm. The goal is to use all degrees of freedom available to efficiently provide enough starting points for the outbound part. The details depend on the variant of Skein considered, the choice of round key additions that are covered by the inbound phase, etc. In the following we describe the technique in a way that is independent of such details.

Let us consider 8 consecutive rounds. The addition of the round key K^s in the middle will be our matching point. We enumerate a large set of internal states both before and after the round key addition such that (1) the expected rotational trail is followed in the 8 rounds, and (2) it is possible to compute a subkey K^s that matches the global constraints set up for later phases of the attack, and connects those two internal states. In experiments we found that by simply forcing a part (less than a quarter of the bits) of the state to a particular value can lead to pairs following a rotational trail with probability 1 for 3-5 rounds in forward direction. For the inbound phase we actually need less. Two rounds in forward direction and backwards direction is enough for both chunks of 4 rounds we operate on independently. In addition, for those two rounds, many differentials exist that allow for manipulation of the outputs of those 4-round chunks in a way that resembles message modification techniques in MD5 or SHA-1 [21,20]. To connect those chunks of 4-round computations, we use the degrees of freedom in the choice of the subkey K_s. The global conditions

On the other hand, note that this does not fully determine the key yet, as the compression function also has a tweak input which serves as another source for degrees of freedom. This leaves some control over subkeys K^{k+1} and K^{k-1}.

4.3 Acceleration Phase

The acceleration phase of the attack may be seen as part of the inbound phase or part of the outbound phase. Technically, starting from here computations

are done in an inside-out manner, yet remaining degrees of freedom are used to accelerate the search for right pairs in the outbound phase.

As soon as we get a right pair of computations for the inbound phase, we produce many more of them from the given one as follows. We follow the simple idea of neutral bits as e.g. applied in the analysis of SHA-0 and SHA-1 [3]. We view them as auxiliary path [9] (also formalized as tunnels or submarines in [11,19,17]) and apply the differences specified by the path to the key and the tweak.

The configuration of the auxiliary path for Skein-256 is given in Table 5. We apply the original path difference to the first execution of the pair, and the rotated path difference to the second execution.

We consider $\oplus$-differences here, so we have to take into account the fact that the tweak and the key are added by the modular addition. Therefore, we choose the difference so that the probability of the carry is low. However, since adjacent bits are often neutral as well, a carry bit may still preserve the rotational pair.

Table 5. Configuration of the auxiliary path for Skein-256. K_i is the i-th word of the first subkey K^0.

Round	Subkey	Subkey words			
45	K^{11}	K_1 0	K_2 0	K_3 δ	K_4 δ
	Tweak	Tweak words			
	T^{11}		$T_\oplus$ 0	T_0 δ	
49	K^{12}	K_2 0	K_3 δ	K_4 δ	K_0 0
	T^{12}		T_0 δ	T_1 δ	
53	K^{13}	K_3 δ	K_4 δ	K_0 0	K_1 0
	T^{13}		T_1 δ	$T_\oplus$ 0	

In Skein-256 we take various δ and apply the resulting auxiliary path $\mathcal{P}_\delta$ to the right pair. We choose δ so that the differences in the subkey K^{12} compensate each other. Then we check whether the modular additions in rounds 43-44 and 53-55 are not affected by the modification. If so, we get another rotational pair for rounds 43-55.

In experiments, we found that 44 of the 64 possible individual bits that result in a local collision of the latter type behave neutral with probability larger than 0.75 for three rounds in forward direction and simultaneously two rounds in backwards direction, 37 consecutive bits of those have a probability very close

to 1[1]. Details for this phase will be found in Appendix in Table 6. Overall, the results mean that every time those five rounds in the outbound phase are computed, and the effort of those is less than 2^{37}, the amortized effort for those computations will be negligible. If the effort for those five rounds is more, the effect of this acceleration phase, the speed-up, still grows roughly exponential with the number of neutral bits used.

4.4 Outbound Phase

We follow the idea of [10], and introduce corrections in the Threefish keys. But unlike [10], we consider modular corrections, i.e. we define the related-key pair by $(K, \overleftarrow{K} + e)$, where e is a low-weight correction, "+" is modular addition, and the rotation amount is fixed to 2 to bypass the key schedule constant. Each 64-bit word w in Skein can be seen as a concatenation of two words w_1, w_2, i.e. $w = w_1||w_2$ where w_1 represent the two most significant bits of w and w_2 the rest 62 bits.

To obtain a high number of rounds in the outbound phase, we carefully choose optimal corrections and fix some of the key bits. More specifically, we found the best values of key bits with the optimized exhaustive search. Now we explain how to optimize the search in Skein-256 (Figure 2).

We consider two rounds of Skein-256 with a subkey addition in between (rounds 4-5, 8-9, etc.). Note that the outer double rounds (6-7, 10-11, etc) simply keep the rotational pairs, so the probability does not depend on the number of round. The outer rounds probability is $2^{-8.5}$ for Skein-256 and 2^{-17} for Skein-512.

We denote the four words of the internal state before the double rounds by (A, B, C, D). Therefore, we have

$$(A, B, C, D) = (a_1||a_2,\ b_1||b_2,\ s_1||s_2,\ t_1||t_2);$$
$$(\overleftarrow{A}, \overleftarrow{B}, \overleftarrow{C}, \overleftarrow{D}) = (a_2||a_1,\ b_2||b_1,\ s_2||s_1,\ t_2||t_1).$$

Similarly, we denote by

$$K = [k_1||k_2,\ k_3||k_4,\ k_5||k_6,\ k_7||k_8]; \quad \overleftarrow{K} \oplus e = [k'_2||k'_1,\ k'_4||k'_3,\ k'_6||k'_5,\ k'_8||k'_7].$$

the rotational pair of subkeys. Then the corrections e_i can be defined as

$$e_i = k'_{2i+1}||k'_{2i+2} - k_{2i+1}||k_{2i+2}.$$

In Figure 2 the pairs are presented one a top of another with the symbol "- - - - - - -" between them. By $C_{z_1,\ldots,z_k}$ we denote the carry from the sum $z_1 + \ldots + z_k$, i.e.

[1] The fact that carries have to behave equivalently for round key additions in both forward and backward direction puts constraints on the inbound phase which are ignored here to keep the exposition simple. This either results in less degrees of freedom available to perform the exhaustive-search part of the attack, or reduces the number of possible combinations of neutral bits, and has to be taken into account in the overall estimate of the time complexity.

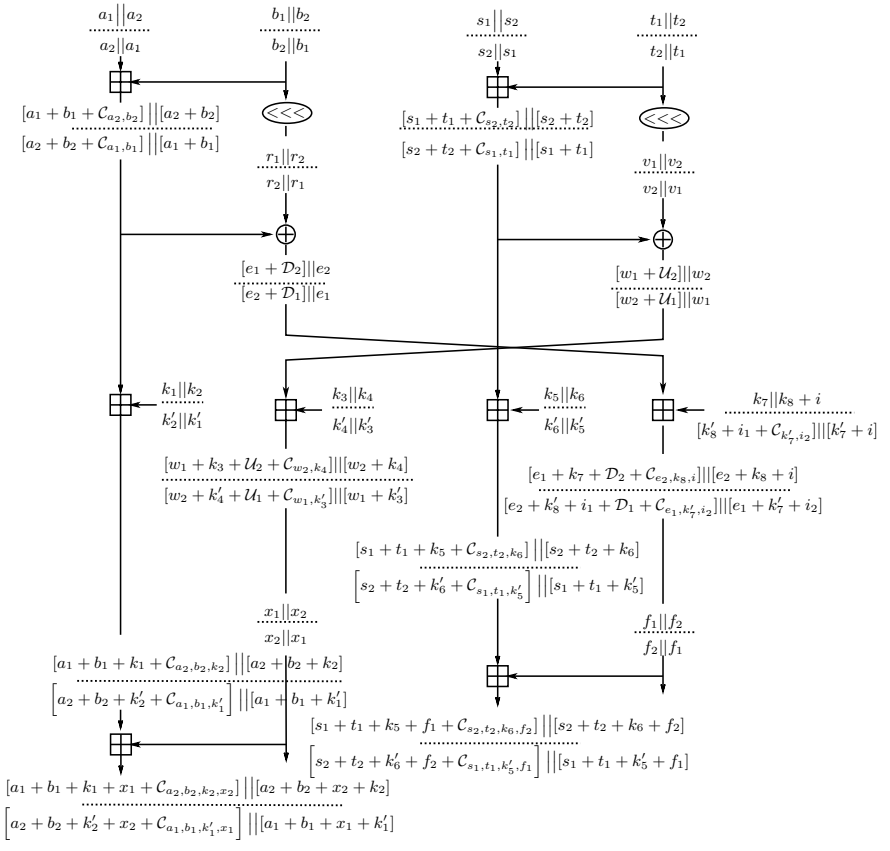


Fig. 2. Rotational pair through two rounds with key addition of Skein-256

when $z_i < 2^r$, then $C_{z_1,\ldots,z_k} = (z_1 + \ldots + z_k) \ggg_r$. The variables $r, v, \mathcal{D}, \mathcal{U}, x, f$ are introduced to maintain the $2+62$ bit representation of the words. With $i = i_1||i_2$ we denote the round counter. Since the rotation preserves the rotational property, we can omit the rotations in the second round of the double subkey rounds, and only require rotational output pairs after the additions in this round. To obtain such pairs for the first output, the following conditions have to hold:

$$\begin{aligned} a_1 + b_1 + k_1 + x_1 + C_{a_2,b_2,k_2,x_2} &= a_1 + b_1 + x_1 + k_1^{'} \\ a_2 + b_2 + x_2 + k_2 &= a_2 + b_2 + k_2' + x_2 + C_{a_1,b_1,k_1',x_1} \end{aligned}$$

Similarly, for the rest 3 outputs, we get the following conditions:

$$\begin{aligned} w_1 + k_3 + \mathcal{U}_2 + C_{w_2,k_4} &= w_1 + k_3' \\ w_2 + k_4 &= w_2 + k_4' + \mathcal{U}_1 + C_{w_1,k_3'} \\ s_1 + t_1 + k_5 + f_1 + C_{s_2,t_2,k_6,f_2} &= s_1 + t_1 + k_5' + f_1 \end{aligned}$$

$$s_2 + t_2 + k_6 + f_2 = s_2 + t_2 + k'_6 + f_2 + C_{s_1,t_1,k'_5,f_1}$$
$$e_1 + k_7 + \mathcal{D}_2 + C_{e_2,k_8,i} = e_1 + k'_7 + i_2$$
$$e_2 + k_8 + i = e_2 + k'_8 + i_1 + \mathcal{D}_1 + C_{e_1,k'_7,i_2}$$

The above 8 equations, can be reduced to:

$$k'_1 - k_1 = C_{a_2,b_2,k_2,x_2} \quad (1)$$
$$k'_2 - k_2 = -C_{a_1,b_1,k'_1,x_1} \quad (2)$$
$$k'_3 - k_3 = C_{w_2,k_4} + \mathcal{U}_2 \quad (3)$$
$$k'_4 - k_4 = -(C_{w_1,k'_3} + \mathcal{U}_1) \quad (4)$$
$$k'_5 - k_5 = C_{s_2,t_2,k_6,f_2} \quad (5)$$
$$k'_6 - k_6 = -C_{s_1,t_1,k'_5,f_1} \quad (6)$$
$$k'_7 - k_7 = C_{e_2,k_8,i} + \mathcal{D}_2 - i_2 \quad (7)$$
$$k'_8 - k_8 = i - i_1 - (C_{e_1,k'_7,i_2} + \mathcal{D}_1) \quad (8)$$

This system gives as a hint how to choose the corrections e_i and the values of some of the subkey bits. For each carry $C_{z_1,\ldots,z_k}$ it holds $0 \le C_{z_1,\ldots,z_k} < k$. Yet the probability that a carry will take a specific value in this range, when z_i are randomly chosen, is not uniformly distributed. When the carries come from sums with 4 terms, the probability is highest for the values 1 and 2. Therefore, for our brute force, we limit the differences $k'_1 - k_1, k_2 - k'_2, k'_5 - k_5, k'_6 - k_6$, only to these two values.

The variables $\mathcal{U}_1, \mathcal{U}_2, \mathcal{D}_1, \mathcal{D}_2$, are determined as follows:

$$\mathcal{U}_1 = ((s_2 + t_2 + C_{s_1,t_1}) \oplus v_2) - ((s_2 + t_2) \oplus v_2)$$
$$\mathcal{U}_2 = ((s_1 + t_1 + C_{s_2,t_2}) \oplus v_2) - ((s_2 + t_2) \oplus v_2)$$
$$\mathcal{D}_1 = ((a_2 + b_2 + C_{a_1,b_1}) \oplus r_2) - ((a_2 + b_2) \oplus r_2)$$
$$\mathcal{D}_2 = ((a_1 + b_1 + C_{a_2,b_2}) \oplus r_1) - ((a_1 + b_1) \oplus r_1)$$

These variables can take only odd values and a zero. Since C_{w_2,k_4} can take $0, 1$ and $\mathcal{U}_2$ can take $0, 1$ it means that $k'_3 - k_3$ (see (3)) can also take 1 and 2 (the same values as the one for the subkeys discussed above). A similar reasoning is applicable to the difference $k_4 - k'_4$. The differences $k'_7 - k_7$, $k_8 - k'_8$ that are left, are the only one that actually depend on the round counter. Yet, since $C_{e_2,k_8,i}$ can take the values $0, 1$[2], i.e. it is not fixed but rather flexible, the whole expression $C_{e_2,k_8,i} + \mathcal{D}_2 - i_2$, for any i_2 can take the values $1, 2$ (recall that $\mathcal{D}_2$ can be any odd value). Therefore the difference $k'_7 - k_7$ can be 1 or 2 (with probability that depends on the round counter i_2). Finally, let us focus on the difference $k'_8 - k_8$ which is determined by the expression $i - i_1 - C_{e_1,k'_7,i_2} - \mathcal{D}_1$. For a specific counter i, when $k'_7 + e_2 = 0$, the carry C_{e_1,k'_7,i_2} is fixed. Hence in this case, the whole expression can take only one value, 1 or 2, but not the both. This limits $k'_8 - k_8$ to only a single value.

[2] It can take the value 2 as well, but the probability is really low because the counter i is only 4-5 bits.

Now recall that k_i, k'_i are the values of the particular subkey words, and not the key words. Once we fix all of the differences in the subkey words of some round, then in the next round, practically the same differences will appear shifted by one index. Also, since the value of the difference in the last key word K_4 is determined from the other words, we would have to fix the values of k_1, k_3, k_5, k_7 and the two least significant bits of k_2, k_4, k_6, k_8 so that the difference in K_4 will be as expected. We fix only two bits because we choose the initial difference to be 1 or 2.

In our brute force search, first we find good values for the differences and the two most significant key bits of each key word. We try all possible differences 1 or 2, and then we fix the key bits values, such that the difference in the two most significant bits of K_4 will also be 1 or 2, and we take into account the limitation on $k'_8 - k_8$ for each counter. Then, we try all possible differences 1 and 2 in the least 62 bits of the each key word. We choose the differences that pass with highest probability through the double subkey rounds. Also, we fix the 2 least significant bits in each key word, so that the difference in the least 62 bits of K_4 will also be 1 or 2. Finally, to increase the probability we fix the values of the bits 60,61 (the next two bits after the 2 most significant bits). This results in fixing the two most significant bits of k_2, k_4, k_6, k_8 which in return increases the probability that the carries take the expected values.

Rather than finding the above values through a theoretically small brute force, we have tested our approach on a real double subkey rounds Skein-256. That is, most of the values, were found and confirmed to be good by taking rotational input pairs of states and rotational input pair of key words with corrections and testing the probabilities on double subkey rounds. In some cases the theoretical probabilities did not coincide with the empirical. This is because there are some hidden dependencies. For example, both $\mathcal{U}_1$ and $k'_5 - k_5$ depend on s_2, t_2. Once we had the optimal corrections (and some bit values) of the keys for the double subkey rounds, we found the probability for 4 consecutive rounds. We start with a random rotational input pair of states and go through three rounds. Then we add the subkeys (with the particular counters) and then we go for an additional round.

We fix 6 bits in K: 4 MSBs and 2 LSBs, and 6 bits in K_B: 2 MSBs and 4 LSBs. The values of these bits are given at Table 3. In Skein-256 the probability to pass rounds 3–42 (i.e. 10 key additions) is 2^{-244}. A detailed table with round-by-round probabilities is given at Table 7 of the Appendix.

Optimal values for the differences and some key bits can be obtained for Skein-512 as well. A property of the double subkey rounds Skein-512 that helps to run the brute force search is that these two double subkey rounds can be split into two non-intercepting halves (see Fig.3 in the Appendix). Then, for each half, the optimal differences can be found independently. Note that this simply speeds up the brute force for optimal differences and values, but has no impact on the actual probability of the inbound phase. Unlike Skein-256, in Skein-512 we could not find empirically the probabilities for 4 consecutive rounds because they were too low. Hence, we considered each 4 rounds as double round + double subkey

round and simply multiplied the probabilities of these two. The values for the optimal 6 bits of each key word in Skein-512 are given in Table 4. In Skein-512 the probability to pass rounds 3–46 is about 2^{-494} (details in Table 8).

4.5 Probabilities in the Khovratovich-Nikolić Analysis

The paper [10] provided the rotational analysis of Threefish on up to 42 rounds. The probability estimates were based on several independence assumptions, which must be corrected as follows:

- The probability of the rotational pair propagation through double rounds without key addition (2-3, 6-7, etc.) is not a multiplication of probabilities for a single round. The problem is that two consecutive modular additions $((a \boxplus b) \boxplus c)$ have lower rotational probability than expected. For example, the rotational probability of one round in Skein-256 is $2^{-3.35}$ for the rotation by 2, but the probability of two rounds is $2^{-8.52}$ instead of $2^{2\cdot(-3.35)} = 2^{-6.7}$.
- The rotational inputs to the round before the key addition (4, 8, etc.) are not uniformly distributed, and this partly compensates the negative effect of the dependency (see above). We note that the non-uniformity of inputs is best approximated with restricting the two most significant bits from the value $\{00\}$.
- The propagation of the rotational inputs through the double round with the key addition in Threefish-256, with the appearance and the correction of errors, can not be considered as two independent events (i.e., as getting rotational pairs in the further MIX operations independently). As a result, the probability of this event can not be computed as a multiplication of other probabilities, and must be computed as a single value.

4.6 Degrees of Freedom Analysis

Now we discuss the following question: How often can this inbound phase be repeated? After fixing the differences and the corrections, for Skein-256 we have

$$256 + 256 + 128 = 640$$

degrees of freedom available to perform the attack. The outbound phase fixes 24 of the 256 bits of the key, (also 12 bits of the 128-bit tweak), and in addition may need up to 256 bits to follow the longest possible trail with high probability. What remains is

$$640 - 36 - 256 = 348$$

degrees of freedom to be spent by the inbound and the acceleration phase. If variants with less rounds are targeted, this number is higher, as less repetitions are needed for the shorter outbound phase. Overall, this is enough for our purposes.

4.7 Summary and Complexity Estimates

We experimentally verified the probabilities of the outbound phase, and took various dependencies into account, and also experimentally verified parts of the acceleration and inbound phase.

Using the Skein-256 compression function as an example, we describe the resulting attack. As illustrated already in Fig. 1, the 8-round inbound part is performed close to the output of the cipher/compression function, the 5 round acceleration area (3 rounds in forward direction and 2 rounds in backward direction) surrounding it. The majority of the inside-out computation is then done in backwards direction, covering about 40 rounds. In total this gives about 53 rounds. Additionally, early stopping techniques will only require the computation of a small number of rounds in the outbound part before another trial is made, saving a factor of the computational complexity that is in the order of the number of rounds.

We estimate the amortized cost for the rounds covered by inbound and acceleration phase for both Skein-256 and Skein-512 by 1, as there are plenty of long ranging neutral bits that cover up costs in solving the right pairs in those inner rounds. In Skein-256, we will spend 2^{244} in the outbound+acceleration phases to find 2^{244} starting pairs for the outbound phase. One such pair will pass this phase with probability close to one. Therefore with an effort that is roughly equivalent to 2^{244} calls to the compression function of Skein-256 we can find one rotational pair of messages and chaining values (with corrections) that produces a rotational pair of updated chaining values. To produce 2^7 such pairs, i.e. to find 2^7-rotational collisions in Skein-256, we only need $2^{7+244} = 2^{251}$ calls. On the other hand, in a random function one has to make at least $2^7 \cdot 2^{\frac{128-2}{128+2}256} \approx 2^{255}$ calls (see Lemma 1).

Similarly, for the compression function of Skein-512, we can create 2^8 rotational collisions with $2^{8+495} = 2^{503}$ compression function calls, while a random function would require $2^8 \cdot 2^{\frac{256-2}{256+2}512} \approx 2^{512}$ calls.

5 Conclusion and Future Work

Our results do not threaten the practical use of full-round Skein or Threefish. However, we show that these constructions behave non-random in settings where all or most inputs can be chosen, and this for more rounds than initially thought. We do not assume any other modifications. We argue that variants of Threefish reduced from 72 to about 53/57 rounds is not an ideal cipher in a similar way as AES-256 was shown not to be an ideal cipher in the first attack on AES [4]. For the Skein compression function a similar argument is made. Since Skein has a very light-weight output transformation, our non-randomness results can also carry over to the actual hash function. There, less degrees of freedom limit, but not prohibit, the applicability of some of our new techniques. To summarize, the following ideas and approaches lead to the improved results:

- The rebound approach as a high-level model for the attack.
- Considering rotational corrections with respect to integer addition instead of XOR.
- Based on analytic reasoning, we find an efficient search method for fixing a subset of input bits before other phases of attacks.
- Using the degrees of freedom in the internal state to efficiently solve for the inner 8-rounds.
- Using the 8-round local collision as long-range neutral bits in an inside-out manner to speed up the outbound phase.

It will be interesting to study how rotational properties found in other constructions, some of which have been reported recently, can also be amplified in a way similar to what we demonstrated in this paper for Skein. The inbound and acceleration techniques we use in our analysis are to a large extent independent of the statistical property that is meant to be produced at the inputs and outputs of Skein. Hence, in addition to the rotational attacks described in this paper, also more traditional differential attacks aiming for collision or near-collision attacks will be able to take advantage of those techniques.

Acknowledgements. This work was sponsored in part by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy), and by the European Commission under contract ICT-2007-216646 (ECRYPT II). Ivica Nikolić is supported by the Fonds National de la Recherche Luxembourg grant TR-PHD-BFR07-031.

References

1. Assche, G.V.: A rotational distinguisher on Shabal's keyed permutation and its impact on the security proofs (2010), `http://gva.noekeon.org/papers/ShabalRotation.pdf`
2. Aumasson, J.-P., Çalik, Ç., Meier, W., Özen, O., Phan, R.C.-W., Varici, K.: Improved cryptanalysis of Skein. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 542–559. Springer, Heidelberg (2009)
3. Biham, E., Chen, R.: Near-collisions of SHA-0. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 290–305. Springer, Heidelberg (2004)
4. Biryukov, A., Khovratovich, D., Nikolic, I.: Distinguisher and related-key attack on the full AES-256. In: Halevi, S. (ed.) Advances in Cryptology - CRYPTO 2009. LNCS, vol. 5677, pp. 231–249. Springer, Heidelberg (2009)
5. Chen, J., Jia, K.: Improved related-key boomerang attacks on round-reduced threefish-512. Cryptology ePrint Archive, Report 2009/526 (2009)
6. Daum, M.: Cryptanalysis of Hash Functions of the MD4-Family. PhD thesis, Ruhr-Universität Bochum (May 2005)
7. Ferguson, N., Lucks, S., Schneier, B., Whiting, D., Bellare, M., Kohno, T., Callas, J., Walker, J.: The Skein hash function family. Submitted to SHA-3 Competition (2008)
8. Ferguson, N., Lucks, S., Schneier, B., Whiting, D., Bellare, M., Kohno, T., Callas, J., Walker, J.: The Skein hash function family - version 2. Submission to NIST (Round 2) (2009)

9. 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)
10. Khovratovich, D., Nikolić, I.: Rotational cryptanalysis of ARX. In: Beyer, I. (ed.) FSE 2010. LNCS, vol. 6147, pp. 333–348. Springer, Heidelberg (2010)
11. Klima, V.: Tunnels in hash functions: MD5 collisions within a minute (2006), http://eprint.iacr.org/2006/105.pdf
12. Lamberger, M., Mendel, F., Rechberger, C., Rijmen, V., Schläffer, M.: Rebound distinguishers: Results on the full Whirlpool compression function. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 126–143. Springer, Heidelberg (2009)
13. Lamberger, M., Mendel, F., Rechberger, C., Rijmen, V., Schläffer, M.: The Rebound Attack and Subspace Distinguishers: Application to Whirlpool. Cryptology ePrint Archive, Report 2010/198 (2010), http://eprint.iacr.org/
14. Matusiewicz, K., Naya-Plasencia, M., Nikolic, I., Sasaki, Y., Schläffer, M.: Rebound attack on the full LANE compression function. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 106–125. Springer, Heidelberg (2009)
15. Mendel, F., Peyrin, T., Rechberger, C., Schläffer, M.: Improved cryptanalysis of the reduced Grøstl compression function, ECHO permutation and AES block cipher. In: Jacobson Jr., M.J., Rijmen, V., Safavi-Naini, R. (eds.) SAC 2009. LNCS, vol. 5867, pp. 16–35. Springer, Heidelberg (2009)
16. Mendel, F., Rechberger, C., Schläffer, M., Thomsen, S.S.: The rebound attack: Cryptanalysis of reduced Whirlpool and Grøstl. In: Dunkelman, O. (ed.) Fast Software Encryption. LNCS, vol. 5665, pp. 260–276. Springer, Heidelberg (2009)
17. Naito, Y., Sasaki, Y., Shimoyama, T., Yajima, J., Kunihiro, N., Ohta, K.: Improved collision search for SHA-0. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 21–36. Springer, Heidelberg (2006)
18. Nikolić, I., Pieprzyk, J., Sokolowski, P., Steinfeld, R.: Rotational cryptanalysis of (modified) versions of BMW and SIMD (2010), https://cryptolux.org/mediawiki/uploads/0/07/Rotational_distinguishers_Nikolic,Pieprzyk,Sokolowski,Steinfeld.pdf
19. Stevens, M.: On collisions for MD5. Master's thesis, Eindhoven University of Technology, Eindhoven, Netherlands (2007)
20. 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)
21. 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)
22. Wu, S.: Semi-free start collision for 12-round Cheetah-256. NIST mailing list (local link) (2009)

A Details

Table 6. Neutral bits in the acceleration phase. These are used in an inside-out manner, with those computations being 8 rounds apart. A single 64-bit word is used, enumeration is from 0 (LSB) to 63 (MSB). The probabilites are measured over 100 right pairs over two rounds backwards and three rounds forwards direction for Skein-256.

bit	prob.	bit	prob.	bit	prob.	bit	prob.	bit	prob.	bit	prob.	bit	prob.	bit	prob.
$7-17$	1.00	18	0.99	19	1.00	20	0.99	21	1.00	22	0.99	23	1.00	24	0.99
25	0.95	26	0.94	27	0.93	28	0.82	31	0.79	33	0.86	36	0.77	$38-45$	1.00
46	0.99	47	1.00	48	0.99	49	0.98	50	0.97	51	0.96	52	0.96	53	0.96
54	0.90	55	0.84												

Table 7. Round-by-round rotational probabilities for Skein-256

Rounds	1-2	3-5	6-9	10-13	14-17	18-21
Prob. log_2	–	−15.13	−21.97	−21.84	−24.44	−24.69
Rounds	22-25	26-29	30-33	34-37	38-41	42
Prob. log_2	−23.83	−26.09	−23.44	−31.75	−27.09	−3.3

Table 8. Round-by-round rotational probabilities for Skein-512

Rounds	1-2	3	4-5	6-7	8-9	10-11	12-13	14-15
Prob. log_2	–	−6.7	−26.35	−17.05	−26.21	−17.05	−24.26	−17.05
Rounds	16-17	18-19	20-21	22-23	24-25	26-27	28-29	30-31
Prob. log_2	−28.26	−17.05	−28.29	−17.05	−23.79	−17.05	−23.56	−17.05
Rounds	32-33	34-35	36-37	38-39	40-41	42-43	44-45	46
Prob. log_2	−27.18	−17.05	−32.23	−17.05	−35.17	−17.05	−31.86	−6.7

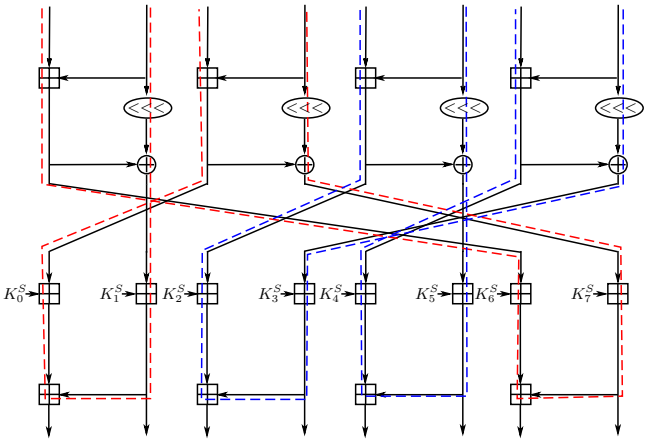


Fig. 3. Double subkey round in Skein-512 divided into two nonintersecting halves – red and blue

Finding Second Preimages of Short Messages for Hamsi-256

Thomas Fuhr

ANSSI, Paris, France and TELECOM-ParisTech, Paris, France
thomas.fuhr@ssi.gouv.fr

Abstract. In this paper we study the second preimage resistance of Hamsi-256, a second round SHA-3 candidate. We show that it is possible to find affine equations between some input bits and some output bits on the 3-round compression function. This property enables an attacker to find pseudo preimages for the Hamsi-256 compression function. The pseudo preimage algorithm can be used to find second preimages of the digests of messages M with complexity $2^{251.3}$, which is lower than the best generic attacks when M is short.

Keywords: hash functions, Hamsi, second preimage.

1 Introduction

Hamsi is a family of hash functions that have been submitted to the NIST SHA-3 competition by Küçük [4]. It contains 4 versions, with respective outputs of 224, 256, 384, and 512 bits. It is based on the Merkle-Damgård domain extender, however its design is rather original as it does not make use of a block cipher in Davies-Meyer mode. The Hamsi compression function uses short message blocks and its security relies on a complex message expansion. Instead of a keyed permutation, a fixed permutation is applied to the concatenation of the incoming chaining variable and the expanded message. The new chaining variable is obtained by truncation of the output of the permutation and feedforward with the previous chaining variable.

Previous work. Several distinguishers on the Hamsi compression function have already been discovered. Some of them rely on the fact that the algebraic degree of the internal permutation is small. In [1], Aumasson noticed that the algebraic degree of 5 rounds of the compression function as a function of the incoming chaining variable is at most 243. Aumasson and Meier then enhanced this observation to find zero-sum distinguishers on a 6-round version of the compression function [2]. Several results of differential cryptanalysis have also been found on the compression function. As a difference on the message has only a small probability to propagate, they concern pseudo near collisions on the compression function [8,9]. Calik and Turan found out that for some given differences in the incoming chaining variables, the difference on one output bit of the compression function can be predicted with probability one, leading to a pseudo preimage attack [3].

M. Abe (Ed.): ASIACRYPT 2010, LNCS 6477, pp. 20–37, 2010.

Our contribution. In this article we describe a weakness of the Hamsi compression function, that can be used to find second preimages for Hamsi-256 with a complexity equivalent to $2^{251.3}$ compression evaluations, improving the best known attack for short messages. This is the first attack that breaks the generic bounds for one of the second round SHA-3 candidates. Our method can be related to cube attacks [5] and AIDA [11]. It is based on an accurate choice of the variables, and on the setting of some initial conditions on the internal state to control the propagation of these variables and to prevent the algbraic degree from growing. We aim at solving a system of polynomial equations, and herefore we set the values of some variables to constants and try to solve the system with the remaining variables. Our main idea consists in setting some conditions on the message block and the chaining variable in order to find affine relations between the output of the compression function and some bits of the incoming chaining variable. These relations can be used to find second preimages for the full hash function.

Related work. Shamir and Dinur independently discovered an algebraic second preimage attack against Hamsi-256 based on cube techniques. Their attack was presented at the Crypto 2010 rump session, and also breaks the complexity of generic attacks against single-pipe Merkle-Damgård hash functions when the initial message is short [10].

Outline of the paper. In Section 2 we briefly describe the hash function Hamsi-256. In Section 3, we display two algebraic properties of the S-box used in Hamsi, and show how to use it to write the result of the first two rounds of the compression function as an affine function of some bits of the chaining variable. After that we show how to extend this property to find affine equations on the full Hamsi-256 compression function in Section 4. Under some conditions on the message block and the incoming chaining variable, we managed to find 14 (resp. 11) output bits of the compression function that can be written as an affine function of 7 (resp. 8) bits of the incoming chaining variable, the message block and the rest of the chaining variable being fixed. In Section 5, we describe how to use these equations to find pseudo preimages for the full Hamsi-256 compression function, along with some optimization techniques and an evalation of the complexity[1]. Then, in Section 6, we show how to use the pseudo preimage algorithm to find second preimages for the full hash function with a complexity equivalent to $2^{251.32}$ compression evaluations, which is our main result. Finally, in Section 7, we study the application of generic techniques on the Merkle-Damgård domain extender variant used in Hamsi. The resulting complexity is slightly higher than in the case of the Merkle-Damgård domain extender, due to the fact that the message blocks have less entropy than required for a direct application of generic techniques. Therefore our second preimage attack is more efficient than generic techniques when the initial message is short.

[1] A pseudo preimage of a chaining variable C^* is a couple (m, C) where m is a message block and C is a chaining variable such that the result of the compression function $\mathcal{F}(C, m)$ is C^*.

Notation. Throughout the paper, variables represented by small letters are 32-bit variables, and capital letters stand for the whole internal state, or messages. The j-th LSB of variable v is denoted $v^{(j)}$.

$\mathcal{H}(M)$ represents the digest of message M by `Hamsi-256`. $\mathcal{F}(C, m)$ stands for the output of the `Hamsi-256` compression function applied to chaining variable C and message block m, and the iteration of the compression function on several message blocks is defined recursively as follows:

$$\mathcal{F}_1(C, m_1) = \mathcal{F}(C, m_1)$$
$$\forall i \geq 2, \mathcal{F}_i(C, m_1, \ldots, m_i) = \mathcal{F}(\mathcal{F}_{i-1}(C, m_1, \ldots, m_{i-1}), m_i)$$

2 Description of `Hamsi-256`

In this article we focus on `Hamsi-256`. Our technique also applies to `Hamsi-224`, however, unlike for `Hamsi-256`, it does not break the generic bounds.

`Hamsi-256` uses a compression function that maps a 256-bit chaining variable H_{i-1} and a 32-bit message block to a new 256-bit chaining variable. It consists of the following operations:

Message expansion. Firstly, the 32-bit message block m is expanded into a 256-bit variable $\mathcal{E}(M) = (m_0, ..., m_7)$. The expansion function is a linear code over $GF(4)$.

Concatenation. The expanded message is then concatenated with the incoming chaining variable $C = (c_0, ..., c_7)$ to produce a 512 state S represented by a 4×4 matrix of 32-bit registers. The concatenation function is the following:

$$\mathcal{C} : (E(M), C) \rightarrow \begin{pmatrix} s_0, & s_1, & s_2, & s_3, \\ s_4, & s_5, & s_6, & s_7, \\ s_8, & s_9, & s_{10}, & s_{11}, \\ s_{12}, & s_{13}, & s_{14}, & s_{15} \end{pmatrix} = \begin{pmatrix} m_0, & m_1, & c_0, & c_1, \\ c_2, & c_3, & m_2, & m_3, \\ m_4, & m_5, & c_4, & c_5, \\ c_6, & c_7, & m_6, & m_7 \end{pmatrix}.$$

Round function. After the concatenation the following round permutation is applied three times (or eight times for the last message block):

$$\mathcal{R} : S \rightarrow \mathcal{L}(\mathcal{S}(\mathcal{A}(S))),$$

where $\mathcal{A}$ consists in adding a constant value and a counter to the state, $\mathcal{S}$ is a substitution layer based on the use of the second 4-bit to 4-bit S-box of `Serpent` and $\mathcal{L}$ is a diffusion layer that operates on 4 sets of 4 32-bit variables in parallel.

More precisely, $\mathcal{S}$ consists in applying, for all $i \in \{0 \ldots 3\}$ and $j \in \{0 \ldots 31\}$, the S-box to bits j of words $s_i, s_{i+4}, s_{i+8}, s_{i+12}$. In other words, the same S-box is applied in parallel to the 128 columns of the internal state.

Table 1. S-box used in Hamsi. Inputs and outputs in hexadecimal, lsb of x corresponds to words $s_0, \ldots, s_3$.

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S[x]$	8	6	7	9	3	C	A	F	D	1	E	4	0	B	5	2

The diffusion layer works as follows. It takes as inputs $(a, b, c, d) = (s_0, s_5, s_{10}, s_{15})$ (resp. $(s_1, s_6, s_{11}, s_{12})$, (s_2, s_7, s_8, s_{13}), (s_3, s_4, s_9, s_{14})) and consists of the following operations:

$$\begin{aligned} a &:= a \lll 13 \\ c &:= c \lll 3 \\ b &:= (b \oplus a \oplus c) \lll 1 \\ d &:= (d \oplus c \oplus (a \ll 3)) \lll 7 \\ a &:= (a \oplus b \oplus d) \lll 5 \\ c &:= (c \oplus d \oplus (b \ll 7) \lll 22 \end{aligned}$$

Truncation and feedforward. After the third round, the output of the compression function is obtained by applying a truncation function to the state and xoring the result to the former chaining value.

$$\begin{aligned} \mathcal{T} : S &\rightarrow \Sigma = (s_0, s_1, s_2, s_3, s_8, s_9, s_{10}, s_{11}) \\ \mathcal{X} : \Sigma &\rightarrow C^* = C \oplus \Sigma. \end{aligned}$$

Domain extender. To build a variable-length hash function, Hamsi makes use of the Merkle-Damgård construction. The padding consists in concatenating to the message a "1" and as many "0"s as necessary to get an integer number of blocks, and then by further concatenating the message length encoded on 64 bits. For the last block, the permutation consists of 8 rounds (instead of 3).

3 An Observation on the Two-Round Hamsi-256 Compression Function

In this Section we focus on a reduced version of the Hamsi-256 compression function, where the internal permutation is reduced to two rounds. The result we get will be used in the following Sections to break the full version of Hamsi-256. We show how to find pseudo preimages for this reduced-round version of the compression function.

3.1 Study of the Hamsi S-Box

On the Hamsi S-box we notice the following properties.

We use the fact that $S[9] = 1$, $S[C] = 0$, $S[B] = 4$, and $S[E] = 5$ to deduce that

$$\forall (x, b) \in \{0, 1\}^2, S[(x, b, \bar{x}, 1)] = (x + b, 0, b, 0). \quad (1)$$

As a result, only one bit of the output depends on x. Similarly we have $S[3] = 9$ and $S[9] = 1$, which leads to

$$\forall x \in \{0,1\}, S[(1,x,0,\bar{x})] = (1,0,0,x). \tag{2}$$

If the input of an S-box depends on only one variable bit, then the output of the S-box can be expressed as an affine function of this bit. with that in mind, properties 1 and 2 have been found according to the following criteria. First, only one output bit of the S-box should depend on x. Second, input bits 0 and 2 or 1 and 3 must not depend on x, so that for an appropriate choice of a first round S-box, only the input bits coming from the chaining variable depend on x.

3.2 An Interesting Set of Variables

Let us now consider any value of the message block m. Without knowing the incoming chaining value, we can compute $s_0, s_1, s_6, s_7, s_8, s_9, s_{14}, s_{15}$ after the first round constant addition. Let us now suppose that the j-th bit of s_{14} is $s_{14}^{(j)} = 1$. Then, independently of the value of $s_6^{(j)}$, if $s_2^{(j)} = x^{(j)}$ and $s_{10}^{(j)} = \overline{x^{(j)}}$, only the first output bit of the j-th S-box of the 3-rd column will depend on $x^{(j)}$ (according to equation 1). Let J be a set of variables that satisfies this property.

We can then define one variable bit $x^{(j)} \in \{0,1\}$ for each j such that $s_{14}^{(j)} = 1$. After the first S-box layer, only the word s_2 depends on the variable set $X = \{x^{(j)}\}_{j \in J}$, through an affine relation. After the first round diffusion layer and the second round constant addition, words s_2,s_7,s_8, and s_{13} depend linearly on X, which means that only one input bit of each S-box of the second substitution layer can depend on X. As a consequence, the output of this layer is also an affine function of X. The second diffusion layer, the truncation and the feedforward cannot increase the degree, so the whole output of the compression function is an affine function of X.

3.3 Building and Solving the Linear System

We can then try to invert the 2-round compression function $\mathcal{F}$, *ie* to find a message block M and a chaining value C that maps to a given value C^*. The idea is to express the output of the compression function as an affine system of a given set of variables, and to solve this system. With an appropriate choice of variables, we know that the system is affine but we first need to compute its coefficients. To achieve it we do the following:

1. Choose a message block M and compute the resulting value of s_{14} before the first substitution layer.
2. Compute the resulting set of variables $X = \{x^{(j)}\}_{j \in J}$. If $|J| < 16$, choose another value for M.
3. Choose a chaining value C such that for all $j \in J$, $s_2^{(j)} \oplus s_{10}^{(j)} = 1$ after the first constant addition. C is then divided into a *v*ariable part (the bits $c_0^{(j)}$ and $c_4^{(j)}$ for $j \in J$) and a *c*onstant part (the other bits).

4. Compute $\mathcal{F}(M, C)$ to get the constant coefficients of the system.
5. For each $j \in J$, derive C_j from C by complementing the values of $c_0^{(j)}$ and $c_4^{(j)}$. Compute $\mathcal{F}(M, C_j) \oplus \mathcal{F}(M, C)$ to get the coefficients of $x^{(j)}$, using an interpolation method.
6. Solve the affine system of 256 equations in |J| unknowns to find a preimage of C^*. If it has no solution, choose another value of the constant part of C. If all the values have been tried, increase the value of M.

We can then assume that the complexity of solving the resulting equation system is smaller that the complexity of one evaluation of the compression function. To avoid useless computation, one can for example try to find solutions to subsystems, and abort as soon as an inconsistency is detected. Each system allows us to test $2^{|J|}$ values of $\mathcal{F}(M, C)$ with a complexity of less than $|J| + 2$ evaluations of the compression function. As $|J| \geq 16$, the total complexity of this algorithm is about $18 \times 2^{256-16} \approx 2^{244}$ compression evaluations.

4 Linear Equations for the Full Hamsi-256 Compression Function

In this Section we show how to apply similar techniques to find linear equations for the full Hamsi-256 compression function.

If we try to use the same property on the S-box as in the previous Section, we cannot find any large set of variables that lead to linear equations. To this end, property 2 is more interesting. If the message block is such that before the first substitution layer, $s_0^{(j)} = 1$ and $s_8^{(j)} = 0$, if we set $s_4^{(j)} = x^{(j)}$ and $s_{12}^{(j)} = \overline{x^{(j)}}$, only the j-th bit of s_{12} depends on $x^{(j)}$ after the S-box layer. The same remark applies to s_1, s_5, s_9, s_{13}, with $s_5^{(j)} = y^{(j)}$, $s_{13}^{(j)} = \overline{y^{(j)}}$, $s_1^{(j)} = 1$, and $s_9^{(j)} = 0$,.

In comparison with the technique used in Section 3, we use more freedom degrees (for each variable, two bits of the message and one bit of the chaining variable). However, as s_{12} is the d input of the diffusion layer, the dependence in $x^{(j)}$ does not propagate fast during the first round.

The internal state S before the permutation rounds can then be divided into three parts:

- **Variable bits:** The sets of variables X, Y.
- **Conditional bits:** Bits of the initial internal state that must take a given value so that the dependence of the internal state in the variables after the first substitution layer are as described in equation 2. Each Variable bit requires the definition of three Conditional bits: two on the message block and one on the incoming chaining variable.
- **Constant bits:** All the other parts of the internal state.

These bits are not necessarily directly bits from the incoming message block or chaining variable: they can be a linear function of such bits. For example, when considering equation 2, the Conditional bits on the incoming chaining value are

the exclusive or of two bits ($s_4^{(j)}$ and $s_{12}^{(j)}$, or $s_5^{(j)}$ and $s_{13}^{(j)}$). The corresponding Variable bit can then be taken as the value of one of these two bits.

As a result, it is possible to find sets of variables $X = \{x^{(j)}\}_{j \in J_x}$ and $Y = \{y^{(j)}\}_{j \in J_y}$ such that some output bits of the whole compression function depend linearly on the Variable bits X, Y, provided the Conditional bits take a given value and once the Constant bits are set.

Algebraic properties of the `Hamsi-256` *S-box.* To find these sets of variables we have to take into account the following properties.

1. Any function f from 1 bit to n bits is affine. It can be defined as $f(b) = f(0) \oplus (f(1) \oplus f(0))b$. Therefore, if all input bits of a 4-bit S-box are constant except one, the output ofthe S-box is an affine function of the remaining input bit. If this input bit is an affine function of the variables in $X \cup Y$, it is also the case for the 4 output bits, as a composition of affine functions.
2. Similarly, if the input of an S-box depends on only one of the variables, its output is an affine function of this variable.
3. If (b_0, b_1, b_2, b_3) is the output of an S-box with input (a_0, a_1, a_2, a_3) : the only nonlinear monomial in the expression of b_0 is a_0a_2, and b_3 only depends on nonlinear monomials $a_0a_1a_2$ and a_1a_3. Therefore, if the monomial a_0a_2 is an affine function of $X \cup Y$, so is b_0. Similarly, if monomials a_1a_3 and $a_0a_1a_2$ are affine in $X \cup Y$, so is b_3.

We will now use these properties in an automated search as sufficient conditions to guarantee that some final and intermediate bits involved in the computation of the compression function are affine functions of a set of variables.

Optimal sets of variables. For our second preimage attack we then have to determine optimal sets of variable bits. In our attack two phases are time-consuming: the generation of the affine equation system, and the test of the solutions. The complexities mainly depend on the number of variables N_{var} and the number of resulting affine equations N_{eq}. For a given number of variables N_{var}, we then look for the choice of the variable set that leads to the largest affine equation system, using an exhaustive search. The cost of the system generation decreases when the number of variables increases, whereas the cost of testing the solutions mainly decreases when the number of equations increases. A precise evaluation of the complexity of the attack is given in section 5. The optimal values for N_{var} and N_{eq} can then be found as a tradeoff between the complexity of these two algorithms.

Furthermore, the equation systems that have been generated can be reused if one tries to find a pseudo preimage for multiple targets, which is the case in some parts of our attack. Therefore, the optimal sets of equations are different in the different parts of the attack.

Finding the optimal sets of variables. For a given value of N_{var}, we determine the set $X \cup Y$ of variables that leads to the maximal value of N_{eq}. We achieve it

through an automated exhaustive search on all the sets of variables $X_i \cup Y_i$ that contain exactly N_{var} elements.

Let us now consider a fixed set $X \cup Y$. We try to determine which output bits can always be expressed as an affine function of the Variables of $X \cup Y$, under the assumption that all the Conditional bits take the right constant value and that all the Constant bits are fixed. We then do the following:

For each pair of variables $\{z, z'\} \in X \cup Y$, we determine the set of output bits $\mathcal{S}_{z,z'}$ that are always affine functions of z and z' when the Conditional bits take the right value and the other parts of the initial state are set to a fixed constant value. How to determine these sets will be depicted below. Once this is done, the bits in the set $\mathcal{S}_{X,Y} = \cap_{\{z,z'\} \in X \cup Y} \mathcal{S}_{z,z'}$ are affine functions of the set of variables $X \cup Y$. If the algebraic expression of an output bit b as a function of the variables in $X \cup Y$ contains a monomial of degree 2 or more, let z and z' be two variables of this monomial. Then b cannot be in $\mathcal{S}_{z,z'}$, because for some assignment of all the other variables, the expression of b contains the monomial zz'.

Let us now describe how to find $\mathcal{S}_{z,z'}$. After the first S-box layer, only one bit depends on each of the variables z and z'. We then study the propagation of these variables through the compression function. The propagation is not always deterministic - it is probabilistic through the S-box layers. For each intermediate bit of the internal state, we then determine if it is independent from z and z', if it can depend linearly on z and/or z' or if it can be quadratic in z and z'. The diffusion layer $\mathcal{L}$ is linear. Therefore a bit of the internal state after the diffusion layer is always affine in z, z' if and only if all the input bits it depends on also are always affine in z, z'. $\mathcal{A}$ does no change the degree of each bit of the internal state. $\mathcal{S}$ is nonlinear and can increase the degree. More precisely, if two different input bits of a given S-box can depend respectively on z and z', some output bits may be quadratic. At the end of the compression function, $\mathcal{T}$ and $\mathcal{X}$ cannot increase the degree.

Let us now consider a fixed set $X \cup Y$. We try to determine which output bits can always be expressed as an affine function of the Variables of $X \cup Y$, under the assumption that all the Conditional bits take the right constant value and that all the Constant bits are fixed. Equivalently, we can try to determine the output bits b_i which polynomial expression as a function of the Variable bits can contain monomials of degree ≥ 2. This means that for some choice of $(z, z') \in X \cup Y$ and for some assignment of all the other variables, the polynomial expression of b_i can contain the monomial zz'. Therefore, for each choice of $(z, z') \in X \cup Y$, we compute which bits of the internal state can contain the monomials z, z', and zz' during the intermediate computation and in the resulting chaining value.

Using this method, we found the following properties for 7 and 8 variables. Provided that the message block and the whole chaining variable except the x and y variables, and under the assumption that N_{cond} conditions on the message block and the chaining value are verified:

- Output bits 9, 18, 44, 88, 144, 152, 183, 185, 188, 193, 219, 221, 228 and 246 depend linearly on $(x_3, x_{26}, x_{30}, y_4, y_6, y_7, y_{15})$, which makes 14 output bits and 7 variables with $N_{cond} = 21$.
- Output bits 11, 39, 46, 185, 188, 195, 218, 220, 230, 248 and 255 depend linearly on $(x_3, x_{28}, x_{29}, y_1, y_6, y_7, y_{15}, y_{31})$, which makes 11 output bits and 8 variables with $N_{cond} = 24$.

Once the Conditional bits are assigned the right value and the Constant bits are assigned any value, the relation between some output bits (denoted **Equation bits**) and the Variable bits can be described as a linear equation system.

5 Pseudo Preimages for the Hamsi-256 Compression Function

In this Section we try to find pseudo preimages of a given value C^* of the chaining variable. We aim at finding m, C such that $\mathcal{F}(C, m) = C^*$. In the first subsection we describe an optimized algorithm that makes the following operations with a reduced complexity. Once we know that N_{eq} output bits $t_0, \ldots, t_{N_{eq}-1}$ are affine functions of N_{var} variable bits $z_0, \ldots z_{N_{var}-1}$, computing the inverse of the compression function can be achieved as follows. We also give here a correspondance between the operations described and the steps of the algorithm that compute them.

- Set the initial value of the chaining variable C and the message block m such that all the conditions are verified (steps 1 and 2).
- Compute the output bits $t_0, \ldots, t_{N_{eq}-1}$ of $\mathcal{F}(C, m)$ (steps 3 to 7).
- The output bits $t_0, \ldots, t_{N_{eq}-1}$ of the compression function is then an affine function of the variables. Compute the coefficients of this function (step 8).
- Solve the resulting system of affine equations (step 9). If it does not have any solution, start again.
- If the linear system has a solution m_i, C_i, compute the compression function to determine whether $\mathcal{F}(C_i, m_i) = C^*$ (step 10). This occurs with probability $2^{N_{eq}-256}$. If not, start again.

5.1 Building and Solving the Equation Systems

A basic idea. The first idea to compute the coefficients of the equation system would be to reuse the idea of Section 3. More precisely we could evaluate the compression function with all the variables set to 0 to get the constant coefficients, and once for each variable to get the coefficients for this variable, by running the compression function.

But to determine the coefficients, we only need to compute the parts of the state that really depend on the Variable bits and impact the Equation bits, which involve less computation than running the whole compression function.

Furthermore, some small changes in the incoming chaining variable do not impact immediately the whole internal state. Some parts of the computation can

then be reused when computing the constant coefficients of different equation systems. We will describe a method to compute these systems below.

A more efficient method. To achieve a more efficient computation of the coefficients, we can use the following ideas:

- The coefficients of each variable only depend on the propagation of the variable through the second and the third diffusion layer. Therefore they can be recovered from the inputs of the affected S-boxes.
- The first two rounds of the Hamsi-256 compression function are an affine function of the variables defined in Section 3.

We then use a set $|J|$ of 8 variables as defined in section 3, denoted *auxiliary variables*, to compute more efficiently 2^8 equation systems. We know from the analysis of section 3 that the whole internal state up to the input of the third S-box layer are affine functions of these variables, provided that some Conditional bits have the apropriate value. Instead of running the whole compression function to get the constant coefficients for each system, we only modify one auxiliary variable from one system to the next one. Therefore, some intermediate values do not need to be computed again.

Once we have computed the intermediate values of the internal state with all the principal and auxiliary variables set to 0, we can deduce all the values of the internal state for any of the 2^8 possible assignments of the auxiliary variables by studying the propagation of the 8 auxiliary variables through the S-box layer of round 2.

We can then improve the attack as follows.

1. Set the value of the Conditional bits from the chaining variable to their appropriate value.
2. Choose the Constant bits of the chaining variable, and the message block m such that all the conditions are verified.
3. Choose a set of 8 auxiliary variables such that the resulting auxiliary conditions are verified. For a random value of the initial internal state, we can find 8 auxiliary variables with a good probability. If not so, go back to step 2.
4. Compute the first two rounds of the compression function with all the Variables and auxiliary variables set to 0. Keep trace of the results of internal operations.
5. Compute the propagation of the auxiliary variables in the first two rounds.
6. For each value of the set of auxiliary variables, recover the inputs of the S-boxes involving the Variables in rounds 2 and 3.
7. Recover the constant coefficients by running the part of the third round that affect the Equation bits.
8. Recover the other coefficients of the system by studying the propagation of the Variables during rounds 2 and 3.
9. Solve the resulting linear equation system. If it does not have any solution, go back to step 2.
10. Set the Variable bits according to one of the solutions of the equation system, and compute the compression function. If the result is not the target C^*, go back to step 2.

5.2 Complexity Evaluation of the Attack

We now aim at evaluating the complexity of the different steps of the attack. As we try to avoid useless computations, we mainly use operations on bits and not on 32-bit registers. We could use parallelism by building several systems at the same time, with different values of the Constant bits of the incoming chaining variable. Therefore we argue that the right metrics for evaluating the complexity of the attack is the number of elementary bitwise operations (AND, OR, XOR) it involves. To compare it to generic attacks, we use the analysis of Shamir and Dinur [10] and evaluate the number of bitwise operations in the `Hamsi`-256 compression function to about 10500.

Steps 1 to 3 are setup steps and have a negligible complexity compared to the other steps. We also argue that the choice of auxiliary variables can be the same for a large range of systems, therefore the study of which parts of the intermediate internal state they impact can be precomputed once and has a neligible complexity.

Step 4 involves the computation of about 2 out of 3 rounds of the compression function. A careful analysis of which output bits of the S-boxes need to be computed and which parts of the linear diffusion layers need to be run leads to 5248 operations for the 7-variable systems and 4852 operations for the 8-variable systems.

Step 5 involves the computation of at most 7 second round S-boxes per auxiliary variable, and at most $7 \times 20 = 140$ XOR operations per variable for the second round diffusion layer, which makes at most 1120 elementary operations for 2^8 systems.

Step 6 consists in xoring the values of the inputs of some S-boxes before rounds 2 and 3 for different values of the auxiliary variables. The values of these variables can be chosen following a Gray code, to minimize the parts of the state that ha to be computed again. Therefore, only 7 input bits of the second S-box layer can be affected. For the third S-box layer, only some S-boxes are useful (45 for the 7-variable systems, 34 for the 8-variable systems). This step then requires $7 + 4 \times 45 = 187$ (resp. $7 + 4 \times 34 = 143$) XORs for the 7-variable (resp. 8-variable) systems.

Step 7 requires to evaluate the constant coefficients of the system. These coefficients can be recovered by computing some parts of the output of the compression function, knowing the output of the second round. This consists in applying the diffusion operations and the feedforward. To compute the feedforward one needs to invert N_{eq} bits of the first round constant addition. This step costs $473 + 54 + 28 = 555$ (resp. $328 + 37 + 22 = 387$) operations per system.

Step 8 consists in recovering the coefficients of degree 1 monomials. This can be achieved by studying the propagation of the variables through the S-boxes. For the 7-variable (resp. 8-variable) systems the inputs of 17 (resp. 20) S-boxes depend on the variables before the second substitution layer. For some of them, only some output bits need to be computed. For each 7-variable (resp. 8-variable) system, this requires 210 (resp. 200) operations. The propagation through the second diffusion layer to the inputs of the useful third round S-boxes requires 60

(resp. 46) XORs. In the third round, the outputs of 45 (resp. 34) S-boxes affect the Equation bits. To evaluate the coefficients of the variables, 3 cases can occur for the third round S-box layer:

1. The input of the S-box does not depend on Variables. Then the output does not depend on the Variables either, and no computation is required. This occurs for 5 (resp. 9) S-boxes.
2. One input bit can a priori depend on one or several Variables. Then its output depend on the same Variables as its input, and computing the coefficients is equivalent to one S-box evaluation. This occurs for 31 (resp. 17) S-boxes, leading to a complexity of 364 (resp. 155) operations.
3. Two input bits can a priori depend on the Variables. As the dependences are not deterministic, 3 different cases of dependences can occur during the actual computation of the system. Each of them leads to a different propagation of the difference. If the adversary uses parallelism, he needs to compute the dependences for the 3 cases, leading to a complexity equivalent to 3 S-box computations. This occurs for 6 (resp. 8) S-boxes, leading to a complexity of 211 (resp. 213).

The linear coefficients can be derived using simple operations from the bits representing the propagation of the variables through the second and the third S-box layer. The overall commplexity to retrieve the coefficients from these bits is then at most 125 (resp. 101) operations.

Putting everything together, the average costs to compute the coefficients of an equation system are:

- $\frac{5248+1120}{2^8} + 187 + 210 + 60 + 555 + 364 + 211 + 125 = 1737$ operations for 7-variable systems,
- $\frac{4852+1120}{2^8} + 143 + 200 + 46 + 387 + 155 + 213 + 101 = 1268$ operations for 8-variable systems,

Overall, the cost to construct the 7 variable system is about $T_{build}^{(7)} = 2^{-2.59}$ compression evaluations. The complexity to build the 8-variable system is $T_{build}^{(8)} = 2^{-3.05}$ compression evaluations.

Step 9 then consists in solving the equation system, which complexity T_{solve} is small compared to the evaluation of the compression function. We use the Gauss algorithm. Therefore the complexity is as follows: for each of the N_{var} variables, for each of the N_{eq} equations, we compute at most $(N_{var} + 1)$ XORs, and the average number of XORs is $N_{var}/2$. This leads to an overall complexity of $N_{var}(N_{var} + 1)N_{eq}/2$ operations per system, which means 392 operations for 7-variable systems and 396 operations for 8-variable systems. One can therefore bound the complexity of this step by $T_{solve}^{(7)} = 2^{-4.74}$ and $T_{solve}^{(8)} = 2^{-4.72}$.

The success probability of step 10 is then $2^{N_{eq}-256} = 2^{-242}$, leading to an overall complexity of $2^{256-N_{eq}}T_{test}$ compression evaluations (the complexity to test one solution is $T_{test} \approx 1$). Each system of equations enables to test $2^{N_{var}}$

values of the chaining variable, therefore one needs to compute about $2^{256-N_{var}}$ systems. The best pseudo-preimage algorithm is then obtained for 8 variables:

$$T^{(8)}_{preimage} = 2^{248}(T^{(8)}_{build} + T^{(8)}_{solve}) + 2^{245}T^{(8)}_{test} \approx 2^{246.2}. \quad (3)$$

Variability. We also need to make sure that the search space is big enough to find the second preimages we need. We can only detect a certain type of pseudo preimages for a given output, that can be defined by the conditions that are imposed on the input message block and chaining variable. For 8 variables, we have 24 such bit conditions (16 on the message block and 8 on the chaining variable). The original search space has a size $2^{256+32} = 2^{288}$, we then expect $2^{288-24} = 2^{264}$ couples (C, m) to fulfill these conditions. We also need to find 8 auxliary variables. An auxiliary variable can be defined when one condition on the message block and one condition on the chaining variable are verified (according to Section 3). As we have 32 potential auxiliary variables, the probability that at least 8 of them can be chosen is at least $1/2$. Therefore we expect at least 2^{263} candidates, among which 2^7 are pseudo-preimages of a given value. This argument confirms that the search space is big enough to make the attack work.

6 Second Preimages for the Full Hamsi-256

As we showed in Section 5, pseudo preimages can be found for the Hamsi-256 compression function with a complexity about $2^{246.2}$ compression evaluations. This threatens the security of Hamsi-256, because one can use a pseudo preimage algorithm to build a second preimage finding algorithm using a basic meet-in-the middle approach. In this section we describe this both this basic method and show how to improve it. The main idea is the following: the complexity of the pseudo-preimage attack is dominated by the complexity of the construction of the equation systems, especially the complexity to recover the coefficients of the equations. In the general second preimage setting, one can then try to invert one of the intermediate chaining variables. As the coefficients of the linear system are the same whatever the value of the chaining variable we try to invert, we can spare some computation.

6.1 A Basic Second Preimage Algorithm

The most natural idea to generate second preimages using our pseudo preimages algorithm consists in using a basic meet-in-the middle approach. The algorithm is the following :

1. Compute $2^{5.9}$ pseudo preimages of the chaining value after the ninth message block.
2. Compute intermediate hash values for sequences of 8 message blocks until reaching one of the values computed in step 1. The expected number of such messages is around $2^{251.1}$.

This would lead to a second preimage attack with a complexity about $2 \times 2^{251.1} = 2^{252.1}$. However, the original message must be contain at least 9 blocks, so as to make sure that we have enough variability to build a second preimage of an equivalent length. An improvement of our technique would lead to an improvement of the best second preimage attacks on Hamsi-256.

6.2 Pseudo Preimages in a Set of Images

In the first step of the basic attack, a large amount of the computation time is consumed to generate the systems. If one has several targets, this computation can be done only once. In this section we will describe another algorithm that benefits from this remark.

We will now describe how to find pseudo preimages of an element of a set of N images, which is an easier problem than finding a pseudo preimage of given element. In our method, the computation of the coefficients of the linear equation systems only depends on the target by xoring it to the constant coefficients. We can therefore use a similar method to compute a preimage of an element of a set by computing the coefficients only once, and trying to solve the system of equations for all the N elements of the set.

The beginning of the resolution of the equation system is also common for all the targets. One aim at solving the equation $y_i = Ax$, where $\{y_i\}, i \in I$ are constant binary vectors of size N_{eq}, x is an unknown binar vector of size N_{var} and A is a fixed binay matrix. One can then begin with the computation of the Gauss algorithm on a basis of the y space. The complexity of this part can be denoted T_{invert}. A similar argument than the one used in previous section allows to estimate it as $N_{eq}N_{var}(N_{eq} + N_{var})/2$. The end of the resolution consists in checking whether the remaining equations are verified. In other words, testing at most N_{eq} linear relations on the ouptut bits, leading to a complexity of at most $T_{check} = N_{eq}^2$ elementary operations. Therefore this step can be overlooked in numerical applications.

As a result, the complexity of the new algorithm is derived from equation 3:

$$T_{set}(N) = \frac{2^{256-N_{var}}}{N}(T_{build} + T_{invert}) + 2^{256-N_{var}}T_{check} + 2^{256-N_{eq}}T_{test} \quad (4)$$

We also have $T_{invert}^{(7)} \approx 1029$ operations, and $T_{invert}^{(8)} \approx 836$ operations, which means $T_{invert}^{(7)} \approx 2^{-3.35}$, ad $T_{invert}^{(8)} \approx 2^{-3.65}$ compression evaluations.

6.3 Second Preimages for Short Messages

We can now consider the following algorithm. It requires that the original message contains at least 10 complete blocks. If this condition is fulfilled, its complexity does not depend on the message length. Therefore it is more efficient than Kelsey and Schneier's attack only for short messages.

We consider a message $M = m_0||\dots||m_9||\dots||m_\ell$ and try to find a second preimage of the digest of M. Therefore we consider the chaining variable $h_{10} =$

$\mathcal{F}_{10}(IV, m_0, \ldots, m_9)$. First, we try to find x pseudo preimages of h_{10}, namely $(h_{9,1}, m_{9,1}), \ldots, (h_{9,x}, m_{9,x})$. We use our 8-variable set. The complexity of this step is about:

$$T_1(x) = x \times (2^{248}(T^{(8)}_{build} + T^{(8)}_{solve}) + 2^{245}T^{(8)}_{test}) \approx 2^{246.2} \times x \tag{5}$$

In a second step, starting from $S = \{h_9, h_{9,1}, \ldots, h_{9,x}\}$ where $h_{9,0} = \mathcal{F}_9(IV, m_0, \ldots, m_8)$, we search y pseudo preimages of one element of the set S, $(h_{8,1}, m_{8,1}), \ldots, (h_{8,y}, m_{8,y})$. For this step we use 7-variable equation systems. The complexity of the second step is:

$$T_2(x, y) = (\frac{2^{249}}{x+1}(T^{(7)}_{build} + T^{(7)}_{invert}) + 2^{242}T^{(7)}_{test}) \times y \approx 2^{247.1}\frac{y}{x+1} + 2^{242}y. \tag{6}$$

Finally, using a probabilistic approach, we try to find $(m_0^*||\ldots||m_7^*) \neq (m_0||\ldots||m_7)$ such that the resulting chaining variable $h_8^* = \mathcal{F}_7(IV, m_0^*, \ldots, m_7^*)$ collides with one of the $h_{8,j}$ with $h_{8,0} = \mathcal{F}_8(IV, m_0, \ldots, m_7)$. The complexity of this step is then:

$$T_3(y) = \frac{2^{256}}{y+1}. \tag{7}$$

Let us denote $m_{8,0} = m_8$ and $m_{9,0} = m_9$. For j as defined above, there exists i such that $\mathcal{F}(h_{8,j}, m_{8,j}) = h_{9,i}$. As a result, $\mathcal{F}_{10}(IV, m_0^*, \ldots, m_7^*, m_{8,j}, m_{9,i}) = h_{10}$, and

$$\mathcal{H}(m_0^*||\ldots||m_7^*||m_{8,j}||m_{9,i}||\ldots||m_\ell) = \mathcal{H}(M). \tag{8}$$

This leads to a second preimage for $\mathcal{H}(m)$ with complexity

$$T(x, y) = T_1(x) + T_2(x, y) + T_3(y) \approx 2^{246.2} \times x + \frac{2^{247.1}y}{x+1} + 2^{242} \times y + \frac{2^{256}}{y+1}. \tag{9}$$

For `Hamsi`-256 the best compromise is found when the complexity of all these steps are almost the same. For $x = 11$ and $y = 71$ we then have :

$$T_1(x) \approx 2^{249.66}, T_2(x, y) \approx 2^{249.66}, T_3(y) \approx 2^{249.83}$$

This leads to a complexity of about $T(x, y) \approx 2^{251.30}$ compression evaluations.

7 The Kelsey-Schneier Second Preimage Attack

In previous sections we described a second preimage attack that runs faster than generic attacks on hash functions. To be exhaustive we also need to argue that it runs faster than generic attacks on the domain extender used to design `Hamsi`.

In [7], Kelsey and Schneier showed a generic attack on single-pipe Merkle-Damgård hash functions. To achieve it, they use either a multicollision finding algorithm created by Joux [6], or fixed points. As `Hamsi`-256 is based on the Merkle-Damgård domain extender, this attack can also be used against `Hamsi`-256. However, it makes use of very short message blocks, that do not give the adversary enough freedom degrees to apply the attack to `Hamsi`-256. Furthermore,

the specific design of the compression function does not enable an adversary to generate fixed points easily.

In this Section we describe a modified version of the attack, so as to make it applicable to Hamsi-256. The modification is trivial, however the complexity of the new attack slightly differs from the complexity of the original attack. The aim of this Section is therefore to find an estimation of the complexity of the best generic attack against Hamsi-256.

7.1 Description of the Attack

Definition 1. *A (p, q) expandable message for a Merkle-Damgård hash function $\mathcal{H}$ is a set of $(q - p + 1)$ messages $(\mu_p, \ldots, \mu_q)$ such that*

1. $\mathcal{H}(\mu_p) = \mathcal{H}(\mu_{p+1}) = \ldots = \mathcal{H}(\mu_q) = h$.
2. $\forall i \in \{p, \ldots, q\}$, μ_i *contains exactly i blocks after the padding.*

The original second preimage attack works as follows. Let us now suppose that we want to find a second preimage of the Hamsi-256 digest of an ℓ-block message $M = m_0||m_1||...||m_{\ell-1}$. We aim at finding a message M' such that $\mathcal{H}(M) = \mathcal{H}(M')$. We look for M' such that M and M' have the same length.

1. Generate a (p, q) expandable message for $\mathcal{H}$, for some appropriate values of p and q that will be discussed later on.
2. Choose the common digest value h as chaining variable, and compute the compression function for random sequences of 8 message blocks, to find $(m_1^*, \ldots, m_8^*)$ such that $\mathcal{F}_8(h, m_1^*, \ldots, m_8^*)$ is one of the chaining values involved in the computation of $\mathcal{H}(M)$, $CV_i = \mathcal{F}_i(IV, m_0, \ldots, m_i)$ for $i \in \{p + 8, \ldots, q + 8\}$.
3. Compute μ_{j-8}. The message $M' = \mu_{j-8}||m_1^*|| \ldots ||m_8^*||m_{j+1}||..||m_{\ell-1}$ is a second preimage of $\mathcal{H}(M)$.

7.2 Expandable Messages for Hamsi-256

Expandable messages are generated using the multicollision algorithm of [6]. Expandable messages of size 2^k can be generated by iterating the following search.

Set $C_0 = IV$ (the initialization vector of Hamsi-256). For all i in $\{0, \ldots, k - 1\}$, find two sequences of message blocks $L_{i,0} = (a_{i,1}, \ldots, a_{i,\alpha_i})$ and $L_{i,1} = (b_{i,1}, \ldots, b_{i,\alpha_i+2^i})$ such that :

$$C_{i+1} = F_\alpha(C_i, a_{i,1}, \ldots, a_{i,\alpha_i}) = F_{\alpha_i+2^i}(C_i, b_{i,1}, \ldots, b_{i,\alpha_i+2^i}).$$

Let $p = \sum_{i=0}^{k-1} \alpha_i$, and $j \in \{p, \ldots, r + 2^k - 1\}$. We can write $j = p + \sum_{i=0}^{k-1} x_i 2^i$, with $x_i \in \{0, 1\}$. Then the sequence $\mu_j = (L_{0,x_0}, \ldots L_{k-1,x_{k-1}})$ has length j, and $\mathcal{F}_j(C_0, \mu_j) = C_k$. In the generic case, Kelsey and Schneier take $\alpha_i = 1$ for all i. The cost of each step of the search is then about $2^{n/2}$ because of the birthday paradox, leading to an overall complexity of about $k2^{n/2}$.

Hamsi-256 has the specific property that the message blocks are small compared to the chaining variables. Therefore, if the attacker chooses $\alpha_i = 1$, he can generate only 2^{32} values for the sequence $L_{i,0}$. In the first iterations, the probality to find a collision is very small, and the cost of iterations for $i \geq 3$ is about $2^{256-32} = 2^{224}$. To keep an equivalent complexity, one then needs to choose $\alpha_i = 4$ for each value of i leading to a $(4k, 4k+2^k-1)$ expandable message after about $k2^{128}$ compression evaluations.

7.3 Complexity Evaluation

In the case of Hamsi-256 we choose $p = 4k$ and $q = 4k + 2^k - 1$ such that $q + 8 \leq \ell - 1$. The last two compression functions of the computation of $\mathcal{H}(M)$ involve message blocks representing the bitlength of m, and the block before contains padding bits so we do not take the resulting chaining value into account.

The cost of the expandable message generation is then about $k2^{128}$ compression function evaluations. The average number of trials for the second step is then about $\frac{2^{256}}{q-p+1} = 2^{256-k}$. The message μ_{j-8} can be recovered easily. The overall complexity of the attack is then:

$$T(k) = k2^{128} + 2^{256-k} \tag{10}$$

The complexity of the attack is the same as the one found by Kelsey and Schneier, but the condition on the message length is slightly different ($\ell \geq 4k + 2^k + 8$ instead of $\ell \geq k + 2^k + 1$). As a result, our attack described in previous Section is more efficient than this generic attack for messages which length is between 10 and 96 blocks.

7.4 Possible Improvements

Some small improvement of our second preimage attack could be obtained by mixing the attack on the domain extender by Kelsey and Schneier with our pseudo preimage finding algorithm. For example, one could try to invert some of the intermediate chaining variables involved in the computation of $\mathcal{H}(m)$ between the two steps of the generic attack, so as to increase the potential number of targets for the second phase. Such an attack could however only be efficient for short messages, as the interest of our pseudo preimage algorithm is that it discards some values of (C, m) due to linear relations. If the target space becomes larger than 2^{14}, almost every value of $\mathcal{F}(m, C)$ will be computed anyway, and applying our technique is pointless.

8 Conclusion and Openings

In this article we displayed the first attack on Hamsi-256 that runs faster than generic attacks on hash functions. Though it has some similarities with differential attacks, such as the study of the propagation of variables or the reduction

of the search space by setting some conditions, it is mainly an algebraic attack. For short messages, our algorithm is faster than generic attacks on the the Merkle-Damgård domain extender as used for Hamsi. While the attack complexity does not represent any practical immediate threat for the use of Hamsi-256, it enlightens some weaknesses in its design.

Acknowledgements

I would like to thank Henri Gilbert and the anonymous reviewers of Asiacrypt 2010 for their helpful comments on earlier versions of this paper. Many thanks to Adi Shamir and Itai Dinur for the ideas we exchanged on Hamsi-256.

References

1. Aumasson, J.-P.: On the pseudorandomness of hamsi. NIST mailing list (local link) (2009)
2. Aumasson, J.-P., Meier, W.: Zero-sum distinguishers for reduced keccak-f and for the core functions of luffa and hamsi. NIST mailing list (2009)
3. Calik, C., Turan, M.S.: Message recovery and pseudo-preimage attacks on the compression function of hamsi-256. Cryptology ePrint Archive, Report 2010/057
4. Küçük, Ö.: The hash function hamsi. Submission to NIST (updated) (2009)
5. Dinur, I., Shamir, A.: Cube attacks on tweakable black box polynomials. In: EUROCRYPT, pp. 278–299 (2009)
6. Joux, A.: Multicollisions in iterated hash functions. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 306–316. Springer, Heidelberg (2004)
7. Kelsey, J., Schneier, B.: Second preimages on n-bit hash functions for much less than 2^n work. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 474–490. Springer, Heidelberg (2005)
8. Wang, K.J.M., Wang, X., Wang, W.: New pseudo-near-collision attack on reduced-round of hamsi-256. Cryptology ePrint Archive, Report 2009/484 (2009), http://eprint.iacr.org/
9. Nikolic, I.: Near collisions for the compression function of hamsi-256. In: CRYPTO rump session (2009)
10. Shamir, A., Dinur, I.: An algebraic attack on hamsi-256 (to appear)
11. Vielhaber, M.: Aida algebraic iv differential attack breaking one.fivium by aida an algebraic iv differential attack (2007)

Non-full-active Super-Sbox Analysis: Applications to ECHO and Grøstl

Yu Sasaki[1], Yang Li[2], Lei Wang[2], Kazuo Sakiyama[2], and Kazuo Ohta[2]

[1] NTT Information Sharing Platform Laboratories, NTT Corporation
3-9-11 Midoricho, Musashino-shi, Tokyo 180-8585 Japan
sasaki.yu@lab.ntt.co.jp
[2] The University of Electro-Communications
1-5-1 Choufugaoka, Choufu-shi, Tokyo, 182-8585 Japan
{liyang,wanglei,saki,ota}@ice.uec.ac.jp

Abstract. In this paper, we present *non-full-active Super-Sbox analysis* which can detect non-ideal properties of a class of AES-based permutations with a low complexity. We apply this framework to SHA-3 round-2 candidates ECHO and Grøstl. The first application is for the full-round (8-round) ECHO permutation, which is a building block for 256-bit and 224-bit output sizes. By combining several observations specific to ECHO, our attack detects a non-ideal property with a time complexity of 2^{182} and 2^{37} amount of memory. The complexity, especially in terms of the product of time and memory, is drastically reduced from the previous best attack which required $2^{512} \times 2^{512}$. Note that this result does not impact the security of the ECHO compression function nor the overall hash function. We also show that our method can detect non-ideal properties of the 8-round Grøstl-256 permutation with a practical complexity, and finally show that our approach improves a semi-free-start collision attack on the 7-round Grøstl-512 compression function. Our approach is based on a series of attacks on AES-based hash functions such as rebound attack and Super-Sbox analysis. The core idea is using a new differential path consisting of only non-full-active states.

Keywords: AES-based permutation, ECHO, Grøstl, SHA-3, Super-Sbox.

1 Introduction

Hash functions are used in the wide range of cryptographic applications. Since the break of MD5 and SHA-1 [1,2], cryptographers have been seeking secure and efficient hash constructions. From these backgrounds, NIST started the competition to determine the future standard hash function called SHA-3 [3].

In the SHA-3 competition, 14 algorithms are being considered as round 2 candidates. At the present time, none of them has been seriously broken in terms of the important security properties such as collision resistance or preimage resistance. However, regarding some candidates, building blocks such as compression functions or internal permutations have been shown that they do not satisfy

M. Abe (Ed.): ASIACRYPT 2010, LNCS 6477, pp. 38–55, 2010.

ideal properties. Although it does not damage the security of hash functions immediately, the analyses against building blocks are useful to know the potential weakness, security margin, validity of the security proof, and so on.

Many of the SHA-3 candidates are based on the design strategy of AES [4,5]. Recently, an outstanding progress in the cryptanalysis against AES-based hash functions or permutations has been made [6,7,8,9,10,11,12,13,14,15,16]. Specifically, *Rebound attack* proposed by Mendel *et al.* at FSE 2009 [7], *Start-from-the-Middle attack* proposed by Mendel *et al.* at SAC 2009 [8], and *Super-Sbox analysis* applied to the rebound attack by Lamberger *et al.* at Asiacrypt 2009 [15] and by Gilbert and Peyrin at FSE 2010 [9] have wide range of their applications and are powerful analytic tools. In fact, the rebound based attack has been applied to several SHA-3-candidates [7,8,9,10,11,12,13,14,17] such as Grøstl [18], ECHO [19], JH [20], Cheetah [21], LANE [22], Twister [23]. It has also been applied to other hash functions [7,8,9,15,16] such as Whirlpool [24] and AES hashing modes.

ECHO [19], designed by Benadjila *et al.*, is one of the round 2 algorithms in the SHA-3 competition using a 2048-bit AES-based permutation. The number of rounds in the permutation is 8 for ECHO-224 and -256, and 10 for ECHO-384 and -512. At FSE 2010, Gilbert and Peyrin showed that the full-round (8-round) ECHO permutation could be distinguished from an ideal permutation with time of 2^{768} and memory of 2^{512} by using the Super-Sbox analysis [9]. After that, Peyrin [25,26] improved this attack which required 2^{512} in both time and memory. Because the 8-round ECHO permutation is a building block to generate 256-bit or 224-bit hash values and compression part from 2048-bits to 256- or 224-bits is not considered, the impact of this attack seems almost negligible. In addition, as long as it is evaluated by the framework of [9], the time or memory cannot be below 2^{512}[1]. To sum up, there is no powerful analysis on the ECHO hash function nor compression function. Even though attacks on the permutation reached full-round, the complexity is too high.

Note that the reduced ECHO compression function is attack by Peyrin [26]. Recently, Schläffer presented the analysis on ECHO [27] and Ideguchi *et al.* presented the analysis on Grøstl [28]. These results are listed in Table 1.

Our Contributions

In this paper, we present *non-full-active Super-Sbox analysis* which can detect non-ideal properties of a class of AES-based permutations with a low complexity. To demonstrate its applicability, we first apply the non-full-active Super-Sbox analysis to the 8-round Grøstl-256 permutation, which is an AES-based permutation consisting of the 8×8 state. This attack can detect a non-ideal property of the 8-round Grøstl-256 permutation with time of 2^{48} and memory of 2^8, while detecting the same property of an ideal permutation requires 2^{96}. We then apply this framework to the full-round (8-round) ECHO permutation by optimizing the attack with taking several properties specific to ECHO into account. This attack

[1] Reasons of this limitation are explained in [9, Section 4.4] and [26, Appendix B].

Table 1. Comparison of attack results on ECHO and on Grøstl

Target	Rounds	Time	Memory	Attack Type	Paper
ECHO-256/-224 Permutation	8 (full)	2^{768}	2^{512}	Distinguisher	[9]
	8 (full)	2^{512}	2^{512}	Distinguisher	[26]
	8 (full)	2^{182}	2^{37}	Distinguisher	Sect. 5.2
	7	2^{128}	2^{32}	Distinguisher	[26]
	7	2^{118}	2^{38}	Distinguisher	Append. A
ECHO-256/-224 Single-pipe Comp. Func.	3	2^{64}	2^{64}	Distinguisher	[26]
	3	2^{32}	2^{38}	Distinguisher	Append. B
Grøstl-256 Permutation	8	2^{112}	2^{64}	Distinguisher	[9]
	8	2^{64}	2^{64}	Distinguisher	[28]
	8	2^{48}	2^{8}	Distinguisher	Sect. 4.4
Grøstl-512 Comp. Function	7	2^{152}	2^{64}	Semi-free-start coll.	[17]
	7	2^{152}	2^{56}	Semi-free-start coll.	Sect. 5.3
ECHO-256 Hash Function	4	2^{64}	2^{64}	Collision	[27]
	5	2^{96}	2^{64}	Distinguisher	[27]
ECHO-256 / -512 Comp. Function	3/3	$2^{64}/2^{96}$	$2^{64}/2^{64}$	Semi-free-start coll.	[26]
	7/7	$2^{107}/2^{106}$	$2^{64}/2^{64}$	Distinguisher	[27]
Grøstl-256 Comp. Func.	10 (full)	2^{192}	2^{64}	Distinguisher	[26]
Grøstl-512 Comp. Func.	11	2^{640}	2^{64}	Distinguisher	[26]

can detect a non-ideal property of the 8-round ECHO permutation with time of 2^{182} and memory of 2^{37}, while detecting the same property of an ideal permutation requires 2^{256}. Note that the 8-round ECHO permutation is a building block for ECHO-256 and ECHO-224. As far as we know, this is the first result on the full-round ECHO permutation which can work with both time and memory (or product of these factors) below 2^{256} (or 2^{224}). Note, however, that the role of the convolution in the ECHO compression function is very important for its security and our distinguisher cannot be extended to the ECHO compression function, nor the hash function. Finally, we show that our approach also improves the amount of memory for the semi-free-start collision attack on the 7-round Grøstl-512 compression function to 2^{56} from 2^{64}. In appendices, we show new results on the reduced-round ECHO permutation and compression function. An attack on the 7-round ECHO permutation and a low complexity distinguisher on the 3-round single-pipe ECHO-256 compression function are included. The attack results are summarized in Table 1. The technical details in this paper are as follows.

Low complexity distinguishers on AES-based permutations. We present a new strategy of the Super-Sbox analysis which can work for a class of AES-based permutations in generic. The core idea is using a differential path whose inbound part, in particular inside the Super-Sbox, consists of only non-full-active states. Regarding non-active bytes, the difference is always 0 through the SubBytes and InverseSubBytes operations regardless of

its value. Hence, attackers can freely choose the value without breaking the differential path. This freedom degrees enable attackers to control values (or differences through the SubBytes operation) of other bytes inside the Super-Sbox to be connected efficiently.

Observations on the property of ECHO permutation. We explain two new observations on the ECHO permutation when dealing with the byte-wise truncated differential path. First, we find that the linearity of the jointed two linear operations (MixColumns inside the BigSB and the following BigMC) should be taken into account in order to correctly calculate the complexity for a certain differential path. Second, there are freedom of the differential paths inside BigSB available to attackers to reduce the complexity.

In Section 2, we describe AES-permutation, ECHO, and Grøstl. In Section 3, we introduce previous work. In Section 4, we present the framework of non-full-active Super-Sbox analysis and show its application to the 8-round Grøstl-256 permutation. In Section 5, we attack the full-round ECHO permutation and the 7-round Grøstl-512 compression function. In Section 6, we conclude this paper. Results on other variants of ECHO are described in appendices.

2 Specifications

AES [4,5] is a 128-bit block-cipher represented by a 4×4 byte state. Here we consider a general AES-based permutation with $r \times r$ state where each element is a c-bit word. The row and column positions of a word/byte is denoted by i and j, respectively where $i, j \in [0, r-1]$. As shown in Fig. 1, the state is updated by four operations in a round of the AES-based permutation.

- SubBytes (SB): non-linear word/byte substitution according to an S-box.
- ShiftRows (SR): each word/byte at row j is rotated to left by j positions.
- MixColumns (MC): multiply each column by a MDS matrix.
- AddRoundKey (AK): bit-wise XOR of the current state and a constant.

2.1 ECHO Permutation

ECHO [19] designed by Benadjila *et al.* is a hash function using a 2048-bit AES-based permutation as its building block. The permutation consists of 8 rounds for ECHO-224 and -256, and 10-rounds for ECHO-384 and -512. The 2048-bit

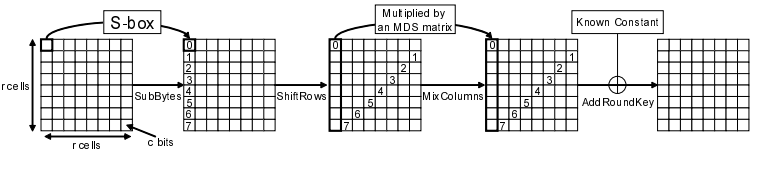


Fig. 1. The operations inside a round of AES-based permutation

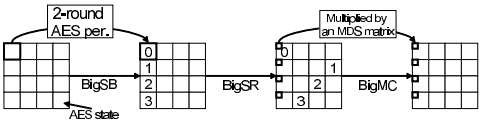


Fig. 2. One round of ECHO permutation

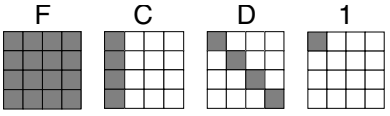


Fig. 3. Notations for ECHO BigWords

internal state can be represented by a 4×4 matrix where each element is a 128-bit AES state called a BigWord. The round operation in the ECHO permutation manipulates 128-bit BigWords instead of 8-bit bytes. One round of the ECHO permutation shown in Fig. 2 has three operations:

- BigSB: substituting each BigWord by applying two AES-rounds.
- BigSR: each BigWord at row j is rotated to left by j positions.
- BigMC: multiply each 4 bytes of the ECHO state by a MDS matrix.

To simplify the dedicated analysis on ECHO, as introduced by [26], we denote 4 types of byte-wise truncated differences of the BigWord as shown in Fig. 3.

2.2 Grøstl Permutation and Compression Function

Grøstl designed by Gauravaram *et al.* [18] is another hash function built upon the AES-based permutations. Grøstl-256 permutation uses an 8×8 state where each element is an 8-bit byte, while Grøstl-512 permutation uses an 8×16 state. The number of rounds in the permutation is 10 for Grøstl-224 and -256, and 14 for Grøstl-384 and -512. The Grøstl-512 uses different ShiftRows operation from Grøstl-256, where the bytes at row 7 are rotated to left by 11 positions.

3 Previous Work

Rebound attack was proposed by Mendel *et al.* at FSE 2009 [7], which is useful to analyze AES-based permutations. It divides a differential path into two parts; inbound and outbound phases. Inbound phase controls the most expensive part of the differential path with a very low average complexity, then outbound path is satisfied probabilistically. It needs to make sure the total number of starting points generated at the inbound phase is enough to fulfill the outbound path.

Start-from-the-Middle attack was proposed by Mendel *et al.* at SAC 2009 [8]. It improves the rebound attack by extending the number of controlled rounds from 2 to 3. The idea is to utilize the independence and the freedom of each search procedure as much as possible. As a result, without increasing the time and memory, 3 rounds of the differential path can be fulfilled efficiently.

Super-Sbox analysis for the rebound attack was independently proposed by Lamberger *et al.* at Asiacrypt 2009 and by Gilbert and Peyrin at FSE 2010 [9]. Super-Sbox combines 2 non-linear layers and 1 diffusion layer to 1 non-linear layer with a larger substitution-box. It can extend the inbound phase by one

more round. As a side effect, attackers need to spend more time and memory to exploit the differential property of the Super-Sbox.

Peyrin proposed a differential path for ECHO with an increased granularity [26]. This can reduce the number of active bytes inside an active BigWord, and thus the attack complexity can be reduced.

4 Framework of Non-full-active Super-Sbox Analysis

In this section, we use the following notations:

r: a number of rows and columns in a state.
c: a number of bits of each cell (word) in a state.
s: a number of non-active columns in the initial state of the differential path.
$Col(x)$: a state where x columns are fully active, namely, $r \times x$ bytes are active.
$SR(Col(x)), SR^{-1}(Col(x))$: a state where $Col(x)$ is passed over SR and SR^{-1}.
F: a state where all bytes are active.
x/y: a state where y bytes become non-active from a state x.

In the Super-Sbox analysis, as long as we follow the strategy of Gilbert and Peyrin [9], the attack complexity is lower-bounded by 2^{rc}. In this section, we present a new framework called non-full-active Super-Sbox analysis which can detect non-ideal properties with a lower complexity. We first make a truncated differential path whose inbound part, in particular inside the Super-Sbox, consists of non-full-active states. For non-active bytes, the differential transition 0 to 0 is always held regardless of its value, and thus attackers can freely choose the value without breaking the path. This gives attackers the freedom degrees to adjust other bytes inside the Super-Sbox.

Non-full-active Super-Sbox analysis can be applied to AES-based permutations. We assume that the MixColumns operation is composed of MDS matrix [5]. Namely, the sum of the number of active bytes in the input and output states is greater than or equal to $r + 1$, otherwise 0.

4.1 Non-full-active Truncated Differential Path

We show a generic description of the non-full-active differential path. The differential path has a parameter s, which is the number of non-active columns in the initial state. The parameter s determines the complexity of the attack. The differential path is depicted in Fig. 4 with instantiating the case $r = 8$ and $s = 3$.

To make the differential path, we start from the state after the 2nd and 5th rounds, whose states are $Col(1)/s$ and $SR^{-1}(Col(1))/(r-(s+1))$, respectively. The differential propagation through the 3rd round in forward and the 5th round in backward are deterministic, which result in $F/Col(s)$ and $F/SR^{-1}(Col(r-(s+1)))$, respectively. We then need to check that the differential propagation through the MixColumns operation in the 4th round is consistent with the MDS property. Because input and output states have $r-s$ and $r-(r-(s+1))$ active bytes in each column respectively, the sum of active bytes in the input and output

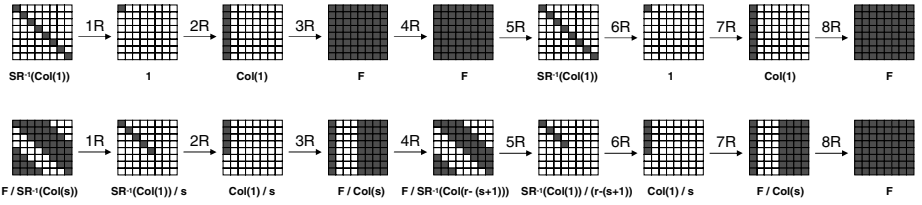


Fig. 4. (Bottom) New differential path for 8-round AES-based permutations with instantiating $r = 8$ and $s = 3$. (Top) Previous path for the Super-Sbox analysis [9].

is $r - s + r - (r - (s + 1)) = r + 1$. Hence, the differential path is consistent with the MDS property. Next, we determine the differential propagation through the 6th round in forward. The number of active bytes should be reduced as much as possible after the 6th round in order to make the target non-ideal property hard for an ideal permutation. Hence, we maximize the number of non-active bytes with satisfying the MDS property, which results in the state $Col(1)/s$. Similarly, we determine the differential propagation through the 2nd round in backward. We make the number of active bytes to be the same as the state after the 6th round[2], which results in $SR^{-1}(Col(1))/s$. The rest of the path is deterministic.

4.2 Low Complexity Inbound Phase

We explain how to compute the inbound phase for our path. Details of states inside the inbound phase are shown in Fig. 5, with denoting each state by $\#i$, where $0 \leq i \leq 8$. The inbound phase starts from the state after the SubBytes in the 3rd round (#0) and the state input to the 6th round (#8). The goal of the inbound phase is finding paired values satisfying the differential path through #0 to #8. We find 2^c such paired values with 2^c computations and 2^c memory.

States #0 and #8 include $r - s$ and $s + 1$ active bytes, respectively. First, we choose and fix the differences of all active bytes in #0 and the differences of s active bytes out of $s + 1$ active bytes in #8. Then, for each 2^c possible differences of the last active byte in #8, we aim to store a corresponding paired value. Due to the linearity of the operations, we can compute the corresponding differences in state #2 and corresponding s-byte differences in each column of #6. The Super-Sbox analysis can be applied between #2 and #6, namely we can compute them column by column independently. Previous Super-Sbox analysis spent 2^{rc} of time and 2^{rc} of memory for this computation, while we efficiently connect these two states by using the freedom degrees of the non-active states. In the following, we only show the Super-Sbox computations in the left most column, which is emphasized with bold squares in Fig. 5. The other columns can be connected with the same procedure.

[2] With a lower probability, the number of active bytes can be smaller. However, this will not lead to any advantage in the distinguishing attack.

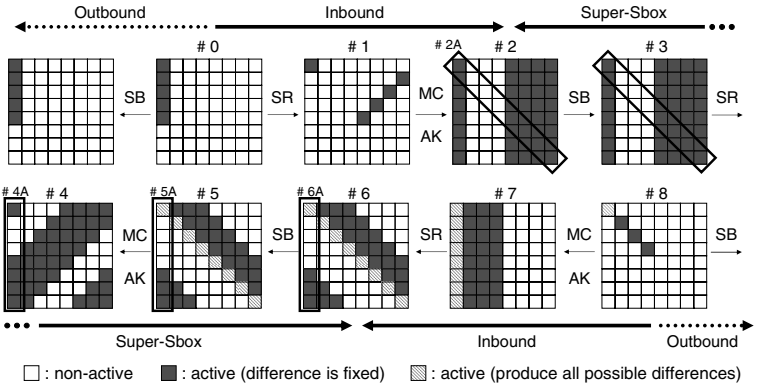


Fig. 5. Inbound phase for the new differential path with non-full-active states

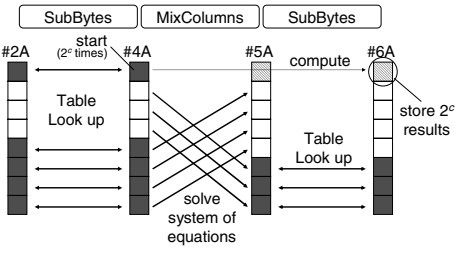


Fig. 6. Computation procedures inside each Super-Sbox

Computation procedure inside the Super-Sbox. The operations inside the Super-Sbox are shown in Fig. 6. Because the ShiftRows operation does not give any impact inside the Super-Sbox, we omit it in Fig. 6. To stress that each Super-Sbox is computed column by column, we denote the states inside the Super-Sbox by #2A, #4A, #5A, and #6A in Fig. 6. The goal of this procedure is efficiently producing 2^c paired values which satisfy the fixed part of the differences of #2A and #6A. This procedure finds 2^c paired values with a time complexity of 2^c and 2^c memory. The attack procedure is as follows.

0. For each active byte whose difference is fixed in #2A and #6A, compute SB and Inverse-SB for all possible 2^c values and a fixed difference. Store these 2^c values and corresponding output differences as a look up table. We sort tables according to the output differences so that table look-up only requires 1 memory access. As a result, $(r-s)+s=r$ look-up tables are prepared.
1. Choose a difference of one active byte in #4A. (The top byte of #4A is chosen in Fig. 6.)
2. We have other $r-s-1$ active bytes in #4A and need to make sure the same number of bytes in #5A are non-active. This is done by solving a system of equations and we will obtain one solution of the system. As a result, differences in #4A and #5A become consistent and are uniquely fixed.

3. From a fixed difference of #4A and the given difference of #2A, for each active byte, we obtain a pair of values which connects these differences by looking up tables generated in Step 0. Do the same for fixed s-byte differences of #6A and #5A. Note that values for non-active bytes are not fixed yet.
4. Then we connect the values of active bytes of #4A and #5A. We use the freedom degrees of non-active bytes to effectively achieve this. There are s non-active bytes in #4A and s active bytes in #5A whose values are fixed in Step 3. By solving a system of equations, we can calculate the values of s non-active bytes in #4A so that the fixed s bytes of #5A can be consistent.
5. With the fixed values in #4A, we compute the non-fixed active byte in #5A, and further compute the corresponding value in #6A. We store entire values and differences of states #2A and #6A in a table.
6. We iterate Step 1 to Step 5 2^c times by changing the difference of the chosen active byte in #4A.

Complexity of inbound phase. We assume r and s are enough small compared to 2^c (e.g. $r = 8, s = 3$, and $2^c = 256$ in Fig. 4). Step 0 requires 2^c computations and 2^c memory. Step 1 to Step 4 can be computed with a complexity of 1 (Based on the assumption, the cost for looking-up r tables and solving systems of equations of size s are ignored). Step 5 uses a memory of 1. Because Steps 1 to 5 are repeated 2^c times in Step 6, the complexity of this procedure is time 2^c and memory 2^c. Note that 2^c values and differences of the non-fixed active byte are stored in the table. Therefore, we obtain 1 solution on average for any difference of the non-fixed byte.

After we finish the computation for all Super-Sboxes, we choose a difference of the non-fixed byte in #8 in Fig 5. For each of its possible 2^c differences, we compute the corresponding difference in #6, and obtain the value which connects #2 to #6 by looking up each Super-Sbox. Note, we obtain one solution on average for any pair of differences in #2 and #6. To sum up, we can obtain 2^c starting points, which are solutions of the inbound phase, with time 2^c and memory 2^c. In other words, we obtain a starting point with time 1 on average.

4.3 Outbound Phase and the Freedom Degrees

After the inbound phase, we compute the outbound phase. The differential path described in Fig. 4 has two probabilistic differential propagations: 1) the backward computation through the 2nd round and 2) the forward computation through the 6th round. In both rounds, the MixColumns or InverseMixColumns operations need to produce s non-active bytes. Therefore, for each of these rounds, the success probability is 2^{-cs}. Finally, this attack requires 2^{2cs} starting points for the outbound, and each starting point is generated with time 1 on average. Hence, with a time 2^{2cs}, we find a pair following the differential path.

We also need to confirm that the available freedom degree is enough. Our attack starts from the states #0 and #8 in Fig. 5. #0 and #8 include $r - s$ and $s+1$ active bytes respectively, and thus we have $2^{c(r+1)}$ freedom degrees in total.

Table 2. The complexity to find a property with our attack and ideal permutation

s	1	2	3	4	5	6	7	8
Ours	2^{2c}	2^{4c}	2^{6c}	2^{8c}	2^{10c}	2^{12c}	2^{14c}	2^{16c}
Ideal	$2^{\frac{cr}{2}}$	2^{cr}	$2^{\frac{3cr}{2}}$	2^{2cr}	$2^{\frac{5cr}{2}}$	2^{3cr}	$2^{\frac{7cr}{2}}$	2^{4cr}

Hence, as long as the parameter s satisfies $2^{c(r+1)} \geq 2^{2cs}$, which is converted to below, we have enough freedom degrees.

$$s \leq \frac{r+1}{2} \tag{1}$$

4.4 Target Class of AES-Based Permutations and an Example

Let us consider the complexity for an ideal permutation. The last MixColumns is not taken into account because it is fully linear. Hence, the problem is regarded as finding a crs-bit collision. A crs-bit collision can be found by the birthday attack because attackers have enough freedom degrees due to Eq. (1). Hence, the complexity for an ideal permutation is $2^{\frac{crs}{2}}$. The comparison of the non-full-active Super-Sbox analysis and the ideal case is shown in Table 2.

From Table 2, we can see $r > 4$ is a condition so that our attack can work. Therefore, our attack cannot be applied to AES ($r = 4$). Note that the ECHO permutation is regarded as an AES-based permutation with $r = 4$ at a BigWord level. However, it has other structures and this enables us to greatly reduce the attack complexity on the ECHO permutation. See Section 5 for details.

Let us consider an application for a real primitive. Grøstl-256 uses an AES-based permutation with $r = c = 8$. In previous Super-Sbox analysis [9], the 8-round permutation is distinguished with time 2^{112} and memory 2^{64}, which is too expensive to be implemented. In our attack, we choose $s = 3$, whose differential path is shown in Fig. 4. Consequently, from Table 2, we can detect a pair of values following the differential path with time 2^{48} and 2^8 memory, while finding a pair of values in an ideal permutation requires 2^{96}, which is infeasible. Choosing other s is also possible as long as $s \leq 4$.

5 Applications to ECHO and Grøstl

5.1 New Observations on ECHO

In this section, we explain several new observations on the ECHO permutation when dealing with the dedicated byte-wise differential path.

Complexity analysis for jointed MixColumns and BigMC. In the ECHO permutation, 2-round AES permutation inside BigSB can be considered as a non-linear layer with Super-Sboxes and a diffusion layer consisting of ShiftRows, MixColumns and AddRoundKey. Note that from the second MixColumns inside

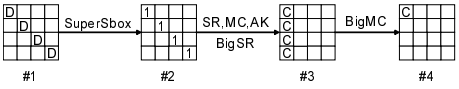


Fig. 7. A differential path for a 1-round ECHO permutation

BigSB to the following BigMC are successively performed. We show that the linearity of jointed MixColumns and BigMC should be considered to correctly compute the complexity for certain differential paths.

As an example, let us check the complexity for the differential path shown in Fig. 7 assuming the differences and real values at state #1 have full freedom. In the previous analysis [26, Appendix B], the complexity for this differential path is likely to be divided into three parts and analyzed independently. State #1 to #2 can be fulfilled when the output of each active Super-Sbox has only 1 active byte. Since there are totally 12 bytes required to be zero, the probability is regarded as 2^{-96}. The complexity from #2 to #3 is 1. And since 12 bytes are required to be zero from #3 to #4, the probability is regarded as 2^{-96}. As a result, the total probability is regarded as $2^{-96\times 2} = 2^{-192}$. However we show that MixColumns and BigMC cannot be considered separately, and thus the correct probability needs to be reconsidered.

We can see that the freedom of the difference for state #2 or #3 is at most 2^{32}, since #2 has only 4 active bytes. As a contradiction for the previous analysis, the freedom of difference at #3 (2^{32}) seems impossible to fulfill the differential propagation to #4 (2^{-96}). However, we show that this propagation is fulfilled only with a probability of 2^{-24}, and thus 2^{32} freedom degrees are enough.

This fact can be understood from two directions. First, for a position-fixed active byte and the fixed MDS matrix used in MixColumns between #2 and #3, the 4 active bytes inside each active BigWord at #3 has a fixed linear relationship. Then if BigMC generates the required difference at #4 for one of 4 active-byte positions with a probability of 2^{-24} (e.g. 4 top-left bytes from 4 active BigWord at #3 generate 1 active byte at the top-left of state #4), the other three active-byte positions become the same differential pattern at #4 with probability 1. Another interpretation is that one can switch the operation order, namely performing BigMC first and MixColumns later. When 4 active bytes in #2 generate only 1 active byte through BigMC with a probability of 2^{-24}, the differential path from #3 to #4 through MixColumns is fulfilled with probability 1. As a result, the total complexity is $2^{96+24} = 2^{120}$ instead of 2^{192}. Note that this fact was independently discovered by [27] as SuperMixColumns.

Freedom of the differential path inside BigSB. We can use the freedom of the differential path inside BigSB to reduce the attack complexity. Our attacks only care about the differences at the start and end states of the permutation. We notice that while keeping the differential path at a BigWord level, attackers can use the freedom of the differential paths at a byte level inside BigSB.

We again use the differential path in Fig. 7 as an example. In order to fulfill the differential path, the 4 active bytes in state #2 must be at the same position

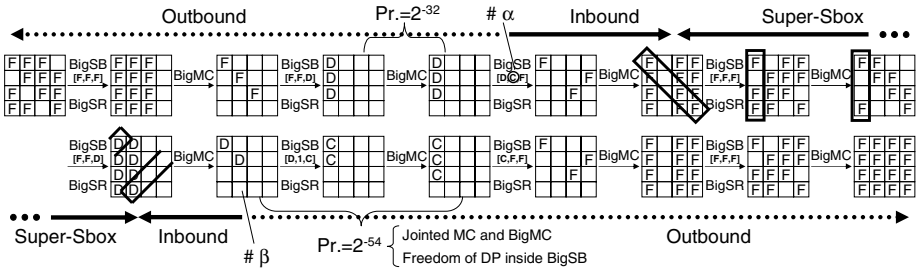


Fig. 8. Non-full active differential path for 8-round ECHO permutation

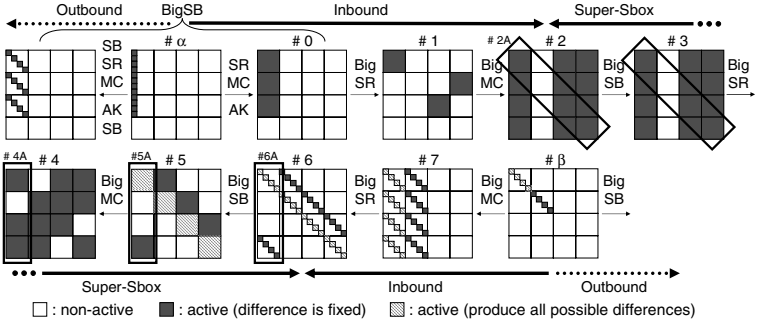


Fig. 9. Inbound phase for 8-round ECHO permutation

inside the leftmost column of each BigWord[3]. As a result, the differential path inside the BigSB has 4 choices for the positions of active bytes, and thus, the complexity for the differential path in Fig. 7 can be reduced from 2^{120} to 2^{118}.

5.2 Attack on Full-Round ECHO Permutation

Truncated Differential path. We use the differential path explained in Section 4.1 with parameter $s = 1$ at a BigWord level, which is shown in Fig. 8. We use the notation BigSB$[x, y, z]$, where $x, y, z \in \{F, D, C, 1\}$ to show that x, which is the input differential pattern to BigSB, changes into y after the 1st AES-round and into z after the 2nd AES-round.

Inbound phase. The detailed differential path for the inbound phase is described in Fig. 9. The inbound phase starts from a middle of BigSB in the 3rd round ($\#\alpha$) and the input state to the 6th round ($\#\beta$), where the differential form in $\#\alpha$ is C. We first choose and fix a difference of $\#\alpha$ and a difference of one of active BigWords of $\#\beta$, and compute the corresponding differences of

[3] If 4 active bytes in state #2 are in different positions inside each BigWord, the path for Fig. 7 becomes impossible. This may be used as a countermeasure of our attack.

#2 and #6. In the inbound phase, for each of 2^{32} possible differences of the non-fixed active BigWord in $\#\beta$, we find a pair of values that satisfies the chosen differences of $\#\alpha$ and $\#\beta$. The attack procedure follows the one explained in Section 4.2 with some optimization specific to ECHO. In the followings, we describe details to compute 1 Super-Sbox of ECHO with the size of 128 bits.

0. Generate a look-up table for each 4 active BigWord with fixed difference in #2A and #6A, With the procedure in Section 4.2, this costs 2^{128} time and 2^{128} memory. [8] pointed out that this could be performed efficiently by looking inside BigSB. The BigSB can be regarded as 4 Super-Sboxes (SB, SR, MC, AK, SB) with the size of 32 bits and the linear part (SR, MC, AK). Then, for a given output difference of BigSB, we can calculate back the corresponding difference of the linear part, and thus values are searched by looking up four 32-bit Super-Sboxes independently. Hence, look-up tables for 4 BigWords can be generated by computing 16 Super-Sboxes, which requires 16×2^{32} in both time and memory.
1. Choose a difference of one active BigWord in #4A.
2. By solving a system of equations, compute differences of 2 active BigWords in #4A so that 2 target BigWords in #5A can be non-active.
3. For each active BigWord with fixed difference, obtain a pair of values which connects differences between #4A and #2A, and between #6A and #5A by looking up tables generated in Step 0.
4. By solving a system of equations, calculate the value of 1 non-active BigWord in #4A so that the fixed value of 1 BigWord in #5A can be consistent.
5. With the fixed paired values in #4A, compute the non-fixed active BigWord in #5A and #6A. Only if the computed difference of #6A has the diagonal form D, store entire values and differences of states #2A and #6A in a table.
6. Iterate Steps 1 to 5 2^{128} times by changing the difference of the chosen BigWord in #4A.

In Step 0, look up tables are generated with 2^{36} time and 2^{36} memory. Steps 1 to 5 are iterated 2^{128} times. In Step 5, the computed difference has the diagonal form D with a probability of $2^{32}/2^{128} = 2^{-96}$, and thus we store 2^{32} data after 2^{128} iterations. Hence, the complexity for 1 Super-Sbox with the size of 128 bits is 2^{128} computations and $2^{36} + 2^{32}$ memory. Note that we need $2^{36} + (4 \times 2^{32}) < 2^{37}$ memory for 4 Super-Sboxes. In the end, the inbound phase generates 2^{32} starting points with 2^{128} computations and 2^{37} memory, which is 2^{96} computations on average to generate 1 starting point.

Success probability and freedom degrees. If details are considered, Step 3 succeeds only probabilistically. Look-up tables for each BigWord consists of 4 Super-Sboxes with the size of 32 bits. Assume that each Super-Sbox has the same property as the AES Sbox. Namely, for a randomly given a pair of input and output differences, with a probability of approximately 2^{-1}, there exists approximately 2 paired values satisfying the differences. In Step 3, we look-up 16 Super-Sboxes. Hence, the success probability is 2^{-16} and we obtain 2^{16} paired values. We compute Steps 4 and 5 for all 2^{16} paired values, and thus they

are computed 2^{128} times in total by the 2^{128} iteration of Step 6. Consequently, the total time and memory for the inbound phase will not change. Note that the estimation by using average numbers is imprecise only if the cost for the outbound phase is cheaper than the inbound phase. Because our attack iterates the inbound phase 2^{54} times, the evaluation with average numbers is valid.

We then check the freedom degrees. In the inbound phase, we can choose up to 2^{96} differences for $\#\alpha$ and 2^{32} differences for the fixed active BigWord in $\#\beta$. Hence, the inbound phase can be iterated 2^{128} times and thus we can generate 2^{160} starting points in maximum, which are enough to satisfy the outbound.

Outbound phase. The differential path shown in Fig. 8 includes two probabilistic differential propagations.

InverseBigMC in the 2nd round. For each of diagonal positions, Inverse MixColumns outputs one non-active byte. This probability is $(2^{-8})^4 = 2^{-32}$.

BigSB and BigMC in the 6th round. Observations explained in Section 5.1 are applied for this part. The probability that the differences in 2 BigWords propagate as $D \rightarrow 1 \rightarrow C$ is $(2^{-24})^2 = 2^{-48}$. By taking the freedom of the differential path inside BigSB into account, the probability becomes $4 \times 2^{-48} = 2^{-46}$. In the BigMC operation, MC is computed for 4 positions. Due to the property of jointed MixColumns and BigMC operations, all of the 4 positions will make 1 non-active byte with a probability of 2^{-8} in total. As a result, the total success probability of the 6th round is $2^{-46} \times 2^{-8} = 2^{-54}$.

In the end, the success probability of the outbound phase is $2^{-32} \times 2^{-54} = 2^{-86}$.

Total complexity and comparison with ideal case. In our attack, we generate 2^{86} starting points and each of them is generated with 2^{96} computations on average. Hence, the total complexity is $2^{86} \times 2^{96} = 2^{182}$. Note that this attack requires 2^{37} memory. On the other hand, for the ideal case, the property is regarded as finding a 512-bit collision. This requires 2^{256}, which is much higher than our attack on ECHO.

5.3 Improving Semi-free-start Collisions on 7-Round Grøstl-512

We improve the semi-free-start collision attack on 7-round Grøstl-512 compression function proposed by Mendel *et al.* [17]. It uses the previous Super-Sbox analysis and thus requires 2^{64} memory. We show the memory can be reduced to 2^{56} with the non-full-active Super-Sbox analysis. Because our outbound phase is the same as [17], we only explain the inbound phase.

In the Super-Sbox analysis with a rectangle state such as $r \times 2r$, several Super-Sboxes include non-active bytes. Hence, the framework explained in Section 4 can be applied and the data stored for each Super-Sbox can be reduced. In the previous differential path [17, Fig.7] shown in Fig. 10, the 9th Super-Sbox at $\#P_3^{SH}$ takes a full-active column as input and output a full-active column, which requires 2^{64} memory. In fact, this is a bottleneck in the entire attack.

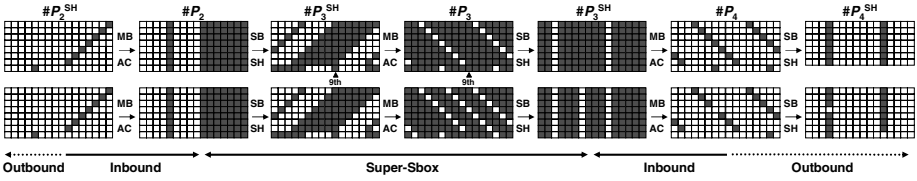


Fig. 10. (Bottom) New differential path for Grøstl-512. (Top) Previous path.

We reduce the number of active bytes where we choose the differences at the initial step of the inbound phase ($\#P_4$). This results in a differential path where each Super-Sbox has at least one non-active byte. The new path is shown in Fig. 10. Each Super-Sbox can be computed based on the procedure explained in Section 4.2, which results in generating 2^{56} starting points with 2^{56} time and 2^{56} memory. Note that the differential propagation from $\#P_3^{SH}$ to $\#P_3$ must be consistent with the MDS property. We confirmed that the amount of memory could not be below 2^{56} due to this limitation.

Because we reduced the number of active bytes, the freedom degree was also reduced. The success probability of the outbound phase is 2^{-152}, and thus we need 2^{152} starting points. Because our attack can choose 22-byte differences (8-byte for $\#P_2^{SH}$ and 14-byte for $\#P_4$) at the initial step, up to $2^{8\times 22} = 2^{176}$ starting points can be produced, which is enough to satisfy the outbound path.

6 Conclusions

We presented the non-full-active Super-Sbox analysis which can detect non-ideal properties of a class of AES-based permutations with a low complexity. The core idea is using a differential path consisting of only non-full-active states. This gives us the freedom to efficiently control inside the Super-Sbox. We then applied this framework to the full-round ECHO permutation by taking properties specific to ECHO into account. Consequently, our attack could detect a non-ideal property with time 2^{182} and memory 2^{37}. Note because of the convolution operation, our attack cannot be extended to the hash or compression function. We then applied our approach to Grøstl to obtain the distinguishing attack on the 8-round Grøstl-256 permutation with a practical cost, and to obtain an improvement on the semi-free-start collision attack on the 7-round Grøstl-512 compression function.

References

1. 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)
2. 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)
3. U.S. Department of Commerce, National Institute of Standards and Technology: Federal Register /Vol. 72, No. 212/Friday, November 2, 2007/Notices (2007)

4. U.S. Department of Commerce, National Institute of Standards and Technology: Specification for the ADVANCED ENCRYPTION STANDARD (AES) (Federal Information Processing Standards Publication 197) (2001)
5. Daemen, J., Rijmen, V.: The design of Rijndael: AES – the Advanced Encryption Standard (AES). Springer, Heidelberg (2002)
6. Peyrin, T.: Cryptanalysis of Grindahl. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 551–567. Springer, Heidelberg (2007)
7. Mendel, F., Rechberger, C., Schläffer, M., Thomsen, S.S.: The rebound attack: Cryptanalysis of reduced Whirlpool and Grøstl. In: Dunkelman, O. (ed.) Fast Software Encryption. LNCS, vol. 5665, pp. 260–276. Springer, Heidelberg (2009)
8. Mendel, F., Peyrin, T., Rechberger, C., Schläffer, M.: Improved cryptanalysis of the reduced Grøstl compression function, ECHO permutation and AES block cipher. In: Jacobson Jr., M.J., Rijmen, V., Safavi-Naini, R. (eds.) SAC 2009. LNCS, vol. 5867, pp. 16–35. Springer, Heidelberg (2009)
9. 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)
10. Rijmen, V., Toz, D., Varici, K.: Rebound attack on reduced-round versions of JH. In: Hong, S., Iwata, T. (eds.) FSE 2010. LNCS, vol. 6147, pp. 286–303. Springer, Heidelberg (2010)
11. Wu, S., Feng, D., Wu, W.: Cryptanalysis of the LANE hash function. In: Jacobson Jr., M.J., Rijmen, V., Safavi-Naini, R. (eds.) SAC 2009. LNCS, vol. 5867, pp. 126–140. Springer, Heidelberg (2009)
12. Wu, S., Feng, D., Wu, W.: Practical rebound attack on 12-round Cheetah-256. In: Lee, D., Hong, S. (eds.) ICISC 2009. LNCS, vol. 5984, pp. 300–314. Springer, Heidelberg (2010)
13. Matusiewicz, K., Naya-Plasencia, M., Nikolić, I., Sasaki, Y., Schläffer, M.: Rebound attack on the full LANE compression function. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 106–125. Springer, Heidelberg (2009)
14. Mendel, F., Rechberger, C., Schläffer, M.: Cryptanalysis of Twister. In: Abdalla, M., Pointcheval, D., Fouque, P.-A., Vergnaud, D. (eds.) ACNS 2009. LNCS, vol. 5536, pp. 342–353. Springer, Heidelberg (2009)
15. Lamberger, M., Mendel, F., Rechberger, C., Rijmen, V., Schläffer, M.: Rebound distinguishers: Results on the full Whirlpool compression function. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 126–143. Springer, Heidelberg (2009)
16. Lamberger, M., Mendel, F., Rechberger, C., Rijmen, V., Schläffer, M.: The rebound attack and subspace distinguishers: Application to Whirlpool. Cryptology ePrint Archive, Report 2010/198 (2010)
17. Mendel, F., Rechberger, C., Schläffer, M., Thomsen, S.S.: Rebound attack on the reduced Grøstl hash function. In: Pieprzyk, J. (ed.) Topics in Cryptology - CT-RSA 2010. LNCS, vol. 5985, pp. 350–365. Springer, Heidelberg (2010)
18. Gauravaram, P., Knudsen, L.R., Matusiewicz, K., Mendel, F., Rechberger, C., Schläffer, M., Thomsen, S.S.: Grøstl addendum. Submission to NIST (updated) (2009)
19. Benadjila, R., Billet, O., Gilbert, H., Macario-Rat, G., Peyrin, T., Robshaw, M., Seurin, Y.: SHA-3 proposal: ECHO. Submission to NIST (updated) (2009)
20. Wu, H.: The hash function JH. Submission to NIST (updated) (2009)
21. Khovratovich, D., Biryukov, A., Nikolić, I.: The hash function Cheetah: Specification and supporting documentation. Submission to NIST (2008)

22. Indesteege, S.: The LANE hash function. Submission to NIST (2008)
23. Ewan Fleischmann, C.F., Gorski, M.: The Twister hash function family. Submission to NIST (2008)
24. Rijmen, V., Barreto, P.S.L.M.: The Whirlpool hashing function. Submitted to NISSIE (2000)
25. Peyrin, T.: Improved differential attacks for ECHO and Grøstl. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 370–392. Springer, Heidelberg (2010)
26. Peyrin, T.: Improved differential attacks for ECHO and Grøstl. Cryptology ePrint Archive, Report 2010/223 (2010) Extended version of the CRYPTO (2010) article
27. Schläffer, M.: Subspace distinguisher for 5/8 rounds of the ECHO-256 hash function. In: Preproceedings of SAC 2010, pp. 379–398 (2010)
28. Ideguchi, K., Tischhauser, E., Preneel, B.: Improved collision attacks on the reduced-round Grøstl hash function. Cryptology ePrint Archive, Report 2010/375 (2010); Appeared in the accepted papers list of ISC (2010)

A Attack Procedures on 7-Round ECHO Permutation

Using the differential path shown in Fig. 11, we present an attack on the 7-round ECHO permutation with time of 2^{118} and memory of 2^{38}.

Step 1. An attacker picks up a difference at state #A (from 2^{128} patterns) and calculates the difference back to #B (state after the second SubBytes).

Step 2. The transformation from #B to #C can be divided into 64 independent 4-byte Super-Sboxes. For each Super-Sbox with fixed output difference, by testing all 2^{32} output values, the attacker can make a table of all possible input values and differences. At the end of Step 2, all the possible pairs at #C are stored in a table named T1 that is composed of 64 small tables each with size 2^{32}. Hence, we need 2^{38} memory for this step.

Step 3. For each active BigWord at #D, the attacker picks up a difference and calculates a corresponding difference of BigColumn at #C. Then attacker checks whether the calculated difference exists in T1. Once it exists, the attacker uses the corresponding real values at #C to calculate back the real values at #D. This test is repeated for all possible differences for each active BigWord of #D, and all possible differences and real values at #D are stored in a new table named T2. The time and memory for Step 3 are both 2^{32}.

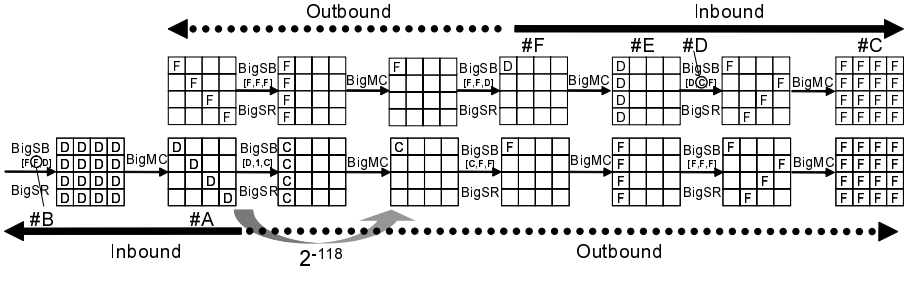


Fig. 11. Differential path for 7-round ECHO permutation

Step 4. For all the possible pairs at each active BigWord at #D, the attacker calculates the pairs at #E and stores the results as a table named T3.

Step 5. For all the possible 2^{32} differences at #F, the attacker calculates the differences at #E and checks whether the calculated difference exists in T3.

When Steps 1 to 5 are applied to Fig. 11, the inbound and backward outbound phases are merged and calculated efficiently. ABy applying the procedure once, with time of 2^{32} and memory of 2^{38}, the attacker gets 2^{32} start points. Note that with the 2^{128} freedom of the differences at #A, the forward outbound phase can be fulfilled. As a result, the total complexity is 2^{118} in time by the observations in Section 5.1 and 2^{38} in memory.

B Attack on 3-Round ECHO-SP Compression Function

Note that, for the attack in Appendix A, there is no specific requirement for the differences at state #A. Using this property we can find a non-ideal property of the 3-round single-pipe ECHO compression function specified in [19].

The differential path is shown in Fig. 12. An attacker makes sure the differences at #A can be cancelled in the compression calculation, *i.e.* for each row of BigWords at #A, the difference labeled as A is the same with the one labeled as B. By applying the procedure in Appendix A, this differential path can be satisfied using 2^{32} time and 2^{38} memory, while it costs 2^{64} for the ideal case.

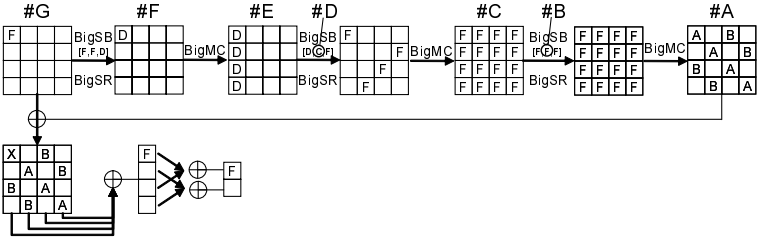


Fig. 12. Differential path of 3-round single-pipe ECHO compression function

Advanced Meet-in-the-Middle Preimage Attacks: First Results on Full Tiger, and Improved Results on MD4 and SHA-2

Jian Guo[1], San Ling[1], Christian Rechberger[2], and Huaxiong Wang[1]

[1] Nanyang Technological University, Singapore
[2] Katholieke Universiteit Leuven, ESAT/COSIC, and IBBT, Belgium
ntu.guo@gmail.com

Abstract. We revisit narrow-pipe designs that are in practical use, and their security against preimage attacks. Our results are the best known preimage attacks on Tiger, MD4, and reduced SHA-2, with the result on Tiger being the first cryptanalytic shortcut attack on the full hash function. Our attacks runs in time $2^{188.8}$ for finding preimages, and $2^{188.2}$ for second-preimages. Both have memory requirement of order 2^8, which is much less than in any other recent preimage attacks on reduced Tiger. Using pre-computation techniques, the time complexity for finding a new preimage or second-preimage for MD4 can now be as low as $2^{78.4}$ and $2^{69.4}$ MD4 computations, respectively. The second-preimage attack works for all messages longer than 2 blocks.

To obtain these results, we extend the meet-in-the-middle framework recently developed by Aoki and Sasaki in a series of papers. In addition to various algorithm-specific techniques, we use a number of conceptually new ideas that are applicable to a larger class of constructions. Among them are (1) incorporating multi-target scenarios into the MITM framework, leading to faster preimages from pseudo-preimages, (2) a simple precomputation technique that allows for finding new preimages at the cost of a single pseudo-preimage, and (3) probabilistic initial structures, to reduce the attack time complexity. All the techniques developed await application to other hash functions. To illustrate this, we give as another example improved preimage attacks on SHA-2 members.

Keywords: Preimage, MD4, Tiger, SHA-2, Hash function, Cryptanalysis.

1 Introduction

After the spectacular collision attacks on MD5 and SHA-1 by Wang *et al.* and follow-up work [12,39,44,45], implementors have reconsidered their choices. While starting a very productive phase of research on the design and analysis of cryptographic hash functions, the impact of these results in terms of practical and worrying attacks turned out to be less than anticipated (exceptions are *e.g.,* [26,38,40]). Instead of collision resistance, another property of hash functions is more crucial for practical security: preimage resistance. Hence, research

M. Abe (Ed.): ASIACRYPT 2010, LNCS 6477, pp. 56–75, 2010.

on preimage attacks and the security margin of hash functions against those attacks seems well motivated, especially if those hash functions are in practical use.

An important ongoing challenge is to find an efficient and trustworthy new hash function for long term use (*e.g.,* in the SHA-3 competition). For new hash functions, an important first step to get confidence in them is to apply known cryptanalysis methods in order to break them. So the cryptanalysts' toolbox needs to be well equipped for this.

The new techniques we present in this paper contribute to both issues at the same time. They give new, generically applicable tools to cryptanalysts for analyzing compression functions and hash functions, and at the same time applications of them improve significantly upon known preimage attacks on hash functions in practical use, like MD4, Tiger, and SHA-256/512. In the following we outline the new tools and new results that will be described later in the paper. We describe them in a way to fit into the meet-in-the-middle (MITM) framework of Aoki and Sasaki as recently developed in a series of papers [6,7,8,36,37], although we note that the basic approach was pioneered by Lai and Massey [23]. Other interesting approaches to preimage attacks appeared in [11,13,21,22,24,25,28,29,34,46].

New methods. New methods described in this paper that are independent of a particular attack or hash functions are the following:

- **Probabilistic initial structure**, compared with (deterministic) initial structure, is found be useful for significantly reducing attack complexity for the first. To improve the time complexity of a MITM preimage attack, the attackers usually need to find more neutral words. This usually reduces the number of attackable steps, due to the fact that the more neutral words, the faster the neutrality is destroyed, and the less step can be covered for independent chunks, initial structure, and partial matching. Hence, there is a tradeoff between the size of neutral words, and attackable steps. In this paper, using MD4 in Section 3 as an example, we show one can use more neutral words, and maintain long initial structure at the same time, with cost of turning the initial structure into a probabilistic one. A similar technique has been used in [37], however there it serves the purpose of better approximating the initial structure, and the attack complexity is not reduced due to limited bits for partial matching.
- **Incorporating multi-target scenarios into the MITM framework**, leading to faster preimage attacks. The MITM framework is the basis for several theoretically interesting results on the preimage resistance of various hash functions, mostly close to brute force search complexities. One reason for this is that in order to exploit all the options of this framework, matching points of the meet-in-the-middle phase can be anywhere in the computation of the compression function, and not necessarily at their beginning or end. Even though this gives an attacker more freedom in the design of a compression function attack, this always leads to big efficiency losses when the attack

on the compression function is converted to an attack on the hash function. Hence, an attacker basically has to choose between a more restricted (and potentially much slower) strategy in the compression function attack that allows more control over the chaining values and in turn allows efficient tree- or graph-based conversion methods, or to fully exploit the freedom given by the latest versions of the MITM framework in the compression function attack at the cost of inefficient conversion methods. In Section 2.2 we describe a way to combine the best of both worlds. Later in the paper, this results in the best known preimage attacks for Tiger and the SHA-2 members.
– A simple **precomputation technique** that allows for finding new preimages at the cost of a single pseudo-preimage. See Section 3 for an application to MD4, where this approach is shown to outperform any point on the time/memory trade-off curve by Hellman [18] (which was proven optimal in [10] in the generic case).

New results in perspective. In addition to the conceptual ideas that contribute to the cryptanalysts' toolbox in general, we also apply those ideas and present concrete results. In fact, we manage to improve the best known preimage attacks on a number of hash functions in practical use. A table of best related works, and the comparison with our main results can be found in [16].

– **Tiger:** One of the few unbroken but time-tested hash functions, designed by Anderson and Biham [5] in 1996, Tiger is sometimes recommended as an alternative to MD4-like designs like SHA-1, especially because it is faster than SHA-1 on common platforms. Tiger is in practical use *e.g.,* in decentralized file systems, or in many file sharing protocols and applications, often in a Merkle-tree construction (also known as TigerTree [3]). The best collision attack on Tiger is on 19 rounds [30].[1] So far the best preimage attack on the Tiger hash function is by Wang and Sasaki [42]. Independently of our work, they applied the MITM preimage attack to Tiger reduced to 23 steps with time complexity higher than ours (1.4×2^{189}) and requirements of 2^{22} units. Our new attack improves those in many aspects and seems to be the *first cryptanalytic shortcut attack on the full Tiger hash function.* Our attack on the full 24 rounds hash functions has time complexity $2^{188.8}$ (compression function attack is $2^{185.4}$) and memory requirements are only in the order of 2^8. These results are obtained using the multi-target technique mentioned above, and a dedicated technique to construct an initial structure in a precomputation.
– **MD4:** Even though very efficient collision search methods exist for MD4 [43,35], this hash function is still in practical use. Examples include password handling in Windows NT, the S/KEY one-time-password system [17],

[1] If an attacker can choose both the difference and the actual values not only of the message, but also of the chaining input, then the full compression function can be attacked, see Mendel and Rijmen [31]. However, this attack cannot be extended on the hash function, whereas all the attacks in this paper can.

integrity checks in popular protocols *e.g., rsync* [2] or file-sharing protocols [1] and applications. The time complexity for the best known compression function attack is reduced from 2^{96} (by Leurent [27]) to 2^{72}. Assuming 2^{128} precomputation using the large computation technique mentioned above, and 2^{81} storage, the effort for finding any new preimage (be it for the same or a different target hash value as a challenge) can now be as low as $2^{78.4}$.

– **SHA-2:** The members of the SHA-2 family of hash functions are probably among the most interesting cryptanalytic targets, not only because of the uptake of its adoption in all places where a hash function is needed (and they are countless), but also because they are used to compare them to candidates of the ongoing SHA-3 competition. We use SHA-2 members as an example to illustrate the effect of using the multi-target scenario. This way we also improve the best known preimage attacks on reduced SHA-256 and reduced SHA-512. They are described in [16, Appendix A].

Outline. This paper is organized as follows. Section 2 describes the MITM preimage attack, four different methods converting the pseudo-preimage to preimage (including two new ones), and also recapitulates techniques to extend MITM based preimage attacks. We apply these new techniques to MD4 and Tiger in Section 3 and Section 4, respectively. Section 5 concludes the paper.

2 The Meet-in-the-Middle Preimage Attack

The general idea of the preimage attack, illustrated in Fig 1, can be explained as follows:

1. Split the compression function into two chunks, where the values in one chunk do not depend on some message word W_p and the values in the other chunk do not depend on another message word W_q ($p \neq q$). We follow the convention and call such words neutral with respect to the first and second chunk, respectively.
2. Fix all other values except for W_p, W_q to random values and assign random values to the chaining registers at the splitting point.
3. Start the computation both backward and forward from the splitting point to form two lists L_p, L_q indexed by all possible values of W_p and W_q, containing the computed values of the chaining registers at the matching point.
4. Compare two lists to find partial matches (match for one or a few registers instead of the full chaining) at the matching point.
5. Repeat the above three steps with different initial configurations (values for splitting point and other message words) until a full match is found.
6. Note that the match gives a pseudo-preimage as the initial value is determined during the attack. However, it is possible to convert pseudo-preimages to a preimage using a generic technique described in [33, Fact 9.99]. One can compute many pseudo-preimages, and then find a message which links the IV to one of the input chaining of the pseudo-preimages.

With the work effort of 2^l compression evaluations (let the space for both W_p and W_q be 2^l), we obtain two lists, each one containing 2^l values of the register to match. When we consider all of the 2^{2l} possible pairs, we expect to get around 2^l matches (assume we match l bits at the matching point). This means that after 2^l computations we get 2^l matches on one register, effectively reducing the search space by 2^l. Leaving all the other registers to chance allows us to find a complete match and thus a pseudo-preimage in 2^{n-l} computations if the chaining is of n bits. We repeat the pseudo-preimage finding $2^{l/2}$ times, which costs $2^{n-l/2}$, and then find a message which links to one of the $2^{l/2}$ pseudo-preimages, this costs $2^{n-l/2}$. So the overall complexity for finding one preimage is $2^{n-l/2+1}$, with memory requirement of order 2^l.

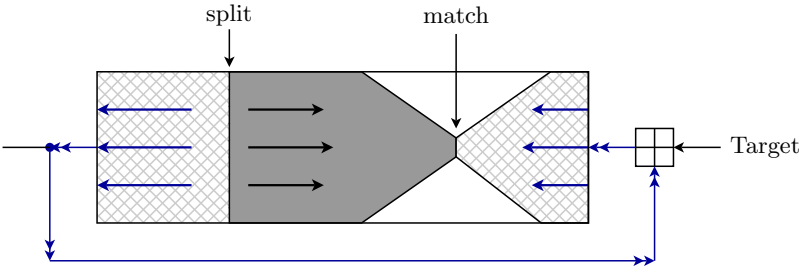


Fig. 1. Meet-in-the-Middle Pseudo-Preimage Attack against Davies-Meyer Hash Functions

Remark on bits for partial matching. Assume we have m bits for partial matching, we expect 2^{2l-m} good candidates with the m-bit matched. However we still need to check if one of the remaining candidates gives a full match (pseudo-preimage), the checking costs about 2^{2l-m} (a bit less indeed, since we can store the computed candidates up to the point before partial matching, and re-check the partial matching portion only). To minimize the time complexity, we require $m \geq l$, so that the partial matching costs $2^{2l-m} \leq 2^l$, which can be neglected.

2.1 Multi-Target Pseudo Preimage (MTPP)

In [27], Leurent provides an unbalanced-tree multi-target pseudo-preimage method to convert the pseudo-preimages to preimage with complexity $(l\ln(2) + 1) \cdot 2^{n-l}$, compared with $2^{n-l/2+1}$ in [33, Fact 9.99]. Suppose the matching point is at the end of compression function. The matching process is to find $l_p + l_q = t$ ($l_p \in L_p$, $l_q \in L_q$, and $t \in T$, the set of known targets). When we are given k targets, the chance to find a match increases by a factor k, *i.e.,* it takes $2^{n-l}/k$ to find a pseudo-preimage which links to one of the k targets. To find 2^k pseudo-preimages, it takes $2^{n-l}/1 + 2^{n-l}/2 + 2^{n-l}/3 + \cdots + 2^{n-l}/2^k \simeq k\ln(2) \cdot 2^{n-l}$. To find a preimage, it is expected to repeat 2^{n-k} blocks finding a message, which links to one of the 2^k targets. Taking the optimal $k = l$, the overall complexity is

$$2^{n-k} + k\ln(2) \cdot 2^{n-l} = (l\ln(2) + 1) \cdot 2^{n-l} \quad . \tag{1}$$

Note this conversion does not necessarily increase the memory requirement, *i.e.,* it can be the same as for finding a pseudo-preimage, since we compute the 2^l pseudo-preimages in sequence.

Enhanced 3-Sum Problem. The above conversion comes with an assumption that the matching can be done within 2^l. Note from each chunk, we have 2^l candidates (denoted as L_p and L_q), and given 2^k targets (denoted as T), we are to find *all* possible (l_p, l_q, t), where $l_p \in L_p$, $l_q \in L_q$ and $t \in T$, such that $l_p + l_q = t$. We call this problem the Enhanced 3-Sum Problem, where the standard 3-sum problem *decides* whether there is a solution [4]. Current research progress [9] shows that the problem can be solved in $O(2^{2l})$ or slightly faster. So this approach expects the matching to be done in 2^{2l} (for $k = l$) instead of the assumed 2^l. However the matching only occurs in the final feed-forward operation ("+" in most of the MD hash families), which is a small portion of the compression. Hence this approach expects 2^{2l} "+" operations to be somewhat equivalent to 2^l compression computations by counting the number of "+" in the compression, when l is relatively small (*e.g.,* ≤ 7 for MD4 and Tiger, since there are about 2^7 "+" in MD4 compression; we simply count the number of operations ("+", "−", "×" and sbox lookup) in the case of Tiger).

2.2 Generic Multi-Target Pseudo Preimage (GMTPP)

The framework of Aoki and Sasaki could not take advantage of a multi-target scenario to speed-up the conversion from pseudo-preimage to preimages. The reason is a rather strong requirement on the compression function attack by the MTPP approach outlined above. By generalizing the setting, we weaken the assumption on the compression function attack, and hence allow the MITM framework to take advantage of new speed-up possibilities.

When the matching point is not at the end of the compression function, we can still make use of the multi-targets. Consider the sum of the size of W_p and W_q to be $2l$, and assume we can re-distribute the $2l$ bits to W_p and W_q freely[2]. Given 2^k targets, we can distribute the $2l$ bits to $l + k/2$ and $l - k/2$, so that we can have $2^{l+k/2}$ candidates for each direction (combining the $2^{l-k/2}$ and 2^k targets to get $2^{l+k/2}$ candidates). In this way, we can find a pseudo-preimage in $2^{n-l-k/2}$ and finding 2^k targets costs $\Sigma_{i=1}^{2^k} 2^{n-l} \cdot i^{-1/2} \simeq 2^{n-l+1+k/2}$ (see [16, Appendix B] for a proof). So we can find the preimage in

$$2^{n-k} + 2^{n-l+1+k/2} = 3 \cdot 2^{n-2l/3} \tag{2}$$

taking the optimal $k = 2l/3$. For this method to work, we will need more matching bits: $4l/3$ bits instead of l (we have $2^{4l/3}$ candidates for both directions). The memory requirement hence increases to $2^{4l/3}$. Here we trade memory for speed from $2^{n-l}/2^l$ (time/memory) to $2^{n-l-k/2}/2^{l+k/2}$ for $k = 0, \ldots, 2l/3$. And we have full control on any other speed/memory balance in-between by making use of the proper number of given targets, *i.e.,* less than 2^k.

[2] This being a very natural assumption is illustrated by the fact that for both MD4 and SHA-2 we can give a useful application that uses this.

Table 1. Comparison of methods converting pseudo-preimage to preimage

Name	Reference	Time	Memory	PM	Assumption
Traditional	Section 2, [33]	$2^{n-l/2+1}$	2^l	l	-
GMTPP	new, Section 2.2	$3 \cdot 2^{n-2l/3}$	$2^{4l/3}$	$4l/3$	redistribute neutral bits
MTPP	Section 2.1, [27]	$(l \ln(2) + 1) \cdot 2^{n-l}$	2^l	$2l$	Enhanced 3-SUM PM at feedforward
FPLP	new, Section 2.3	2^{n-l}	$\max(2^z, 2^l)$	l	2^n precomputation subset of chaining of size 2^z

2.3 Finding Preimages Using Large Precomputation (FPLP)

Here, we describe a simple technique to turn a large class of pseudo-preimage attacks into preimage attacks without any speed loss. The method requires an initial large precomputation of order 2^n and hence needs to be compared with the time/memory trade-off proposed by Hellman [18]. This means that the time and memory requirements of a dedicated attack need to be below the $TM^2 = N^2$ tradeoff curve in order to be considered as an improvement over the generic attack.

The approach may be described as follows: in the precomputation phase, one tries to find messages for all possible chaining outputs, *i.e.,* find m_i such that $hash(m_i) = h_T$ for (almost) all possible target hash values h_T, but only store those messages m_i in a table together with the output, which can actually "happen" in the pseudo-preimage attack. In the online phase, after the pseudo-preimage attack is carried out, a simple lookup into this memory is enough to find the right linking message. The memory requirement depends on the subset of all possible chaining inputs the pseudo-preimage attack can possibly result in. If this subset can be restricted enough, and the pseudo-preimage attack is fast enough, the approach may outperform the generic method. In Section 3.3, we give an actual example where this is the case for MD4, which seems to be the first result of this kind.

Four different conversion techniques are summarized in Table 1. Our point here is to illustrate and compare various approaches and the *assumptions* they make on the compression function attack. For simplicity, other conversion methods somewhat similar to MTPP (tree construction in [32], alternative tree and graph construction in [13]) are not listed. As an example, the new attack on the MD4 compression function satisfies only assumptions of the traditional and the FPLP approach, the new attack on the Tiger compression function and the SHA-2 compression function satisfy the assumption made by the GMTPP approach.

3 Improved Preimage Attack against MD4

3.1 Description of MD4

MD4 follows the traditional MD-strengthening, the original message is padded by 1, followed by many 0's and the 64-bit length information so that the length of

padded message becomes a multiple of 512. Then divide the padded message into blocks of 512 bits and feed into the compression function iteratively. Output of the final compression is the hash. The compression function follows the Davies-Meyer construction, and comes with two major parts: message scheduling and step function. Message scheduling divides the 512-bit message block into 16 words (32 bit each) and expands them into 48 using permutations, as shown in following table.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	4	8	12	1	5	9	13	2	6	10	14	3	7	11	15
0	8	4	12	2	10	6	14	1	9	5	13	3	11	7	15

Starting from input chaining, the expanded words are fed into the step function iteratively. The output of the last step is added with the input chaining to give the output of the compression function. The step function takes four registers as input, and update one as $Q_i = (Q_{i-4} + \mathsf{F}_i(Q_{i-1}, Q_{i-2}, Q_{i-3}) + M_{\pi(i)} + C_i) \lll r_i$ for $i = 0, \ldots, 47$, where C_i and r_i are predefined constants, π is a permutation defined in above table, and the functions F_i are defined as in the following table. We use typewriter font to denote the hex numbers, such as `5A827999`, $\mathbb{1}$ for `FFFFFFFF`, and $\mathbb{0}$ for `00000000`.

First pass	$0 \le i < 16$	$\mathsf{F}_i = \mathsf{IF}$	$C_i = K_0 = \mathbb{0}$
Second pass	$16 \le i < 32$	$\mathsf{F}_i = \mathsf{MAJ}$	$C_i = K_1 =$ `5A827999`
Third pass	$32 \le i < 48$	$\mathsf{F}_i = \mathsf{XOR}$	$C_i = K_2 =$ `6ED9EBA1`

3.2 Faster Pseudo Preimage Attack

In this section, we present a pseudo-preimage attack in 2^{72}. Separation of chunks is: steps $\{10, \ldots, 26\}$ for the initial structure, steps $\{40, \ldots, 47, 0, \ldots, 9\}$ for the first chunk, steps $\{27, \ldots, 36\}$ for the second chunk, steps $\{37, 38, 39\}$ for partial matching. We choose (M_9, Q_6) as W_p and (M_{14}, Q_{26}) as W_q. The initial structure covers 17 steps from Step 10 to Step 26, as shown in Fig 2 with $a = b = \mathbb{1}$. Note that every register and message words within the initial structure except $Q_6, M_{10}, M_{14}, M_9, Q_{26}$ are fixed to some random values. The concept of 4-cycle local-collision path has been used in [41,14,27]. However, none of those paths help in our MITM preimage attack, since we cannot find more proper choices of neutral words. In our initial structure, the relation between Q_6 and Q_{26} satisfies

$$Q_{26} - Q_6 = \varphi(M_9, M_{10}, M_{14}) \tag{3}$$

for some function φ. Note φ is fixed when all other registers/message words are fixed.

We fix all other registers in Fig 2 in such a way that the influence of the registers in the bold line is absorbed when passing through the F function (this is called *cross absorption property*). Details can be found in [16]. All required values are shown in Fig 2. However, this setting results in no solution, since it is over-constrained on M_{12} and M_{13}. To overcome this problem, we propose a *probabilistic initial structure*.

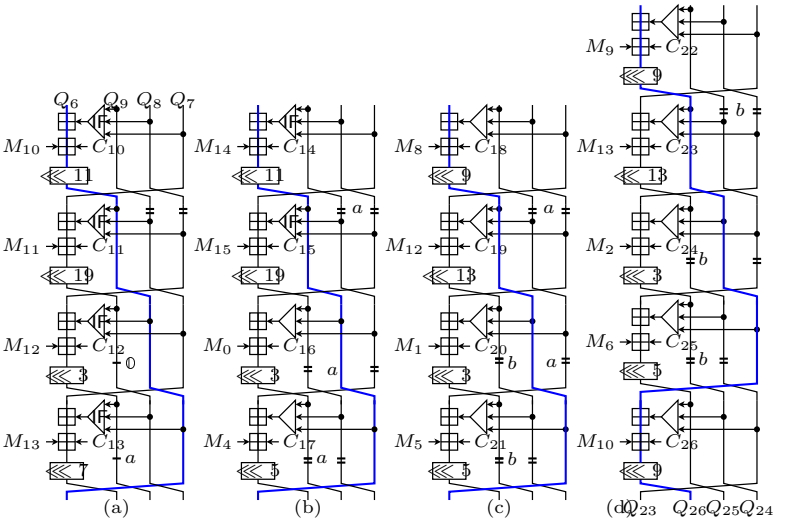


Fig. 2. 17-Step Probabilistic Initial Structure for MD4

Probabilistic Initial Structure. Consider the probability for $a = \mathsf{IF}(b, a, x)$, where a, b are fixed constants, and x is a random value in $\mathbb{F}_{2^{32}}$. The equation does not always hold for all x. However, if $|b|$ (Hamming weight) is very close to 32, then we can expect high probability for the equation to hold. Instead of setting inputs of IF to be strictly $\mathbb{1}$ or $\mathbb{0}$, we use some other values which are close to $\mathbb{1}$ or $\mathbb{0}$ (similarly, we force two inputs of MAJ to be very close), which enables us to find some solutions for the initial structure, as shown in Fig 2, where a, b are variables to be decided later.

We list the equations of the constraints here:

$$
\begin{array}{lrcr}
\text{Step 11:} & Q_9 & = Q_8 & \\
\text{Step 12:} & Q_{11} = \mathbb{0} & \Leftrightarrow & Q_7 + Q_8 + M_{11} = \mathbb{0} \\
\text{Step 13:} & Q_{12} = a & \Leftrightarrow & (Q_8 + Q_9 + M_{12}) \lll 3 = a \\
\text{Step 15:} & Q_{13} = Q_{12} = a & \Leftrightarrow & (Q_9 + M_{13}) \lll 7 = a \\
\text{Step 16:} & Q_{15} = Q_{13} = a & \Leftrightarrow & (a + M_{15}) \lll 19 = a \\
\text{Step 17:} & Q_{16} = Q_{15} = a & \Leftrightarrow & (a + a + M_0 + K_1) \lll 3 = a \\
\text{Step 18:} & Q_{17} = Q_{16} = a & \Leftrightarrow & (a + a + M_4 + K_1) \lll 5 = a \\
\text{Step 19:} & Q_{19} = b & \Leftrightarrow & (a + a + M_{12} + K_1) \lll 13 = b \\
\text{Step 20:} & Q_{20} = Q_{19} = b & \Leftrightarrow & (a + a + M_1 + K_1) \lll 3 = b \\
\text{Step 22:} & Q_{21} = Q_{20} = b & \Leftrightarrow & (a + b + M_5 + K_1) \lll 5 = b \\
\text{Step 24:} & Q_{23} = Q_{21} = b & \Leftrightarrow & (b + b + M_{13} + K_1) \lll 13 = b \\
\text{Step 25:} & Q_{24} = Q_{23} = b & \Leftrightarrow & (b + b + M_2 + K_1) \lll 3 = b
\end{array}
\tag{4}
$$

The above system of equations allows us to have choices for a and b. Note that we used two probabilistic approximations in two places, *i.e.,* $\mathsf{IF}(a, \mathbb{0}, Q_{10}) = \mathbb{0}$ at Step 13, and $\mathsf{MAJ}(b, Q_{18}, a) = a$ at Step 20. Each happens with probability $2^{|a|-32}$ and $2^{-|a \oplus b|}$, respectively (assume Q_{10} and Q_{18} are uniformly distributed).

To have high probability, we search the a, b which maximize $prob = 2^{|a|-|a\oplus b|-32}$. We found $a =$ `EFFFBFEF`, and $b =$ `EFCF1F6F`, which give $prob = 2^{-8}$. Solving (4) leaves $M_0 =$ `C37DFE86`, $M_1 =$ `C377EA76`, $M_2 =$ `C3D92B76`, $M_4 =$ `44FE0488`, $M_5 =$ `452D2004`, $M_{12} =$ `C0FD8501`, $M_{13} =$ `C15EC601`, $M_{15} =$ `07FE3E10`, $Q_8 = Q_9 =$ `1E81397E`, and $Q_7 + M_{11} =$ `E17EC682`. To ensure this works as expected, we verified the probability using a C program [15] , and the experiment confirms the result.

The pseudo-preimage algorithm

1. Fix all mentioned message words/registers as above.
2. Randomly assign all other message words, except M_9, M_{10} and M_{14}.
3. Compute (Q_7, Q_8, Q_9) and (Q_{23}, Q_{24}, Q_{25}).
4. For all (Q_{26}, M_{14}) compute forward from step 27 up to step 36, and obtain the list (L_q, Q_{26}, M_{14}) (expected size 2^{64}).
5. For all (Q_6, M_9), compute backward from step 9 up to step 0, and obtain the list (L_p^i, Q_6, M_9) (expected size 2^{64}).
6. Do feedforward and add the target, continue computing backwards up to step 40, and obtain the list (L_p, Q_6, M_9) (expected size 2^{64}).
7. Do partial matching with Q_{36} and Q_{39} ($2^{64+64-64} = 2^{64}$ pairs left), then match with Q_{38} ($2^{64-32} = 2^{32}$ pairs left).
8. For each pair left, compute the right M_{10}, such that Q_{37} is also matched (we have 2^{32} pairs (M_{14},Q_{26}, M_9, Q_6, M_{10}) fully matched).
9. Check if any pair left satisfies Eqn (3), if yes, output the pseudo-preimage; otherwise repeat the above process until a pseudo-preimage is found ($2^{32+8-32} = 2^8$ repetitions expected).

Clearly, the complexity is 2^{72} with memory requirement 2^{64}. There are some other additional properties. Note that given a new target, we can reuse the two lists L_p^i and L_q, so that the computation starts from Step 6 in the algorithm, which results in slightly faster pseudo-preimage in $2^{69.4}$. Furthermore, such an attack gives pseudo-preimage with chaining limited to the set L_p^i only.

3.3 Preimage Attack on the MD4 Hash Function

To find preimage using the pseudo-preimage attack above, we need to correct the padding. Note that M_{13} is precomputed, hence the length of last block is fixed, we need to fix the least significant 9 bits of M_{14} accordingly, *i.e.,* 447 (`1BF` in hex). Note that adding more blocks will only affect the length by a multiple of 512 (2^9). We leave the number of additional blocks for chance as done in the algorithm in Section 3.2. A small modification on the algorithm (computing 2^{55} candidates for each direction during each repetition, and $2^{128-55\times 2+8} = 2^{26}$ repetitions are needed, hence the size of L_p^i increases to $2^{55+26} = 2^{81}$) will result in pseudo-preimage in $2^{69.4+9} = 2^{78.4}$ with memory requirement 2^{55}. This can be further converted into a preimage in $2^{99.7}$ using the traditional conversion (link to input chaining of the last padded block), as the number of blocks can be resolved by expandable message (we compute a pseudo-preimage following the

padding rule in $2^{78.4}$, then apply the traditional conversion. Now, padding is no longer a problem when inverting the second last block etc.).

The resulting message of this attack has at least 2^{50} blocks, due to the fact that M_{15} is the most significant word of the length ($M_{15}||M_{14}$ denotes the length) and we have preset M_{15} to `07FE3E10`.

Precomputation. Similarly we can restrict the input chaining to a subset of size 2^{81}, by re-using the lists whenever looking for a new pseudo-preimage. So the pseudo-preimage can also be converted to preimage in $2^{78.4}$, when large precomputation is allowed. To achieve this, we precompute about 2^{128} different message blocks (prefixed by the expandable message) and store those with output in the restricted subset. This requires storage of order 2^{81} and precomputation effort 2^{128}. Given a target, we compute a pseudo-preimage (with padding done), and it can be converted to a preimage by looking up the stored chaining values. Hence this requires online computation $2^{78.4}$ only. Using a similar 2^{128} precomputation, the generic Hellman tradeoff would either require almost $2^{7.8}$ times more memory ($2^{88.8}$) to achieve the same runtime, or would lead to online computation that is almost $2^{15.6}$ times slower (2^{94}) if the same memory would be available.

3.4 Second-Preimage Attack on the MD4 Hash Function

In contrast to finding preimages, we can avoid the padding issues when finding second-preimages by finding pseudo-preimages for second last block etc., as done in [27]. Given 2^{128} precomputation, the complexity of this second-preimage attack is in $2^{69.4}$ with 2^{72} memory when $k \geq 2$, *i.e.,* it works for all messages with original length before padding at least 2 blocks (1024 bits, at least 3 blocks after padding). Similarly, it works in time $2^{99.7}$ and 2^{64} memory without precomputation. Although a faster second-preimage attack exists [46], the attack only works for very long messages, *i.e.,* at least 2^{56} blocks. For comparison, a second preimage can be found in 2^{n-k}, if the given message is of more than 2^k blocks, due to Kelsey and Schneier [20] (2^{64} for both time and memory if the optimal $k = 64$ can be achieved).

4 Preimage Attack against Tiger

Before presenting the result, we give some notations used in this Section. Let X^o and X^e denote the odd bytes and even bytes from register X, respectively. More generally, let us denote X^s so that those bits indexed by the set s are the same as in X and the rest are set to 0. To be consistent, we can define $e = \{0, \ldots, 7, 16, \ldots, 23, 32, \ldots, 39, 48, \ldots, 55\}$ and $o = \{8, \ldots, 15, 24, \ldots, 31, 40, \ldots, 47, 56, \ldots, 63\}$.

4.1 Description of Tiger

Tiger is an iterative hash function based on the MD structure. The message is padded followed by the 64-bit length of the original message so that the length of

the padded message becomes a multiple of 512. Then it is split into blocks of 512 bits and fed into the compression function iteratively. The compression of Tiger takes 3 chaining words and 8 message words (each word is of 64 bits) as input and produces the updated 3 chaining words as output. It consists of two parts: message expansion and step function. The input chaining is fed forward, together with output of last step function, to produce the output of the compression function, which is a variant of the Davies-Meyer construction. We introduce the step function and message expansion in details as follows.

Step Function. We name the three input chaining words of compression function as A, B and C. These three registers are updated as follows. $C \leftarrow C \oplus X$; $A \leftarrow A - \mathsf{even}(C)$; $B \leftarrow (B + \mathsf{odd}(C)) \times mul$. The result is then shifted around so that A, B, C become C, A, B. Here $+$, $-$, $\times$ are addition, subtraction and multiplication, in $\mathbb{Z}_{2^{64}}$, respectively. The two non-linear function even and odd are defined as follows.

$$\mathsf{even}(x) = T_1[x_B^0] \oplus T_2[x_B^2] \oplus T_3[x_B^4] \oplus T_4[x_B^6] \ ,$$
$$\mathsf{odd}(x) = T_4[x_B^1] \oplus T_3[x_B^3] \oplus T_2[x_B^5] \oplus T_1[x_B^7] \ ,$$

where $T_1, \ldots, T_4$ are four S-boxes defined on $\{0,1\}^8 \rightarrow \{0,1\}^{64}$, and x_B^i denotes the i-th least significant *Byte* of x, the details can be found in [5]. mul is 5, 7, 9 for the three passes, respectively.

Message Expansion. The 512-bit message block is split into 8 message words $X_0, \ldots, X_7$, each of 64 bits. The key scheduling function takes $X_0, \ldots, X_7$ as input and produces message words $\{X_8, \ldots, X_{15}\}$ and $\{X_{16}, \ldots, X_{23}\}$ recursively as follows. $(X_8, \ldots, X_{15}) = \mathsf{KSF}(X_0, \ldots, X_7)$, $(X_{16}, \ldots, X_{23}) = \mathsf{KSF}(X_8, \ldots, X_{15})$, where the key scheduling function KSF is defined as follows. We use $(X_8, \ldots, X_{15}) = \mathsf{KSF}(X_0, \ldots, X_7)$ as an example here.

First Step:

$$\begin{aligned}
Y_0 &= X_0 - (X_7 \oplus K_3) \\
Y_1 &= X_1 \oplus Y_0 \\
Y_2 &= X_2 + Y_1 \\
Y_3 &= X_3 - (Y_2 \oplus (\overline{Y}_1 \ll 19) \\
Y_4 &= X_4 \oplus Y_3 \\
Y_5 &= X_5 + Y_4 \\
Y_6 &= X_6 - (Y_5 \oplus (\overline{Y}_4 \gg 23)) \\
Y_7 &= X_7 \oplus Y_6
\end{aligned}$$

Second Step:

$$\begin{aligned}
X_8 &= Y_0 + Y_7 \\
X_9 &= Y_1 - (X_8 \oplus (\overline{Y}_7 \ll 19)) \\
X_{10} &= Y_2 \oplus X_9 \\
X_{11} &= Y_3 + X_{10} \\
X_{12} &= Y_4 - (X_{11} \oplus (\overline{X}_{10} \gg 23)) \\
X_{13} &= Y_5 \oplus X_{12} \\
X_{14} &= Y_6 + X_{13} \\
X_{15} &= Y_7 - (X_{14} \oplus K_4)
\end{aligned}$$

with $K_3 = \texttt{A5A5A5A5A5A5A5A5}$, $K_4 = \texttt{0123456789ABCDEF}$, and $\overline{Y}$ denotes bitwise complement of Y.

Attack Preview. The MITM preimage attack has been applied to Tiger, however for variants reduced to 16 and 23 steps [19,42], out of 24 in full Tiger. The

difficulty lies on finding good neutral words, longer initial structure and partial matching. In our attack, we find a 4-step initial structure, extend the partial matching to 5 steps and provide choice of neutral words achieving this. However each of them comes with constraints posed on message words/registers, due to the very complicated message scheduling used in Tiger. Throughout the description of the attack, we will explicitly give all those constraints, and explain how they can be fulfilled using the multi-word technique, *i.e.,* utilizing the degrees of freedom of most message words and registers to fulfill these constraints, which are usually left as random in the original MITM preimage attacks.

4.2 Precomputed Initial Structure

The original initial structure does not apply to Tiger, since the message words are xor-ed into the chaining, followed by addition/subtraction operations. One cannot swap the order of xor and addition/subtraction, unless the chaining values are within certain range so that we can either approximate xor by addition, or approximate addition by xor. We can either restrict one of the inputs to $\mathbb{0}$, or force the output to be $\mathbb{1}$, *e.g.,* $X \oplus \mathbb{0} = X + \mathbb{0}$, and $X \oplus Y = \mathbb{1}$ if and only if $X + Y = \mathbb{1}$. Under this restriction, we are able to have a 4-step initial structure as shown in Fig 3(a), which comes with the following three constraints.

Constraint 1. *Variables from X_i lie on the odd bytes only, so that (X_i^e) is fixed.*

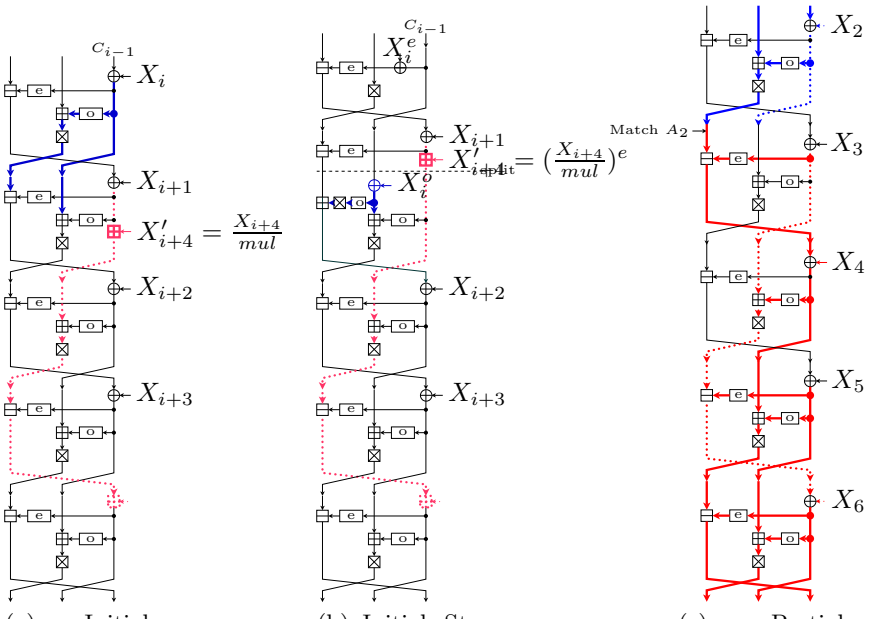
Constraint 2. *Assume we have control over X_{i+4} on those bits so that $(\frac{X_{i+4}}{mul})^o$ is fixed, and there is no carry from even bytes to odd bytes so that we can eventually move the X'_{i+4} further up above the* odd *function in step $i+1$. The idea is to keep the input to the* odd *function unchanged when we move the $(\frac{X_{i+4}}{mul})^e$ as shown in Fig 3(b).*

Constraint 3. *$C_{i+3} \oplus X_{i+4}$ should be 1 for those bits, where variables from X_{i+4} lie.*

After the precomputed initial structure (PIS) is formed, we essentially swap the order of X_i^e and $(\frac{X_{i+4}}{mul})^o$, which are 4 steps away from each other originally.

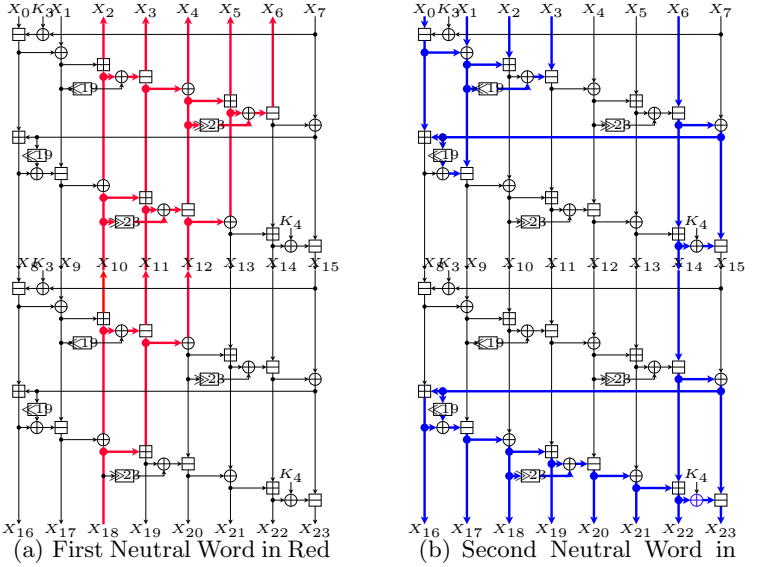
4.3 Message Compensation

The length of each independent chunk is at most 7 steps, due to the fact that any consecutive 8 message words can generate all other words (*i.e.,* related to all other words). Message compensation is used to achieve the maximum length (or close to maximum) for each chunk. Since we are able to have 4-step PIS, we would have $7 + 4 + 1 + 7 = 19$ steps for two chunks. Details are shown in Fig 4. Where $X_7, \ldots, X_{13}$ form the first chunk (7 steps), $X_{14}, \ldots, X_{18}$ may be dealt with using precomputed initial structure as shown above, and $X_{19}, \ldots, X_{23}$, X_0, X_1 are the second chunk (7 steps). In this way, we have 19-step extended chunks.



(a) Initial Structure (b) Initial Structure 2 (c) Partial Matching

Fig. 3. 4-Step Initial Structure and 5-step Partial Matching for Tiger



(a) First Neutral Word in Red (b) Second Neutral Word in Blue

Fig. 4. The neutral words with key scheduling function for Tiger

For the first chunk, we use a few bits of X_{18} as the neutral word (we will discuss which bits are to be used later). We force X_{18} to be the only one affected in the third pass (*i.e.,* $X_{16}, \ldots, X_{23}$). We come up with such a configuration following the rule that there are as few words affected in the current pass as possible. In summary, we have $\{X_2, \ldots, X_6, X_{10}, X_{11}, X_{12}, X_{18}\}$ affected as shown in Fig 4(a). Note this comes with

Constraint 4. *We use at most the least significant 23 bits of* X_{18} *so that these bits disappear when* $(X_{18} \gg 23)$ *is done (as shown in Fig. 4(a)), hence it does not affect* X_{20} *etc.*

For the second chunk, we use a few bits of X_{14} as the neutral word and avoid difference in X_7 in the first pass. Meanwhile, we avoid differences in $X_8, \ldots, X_{13}$ and X_{15} for the second pass. In the end, we have $\{X_0, \ldots, X_3, X_{14}, X_{16}, \ldots, X_{23}\}$ affected as shown in Fig 4(b). Note this comes with a constraint.

Constraint 5. X_{15} *remains constant.*

The two neutral words affect some common message words, *i.e.,* X_2, X_3, X_6 and X_{18}. We will need to choose the bits from two neutral words X_{14} and X_{18} properly, so that

Constraint 6. X_{14} *and* X_{18} *will not affect any common bits of any word simultaneously, i.e., for* X_2, X_3, X_6 *and* X_{18}.

We are left with the choices of the neutral bits for minimizing the attack complexity, which will be discussed later in Section 4.5.

4.4 Partial Matching and Partial Fixing

The direct partial matching works for 3 steps by computing backwards. Furthermore, by fixing the even bytes of the first message word (partial fixing technique) in forward direction, Isobe and Shibutani [19] are able to achieve 4-step, and 5-step by Wang and Sasaki [42]. In addition to the 4-step initial structure, we further post more conditions on message words in order to achieve 5-step partial matching (different from [42]), as shown in Fig 3(c), it covers step 2 to step 6.

Constraint 7. *The partial information below* X_3 *as in Fig 3(c) computed from* X_6 *should cover all even bytes so that we can compute the* even *function in step 3;*

Constraint 8. X_2^o *should be related to* X_{14} *only, so that we can compute the* odd *function at step 2 independently of* X_{18}.

To summarize, we are to use $\{X_7, \ldots, X_{13}\}$ as one chunk, $\{X_{19}, \ldots, X_{23}, X_1, X_2\}$ as the other chunk; precomputed initial structure covers steps using $\{X_{14}, \ldots, X_{18}\}$ ($i = 14$ for Section 4.2); and partial matching works for $\{X_2, \ldots, X_6\}$. Hence, the full Tiger of all 24 steps is covered.

4.5 Attack Description and Complexity Analysis

In this section, we show how to set the message words and registers for the PIS in order to have all constraints fulfilled. We also give algorithms with complexity evaluations, when necessary, to demonstrate how the attack works.

Fulfilling all Constraints. To have constraints about X_{18} fulfilled (*i.e.,* Constraints 2, 4, and 8), we choose neutral bits from $X_{18}^{s_b}$, where $s_b = \{0, \ldots, 7, 16, \ldots, 22\}$. Similarly, to have Constraint 1 on X_{14} fulfilled, we restrict the neutral bits from bytes 3, 5, 7 of X_{14}, *i.e.,* $X_{14}^{s_f}$ with $s_f = \{24, \ldots, 31, 40, \ldots, 47, 57, \ldots, 63\}$ (bit 56 is reserved for padding). Due to the fact that addition/subtraction will only propagate differences towards MSB, the least significant bits of $X_{14}^{s_f}$ that may affect on X_2, X_3, X_6, X_{18} are 43 (due to $\ll 19$), 62 (due to $\ll 19$ twice), 24, and 24, respectively. However, $X_{18}^{s_b}$ has very low chance ($\simeq 0$) of affecting up to bit 43 of X_2, bit 62 of X_3, bit 24 of X_{18}, and we will filter candidates so that the influence on X_6 is limited to up to bit 23. Hence, Constraint 6 can be fulfilled. To fulfill Constraint 5, we force $Y_6^{s_f} = X_{14}^{s_f}$ (through setting $X_{13}^{s_f} = \mathbb{0}$), and $X_7^{s_f} = K_4^{s_f}$. We leave Constraint 3 for PIS setup, and Constraint 7 for partial matching, to be addressed later.

Precomputed Initial Structure. For the precomputed initial structure to work, we have to preset several message words. Besides $X_{13}^{s_f} = 0$ and $X_7^{s_f} = K_4^{s_f}$, we still need to take care of the padding. We set $X_6^{56} = 1$, *i.e.,* the length of original message in last block is 447 ($7 \times 64 - 1$). Hence, we need to set $X_7^{\{0,\ldots,8\}} = 447$. Note that adding more blocks will affect the length by a multiple of 2^9, which has no effect on the 9 LSBs of X_7. To reduce the influence of $X_{14}^{s_f}$ on X_6, we further set $(\overline{Y}_4 \gg 23 \oplus Y_5)^{s_f} = \mathbb{0}$, so that only $X_6^{s_f}$ out of X_6 will be affected. Note the PIS can be done in 2^{15} evaluations of key scheduling (leaving restriction on $X_{14}^{s_f}$ for probability only). This is negligible since we can reuse the PIS for at least 2^{16} times, to be discussed later.

Finding good candidates - Backward. We use bits from $X_{18}^{s_b}$ to compute the good candidates for backward direction. Constraint 2 further restricts us to choose values such that $X_{18}^{\{0,\ldots,7\}}$ and $X_{18}^{\{16,\ldots,23\}}$ are multiple of 9 ($mul = 9$ for third pass). Hence, we can have $\lceil 2^8/9 \rceil \times \lceil 2^7/9 \rceil = 2^{8.8}$ good candidates. Finally, we filter out candidates which do not fulfill Constraint 6. Experiments show that the remaining good candidates are about 2^8. Note these good candidates need to be computed under the constrainted PIS, we use message modification techniques to fulfill the constraints for PIS, and to get the 2^8 good candidates in less than 2^{19} key scheduling evaluations. Details can be found in [15].

Finding good candidates - Forward. We use bits from $X_{14}^{s_f}$ to compute the good candidates for backward direction. To have Constraint 3 fulfilled, we need to filter the candidates, such that it gives $\mathbb{1}$ for $C_{i+3}^{s_b}$ as in Fig 3(b), this reduces the number of candidates to $2^{23-15} = 2^8$. Note that this part can be re-used for many different (at least 2^{16}) C_i, by changing the even bytes, which we can freely set at the very beginning of the MITM preimage attack. Hence, the time complexity for this part is also negligible.

Probabilistic Partial Matching. Partial matching matches A_2 from both sides, where we can compute A_2 in the forward direction without any problem. However, in the backward direction, we only know information of bytes 0, 1, 2, 4, 6 of X_6 (red), as to compute B_3^e. Note that $B_3 = (B_6 \oplus X_6 + \mathsf{even}(B_6))/5 - \mathsf{odd}(B_5)$ ($mul = 5$ for first pass), where B_5 and B_6 are known. We rewrite it to $B_3 = (B_6 \oplus X_6)/5 + K_5$, where $K_5 = \mathsf{even}(B_5)/5 - \mathsf{odd}(B_4)$. We can compute bytes 0, 1, 2 of B_3, yet we still need bytes 4, 6 from information of bytes 4, 6 of X_6 only. Note that $B_3^{\{32,\ldots,39\}} = (B_6^{\{32,\ldots,39\}} \oplus X_6^{\{32,\ldots,39\}} - \mathsf{Bo} \times 2^{32})/5 + K_5^{\{32,\ldots,39\}} + \mathsf{Ca} \times 2^{32}$, where $\mathsf{Bo} \in \{0, \ldots, 4\}$ denote borrow from bit 31 when '/5' is carried out, and $\mathsf{Ca} \in \{0, 1\}$ denote the carry for the '+' from bit 31. We deal with the Bo by computing all possible choices, and guess the $\mathsf{Ca} = K_5^{31}$ which results in a probability 3/4 for the Ca to be correct. This gives an example for byte 4, and we can deal with byte 6 similarly. The process results in 25 times more computations for partial matching, together with probability 9/16. However, we shall only need to repeat the even and the '$-$' at Step 3, so that the essential repetition is equivalent to less than 2^{-1} compression computations per candidate.

Complexity of Finding a (Second) Preimage. Following the MITM preimage attack framework, the pseudo-preimage attack works as follows.

1. Randomly choose A_{14}, B_{14}, C_{14}.
2. Compute precomputed initial structure.
3. Compute candidates in backward and forward directions.
4. Repeat for 2^{16} values of C_{14} by looping all values in byte 4 and 6 (this step is to make time complexity for first three steps negligible):
   (a) For each candidate for backward and forward directions, compute A_2 independently.
   (b) Carry out probabilistic partial matching. If a full match on A_2 is found, further re-check if the "guess" is correct.
5. Repeat 1-4 until a pseudo-preimage is found.

The pseudo-preimage attack works in time $2^{185.4}$ ($2^{192-8} \times 1.5 \times (3/4)^{-2}$), which can be reduced to $2^{182.4}$ when more than 2^4 targets are available (by using targets as part of backward candidates as in GMTPP). The pseudo-preimage can be converted to preimage attack with time complexity $2^{189.7}$ using the traditional conversion, with memory requirement of order 2^8. Following the GMTPP framework, the time complexity can be further reduced to $2^{188.8}$ (by computing 24 pseudo preimages and $2^{192}/24$ linking messages), with the same memory requirement. Similarly, the second-preimage attack works in $2^{188.2}$, when the given message is of more than 2^4 blocks.

5 Concluding Discussion

We conclude with a discussion of results and some open problems that are independent of particular hash functions. In this paper we have extended the framework around meet-in-the-middle attacks that is currently being developed by the community with a number of general approaches. We illustrated those extensions

with improved preimage attacks on various time-tested hash functions, with the first cryptanalytic attack on the full Tiger hash function probably being the most interesting example. Other examples include various improved preimage attacks on MD4 and step-reduced SHA-2.

One of the generic ideas presented was the following. Under the meet-in-the-middle preimage attack framework, we presented new techniques to convert pseudo-preimage into preimage faster than the traditional method, *i.e.,* the Generic Multi-Target Pseudo Preimage and a simple precomputation technique. It will be interesting to see if an algorithm solving the Enhanced 3-Sum Problem faster than 2^{2n} for a set size of 2^n exists, so that the MTPP can be valid for any l. On the other hand, we found pseudo-preimage for MD4 in 2^{72}, it will be interesting to see if any of the new conversion techniques or other unknown techniques works when converting pseudo-preimage to preimage for MD4.

We expect the techniques outlined in this paper to also improve existing preimage attacks on well studied hash functions like MD5, SHA-1, HAVAL, and others. Also, the narrow-pipe SHA-3 candidates seem to be natural targets.

Acknowledgements. We would like to thank Kazumaro Aoki, Gaëtan Leurent, Yu Sasaki, Kyoji Shibutani, Lei Wang, and the anonymous reviewers of Eurocrypt 2010 and Asiacrypt 2010 for interesting discussions. Part of this work was done while the third author was with Graz University of Technology, and the first author was visiting. The work in this paper is supported in part by the Singapore National Research Foundation under Research Grant NRF-CRP2-2007-03, the Singapore Ministry of Education under Research Grant T206B2204, by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy), and by the European Commission under contract ICT-2007-216646 (ECRYPT II).

References

1. Multisource File Transfer Protocol, `http://en.wikipedia.org/wiki/Multisource_File_Transfer_Protocol`
2. Rsync, `http://rsync.samba.org/`
3. TigerTree Hash Code, `http://tigertree.sourceforge.net/`
4. 3-Sum Problem, `http://en.wikipedia.org/wiki/3SUM`
5. Anderson, R.J., Biham, E.: TIGER: A Fast New Hash Function. In: Gollmann, D. (ed.) FSE 1996. LNCS, vol. 1039, pp. 89–97. Springer, Heidelberg (1996)
6. 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)
7. Aoki, K., Sasaki, Y.: Meet-in-the-Middle Preimage Attacks Against Reduced SHA-0 and SHA-1. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 70–89. Springer, Heidelberg (2009)
8. Aoki, K., Sasaki, Y.: Preimage Attacks on One-Block MD4, 63-Step MD5 and More. In: Avanzi, R., Keliher, L., Sica, F. (eds.) SAC 2008. LNCS, vol. 5381, pp. 103–119. Springer, Heidelberg (2009)
9. Baran, I., Demaine, E.D., Patrascu, M.: Subquadratic algorithms for 3SUM. Algorithmica 50(4), 584–596 (2008)

10. Barkan, E., Biham, E., Shamir, A.: Rigorous Bounds on Cryptanalytic Time/Memory Tradeoffs. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 1–21. Springer, Heidelberg (2006)
11. Biham, E.: New Techniques for Cryptanalysis of Hash Functions and Improved Attacks on Snefru. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 444–461. Springer, Heidelberg (2008)
12. 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)
13. De Cannière, C., Rechberger, C.: Preimages for Reduced SHA-0 and SHA-1. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 179–202. Springer, Heidelberg (2008)
14. Dobbertin, H.: The First Two Rounds of MD4 are Not One-Way. In: Vaudenay, S. (ed.) FSE 1998. LNCS, vol. 1372, pp. 284–292. Springer, Heidelberg (1998)
15. Guo, J.: The C Program Verifies the Preimage Attacks against Tiger and MD4 (2010), `http://www.jguo.org/docs/Tiger-MD4-AC10.zip`
16. Guo, J., Ling, S., Rechberger, C., Wang, H.: Advanced Meet-in-the-Middle Preimage Attacks: First Results on Full Tiger, and Improved Results on MD4 and SHA-2. Cryptology ePrint Archive, Report 2010/016 (2010), `http://eprint.iacr.org/2010/016.pdf`
17. Haller, N.: RFC1760 - The S/KEY One-Time Password System (1995)
18. Hellman, M.E.: A Cryptanalytic Time - Memory Trade-Off. IEEE Transactions on Information Theory 26(4), 401–406 (1980)
19. Isobe, T., Shibutani, K.: Preimage Attacks on Reduced Tiger and SHA-2. In: Dunkelman, O. (ed.) Fast Software Encryption. LNCS, vol. 5665, pp. 139–155. Springer, Heidelberg (2009)
20. Kelsey, J., Schneier, B.: Second Preimage on n-bit hash functions for much less than 2^n work. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 474–490. Springer, Heidelberg (2005)
21. Khovratovich, D., Nikolic, I., Weinmann, R.-P.: Meet-in-the-Middle Attacks on SHA-3 Candidates. In: Dunkelman, O. (ed.) Fast Software Encryption. LNCS, vol. 5665, pp. 228–245. Springer, Heidelberg (2009)
22. Knudsen, L.R., Mathiassen, J.E., Muller, F., Thomsen, S.S.: Cryptanalysis of MD2. Journal of Cryptology 23(1), 72–90 (2010)
23. Lai, X., Massey, J.L.: Hash Function Based on Block Ciphers. In: Rueppel, R.A. (ed.) EUROCRYPT 1992. LNCS, vol. 658, pp. 55–70. Springer, Heidelberg (1993)
24. Lamberger, M., Pramstaller, N., Rechberger, C., Rijmen, V.: Second Preimages for SMASH. In: Abe, M. (ed.) CT-RSA 2007. LNCS, vol. 4377, pp. 101–111. Springer, Heidelberg (2006)
25. Lamberger, M., Pramstaller, N., Rechberger, C., Rijmen, V.: Analysis of the Hash Function Design Strategy Called SMASH. IEEE Transactions on Information Theory 54(8), 3647–3655 (2008)
26. Leurent, G.: Message Freedom in MD4 and MD5 Collisions: Application to APOP. In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 309–328. Springer, Heidelberg (2007)
27. Leurent, G.: MD4 is Not One-Way. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 412–428. Springer, Heidelberg (2008)
28. Mendel, F., Pramstaller, N., Rechberger, C.: A (Second) Preimage Attack on the GOST Hash Function. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 224–234. Springer, Heidelberg (2008)

29. Mendel, F., Pramstaller, N., Rechberger, C., Kontak, M., Szmidt, J.: Cryptanalysis of the GOST Hash Function. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 162–178. Springer, Heidelberg (2008)
30. Mendel, F., Preneel, B., Rijmen, V., Yoshida, H., Watanabe, D.: Update on Tiger. In: Barua, R., Lange, T. (eds.) INDOCRYPT 2006. LNCS, vol. 4329, pp. 63–79. Springer, Heidelberg (2006)
31. Mendel, F., Rijmen, V.: Cryptanalysis of the Tiger Hash Function. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 536–550. Springer, Heidelberg (2007)
32. Mendel, F., Rijmen, V.: Weaknesses in the HAS-V Compression Function. In: Nam, K.-H., Rhee, G. (eds.) ICISC 2007. LNCS, vol. 4817, pp. 335–345. Springer, Heidelberg (2007)
33. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press, Boca Raton (1996)
34. Muller, F.: The MD2 Hash Function Is Not One-Way. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 214–229. Springer, Heidelberg (2004)
35. Naito, Y., Sasaki, Y., Kunihiro, N., Ohta, K.: Improved Collision Attack on MD4 with Probability Almost 1. In: Won, D.H., Kim, S. (eds.) ICISC 2005. LNCS, vol. 3935, pp. 129–145. Springer, Heidelberg (2006)
36. Sasaki, Y., Aoki, K.: Preimage attacks on 3, 4, and 5-pass HAVAL. In: Pieprzyk, J.P. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 253–271. Springer, Heidelberg (2008)
37. Sasaki, Y., Aoki, K.: Finding Preimages in Full MD5 Faster than Exhaustive Search. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 134–152. Springer, Heidelberg (2010)
38. Sasaki, Y., Wang, L., Ohta, K., Kunihiro, N.: Security of MD5 Challenge and Response: Extension of APOP Password Recovery Attack. In: Malkin, T. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 1–18. Springer, Heidelberg (2008)
39. Stevens, M., Lenstra, A.K., de Weger, B.: Chosen-Prefix Collisions for MD5 and Colliding X.509 Certificates for Different Identities. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 1–22. Springer, Heidelberg (2007)
40. 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)
41. Vaudenay, S.: On the Need for Multipermutations: Cryptanalysis of MD4 and SAFER. In: Preneel, B. (ed.) FSE 1994. LNCS, vol. 1008, pp. 286–297. Springer, Heidelberg (1995)
42. Wang, L., Sasaki, Y.: Finding Preimages of Tiger Up to 23 Steps. In: Hong, S., Iwata, T. (eds.) FSE 2010. LNCS, vol. 6147, pp. 116–133. Springer, Heidelberg (2010)
43. Wang, X., Lai, X., Feng, D., Chen, H., Yu, X.: Cryptanalysis of the Hash Functions MD4 and RIPEMD. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 1–18. Springer, Heidelberg (2005)
44. 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)
45. 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)
46. Yu, H., Wang, G., Zhang, G., Wang, X.: The Second-Preimage Attack on MD4. In: Desmedt, Y., Wang, H., Mu, Y., Li, Y. (eds.) CANS 2005. LNCS, vol. 3810, pp. 1–12. Springer, Heidelberg (2005)

Collision Attacks against the Knudsen-Preneel Compression Functions*

Onur Özen** and Martijn Stam

LACAL, EPFL
Station 14, CH-1015 Lausanne, Switzerland
{onur.ozen,martijn.stam}@epfl.ch

Abstract. Knudsen and Preneel (Asiacrypt'96 and Crypto'97) introduced a hash function design in which a linear error-correcting code is used to build a wide-pipe compression function from underlying blockciphers operating in Davies-Meyer mode. Their main design goal was to deliver compression functions with collision resistance up to, and even beyond, the block size of the underlying blockciphers. In this paper, we present new collision-finding attacks against these compression functions using the ideas of an unpublished work of Watanabe and the preimage attack of Özen, Shrimpton, and Stam (FSE'10). In brief, our best attack has a time complexity strictly smaller than the block-size for all but two of the parameter sets. Consequently, the time complexity lower bound proven by Knudsen and Preneel is incorrect and the compression functions do not achieve the security level they were designed for.

Keywords: Collision attack, coding theory, compression function.

1 Introduction

Hash functions are currently at the centre of the cryptographic community's attention. While most of this attention is geared directly towards the SHA-3 competition (by analysing its remaining candidates), other, arguably more fundamental questions regarding hash function design should not be forgotten. After all, the study of the underlying principles of hash function design are potentially beneficial for the SHA-3 decision process.

The two most revered principles in hash function design are (i) the Merkle-Damgård iterative construction, or more generally the principle of designing a secure *compression* function and (ii) the Davies-Meyer construction, or more generally the principle of using a *blockcipher* as underlying primitive. Indeed, the currently standardized hash functions from the SHA family follow this approach (as did their predecessor MD5) as well as several of the SHA-3 candidates.

It was already recognized early on that the output sizes of traditional blockciphers are insufficient to yield a secure compression function [16]. This still

* This work has been supported in part by the European Commission through the ICT programme under contract ICT-2007-216676 ECRYPT II.
** Supported by a grant of the Swiss National Science Foundation, 200021-122162.

M. Abe (Ed.): ASIACRYPT 2010, LNCS 6477, pp. 76–93, 2010.

holds true today: for all the (optimally secure) PGV blockcipher-based compression functions [10,1,12] based on an n-bit blockcipher, the (time) complexity of collision- and preimage-finding attacks is at most $2^{n/2}$, resp. 2^n; when $n = 128$ (e.g. AES) the resulting bounds are mostly unacceptable for current practice (in particular for collision resistance).

To achieve acceptable security (based on small block sizes) it is necessary to output a multiple of the block-length. In the 1990s many constructions were proposed for this goal, mostly outputting $2n$ bits with the explicit collision resistance target of 2^n (see [3,9] for an overview). The standard goal for these constructions has been optimal collision-resistance: a target output size is fixed and the compression function should be collision resistant up to the birthday bound for that digest size. In three papers [4,5,6], Knudsen and Preneel adopted a different approach, namely to fix a particular security target and let the output size (and relatedly the number of blockcipher calls) vary as needed in order to guarantee a particular security target *without* imposing optimal security.

Specifically, given r independent ideal compression functions $f_1, \ldots, f_r$, each mapping cn bits to n bits, they create a new 'bigger' compression function outputting rn bits. Following principles (i) and (ii) already mentioned, they then propose to instantiate the underlying ideal compression functions with a blockcipher run in Davies-Meyer mode and to iterate the compression function to obtain a full blockcipher-based hash function. However, they derive their security from the compression function, so that is where we will focus our attention.

The $f_1, \ldots, f_r$ are run in parallel where each of their inputs is some linear combination of the blocks of message and chaining variable that are to be processed; the rn-bit output of their construction is the concatenation of the outputs of these parallel calls. The elegance of the KP construction is in *how* the inputs to $f_1, \ldots, f_r$ are computed. They use the generator matrix of an $[r, k, d]$ error-correcting code over $\mathbb{F}_{2^c}$ to determine how the ck input blocks of the 'big' compression function are xor'ed together to form the inputs to the underlying r functions. (In a generalization they consider the f_i as mapping from bcn' to bn' bits instead and use a code over $\mathbb{F}_{2^{bc}}$.)

The (deliberate) effect of this design is that when two inputs to the 'big' compression function differ, the corresponding inputs for the underlying functions will differ for at least d functions. In particular, when using a systematic generator, a change in the systematic part of the input results in at least $d - 1$ so-called *active* functions in the non-systematic part. Intuitively this means that one has to find a preimage, resp. a collision for the $d - 1$ active functions in parallel. Based on this observation, Knudsen and Preneel claim that (under an assumption) any collision attack needs time at least $2^{(d-1)n/2}$ (and as many f_i evaluations) and they conjecture that a preimage attack will require time at least $2^{(d-1)n}$. Additionally, they give preimage and collision attacks (sometimes matching their lower bounds).

Watanabe [14] already pointed out a collision attack beating the one given by Knudsen and Preneel for many of the parameter sets. In particular, his differential attack works whenever $r < 2k$ and has a query and time complexity of

Table 1. A summary of collision attacks on the Knudsen-Preneel compression functions, with constant and polynomial factors (in n) ignored. Non-MDS parameters are in italic, for $e \in \{2, 4\}$ the underlying primitive is $f_i : \{0,1\}^{2n} \to \{0,1\}^n$, and for $e = 3$ it is $f_i : \{0,1\}^{3n} \to \{0,1\}^n$.

Code	Algorithm 4, [8] Complexity Query	Algorithm 4, [8] Complexity Time	Algorithm 2 Complexity Time/Query	ÖSS Complexity Query	ÖSS Complexity Time	Knudsen-Preneel Lower Bound	Knudsen-Preneel Attack Time
$[r,k,d]_{2^e}$	$2^{kn/(3k-r)}$	Thm. 4, [8]	$2^{dn/(d+1)}$	$2^{rn/(2k)}$	[8]	$2^{(d-1)n/2}$	[6]
$[5,3,3]_4$	$2^{3n/4}$	$2^{3n/4}$	$2^{3n/4}$	$2^{5n/6}$	$2^{4n/3}$	2^n	$2^{4n/3}$
$[8,5,3]_4$	$2^{5n/7}$	$2^{5n/7}$	$2^{3n/4}$	$2^{4n/5}$	$2^{7n/5}$	2^n	$2^{3n/2}$
$[12,9,3]_4$	$2^{3n/5}$	$2^{3n/5}$	$2^{3n/4}$	$2^{2n/3}$	$2^{4n/3}$	2^n	$2^{3n/2}$
$[9,5,4]_4$	$2^{5n/6}$	$2^{5n/6}$	$2^{4n/5}$	$2^{9n/10}$	$2^{11n/5}$	$2^{3n/2}$	2^{2n}
$[16,12,4]_4$	$2^{3n/5}$	$2^{4n/5}$	$2^{4n/5}$	$2^{2n/3}$	$2^{7n/3}$	$2^{3n/2}$	2^{2n}
$[6,4,3]_{16}$	$2^{2n/3}$	$2^{2n/3}$	$2^{3n/4}$	$2^{3n/4}$	$2^{3n/2}$	2^n	$2^{5n/4}$
$[8,6,3]_{16}$	$2^{3n/5}$	$2^{3n/5}$	$2^{3n/4}$	$2^{2n/3}$	$2^{4n/3}$	2^n	$2^{7n/6}$
$[12,10,3]_{16}$	$2^{5n/9}$	$2^{5n/9}$	$2^{3n/4}$	$2^{3n/5}$	$2^{6n/5}$	2^n	$2^{11n/10}$
$[9,6,4]_{16}$	$2^{2n/3}$	$2^{2n/3}$	$2^{4n/5}$	$2^{3n/4}$	2^{2n}	$2^{3n/2}$	$2^{3n/2}$
$[16,13,4]_{16}$	$2^{13n/23}$	$2^{20n/23}$	$2^{4n/5}$	$2^{8n/13}$	2^{2n}	$2^{3n/2}$	$2^{3n/2}$
$[4,2,3]_8$	2^n	2^n	$\times$	2^n	2^{2n}	2^n	$2^{3n/2}$
$[6,4,3]_8$	$2^{2n/3}$	$2^{2n/3}$	$2^{3n/4}$	$2^{3n/4}$	$2^{3n/2}$	2^n	$2^{5n/4}$
$[9,7,3]_8$	$2^{7n/12}$	$2^{7n/12}$	$2^{3n/4}$	$2^{9n/14}$	$2^{8n/7}$	2^n	$2^{8n/7}$
$[5,2,4]_8$	$\times$	$\times$	$\times$	$2^{5n/4}$	2^{3n}	$2^{3n/2}$	$2^{7n/4}$
$[7,4,4]_8$	$2^{4n/5}$	$2^{4n/5}$	$2^{4n/5}$	$2^{7n/8}$	$2^{9n/4}$	$2^{3n/2}$	$2^{3n/2}$
$[10,7,4]_8$	$2^{7n/11}$	$2^{9n/11}$	$2^{4n/5}$	$2^{5n/7}$	2^{2n}	$2^{3n/2}$	$2^{3n/2}$

essentially $k2^n$. Thus he demonstrated that the *proven* collision resistance *lower* bound given by Knudsen and Preneel is incorrect whenever $r < 2k$ and $d > 3$. For a code with minimum distance $d = 3$ he matches the Knudsen-Preneel 2^n collision-resistance lower bound, but does not violate it; the two codes proposed by Knudsen and Preneel with $r \geq 2k$ (namely $[4,2,3]_8$ and $[5,2,4]_8$) seem beyond reproach. Yet this was the first indication that something is amiss with the claim by Knudsen and Preneel. A second indication arrived at FSE'10, when Özen, Shrimpton, and Stam [7] demonstrated a remarkably efficient preimage attack that, for 9 out of 16 cases, runs in time $2^{rn/k}$ which was shown optimal. Moreover, using a yield-based argument, they showed that an information-theoretic adversary in principle should be able to find collisions after only $2^{rn/(2k)}$ queries.

Our contribution. In this paper we deal what we believe will be the final blow against the Knudsen-Preneel compression functions. Our contribution is fourfold, with a summary provided in Table 1. For completeness, we have also investigated the time complexity that one would obtain by straightforward adaptation of the ideas and query complexities of ÖSS; we refer to the full version of this paper [8] for the details.

The *mise en place* in Section 4 provides a detailed mathematical characterization of the Knudsen-Preneel compression function's preprocessing. As a first

simple result, this allows us in Section 5.1 to revise the attack of Watanabe in a way that slightly reduces the time requirements, yet significantly increases the number of collisions it can produce. More precisely, after an initial effort of $d2^n$, we can generate (up to) $2^{(k-d)n}$ collisions in constant time (for $k > d$).

In our revised version of Watanabe's attack, we fix a pure tensor to create a differential. By adaptively looking for an arbitrary tensor and using the same type of queries as Özen, Shrimpton, and Stam we arrive in Section 5.2 at a new, symbiotic collision-finding attack with time complexity $d2^{dn/(d+1)}$. The attack works whenever $d \leq k$ (as in Watanabe's case). Even more amazing is that if the inequality is strict, that is if $d < k$, the adversary can create further collisions (like our revised attack) in *constant* time (up to $2^{(k-d)n}$ collisions).

Thirdly, in Section 6.1 we introduce a parametrized information-theoretic collision attack. It turns out that the new symbiotic attack and the old ÖSS information-theoretic collision attack are both on opposite ends of the spectrum of this parametrized attack, yet optimality is typically achieved somewhere in the middle—with $\mathsf{KP}([4,2,3]_8)$ and $\mathsf{KP}([5,2,4]_8)$ again as exceptions—yielding query complexity $2^{kn/(3k-r)}$.

Our final contribution is a reduced-time variant of our optimized attack above. For this we use the same ideas as ÖSS, but with a crucial twist: where they used the dual code to look for preimages efficiently, we will use the dual *shortened* code to search for collisions efficiently. As a result, for 12 out of 16 suggested parameters we can mount a collision attack whose time complexity matches its query complexity (ignoring constants and logarithmic factors). Even better, only for $\mathsf{KP}([5,2,4]_8)$ we are unable to beat the time-complexity of any prior attack we are aware of, for the rest we set new records.

2 Preliminaries

With some minor modifications, we will adhere to the notation also used by Özen, Shrimpton, and Stam.

Linear error correcting codes. An $[r,k,d]_{2^e}$ linear error correcting code $\mathcal{C}$ is the set of elements (codewords) in a k-dimensional subspace of $\mathbb{F}_{2^e}^r$ (for $r \geq k$), where the minimum distance d is defined as the minimum Hamming weight (taken over all nonzero codewords in $\mathcal{C}$). The dual code $[r, r-k, d^\perp]_{2^e}$ is the set of all elements in the $r-k$-dimensional subspace orthogonal to $\mathcal{C}$ (with respect to the usual inner product), and its minimum distance is denoted $d^\perp$. The Singleton bound puts a limit on the minimum distance: $d \leq r-k+1$. Codes matching the Singleton bound are called maximum distance separable (MDS). An important property of an MDS code is that its duals is MDS as well, so $d^\perp = k+1$.

An $[r,k,d]_{2^e}$ code $\mathcal{C}$ can be generated by a matrix $G \in \mathbb{F}_{2^e}^{k\times r}$, meaning that $\mathcal{C} = \{x \cdot G | x \in \mathbb{F}_{2^e}^k\}$ (using row vectors throughout). A generator matrix G is called systematic iff it has the form $G = [I_k|P]$ for $P \in \mathbb{F}_{2^e}^{k\times(r-k)}$ and I_k the identity matrix in $\mathbb{F}_{2^e}^{k\times k}$. Furthermore, G is the generator matrix of an MDS code iff any k columns are linearly independent. For an index set $\mathcal{I} \subseteq \{1,\ldots,r\}$ we define $G_\mathcal{I} \in \mathbb{F}_{2^e}^{k\times|\mathcal{I}|}$ as the restriction of G to those columns indexed by $\mathcal{I}$.

For a code and any index set $\mathcal{I} \subseteq \{1, \ldots, r\}$, we want to define $\tilde{\mathcal{I}} \subset \{1, \ldots, r\}$ such that $G_{\tilde{\mathcal{I}}}$ is invertible (thus in particular $|\tilde{\mathcal{I}}| = k$) and $\tilde{\mathcal{I}} \subseteq \mathcal{I}$ or $\mathcal{I} \subseteq \tilde{\mathcal{I}}$. For MDS codes, the existence of such an $\tilde{\mathcal{I}}$ can be shown easily (and we can impose uniqueness e.g. by virtue of an ordering). For non-MDS codes there exist some $\mathcal{I}$ for which such an $\tilde{\mathcal{I}}$ does not exist (for example the $\mathcal{I} \subset \{1, \ldots, r\}$ for which $|\mathcal{I}| = k$ but $G_{\mathcal{I}}$ is not invertible), however for any target cardinality it is possible to find an $\mathcal{I}$ (of that cardinality) that does have an $\tilde{\mathcal{I}}$ (e.g. by first going through the systematic columns); we call such an $\mathcal{I}$ *admissible*.

A given $[r, k, d]_{2^e}$ code $\mathcal{C}$ can be *shortened* to obtain a new, derived code $\mathcal{C}'$. Let $i \in \{1, \ldots, r\}$, then consider the set of all codewords in $\mathcal{C}$ that are 0 on position i. The new code $\mathcal{C}'$ consists of these codewords with position i dropped, however we sometimes 'quasi-shorten' and keep the superfluous zeroes present (we always keep the original indexing). It is easy to see that $\mathcal{C}'$ is an $[r-1, k-1, d]_{2^e}$ code unless all codewords in $\mathcal{C}$ had a 0 on position i or $k = 1$ (in the latter case the shortening might result in the trivial one-codeword code $\{0^{r-1}\}$). The shortening of an MDS code is an MDS code itself. By repeated application one can shorten by any index set $\mathcal{I}_0 \subset \{1, \ldots, r\}$ for which $\theta = |\mathcal{I}_0| < k$ to obtain a derived $[r - \theta, k - \theta, d]$ MDS code $\mathcal{C}'$. If G is systematic and $\mathcal{I}_0 = \{1, \ldots, \theta\}$ we can generate the shortened code by dropping the first θ rows of $G_{\mathcal{I}}$, where $\mathcal{I} = \{1, \ldots, r\} \backslash \mathcal{I}_0 = \{\theta + 1, \ldots, r\}$. (For the four non-MDS codes used by Knudsen and Preneel we will perform a separate analysis on repeated shortening.)

Blockwise-linear compression functions. A *compression function* is a mapping $H : \{0,1\}^{tn} \to \{0,1\}^{sn}$ for some blocksize $n > 0$ and integer parameters $t > s > 0$. For positive integers c and n, we let $\mathrm{Func}(cn, n)$ denote the set of all functions mapping $\{0,1\}^{cn}$ into $\{0,1\}^n$. A compression function is *Public Random Function (PuRF)-based* if its mapping is computed by a program with oracle access to a finite number of specified oracles $f_1, \ldots, f_r$, where $f_1, \ldots, f_r \xleftarrow{\$} \mathrm{Func}(cn, n)$. When a PuRF-based compression function operates on input W, we write $H^{f_1,\ldots,f_r}(W)$ for the resulting value. Of primary interest for us will be *single-layer* PuRF-based compression functions without feedforward. These call all oracles in parallel and compute the output based only on the results of these calls; in particular, input to the compression function is not considered.

Most PuRF-based (and blockcipher-based) compression functions are of a special type. Instead of arbitrary pre- and postprocessing, one finds only functions that are blockwise linear. The Knudsen-Preneel construction is also blockwise linear, so let us recall from [7] what is a *blockwise-linear scheme*.

Definition 1 (Blockwise-linear scheme). *Let r, c, b, t, s be positive integers and let matrices $\mathsf{C}^{\mathrm{PRE}} \in \mathbb{F}_2^{rcb \times tb}$, $\mathsf{C}^{\mathrm{POST}} \in \mathbb{F}_2^{sb \times rb}$ be given. We define $H = \mathsf{BL}^b(\mathsf{C}^{\mathrm{PRE}}, \mathsf{C}^{\mathrm{POST}})$ to be a family of single-layer PuRF-based compression functions $H_n : \{0,1\}^{tn} \to \{0,1\}^{sn}$, for all positive integers n with $b|n$. Specifically, let $n'b = n$, and $f_1, \ldots, f_r \in \mathrm{Func}(cn, n)$. Then on input $W \in \{0,1\}^{tn}$ (interpreted as column vector), ${H_n}^{f_1 \ldots f_r}(W)$ computes the digest $Z \in \{0,1\}^{sn}$ as follows:*

1. *Compute* $X \leftarrow (\mathsf{C}^{\text{PRE}} \otimes I_{n'}) \cdot W$;
2. *Parse* $X = (x_i)_{i=1\ldots r}$ *and for* $i = 1\ldots r$ *compute* $y_i = f_i(x_i)$;
3. *Parse* $(y_i)_{i=1\ldots r} = Y$ *and output* $Z = (\mathsf{C}^{\text{POST}} \otimes I_{n'}) \cdot Y$.

where $\otimes$ *denotes the Kronecker product and* $I_{n'}$ *the identity matrix in* $\mathbb{F}_2^{n' \times n'}$.

In the definition above we silently identified $\{0,1\}^n$ with the vector space $\mathbb{F}_2^n$. The map corresponding to $(\mathsf{C}^{\text{PRE}} \otimes I_{n'})$ will occasionally be denoted C^{PRE} and its image $\Im(C^{\text{PRE}}) \subseteq \{0,1\}^{rcn}$. It will be convenient for us to write the codomain of C^{PRE} as a direct sum, so we identify $\{0,1\}^{rcn}$ with $\bigoplus_{i=1}^{r} V_i$ where $V_i = \mathbb{F}_2^{cn}$ for $i = 1, \ldots, r$. If $x_1 \in V_1$ and $x_2 \in V_2$, then consequently $x_1 + x_2$ will be in $V_1 \oplus V_2$. (This extends naturally to $L_1 + L_2$ when $L_1 \subset V_1, L_2 \subset V_2$.)

Knudsen-Preneel compression functions. Knudsen and Preneel [4,5] introduced a family of hash functions employing error correcting codes. (We use the journal version [6] as our frame of reference). Although their work was ostensibly targeted at blockcipher-based designs, the main technical thread of their work develops a transform that extends the range of an 'ideal' compression function (blockcipher-based, or not) in a manner that delivers some target level of security. As is nowadays typical, we understand an ideal compression function to be a PuRF. In fact, the KP transform is a special instance of a blockwise-linear scheme (Definition 1), in which the inputs to the PuRFs are determined by a linear code over a binary field with extension degree $e > 1$, i.e. $\mathbb{F}_{2^e}$, and with C^{POST} being the identity matrix over $\mathbb{F}_2^{rb \times rb}$ (corresponding to concatenating the PuRF outputs). The extension field itself is represented as a subring of the matrix ring (of dimension equalling the extension degree) over the base field. We formalize this by an injective ring homomorphism $\varphi : \mathbb{F}_{2^e} \to \mathbb{F}_2^{e \times e}$ and let $\bar{\varphi} : \mathbb{F}_{2^e}^{r \times k} \to \mathbb{F}_2^{re \times ke}$ be the component-wise application of φ and subsequent identification of $(\mathbb{F}_2^{e \times e})^{r \times k}$ with $\mathbb{F}_2^{re \times ke}$ (we will use $\bar{\varphi}$ for matrices over $\mathbb{F}_{2^e}$ of arbitrary dimensions). For completeness, there is also a group homomorphism $\psi : \mathbb{F}_{2^e} \to \mathbb{F}_2^e$ such that for all $g, h \in \mathbb{F}_{2^e}$ it holds that $\psi(gh) = \varphi(g) \cdot \psi(h)$.

Definition 2 (Knudsen-Preneel transform). *Let* $[r, k, d]$ *be a linear code over* $\mathbb{F}_{2^e}$ *with generator matrix* $G \in \mathbb{F}_{2^e}^{k \times r}$. *Let* $\varphi : \mathbb{F}_{2^e} \to \mathbb{F}_2^{e \times e}$ *be an injective ring homomorphism and let* b *be a positive divisor of* e *such that* $ek > rb$. *Then the Knudsen-Preneel compression function* $H = \mathsf{KP}^b([r, k, d]_{2^e})$ *equals* $H = \mathsf{BL}^b(\mathsf{C}^{\text{PRE}}, \mathsf{C}^{\text{POST}})$ *with* $\mathsf{C}^{\text{PRE}} = \bar{\varphi}(G^T)$ *and* $\mathsf{C}^{\text{POST}} = I_{rb}$.

If $H = \mathsf{KP}^b([r, k, d]_{2^e})$, then $H_n : \{0,1\}^{kcn} \to \{0,1\}^{rn}$ with $c = e/b$ is defined for all n for which b divides n. Moreover, H_n is based on r PuRFs in $\text{Func}(cn, n)$. For use of H in an iterated hash function, note that per invocation (of H) one can compress $(ck - r)$ message blocks (hence the requirement $ek > rb$ ensures actually compression is taking place), and the rate of the compression function is $ck/r - 1$. We will concentrate on the case $(b, e) \in \{(1, 2), (2, 4), (1, 3)\}$ and then in particular on the 16 parameter sets given by Knudsen and Preneel.[1] Since b is uniquely determined given e (and c), we will often omit it.

[1] We note that our analysis is also valid for $c = 5$ (mimicking the MD4/5 situation).

Security notions. A *collision-finding adversary* is an algorithm whose goal is to find two distinct inputs W, W' that hash to the same value, so $H(W) = H(W')$. We will consider adversaries in two scenarios: the information-theoretic one and a more realistic concrete setting. For information-theoretic adversaries the only resource of interest is the number of queries made to their oracles. Otherwise, these adversaries are considered (computationally) unbounded. In the concrete setting, on the other hand, we are interested in the actual runtime of the algorithm and, to a lesser extent, its memory consumption (and code-size).

3 Prior Art on the Knudsen-Preneel Hash Functions

Knudsen and Preneel's security claims. Knudsen and Preneel concentrate on the collision resistance of their compression function in the complexity theoretic model. Under a fairly generous (but plausible) assumption, they essentially show that if $H = \mathsf{KP}^b([r, k, d]_{2^e})$, then finding collisions in H_n takes time at least $2^{(d-1)n/2}$. For preimage resistance Knudsen and Preneel do not give a corresponding theorem and assumption, yet they do *conjecture* it to be essentially the square of the collision resistance.

Knudsen and Preneel also present two attacks, one for finding preimages [6, Proposition 3] and one for finding collisions [6, Proposition 4] (see results in Table 1). Both attacks revolve around finding multi-preimages for the systematic part of the construction in sufficient numbers to make it likely that completion to the non-systematic part will yield a full preimage respectively a full collision.

Watanabe's collision-finding attack. Knudsen and Preneel left a considerable gap between the actual complexity of attacks and their lower bounds in the case of collision resistance. Watanabe [14] has pointed out a collision attack that runs in time $k2^n$ (and as many PuRF evaluations). Thus, for many of the parameter sets, it beats the one given by Knudsen and Preneel. More interestingly, his attack serves as proof that the *lower* bound given by Knudsen and Preneel is *incorrect* for a large class of parameters: whenever $r < 2k$ and $d > 3$, which involves 6 out of 16 parameter sets. (See also Table 1.)

Assume that the code's generator matrix is systematic, that is $G = (I_k|P)$ with $P \in \mathbb{F}_{2^e}^{k\times(r-k)}$. Then the goal is to generate, for each $i \in \{1, \ldots, k\}$, a colliding pair of inputs $x_i \neq x_i'$ (and $f_i(x_i) = f_i(x_i')$) in such a way that their completion to full 'codewords' satisfies $x_i = x_i'$ for $i \in \{k+1, \ldots, r\}$. This is done by ensuring that $x_i \oplus x_i' = \Delta_i$ where $\Delta = \sum_{i=1}^{k} \Delta_i \in \mathbb{F}_2^{ken'} \setminus \{0\}$ is in the kernel of $\bar{\varphi}(P^T) \otimes I_{n'}$ (since $r - k < k$ the kernel is guaranteed to contain a non-trivial element). Mutual independence of the inputs to the PuRFs corresponding to the code's systematic part allow the initial collision searches to be mounted independently. Unfortunately, since the collisions need to be rather special (due to fixed Δ_i's), the birthday paradox does not apply and a collision search costs about 2^n queries and time (per PuRF). On the plus side, the attack is trivially memoryless and parallellizable.

Özen-Shrimpton-Stam preimage-finding attack. An extensive security analysis for the preimage resistance of KP-constructions, falsifying the designers' conjectured lower bound, has been provided by Özen, Shrimpton, and Stam [7]. Additionally, they also provided a related collision-finding attack with a surprisingly low query complexity: $2^{rn/(2k)}$ (but no analysis of its time complexity).

At the core of the Özen-Shrimpton-Stam attacks is the simple observation that $(0^a \,||\, x_1) \oplus (0^a \,||\, x_2)$ yields a string of the form $(0^a \,||\, X)$. More generally, any linear combination of strings with the same pattern of fixed zero bits will yield a string with the same form. By restricting PuRF queries to strings with the same (blockwise) pattern one can optimize the *yield* of these queries (i.e. the maximum number of KP compression function evaluations an adversary can compute for a given number of queries). Matching the yield with the size of the codomain (resp. its square root) gives rise to an information-theoric preimage (resp. collision) attack.

A second observation is that, in the case of a preimage attack, the dual code can be used to find the preimage far more efficiently than a naive method. Direct application of this method however is disappointing (see Table 1). The resulting time complexities are typically much higher than the corresponding query complexities and the attack is seldom competitive with that of Knudsen and Preneel, let alone with that of Watanabe.

4 Decoding the Knudsen-Preneel Preprocessing

An important property that is exploited by both Watanabe and ÖSS is linearity of C^{PRE}. Indeed, the image $\Im(C^{\text{PRE}})$ itself can be regarded as an ekn'-dimensional subspace of $\mathbb{F}_2^{ern'}$, or equivalently as an $[ern', ekn', d']_2$ code $\mathcal{C}^{\otimes}$ (where the minimum distance d' is largely irrelevant; it satisfies $d \leq d' \leq de$). This has the consequence that if $X = C^{\text{PRE}}(W)$ and $X' = C^{\text{PRE}}(W')$ collide, it is guaranteed that $\Delta = X \oplus X' \in \Im(C^{\text{PRE}})$, i.e. the difference Δ itself is a (nonzero) codeword in $\mathcal{C}^{\otimes}$. Below we will give a more detailed mathematical characterization of $\Im(C^{\text{PRE}})$, with a special eye towards the improved collision-finding algorithms we will give later on. Most of the results below are mathematically rather straightforward (and the proofs are left to the full version); the machinery is mainly needed to ensure that, when using canonical bases for the various vector spaces, everything lines up correctly and consistently with the actual Knudsen-Preneel compression function.

Recall that we are given an injective ring homomorphism $\varphi : \mathbb{F}_{2^e} \to \mathbb{F}_2^{e \times e}$ and a group isomorphism $\psi : \mathbb{F}_{2^e} \to \mathbb{F}_2^e$ that satisfy $\varphi(g)\psi(h) = \psi(gh)$ for all $g, h \in \mathbb{F}_{2^e}$. Let $[r, k, d]_{2^e}$ be a linear code with generator matrix $G \in \mathbb{F}_{2^e}^{k \times r}$, let b be a positive divisor of e such that $ek > rb$ and finally let $n = bn'$ be a multiple of b. Then the input processing $C^{\text{PRE}} : \{0,1\}^{ekn'} \to \{0,1\}^{ern'}$ of the Knudsen-Preneel compression function is defined by $C^{\text{PRE}}(W) = (\bar{\varphi}(G^T) \otimes I_{n'}) \cdot W$ (and note that $ern' = rcn$).

Characterization of $\Im(C^{\text{PRE}})$ as a sum. We have already written the codomain of C^{PRE} as a direct sum of *PuRF inputs* by identifying $\{0,1\}^{ren'}$ with $\bigoplus_{i=1}^{r} V_i$

where $V_i = \mathbb{F}_2^{en'}$ for $i = 1, \ldots, r$. Here we will use a second interpretation that emphasizes the *code*. We will consider $\bigoplus_{j=1}^{n'} U_j$ where $U_j = \mathbb{F}_{2^e}^r$ for $j = 1, \ldots, n'$. Since $\mathbb{F}_{2^e}^r$, and by extension $\bigoplus_{j=1}^{n'} U_j$, is a vector space over $\mathbb{F}_{2^e}$, whereas $\{0,1\}^{ern'}$ is a stand-in for the vector space $\mathbb{F}_2^{ern'}$ over $\mathbb{F}_2$, we cannot find a vector space isomorphism (as for the earlier direct sum). Nonetheless we can find a suitable group isomorphism from $\bigoplus_{j=1}^{n'} U_j$ to $\{0,1\}^{ern'}$.

To define the group isomorphism we exploit that, luckily, the underlying $\mathbb{F}_{2^e}$ arithmetic is essentially preserved by $C^{\text{PRE}} : \{0,1\}^{ekn'} \to \{0,1\}^{ern'}$, even though the '$\otimes I_{n'}$' in $C^{\text{PRE}}(W) = (\bar{\varphi}(G^T) \otimes I_{n'}) \cdot W$ garbles things up. To formalize this, let $\rho : \mathbb{F}_{2^e}^{n'} \to \mathbb{F}_2^{en'}$ be the group isomorphism such that $\rho(g\delta) = (\varphi(g) \otimes I_{n'}) \cdot \rho(\delta)$ for all $\delta \in \mathbb{F}_{2^e}^{n'}$ and $g \in \mathbb{F}_{2^e}$.

As usual, we will extend ρ to e.g. r-tuples of elements in $\mathbb{F}_{2^e}^{n'}$ (and hence to vectors in $\mathbb{F}_{2^e}^{n'r}$) by component-wise application, i.e. $\bar{\rho} : \mathbb{F}_{2^e}^{n'r} \to \mathbb{F}_2^{en'r}$. This suffices for a group isomorphism from $\bigoplus_{j=1}^{n'} U_j$ to $\{0,1\}^{ern'}$ as well.

Lemma 1. *Let $\mathcal{I}_0 \subset \{1, \ldots, r\}$, let $\mathcal{C}'$ be the (quasi) shortening of $\mathcal{C}$ on $\mathcal{I}_0$ and let $\mathcal{C}'_j = \mathcal{C}' \subseteq U_j$ for $j = 1, \ldots, n'$. Then $X = \sum_{i=1}^r x_i \in \Im(C^{\text{PRE}})$ with $x_i = 0$ for all $i \in \mathcal{I}_0$ iff $\exists\,!\ \forall_{j=1,\ldots,n'}\ g_j = \sum_{i=1}^r g_{ji} \in \mathcal{C}'_j$ such that $x_i = \rho(\sum_{j=1}^{n'} g_{ji})$.*

The following proposition develops the key idea on how to *recognize* that a given $X \in \mathbb{F}_2^{ern'}$ is an element of $\Im(C^{\text{PRE}})$. This result is exploited in [7] to efficiently find preimages for Knudsen-Preneel compression functions.

Proposition 1. *Let $H = \mathsf{KP}^b([r,k,d]_{2^e})$, $M \in \mathbb{F}_2^{e \times re/b}$ and a nonzero $X \in \mathbb{F}_2^{ern'}$ be given. Suppose that $M = \bar{\varphi}(h^T)$ for some $h \in \mathcal{C}^\perp$, then $X \in \Im(C^{\text{PRE}})$ iff for all positive integers n' it holds that $(M \otimes I_{n'}) \cdot X = 0$.*

Since $\mathbb{F}_{2^e}^{n'r}$ is isomorphic (as vector space over $\mathbb{F}_{2^e}$) to the tensor product $\mathbb{F}_{2^e}^r \otimes \mathbb{F}_{2^e}^{n'}$ this leads in a natural way to a function from $\mathbb{F}_{2^e}^r \times \mathbb{F}_{2^e}^{n'}$ to $\{0,1\}^{ren'}$ by considering pure tensors $g \otimes \delta$ with $g \in \mathbb{F}_{2^e}^r$ and $\delta \in \mathbb{F}_{2^e}^{n'}$. Note that we do not discriminate between different representatives, that is for nonzero $\beta \in \mathbb{F}_{2^e}$ we have that $g \otimes \delta = (\beta g) \otimes (\beta^{-1}\delta)$.

Lemma 2. *If $g \in \mathbb{F}_{2^e}^r$ and $\delta \in \mathbb{F}_{2^e}^{n'}$ then $\bar{\rho}(g \otimes \delta) \in \Im(C^{\text{PRE}})$ iff $g \in \mathcal{C}$ or $\delta = 0$.*

The following lemma states that invertibility of $G_{\tilde{\mathcal{I}}}$ suffices to invert C^{PRE}.

Lemma 3. *Let G be a generator matrix for an $[r,k,d]_{2^e}$ code. Let $\tilde{\mathcal{I}} \subset \{1, \ldots, r\}$ be such that $G_{\tilde{\mathcal{I}}}$ is invertible, with transposed inverse $G_{\tilde{\mathcal{I}}}^{-T}$. Let n' be an integer and, for $i = 1, \ldots, r$, let $V_i = \mathbb{F}_2^{en'}$ be a direct-sum-decomposition of $\mathbb{F}_2^{ern'}$ as before. If given $x_i \in V_i$ for $i \in \tilde{\mathcal{I}}$, or equivalently $\tilde{X} = \sum_{i \in \tilde{\mathcal{I}}} x_i$, then*

$$W = (\bar{\varphi}(G_{\tilde{\mathcal{I}}}^{-T}) \otimes I_{n'}) \cdot \tilde{X}$$

is the unique element for which $X' = C^{\text{PRE}}(W)$ satisfies $x'_i = x_i$ for $i \in \tilde{\mathcal{I}}$.

Algorithm 1 (Revised Watanabe Collision Attack).

Input: $H = \mathsf{KP}^b([r,k,d]_{2^e})$ satisfying $d \leq k$, a nonzero $g \in \mathcal{C} \subseteq \mathbb{F}_{2^e}^r$ with $|\chi(g)| \leq k$, a block size $n = bn'$, and an arbitrary nonzero $\delta \in \mathbb{F}_{2^e}^{n'}$.

Output: A colliding pair $(W, W') \in \left(\{0,1\}^{ekn'}\right)^2$ such that $H_n(W) = H_n(W')$, $W \neq W'$ and $C^{\text{PRE}}(W) \oplus C^{\text{PRE}}(W') = \bar{\rho}(g \otimes \delta)$.

1. Initialization. Compute $\Delta \leftarrow \bar{\rho}(g \otimes \delta)$, set $\mathcal{I} \leftarrow \chi(g)$ and determine $\tilde{\mathcal{I}} \supseteq \mathcal{I}$ for which $G_{\tilde{\mathcal{I}}}$ is invertible.
2. Query Phase. For $i \in \mathcal{I}$ do
   a. Generate a random $x_i \xleftarrow{\$} V_i (= \mathbb{F}_2^{en'})$ and set $x'_i \leftarrow x_i \oplus \Delta_i$;
   b. Query $y_i \leftarrow f_i(x_i)$ and $y'_i \leftarrow f_i(x'_i)$;
   c. If $y_i = y'_i$ then keep (x_i, x'_i) and proceed to next i, else return to a.
3. Degrees of Freedom. For $i \in \tilde{\mathcal{I}} \backslash \mathcal{I}$ pick $x_i \xleftarrow{\$} V_i$ and set $x'_i \leftarrow x_i$.
4. Finalization. Output (W, W') where

$$W \leftarrow (\bar{\varphi}(G_{\tilde{\mathcal{I}}}^{-T}) \otimes I_{n'}) \cdot (\sum_{i \in \tilde{\mathcal{I}}} x_i) \qquad \text{and} \qquad W' \leftarrow (\bar{\varphi}(G_{\tilde{\mathcal{I}}}^{-T}) \otimes I_{n'}) \cdot (\sum_{i \in \tilde{\mathcal{I}}} x'_i)\,.$$

5 A New Symbiotic Collision-Finding Attack

5.1 Revising Watanabe's Attack

Watanabe's attack has complexity $k2^n$, requires $k > r - k$ and essentially finds a single collision. Below we give a revised and improved version of his algorithm. It only has complexity $d2^n$, requires $k \geq d$ and it potentially results in many, many collisions. More precisely, if $k > d$ then after the initial effort (of $d2^n$) we can find a new collision in *constant* time, for up to a whopping $2^{(k-d)n}$ collisions.

In his note, Watanabe describes his attack as a differential attack. Where originally Δ was computed as some non-trivial kernel element, we will compute it based on a codeword $g \in \mathcal{C}$ of sufficiently low weight and an arbitrary (nonzero) 'block multiplier' δ. In particular, we will set $\Delta = \bar{\rho}(g \otimes \delta)$. By using a minimal weight codeword the attack performs best.

For the revised attack to work, we need one further ingredient. Watanabe assumes a systematic code and exploits that, when $k < r - k$, there exists a nonzero codeword $g \in \mathcal{C}$ for which $\chi(g) \subseteq \{1, \ldots, k\}$. This allows easy completion of a partial collision to a full collision. Our revised version allows an arbitrary (nonzero) codeword g of weight at most k (existence of which requires $d \leq k$). Thus $\chi(g)$ might no longer map to the systematic part of the code. Luckily, Lemma 3 provides completion to a full collision, provided $\mathcal{I} = \chi(g)$ is *admissible*. For MDS codes all codewords are admissible; for the four non-MDS codes proposed by Knudsen and Preneel it can be verified that the minimum distance codewords are admissible.

Theorem 1 (Revised Watanabe attack). *Let $H = \mathsf{KP}^b([r,k,d]_{2^e})$ be given with $d \leq k$. Consider H_n (with $b|n$). Then Algorithm 1 using a minimum-weight*

Algorithm 2 (New Symbiotic Collision Attack).

Input: $H = \mathsf{KP}^b([r,k,d]_{2^e})$ satisfying $d \leq k$, a $g \in \mathcal{C} \subseteq \mathbb{F}_{2^e}^r$ with $|\chi(g)| = d$, and a block size $n = bn'$.

Output: A colliding pair $(W, W') \in \left(\{0,1\}^{ekn'}\right)^2$ such that $H_n(W) = H_n(W')$, $W \neq W'$ and $C^{\text{PRE}}(W) \oplus C^{\text{PRE}}(W') = \bar{\rho}(g \otimes \delta)$ for some nonzero $\delta \in \mathbb{F}_{2^e}^{n'}$ to be determined.

1. Initialization. Set $\alpha = d/(d+1), \mathcal{I} = \chi(g)$ and determine $\tilde{\mathcal{I}}$. Let $g = (g_1, \ldots, g_r)$ with $g_i \in \mathbb{F}_{2^e}$ for $i = 1, \ldots, r$.
2. Query Phase. Define
$$\mathcal{X} = (\{0\}^{\frac{n}{b} - \frac{\alpha n}{e}} \times \{0,1\}^{\frac{\alpha n}{e}})^e$$
and, for $i \in \mathcal{I}$ let $Q_i = \mathcal{X} \subset V_i$. Query $f_i \; \forall \; x_i \in Q_i$ and store the results.
3. Local Collision Detection. For $i \in \mathcal{I}$ create a list L_i of all tuples $(g_i^{-1} \cdot \rho^{-1}(x_i \oplus x_i'), x_i, x_i')$ satisfying $x_i, x_i' \in Q_i, x_i \neq x_i'$ and $f_i(x_i) = f_i(x_i')$.
4. Global Collision Detection. Find a set of $|\chi(g)|$ tuples in the respective L_i that all share the same first element. That is, for some $\delta \in \mathbb{F}_{2^e}^{n'}$ and $(x_i, x_i')_{i \in \mathcal{I}}$ it holds for all $i \in \mathcal{I}$ that $(\delta, x_i, x_i') \in L_i$.
5. Degrees of Freedom. For $i \in \tilde{\mathcal{I}} \backslash \mathcal{I}$ pick $x_i \xleftarrow{\$} V_i$ and set $x_i' \leftarrow x_i$.
6. Finalization. Output (W, W') where
$$W \leftarrow (\bar{\varphi}(G_{\tilde{\mathcal{I}}}^{-T}) \otimes I_{n'}) \cdot (\sum_{i \in \tilde{\mathcal{I}}} x_i) \qquad \text{and} \qquad W' \leftarrow (\bar{\varphi}(G_{\tilde{\mathcal{I}}}^{-T}) \otimes I_{n'}) \cdot (\sum_{i \in \tilde{\mathcal{I}}} x_i') \; .$$

codeword g (and an arbitrary nonzero δ) finds collisions for H_n in expected time $d2^n$ (using as many PuRF evaluations).

5.2 A New Symbiotic Attack

Our revised version of Watanabe's attack clearly shows that an attacker potentially has a lot of freedom. Below we transform some of this freedom into a faster attack. More to the point, as in the revised Watanabe attack we still look for a collision with differential $\Delta = \bar{\rho}(g \otimes \delta)$ and fix the codeword $g \in \mathcal{C}$, but we do *not* fix the multiplier δ up front. Instead we determine it based on the outcomes of the queries we make. To increase our success probability, we restrict to the same kind of queries as Özen, Shrimpton, and Stam did.

Theorem 2 (Symbiotic attack). *Let $H = \mathsf{KP}^b([r,k,d]_{2^e})$ be given with $k \geq d$. Consider H_n (with $b|n$). Then Algorithm 2 finds collisions for H_n in $d2^{dn/(d+1)}$ time (using as many PuRF evaluations) and memory (expressed in n-bit blocks).*

Proof (Sketch). We will leave showing the correctness of Algorithm 2 to the full version of this paper [8] and only prove here that a collision is expected and that the query and time complexities are as claimed.

Since $|\mathcal{X}| = 2^{\alpha n}$ by construction, the attack has the stated query complexity (per PuRF) for $\alpha = d/(d+1)$ since all queries are made during the Query

Phase. Using a naive approach, Local Collision Detection step can be performed in roughly $2^{dn/(d+1)}$ comparisons resulting in partial collision lists of expected cardinality $|L_i| \approx 2^{(2\alpha-1)n}$ for $i \in \mathcal{I}$.

For Global Collision Detection, we just enumerate one partial collision list and check for membership against the others. Assuming constant time memory access, the time complexity of this step is at most $(d-1)\max_{i\in\mathcal{I}}|L_i|$. Since $\alpha < 1$ it follows that $2\alpha - 1 < \alpha$ making the Query Phase dominant with its time complexity of $2^{\alpha n}$.

Since we have d active PuRFs in total, the probability of finding a common element among d such lists is then $(\prod_i |L_i|)/|\mathcal{X}|^{d-1}$, or $2^{((2\alpha-1)d-\alpha(d-1))n}$. To ensure an expected number of collisions of one, we need the second exponent to be at least zero, and indeed, solving for zero gives the desired $\alpha = d/(d+1)$. □

6 A Parametrized Collision-Finding Attack

6.1 Optimizing the Query Complexity

The symbiotic attack and the information-theoretic attack by Özen, Shrimpton, and Stam have completely different query complexities and which one is the best seems very parameter dependent. However, it turns out that both attacks are the extreme cases of a more general parametrized attack, as given by Algorithm 3.

Theorem 3. *Let $H = \mathsf{KP}^b([r,k,d]_{2^e})$ be given. Consider H_n (with $b|n$). Then collisions for H_n can be found with Alg. 3 using $2^{\alpha n}$ queries (per PuRF) where*

$$\alpha = \begin{cases} (r-\theta)/(2k-\theta) & \text{for } 0 \le \theta \le \min(r-d, r-k)\ ; \\ (r-\theta)/(r+k-2\theta) & \text{for } r-k \le \theta \le r-d\ . \end{cases}$$

Proof. That the attack has the stated query complexity follows readily from the usual observation that $|\mathcal{X}| = 2^{\alpha n}$ combined with the computation of α exactly matching the theorem statement. What remains to show is that collisions are indeed output and expected with good probability.

For correctness, let (W, W') be output by the algorithm and consider $X = C^{\text{PRE}}(W)$ and $X' = C^{\text{PRE}}(W')$. First, notice that Lemma 3 implies that projecting $(X \oplus X', X, X')$ onto $\bigoplus_{i\in\tilde{\mathcal{I}}} V_i$ is in $L_{\tilde{\mathcal{I}}}$. Now, either of the steps Degrees of Freedom, Filtering or Skip ensures that $(\tilde{\Delta}, \tilde{X}, \tilde{X}') \in L_{\mathcal{I}}$. Finally, since $L_{\mathcal{I}} \subseteq \tilde{L}_{\mathcal{I}}$ it follows that $(x_i, x_i') \in L_i$ for $i \in \mathcal{I}$ and hence by construction (Local Collision Detection) we have $f_i(x_i) = f_i(x_i')$ for those i.

Moreover Collision Pruning guarantees that $\tilde{\Delta} + 0 \in \Im(C^{\text{PRE}})$ and Degrees of Freedom ensures that the projections of $\tilde{\Delta} + 0$ and $X \oplus X'$ onto $\bigoplus_{i\in\tilde{\mathcal{I}}} V_i$ are equal. Hence, $x_i = x_i'$ for all $i \in \mathcal{I}_0$.

Let us move on to the number of expected collisions output. Since $|\mathcal{X}| = 2^{\alpha n}$, the expected number of local collisions found per active PuRF for $i \in \mathcal{I}$ is $|L_i| \approx |\mathcal{X}|^2/2^n = 2^{(2\alpha-1)n}$. Using that $|\mathcal{I}| = r - \theta$ we arrive at a total number of potential collisions of $|\tilde{L}_{\mathcal{I}}| \approx 2^{(2\alpha-1)(r-\theta)n}$. For a true collision to occur, we need

Algorithm 3 (Parameterized Collision Attack).

Input: $H = \mathsf{KP}^b([r,k,d]_{2^e})$, an index set $\mathcal{I}_0 \subset \{1,\ldots,r\}$ with $\theta = |\mathcal{I}_0|$ and $0 \le \theta \le r-d$, and a block size $n = bn'$.

Output: A colliding pair $(W,W') \in \left(\{0,1\}^{ekn'}\right)^2$ such that $H_n(W) = H_n(W'), W \neq W'$, and if $X = C^{\text{PRE}}(W)$ and $X' = C^{\text{PRE}}(W')$ then for all $i \in \mathcal{I}_0$ it holds that $x_i = x'_i$.

1. Initialization. Set $\mathcal{I} \leftarrow \{1,\ldots,r\}\backslash\mathcal{I}_0$, determine $\tilde{\mathcal{I}}$, and set
$$\alpha \leftarrow \begin{cases} (r-\theta)/(2k-\theta) & \text{for } 0 \le \theta \le \min(r-k, r-d)\ ; \\ (r-\theta)/(r+k-2\theta) & \text{for } r-k \le \theta \le r-d\ . \end{cases}$$
2. Query Phase. As in Algorithm 2.
3. Local Collision Detection. For $i \in \mathcal{I}$ create a list L_i of all tuples $(x_i \oplus x'_i, x_i, x'_i)$ satisfying $x_i, x'_i \in Q_i, x_i \neq x'_i$ and $f_i(x_i) = f_i(x'_i)$.
4. Merge Phase. Create $\tilde{L}_{\mathcal{I}} = \left\{\sum_{i\in\mathcal{I}}(\Delta_i, x_i, x'_i) \,|\, (\Delta_i, x_i, x'_i) \in L_i\right\}$.
5. Collision Pruning. Create $L_{\mathcal{I}}$ consisting precisely of those elements of $\tilde{L}_{\mathcal{I}}$ whose first vector (when mapped to the full space) is in $\Im(C^{\text{PRE}})$;
$$L_{\mathcal{I}} = \left\{(\tilde{\Delta}, \tilde{X}, \tilde{X}') | (\tilde{\Delta}, \tilde{X}, \tilde{X}') \in \tilde{L}_{\mathcal{I}} \wedge \tilde{\Delta} + 0 \in \Im(C^{\text{PRE}})\right\}\ .$$
6. Filtering. If $\tilde{\mathcal{I}} \subset \mathcal{I}$ then only select $(\tilde{\Delta}, \tilde{X}, \tilde{X}') \in L_{\mathcal{I}}$ for which $\tilde{X}$ is in the projection of $\Im(C^{\text{PRE}})$ onto $\bigoplus_{i\in\mathcal{I}} V_i$. Create $L_{\tilde{\mathcal{I}}}$ by projecting the selected elements in $L_{\mathcal{I}}$ to the subspace $\bigoplus_{i\in\tilde{\mathcal{I}}} V_i$.
7. Degrees of Freedom. If $\mathcal{I} \subset \tilde{\mathcal{I}}$, then for $i \in \tilde{\mathcal{I}}\backslash\mathcal{I}$ pick $x_i \xleftarrow{\$} V_i$ and set $x'_i \leftarrow x_i$. Create $L_{\tilde{\mathcal{I}}}$ by adding $\sum_{i\in\tilde{\mathcal{I}}\cap\mathcal{I}_0}(0, x_i, x'_i)$ to all elements in $L_{\mathcal{I}}$.
8. Skip. If $\tilde{\mathcal{I}} = \mathcal{I}$ set $L_{\tilde{\mathcal{I}}} \leftarrow L_{\mathcal{I}}$.
9. Finalization. For some $(\tilde{\Delta}, \tilde{X}, \tilde{X}') \in L_{\tilde{\mathcal{I}}}$ output (W, W') where
$$W \leftarrow (\bar{\varphi}(G_{\tilde{\mathcal{I}}}^{-T}) \otimes I_{n'}) \cdot \tilde{X} \qquad \text{and} \qquad W' \leftarrow (\bar{\varphi}(G_{\tilde{\mathcal{I}}}^{-T}) \otimes I_{n'}) \cdot \tilde{X}'$$

to find a tuple $(x_i, x'_i)_{i\in\tilde{\mathcal{I}}}$ such that both $\sum_{i\in\tilde{\mathcal{I}}} x_i$ and $\sum_{i\in\tilde{\mathcal{I}}} x'_i$ can be completed to codewords subject to the constraint that $x_i = x'_i$ for $i \in \mathcal{I}_0$.

If the eventual collision consists of (X, X'), then $\Delta = X \oplus X'$ is a codeword as well and the above implies that $\Delta_i = 0$ for $i \in \mathcal{I}_0$. Hence, Lemma 1 applies and $\tilde{\Delta} = \sum_{i\in\mathcal{I}} \Delta_i$ is somehow 'spanned' by the shortened code. The restriction $\theta \le r-d$ ensures nontriviality of the shortened code (shortening any further and the shortened code would consist of the zero codeword only resulting in $W = W'$, so no collision). In case of MDS codes, the shortened code has parameters $[r-\theta, k-\theta, d']_{2^e}$, in particular it has dimension $k-\theta$. (For non-MDS codes it is possible that a higher dimension is achieved.)

As a result, a fraction $2^{(k-r)\alpha n}$ of the differentials will be satisfactory, leading to an expected number of $|L_{\mathcal{I}}| \approx 2^{((2\alpha-1)(r-\theta)-\alpha(r-k))n}$. If $\mathcal{I} \subseteq \tilde{\mathcal{I}}$ or equivalently $r-\theta \le k$ we are done whenever $|L_{\mathcal{I}}| \ge 1$. Since $r-\theta \le k$ can be rewritten to $\theta \ge r-k$ we are in the second case, with $\alpha = (r-\theta)/(r+k-2\theta)$. Writing

$F = (\lg |L_{\mathcal{I}}|)/n$ and substitution lead to $F \approx (2\alpha - 1)(r - \theta) - \alpha(r - k) = \alpha(2r - 2\theta - r + k) - (r - \theta) = 0$ or $|L_{\mathcal{I}}| \approx 1$ as desired.

If on the other hand $\tilde{\mathcal{I}} \subset \mathcal{I}$, further filtering is needed. In particular, given a potential 'half' of a collision X we need to check if it can correspond to a codeword. Since $\tilde{\mathcal{I}} \subset \mathcal{I}$, we can uniquely complete X to a codeword given k of its elements (all within $\mathcal{I}$). The remaining $|\mathcal{I}| - k$ coordinates need to be in sync. Per remaining element, this occurs with probability $2^{-\alpha n}$, leading to $|\tilde{L}_{\tilde{\mathcal{I}}}| \approx |L_{\mathcal{I}}| \cdot 2^{-\alpha n(r-\theta-k)}$. Now we are in the first case since $0 \leq \theta \leq r - k$. Writing $F = (\lg \tilde{L}_{\tilde{\mathcal{I}}})/n$, we obtain $F \approx ((2\alpha-1)(r-\theta)-\alpha(r-k))-\alpha(r-\theta-k) = \alpha(2k-\theta) - (r-\theta)$. Since we aim for $F = 0$, $\alpha = (r-\theta)/(2k-\theta)$ as desired. □

Corollary 1. *Assuming $d \leq k$, substitution of $\theta = r - k$ in Theorem 3 gives $\alpha = k/(3k - r)$. This is optimal (for Algorithm 3) whenever $r \leq 2k$.*

Proof. That the substition does what it says can be readily verified, so we restrict ourselves to prove the optimality here. Let $f_1(\theta) = (r - \theta)/(2k - \theta)$ and $f_2(\theta) = (r - \theta)/(r + k - 2\theta)$ be two real valued functions defined over closed intervals $0 \leq \theta \leq r - k$ and $r - k \leq \theta \leq r - d$ respectively. Note that both $f_1(\theta)$ and $f_2(\theta)$ are continuous in their respective domains (since their respective poles fall outside the domains). So both $f_1(\theta)$ and $f_2(\theta)$ attain their maximum and minimum in the closed intervals $[0, r - k]$ and $[r - k, r - d]$ respectively. Since $f_1'(\theta) = (r-2k)/(2k-\theta)^2 \leq 0$ (for $r \leq 2k$) and $f_2'(\theta) = (r-k)/(r+k-2\theta)^2 \geq 0$ we can conclude that $f_1(\theta)$ is decreasing and $f_2(\theta)$ is increasing. Therefore, they both attain their minimum at their shared boundary $\theta = r - k$. □

Remark 1. The only two parameter sets proposed by Knudsen and Preneel not satisfying the conditions of the corollary above are $[4, 2, 3]_8$ and $[5, 2, 4]_8$. In both cases $d > k$ and only $f_1(\theta)$ is applicable. For $[5, 2, 4]_8$ one can check that $2k < r$ and $f_1'(\theta) \geq 0$. Hence, the minimum α is attained at $\theta = 0$. For $[4, 2, 3]_8$ it holds that $2k = r$, so that $f_1(\theta)$ is in fact a constant function and both $\theta = 0$ and $\theta = 1$ lead to the same α.

Remark 2. Substitution of $\theta = 0$ in Theorem 3 gives $\alpha = r/(2k)$ and the resulting query complexity coincides with that reported by Özen, Shrimpton, and Stam. On the other extreme, substitution of $\theta = r - d$ gives $\alpha = d/(2d - r + k)$ (assuming $d \leq k$). For MDS codes this simplifies to $\alpha = d/(d+1)$, this time duly coinciding with our symbiotic attack. For non-MDS codes there seems to be a slight mismatch. The reason is that if a non-MDS code is maximally shortened (by $\theta = r - d$), the shortened code has dimension 1, whereas in the derivation of Theorem 3 we pessimistically assumed $k - \theta = 0$ (at least for the KP non-MDS codes that satisfy $r - d = k$). Correcting for this looseness would result in a match with the symbiotic attack.

6.2 Generic Collision Attack against MDS Constructions

If we want to run Algorithm 3 (with fixed $\theta = r-k$ and $\alpha = k/(3k-r)$ as obtained in Corollary 1) we ideally want a time complexity almost coinciding with the

targeted query complexity. For $\theta = r - k$ it holds that $\mathcal{I} = \tilde{\mathcal{I}}$, obviating the need for the steps Filtering and Degrees of Freedom. We have already seen that Local Collision Detection costs at most a small logarithmic factor, which leaves only the Merge Phase and Collision Pruning to worry about. Together, these two steps are designed to produce $L_{\mathcal{I}}$. A naive approach would enumerate all elements in the much larger $\tilde{L}_{\mathcal{I}}$, which is wasteful. Our task is therefore, given the lists of partial collisions L_i for $i \in \mathcal{I}$, to create $L_{\mathcal{I}}$ more efficiently.

In the sequel, we will follow in the footsteps of Özen, Shrimpton, and Stam who used the dual code in a similar problem related to their preimage-finding attack. An important innovation for the collision-finding attack stems from the realization that Δ can be regarded as belonging to the (quasi) shortened code. This allows the use of the dual of the shortened code to speed up the search. As the minimum distance of the dual code is an important parameter in determining the overall time-complexity and shortening a code reduces the minimum distance of its dual accordingly, we make a significant efficiency gain this way.

Road map. We present our collision attack against Knudsen-Prennel compression functions whose C^{PRE} is based on MDS codes in Alg. 4, whereas its analysis is given in Thm. 4. We leave the generalization of our attack to (KP-suggested) non-MDS parameters together with the proof of Thm. 4 to the full version of this work where we also investigate a more space efficient version of Alg. 4.

Reducing the Time Complexity. Since $\mathcal{I} = \tilde{\mathcal{I}}$ and $\theta = r - k$, we know from Algorithm 3 that it is enough to find a nonzero $\Delta \in \Im(C^{\text{PRE}})$ of the form $\Delta = \Delta' + 0$ for $\Delta' = \sum_{i \in \tilde{\mathcal{I}}} \Delta_i$ to complete the collision. Now notice that Δ' is lying in a smaller space $\Im(C'^{\text{PRE}})$ identified by $\mathcal{C}'$ that is the $[r - \theta, k - \theta, d']$ shortened code obtained from $\mathcal{C}$ (by dropping the zeroes of the codewords from all the positions appearing in $\mathcal{I}_0$). This observation allows us to guarantee that $\Delta \in \Im(C^{\text{PRE}})$ once we determine that a candidate Δ' is in $\Im(C'^{\text{PRE}})$. Hence, it is enough for our purposes to limit ourselves to $\Im(C'^{\text{PRE}})$ rather than looking for membership in the larger space $\Im(C^{\text{PRE}})$.

To this end, we first identify an index set $\mathcal{I}_{h'} \subseteq \{1, \ldots, r\}$ (the role of h' will be explained momentarily) defining a subspace $\bigoplus_{i \in \mathcal{I}_{h'}} V_i$ for which $\Im(C'^{\text{PRE}})$ when restricted to this subspace, is not surjective. As a consequence, we will be able to prune significantly the total collection of candidate Δ's keeping only those that are possibly in $\Im(C'^{\text{PRE}})$ (restricted to $\bigoplus_{i \in \mathcal{I}_{h'}} V_i$). In the sequel, we will show how to *efficiently* find an index set $\mathcal{I}_{h'}$, and how to *efficiently* prune.

An important parameter determining the runtime of our collision attack is $d'^{\perp}$, the minimum distance of the dual shortened code. Let χ be the function that maps $h' \in \mathbb{F}_{2^e}^{r-\theta}$ to the set of indices of non-zero entries in h'. Thus, $\chi(h') \subseteq \{1, \ldots, r\}$ and $|\chi(h')|$ equals the Hamming weight of the codeword h'.

An easy adaptation of Proposition 1 shows that if we are given a codeword $h' \in \mathcal{C}'^{\perp}$ and an element $\Delta' \in \mathbb{F}_2^{(r-\theta)en'}$, then Δ' can only be in $\Im(C'^{\text{PRE}})$ if $(\bar{\varphi}(h'^T) \otimes I_{n'}) \cdot \Delta' = 0$, where the only parts of Δ' relevant for this check are those lining up with the nonzero entries of h'. Indeed, an element $\Delta'_{h'} \in$

Algorithm 4 (Collision Attack against MDS-based schemes).

Input: $H = \mathsf{KP}^b([r,k,d]_{2^e})$, an index set $\mathcal{I}_0 \subset \{1,\ldots,r\}$ with $\theta = |\mathcal{I}_0| = r-k$ and a block size $n = bn'$.

Output: A colliding pair $(W,W') \in \left(\{0,1\}^{ekn'}\right)^2$ such that $H_n(W) = H_n(W'), W \neq W'$, and if $X = C^{\text{PRE}}(W)$ and $X' = C^{\text{PRE}}(W')$ then for all $i \in \mathcal{I}_0$ it holds that $x_i = x'_i$.

1. Initialization. Set $\mathcal{I} \leftarrow \{1,\ldots,r\}\backslash\mathcal{I}_0$ (with $|\mathcal{I}| = k$), $\mathcal{I} = \tilde{\mathcal{I}}$, and set $\alpha \leftarrow k/(3k-r)$. Obtain $\mathcal{C}'$ consisting of codewords $g' \in \mathcal{C}'$ that are constructed from $g \in \mathcal{C}$ by dropping zeroes of g from all the positions appearing in $\mathcal{I}_0$.
2. Query Phase. As in Algorithm 3.
3. Local Collision Detection. As in Algorithm 3.
4. Merge Phase. Find a nonzero codeword $h' \in {\mathcal{C}'}^{\perp}$ of minimum Hamming weight ${d'}^{\perp} = 2k - r + 1$. Let $h' = h'_0 + h'_1$ with $\chi(h'_0) \cap \chi(h'_1) = \emptyset$ and of Hamming weights $\lceil {d'}^{\perp}/2 \rceil$ and $\lfloor {d'}^{\perp}/2 \rfloor$ respectively. Create for $j = 0, 1$,
$$\tilde{L}_{h'_j} = \left\{ \left(\Delta'_{h'_j}, X_j, X'_j, (\bar{\varphi}(h'_j) \otimes I_{n'}) \cdot (\Delta'_{h'_j} + 0) \right) \mid (\Delta'_{h'_j}, X_j, X'_j) \in \sum_{i \in \chi(h'_j)} L_i \right\}$$
both sorted on their fourth component.
5. Join Phase. Create $L_{h'}$ consisting exactly of those elements $\Delta'_{h'_0} + \Delta'_{h'_1}$ for which $(\Delta'_{h'_0}, X_0, X'_0, Y_0) \in \tilde{L}_{h'_0}, (\Delta'_{h'_1}, X_1, X'_1, Y_1) \in \tilde{L}_{h'_1}$ and $Y_0 = Y_1$.
6. Collision Pruning. For all $(\Delta'_{h'}, X, X') \in L_{h'}$ create the unique Δ' corresponding to it and check whether it results in $\Delta_i \in L_i$ for all $i \in \mathcal{I} (= \tilde{\mathcal{I}})$. If so, keep $\Delta' = \sum_{i \in \tilde{\mathcal{I}}} \Delta_i$ in $L_{\mathcal{I}}$. Formally
$$L_{\mathcal{I}} = \left\{ (\Delta', \tilde{X}, \tilde{X}') = (\Delta'_{h'}, X, X') \in L_{h'} + \sum_{i \in \tilde{\mathcal{I}} \backslash \chi(h')} L_i | \Delta' \in \Im({C'}^{\text{PRE}}) \right\} .$$
7. Skip. & 8. Finalization. As in Algorithm 3.

$\sum_{i \in \chi(h')} L_i$ can be completed to an element in the range of derived mapping ${C'}^{\text{PRE}}$ iff $(\bar{\varphi}({h'}^T) \otimes I_{n'}) \cdot (\Delta'_{h'} + 0) = 0$. Efficient creation of
$$L_{h'} = \left\{ (\Delta'_{h'}, X, X') \in \sum_{i \in \chi(h')} L_i \mid (\bar{\varphi}({h'}^T) \otimes I_{n'}) \cdot (\Delta'_{h'} + 0) = 0 \right\}$$
can be done adapting standard techniques [2,13,11] by splitting the codeword in two and looking for all collisions in respective entries. That is, assume that $h' = h'_0 + h'_1$ with $\chi(h'_0) \cap \chi(h'_1) = \emptyset$, and define, for $j = 0, 1$
$$\tilde{L}_{h'_j} = \left\{ \left(\Delta'_{h'_j}, X_j, X'_j, (\bar{\varphi}({h'_j}^T) \otimes I_{n'}) \cdot (\Delta'_{h'_j} + 0) \right) \mid (\Delta'_{h'_j}, X_j, X'_j) \in \sum_{i \in \chi(h'_j)} L_i \right\}$$

Then $L_{h'}$ consists of those elements $\Delta'_{h'_0} + \Delta'_{h'_1}$ for which $(\Delta'_{h'_0}, X_0, X'_0, Y_0) \in \tilde{L}_{h'_0}, (\Delta'_{h'_1}, X_1, X'_1, Y_1) \in \tilde{L}_{h'_1}$ and $Y_0 = Y_1$.

By sorting the two $\tilde{L}$'s the time complexity of creating $L_{h'}$ is then roughly the maximum cardinality of the two sets $\tilde{L}_{h'_0}$ and $\tilde{L}_{h'_1}$. Hence, the main trick to reduce the time complexity is to minimize the Hamming weights of h'_0 and h'_1, which is done by picking a codeword $h' \in \mathcal{C}'^{\perp}$ of minimum distance $d'^{\perp}$ and splitting it (almost) evenly. As a result, for the partial collision lists of (almost) same cardinality S, $L_{h'}$ can be constructed in $S^{\lceil d'^{\perp}/2 \rceil}$ time using $S^{\lfloor d'^{\perp}/2 \rfloor}$ memory (ignoring inconsequential factors). We summarize our analysis in Thm. 4.

Theorem 4. *Let $H = \mathsf{KP}^b([r, k, d]_{2^e})$ be given and $\mathcal{C}'$ be a shortened $[r - \theta, k - \theta, d]_{2^e}$ code derived from $\mathcal{C}$ for $\theta = r - k$. Let $d'^{\perp}$ be the minimum distance of the dual code of $\mathcal{C}'$. Suppose $\mathcal{C}$ is MDS (so is $\mathcal{C}'$ with $d'^{\perp} = 2k - r + 1$) and consider the collision attack described in Alg. 4 run against H_n using $q = 2^{\alpha n}$ queries for $\alpha = k/(3k - r)$. Then the expected number of collision outputs is equal to one and the expectations for the internal list sizes are (for $i \in \mathcal{I}$):*

$$|L_i| = 2^{(2\alpha-1)n} \ , \ |L_{h'}| = 2^{((2\alpha-1)d'^{\perp}-\alpha)n} \ ,$$
$$|\tilde{L}_{h'_0}| = 2^{(2\alpha-1)\lceil \frac{d'^{\perp}}{2} \rceil n} \ , \ |\tilde{L}_{h'_1}| = 2^{(2\alpha-1)\lfloor \frac{d'^{\perp}}{2} \rfloor n}$$

The average case time complexity of the algorithm is $\max\left(q, |\tilde{L}_{h'_0}|, |L_{h'}|\right)$ *with a memory requirement of* $\max\left(q, |\tilde{L}_{h'_1}|\right)$ *(expressed in cn-bit blocks).*

7 Conclusion

In this paper we provide an extensive security analysis of the Knudsen-Preneel compression functions by focusing on their collision resistance. We present three improved collision attacks namely the revised Watanabe, symbiotic collision and parametrized collision attacks. Our new attacks work with the least number of queries reported so far. Moreover, except for only one out of 16 suggested parameters, these attacks beat the time-complexity of any prior attack we are aware of.

Acknowledgments. We thank Joachim Rosenthal and Thomas Shrimpton for their useful comments and suggestions.

References

1. Black, J., Rogaway, P., Shrimpton, T.: Black-box analysis of the block-cipher-based hash-function constructions from PGV. In: Yung [15], pp. 320–335
2. Chose, P., Joux, A., Mitton, M.: Fast correlation attacks: An algorithmic point of view. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 209–221. Springer, Heidelberg (2002)

3. Knudsen, L., Muller, F.: Some attacks against a double length hash proposal. In: Roy, B.K. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 462–473. Springer, Heidelberg (2005)
4. Knudsen, L.R., Preneel, B.: Hash functions based on block ciphers and quaternary codes. In: Kim, K., Matsumoto, T. (eds.) ASIACRYPT 1996. LNCS, vol. 1163, pp. 77–90. Springer, Heidelberg (1996)
5. Knudsen, L.R., Preneel, B.: Fast and secure hashing based on codes. In: Burt Kaliski, J., Burton, S. (eds.) CRYPTO 1997. LNCS, vol. 1294, pp. 485–498. Springer, Heidelberg (1997)
6. Knudsen, L.R., Preneel, B.: Construction of secure and fast hash functions using nonbinary error-correcting codes. IEEE Transactions on Information Theory 48(9), 2524–2539 (2002)
7. Özen, O., Shrimpton, T., Stam, M.: Attacking the Knudsen-Preneel compression functions. In: Hong, S., Iwata, T. (eds.) FSE 2010. LNCS, vol. 6147, pp. 94–115. Springer, Heidelberg (2010)
8. Özen, O., Stam, M.: Collision attacks against Knudsen-Preneel compression functions, full version of this paper (2010) (manuscript)
9. Özen, O., Stam, M.: Another glance at double-length hashing. In: Parker, M.G. (ed.) Cryptography and Coding. LNCS, vol. 5921, pp. 176–201. Springer, Heidelberg (2009)
10. Preneel, B., Govaerts, R., Vandewalle, J.: Hash functions based on block ciphers: A synthetic approach. In: Stinson, D. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 368–378. Springer, Heidelberg (1994)
11. Schroeppel, R., Shamir, A.: A $T = O(2^{n/2})$, $S = O(2^{n/4})$ algorithm for certain NP-complete problems. SIAM Journal on Computing 10, 456–464 (1981)
12. Stam, M.: Blockcipher-based hashing revisited. In: Dunkelman, O. (ed.) Fast Software Encryption. LNCS, vol. 5665, pp. 67–83. Springer, Heidelberg (2009)
13. Wagner, D.: A generalized birthday problem. In: Yung [15] pp. 288–303
14. Watanabe, D.: A note on the security proof of Knudsen-Preneel construction of a hash function (2006) (unpublished manuscript), http://csrc.nist.gov/groups/ST/hash/documents/WATANABE_kp_attack.pdf
15. Yung, M. (ed.): CRYPTO 2002. LNCS, vol. 2442. Springer, Heidelberg (2002)
16. Yuval, G.: How to swindle Rabin. Cryptologia 3, 187–189 (1979)

Improved Generic Attacks on Unbalanced Feistel Schemes with Expanding Functions

Emmanuel Volte[1], Valérie Nachef[1], and Jacques Patarin[2]

[1] Department of Mathematics, University of Cergy-Pontoise, CNRS UMR 8088
2 avenue Adolphe Chauvin, 95011 Cergy-Pontoise Cedex, France
[2] PRISM, University of Versailles
45 avenue des Etats-Unis, 78035 Versailles Cedex, France
emmanuel.volte@u-cergy.fr, valerie.nachef@u-cergy.fr,
jacques.patarin@prism.uvsq.fr

Abstract. "Generic" Unbalanced Feistel Schemes with Expanding Functions are Unbalanced Feistel Schemes with truly random internal round functions from n bits to $(k-1)n$ bits with $k \geq 3$. From a practical point of view, an interesting property of these schemes is that since $n < (k-1)n$ and n can be small (8 bits for example), it is often possible to store these truly random functions in order to design efficient schemes (example: CRUNCH cf [6]). Attacks on these generic schemes were studied in [7] and [18]. As pointed in [7] and [18], there are surprisingly much more possibilities for these attacks than for generic balanced Feistel schemes or generic unbalanced Feistel schemes with contracting functions. In fact, this large number of attack possibilities makes the analysis difficult. In this paper, we shall methodically analyze again these attacks. We have created a computer program that systematically analyze all the possible attacks and detect the most efficient ones. We have detected a condition on the internal variables that was not clearly analyzed in [18], and we have found many new improved attacks by a systematic study of all the "rectangle attacks" when $k \leq 7$, and then we have generalized these improved attacks for all k. Many simulations on our improved attacks have also been done and they confirm our theoretical analysis.

Keywords: Unbalanced Feistel permutations, pseudo-random permutations, generic attacks on encryption schemes, Block ciphers.

1 Introduction

A classical way to construct permutation $\{0,1\}^N$ to $\{0,1\}^N$ is to use Feistel schemes with d rounds built with round functions $f_1, \ldots, f_d$. In order to get "Random Feistel Scheme", these round functions need to be randomly chosen. "Generic attacks" on these schemes are attacks that are valid for most of the round functions.

The most usual Feistel schemes are when $N = 2n$ and the functions f_i are from $\{0,1\}^n$ to $\{0,1\}^n$. Such schemes are called "balanced Feistel Schemes" and they have been studied a lot since the famous paper by M.Luby and C.Rackoff

M. Abe (Ed.): ASIACRYPT 2010, LNCS 6477, pp. 94–111, 2010.

[12]. Many results have been obtained on the security of such classical Feistel schemes (see [13] for an overview of these results). When the number of rounds is lower than 5, we know attacks with less than $2^N (= 2^{2n})$ operations: for 5 rounds, an attack in $O(2^n)$ operations is given in [16] and for 3 or 4 rounds an attack in $\sqrt{2^n}$ is given in [1],[14]. When the functions are permutations, similar attacks for 5 rounds are given in [8] and [10]. Therefore, for security, at least 6 rounds are recommended, i.e. each bit will be changed at least 3 times.

When $N = kn$ and when the round functions are from $(k-1)n$ bits to n bits, we obtain what is called an "Unbalanced Feistel Scheme with contracting functions". In [13], M.Naor and O.Reingold give security when for the first and the last rounds pairwise independent functions are used instead of random contracting functions. In [20] security proofs for these schemes are also proved. At Asiacrypt 2006 ([17]) generic attacks on such schemes have been studied.

When $N = kn$ and when the round functions are from n bits to $(k-1)n$ bits, we obtain what is called an "Unbalanced Feistel Scheme with expanding functions", also called "complete target heavy unbalanced Feistel networks"(see [19]). Generic attacks on Unbalanced Feistel Schemes with expanding functions is the theme of this paper. One advantage of these schemes is that it requires much less memory to store a random function of n bits to $(k-1)n$ bits than a random function of $(k-1)n$ bits to n bits. Unbalanced Feistel Schemes with expanding functions together with the Xor of random permutations have been used in the construction of the hash function CRUNCH for the cryptographic hash algorithm competition organized by NIST in 2008 (cf [6]). Our results give a lower bound for the number of rounds used to construct this hash function.

Other kinds of Feistel Schemes are used for well known block ciphers. For example, BEAR and LION [2] are two block ciphers which employ both expanding and contracting unbalanced Feistel networks. The AES-candidate MARS is also using a similar structure.

Attacks on Unbalanced Feistel Schemes with expanding functions have been previously studied by C.S. Jutla ([7]) and improved attacks were given in [18]. However some of the attacks presented in [18] need too many conditions on the internal variables. These attacks work, but with weak keys. In this paper, we make a systematic study of the equations between the internal variables to avoid unlikely collisions on the round functions. Thus we get additional conditions. Nevertheless, with more conditions, we show that it is still possible to attack the same number of rounds as in [18]. In Known Plaintext Attacks (KPA), we obtain the same complexity except for $d = 3k-1$ where our complexity is slightly greater than in [18] but we do not have too many conditions on the internal variables. For Non-Adaptive Chosen Plaintext Attacks (CPA-1), we give a general method to obtain CPA-1 from KPA. Then we get complexities that are, most of the time, better than the ones in [18]. We also show that the best CPA-1 are not derived from the best KPA. For $k \leq 7$, we have generated all the possible attacks, thus the attacks presented here are the best possible attacks. We believe that the generalization of these attacks for any k still gives the best possible attacks. We also provide simulation results for $k = 3$.

The paper is organized as follows. First we introduce some notation and definitions. Then we give an overview of the attacks. In Section 4, we show how we have generated all the possible attacks for $k \leq 7$. In Section 5, we introduce the different kinds of attacks we will used. These attacks named TWO, R1, R2, R3 and R4 generalize the attacks of [18]. Then in Section 6, we present R1, R2 KPA attacks. In Section 7, we show how to get CPA-1 from KPA. In Section 8, we study R1 and R2 CPA-1 and we give the results of our simulations. Finally, all the results are summarized in Section 9.

2 Notation

2.1 Unbalanced Feistel Schemes Notation

We first describe Unbalanced Feistel Scheme with Expanding Functions F_k^d and introduce some useful notations. F_k^d is a Feistel scheme of d rounds that produces a permutation from kn bits to kn bits. At each round j, we denote by f_j the round function from n bits to $(k-1)n$ bits. f_j is defined as $f_j = (f_j^{(1)}, f_j^{(2)}, \ldots, f_j^{(k-1)})$, where each function $f_j^{(i)}$ is defined from $\{0,1\}^n$ to $\{0,1\}^n$. On some input $[I_1, I_2, \ldots, I_k]$, F_k^d produces an output denoted by $[S_1, S_2, \ldots, S_k]$ by going through d rounds. At round j, the first n bits of the round entry are called X^{j-1}. We can notice that $I_1 = X^0$. We compute $f_j(X^{j-1})$ and obtain $(k-1)n$ bits. Those bits are xored to the $(k-1)n$ last bits of the round entry and the result is rotated by n bits.

The first round is represented on Figure 1 below:

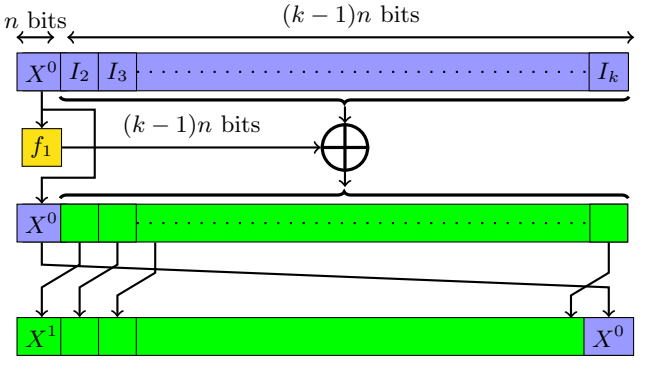


Fig. 1. First Round

We have

$$X^0 = I_1$$
$$X^1 = I_2 \oplus f_1^{(1)}(I_1)$$
$$X^2 = I_3 \oplus f_1^{(2)}(I_1) \oplus f_2^{(1)}(X^1)$$
$$X^3 = I_4 \oplus f_1^{(3)}(I_1) \oplus f_2^{(2)}(X^1) \oplus f_3^{(1)}(X^2)$$

More generally, we can express the X^j recursively:

$$\forall \xi < k,\ X^{\xi} = I_{\xi+1} \bigoplus_{i=1}^{\xi} f_i^{(\xi-i+1)}(X^{i-1})$$
$$\forall \xi \geq 0,\ X^{k+\xi} = X^{\xi} \bigoplus_{i=2}^{k} f_{\xi+i}^{(k-i+1)}(X^{\xi+i-1})$$

After d rounds ($d \geq k+1$), the output $[S_1, S_2, \ldots, S_k]$ can be expressed by using the introduced values X^j:

$$\begin{aligned}
S_k &= X^{d-1} \\
S_{k-1} &= X^{d-2} \oplus f_d^{(k-1)}(X^{d-1}) \\
S_{k-2} &= X^{d-3} \oplus f_{d-1}^{(k-1)}(X^{d-2}) \oplus f_d^{(k-2)}(X^{d-1}) \\
&\ldots \\
S_{\xi} &= X^{d-1-k+\xi} \bigoplus_{i=d-k+\xi}^{d-1} f_{i+1}^{(\xi+d-i-1)}(X^i) \\
&\ldots \\
S_1 &= X^{d-k} \bigoplus_{i=d-k+1}^{d-1} f_{i+1}^{(d-i)}(X^i)
\end{aligned}$$

We don't need another notation, but for a better understanding we introduce a notation for the intermediate values. After round p, we obtain $[M_1^p, M_2^p, \ldots, M_k^p]$. So we have $M_1^p = X^p$, and for all $i \in \{1, 2, \ldots, k\}$ $M_i^0 = I_i$ and $M_i^d = S_i$.

2.2 Differential Attack Notation

Our attacks use sets of points. A point is a plaintext/ciphertext pair. The total number of points gives us the complexity of the attack. From the set of points we extract all the φ-tuple of distinct points $P(1)$, $P(2)$ $\ldots P(\varphi)$, and we count how many φ-tuple verify some equalities (see Figure 2 for an example).

Now, we can describe an attack with a differential path. With the path we can explain why the number of φ-tuples that match the conditions is more important for a F_k^d scheme than for a random permutation. We introduce more definition.

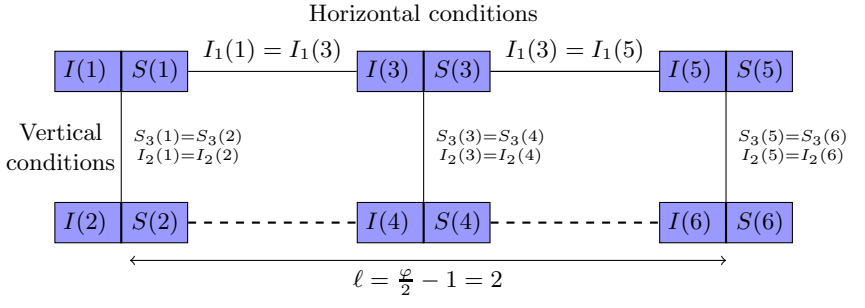


Fig. 2. Example of equalities for $\varphi = 6$

After p rounds, we define "horizontal equalities" on part M_i of the output M as $M_i^p(1) = M_i^p(3) = \ldots = M_i^p(\varphi - 1)$ and $M_i^p(2) = M_i^p(4) = \ldots = M_i^p(\varphi)$. Let $\ell = \frac{\varphi}{2} - 1$. "Vertical equalities" on part M_i are given by $\forall j,\ 0 \leq j \leq \ell$, $M_i^p(2j+1) = M_i^p(2j+2)$. We also define "differential equalities" on part M_i by $\forall j,\ 0 \leq j \leq \ell - 1$, $M_i^p(2j+1) \oplus M_i^p(2j+2) = M_i^p(2j+3) \oplus M_i^p(2j+4)$. Notice that when we have the differential equalities, in order to get the horizontal equalities, it is enough to have the first sequence of equalities, and for the vertical equalities, it is enough to get only the first one. When we impose some equalities, we call them **conditions** (they are satisfied with probability $\frac{1}{2^n}$). This may imply that other equalities will be satisfied with probability 1. On the input and output variables we will always have ℓ differential conditions and either horizontal or vertical conditions. On the internal variables, we will get horizontal or vertical equalities and moreover we will impose more vertical or horizontal conditions. We need to always have differential equalities. When we impose new conditions on the internal variables, we must check that we do not add too many of them. We now give an example with an attack over the F_3^6 scheme. See Table 1.

Table 1. F_3^6 attack

i (round)	$M_1^i(2j+1) \oplus M_1^i(2j+2)$	$M_2^i(2j+1) \oplus M_2^i(2j+2)$	$M_3^i(2j+1) \oplus M_3^i(2j+2)$
0	**0**	**0**	Δ_1
1	0	Δ_1	0
2	$\bullet\Delta_1$	$\bullet 0$	0
3	$.\Delta_2$	Δ_3	$.\Delta_1$
4	**0**	**.0**	$.\Delta_2$
5	0	Δ_2	0
6	Δ_2	0	0

The "." in this table means that there are horizontal equalities or conditions. The "0" in the table means that there are vertical equalities or conditions. This notation will be used for any attack. We can count the total number of conditions for the different part: $n_I = 3\ell + 2$ (number of input conditions), $n_X = 2\ell + 2$ (number of internal conditions), $n_S = 3\ell + 2$ (number of output conditions). If a φ-tuple follow the path, i.e. if it satisfies both the input and the internal conditions, then it will verify the output conditions. But there exist other ways to verify both these output conditions and the input conditions. So, we can prove that the number of φ-tuple will be greater for a F_k^d permutation.

3 Example: CPA-1 Attack on F_3^6

We present here a first example where we have obtained a new and better attack than previously known for F_3^6. In the next sections a complete analysis will be given for more general parameters. This attack is the one described in Table 1 with $\varphi = 4$ and so $\ell = 1$. Figure 3 illustrates this attack. It explains the terms of horizontal and vertical equalities. Moreover, conditions are represented by a solid edge and equalities that are automatically satisfied by a dotted edge.

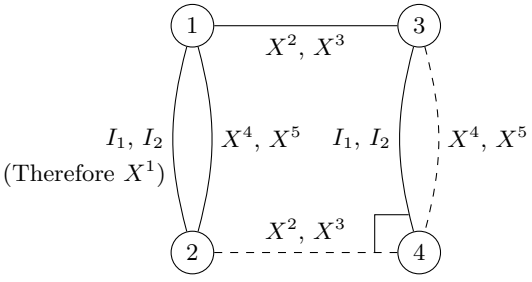


Fig. 3. The F_3^6 attack

We will generate all the possible messages $[I_1, I_2, I_3]$ such that $I_1 = 0$ and the first $n/2$ bits of I_2 are 0. So, we will generate exactly $m = 2^{3n/2}$ messages. How many 4-tuple of points will verify the input conditions ? For the first message we have m possibilities. For the second we have only 2^n possibilities because I_1 and I_2 are imposed by the first message. For the third point we have again m possibilities, and then we have no choice for the last point. Therefore there are $m^2 \times 2^n = 2^{4n}$ 4-tuple of points that satisfy all the input conditions. For a F_3^6 scheme, each of these tuple will satisfy at random the 4 internal conditions with a probability equal to $1/2^{4n}$. So, the expected number of 4-tuples that satisfy also the output conditions will be approximatively 1. Since there are 5 output conditions, the expected number of 4-tuple that satisfy the input conditions and the output conditions will be much lower for a random permutation. So, this CPA-1 attack will succeed with a high probability. We have found here a CPA-1 attack with $O(2^{3n/2})$ complexity and $O(2^{3n/2})$ messages. This is better than the $O(2^{5n/3})$ found in [18]. To find this complexity we can also use Table 3 with $r = 2$, $n_X = 4$, $\ell = 1$ and $k = 3$.

Moreover we have checked that all the other path conditions are verified (see Section 4) and this attack has been simulated by computer. For example, with $n = 10$ and 1000 attacks, we were able to distinguish 575 F_3^6 schemes from a random permutation, so the percentage of success is about 57.5%.

4 Generation of All Possible Attacks for $k \leq 7$

In this section we describe the way we generate all the possible attacks for $k \leq 7$. First we choose a value for k, then we increase the value of d, beginning with $d = 1$, until we find no possible attacks. All the attacks (or sometimes only the best attacks when the number is too much important) are put in a specific file corresponding to the values of k and d.

To find an attack, we need to construct all the differential paths. There are two constraints for this construction:

- In the same round, it's not possible to have k vertical conditions, because it leads to a collision between the points, i.e. $P(1) = P(3) = \cdots = P(\varphi - 1)$ and $P(2) = P(4) = \cdots = P(\varphi)$.

– In the same round, it's not possible to have k horizontal conditions, because it also lead to a collision between the points, i.e. $P(1) = P(2)$ and $P(3) = P(4)$ and ... $P(\varphi - 1) = P(\varphi)$.

When the path is constructed, we look if the attack is valid. To be valid, an attack must overcome five constraints.

1. The complexity of the attack must be smaller than the total number of possible messages: $\frac{n_I + n_X}{2\ell + 2} \leq k$.
2. There must be less internal conditions than output conditions: $n_X \leq n_S$.
3. If $n_X = n_S$ then n_S must be different from the number of final consecutive vertical conditions in the output conditions. If not, it is easy to prove that the output conditions are completely equivalent to the internal conditions. So, the output conditions will not happen more often than for a random permutation.
4. The number of equalities inside the path must be smaller than the number of variables included in them. Moreover we do not consider equalities when a variable occurs only once for all the equalities.
   Let us take an example. The F_3^6 attack given in section 2.2. The equations are: $f_3^{(1)}(X_2 \oplus \Delta_1) \oplus f_3^{(1)}(X_2) = \Delta_2$, $f_3^{(2)}(X_2 \oplus \Delta_1) \oplus f_3^{(2)}(X_2) = \Delta_3$,$f_4^{(1)}(X_3 \oplus \Delta_2) \oplus f_4^{(1)}(X_3) = \Delta_3$, $f_4^{(2)}(X_3 \oplus \Delta_2) \oplus f_4^{(2)}(X_3) = \Delta_1$. We have 4 equations and 5 variables X_2, Δ_1, Δ_2, Δ_3, X_3. All the variables are used at least in 2 equalities, so we cannot simplify.
5. There is no bottleneck in the equalities, i.e. any subset of equalities must have a greater number of variables. If it is not the case, the attack will only work with very particular functions (weak keys). This last point is very difficult to carry out without the help of a computer.

Finally, all the possible attacks are sorted in function of their complexity (KPA or CPA-1). For example there is 71 different attacks on the F_3^6 scheme, and 20 attacks with a CPA-1 complexity equal to $2^{3n/2}$.

All possible attacks are given in an extended version of this paper. In the next sections, we generalize for any k the best attacks (KPA and CPA-1) obtained for $k \leq 7$.

5 Different Kinds of Attacks: TWO, R_1, R_2, $R3$ and $R4$

5.1 TWO Attacks

The TWO attack consists in using m plaintext/ciphertexts pairs and in counting the number $\mathcal{N}_{F_k^d}$ of couples of these pairs that satisfy the relations between the input and output variables. We then compare $\mathcal{N}_{F_k^d}$ with $\mathcal{N}_{perm}$ where $\mathcal{N}_{perm}$ is the number of couples of pairs for a random permutation instead of F_k^d. The attack is successful, i.e. we are able to distinguish F_k^d from a random permutation

if the difference $|E(\mathcal{N}_{F_k^d}) - E(\mathcal{N}_{perm})|$ is much larger than the standard deviation σ_{perm} and than the standard deviation $\sigma_{F_k^d}$, where E denotes the expectancy function.

These attacks give the best attacks from 1 round to $k+2$ rounds. They are studied in [18]. Their complexity is summarized in Section 9.

5.2 R1 Attacks

Here we have vertical conditions on the input and output variables. These attacks are more general than the attacks named R1 in [18] since we allow more vertical conditions on the input and output variables. These attacks were first described by Jutla ([7]). With our differential notation, we have:

	I_1	...	I_r	I_{r+1}	...	I_k		S_1	...	S_{k-v}	S_{k-v+1}	...	S_k
Round 0	0	...	0	Δ^0_{r+1}	...	Δ^0_k	Round d	Δ^d_1	...	Δ^d_{k-v}	0	...	0

Thus, $n_I = k\ell + r$, $n_X = t\ell + w$, $n_S = k\ell + v$. Here n_I denotes the conditions on the input variables. $\ell = \frac{\varphi}{2} - 1$. The number of vertical conditions on the input variables is r. n_X denotes the number of conditions on the internal variables. We use t for horizontal conditions and w for vertical conditions. Similarly, n_S and v denote respectively the number of conditions and the number of vertical conditions on the output variables. Then the number of rounds is given by $r + t + w$. When $n_X \leq n_S$, we can easily obtain a sufficient condition of success (without computing the standard deviation), since in that case we will have for most permutation about 2 times more solutions with F_k^d than with a random permutation. Here this gives the condition: $(k-t)\ell \geq w - v$. In order to avoid weak keys, the number of equations with the internal variables must be smaller than or equal to the number of internal variables. This condition was not always satisfied in [18]. For R1 attacks, it is easy to check that the number of equations is given by $t(k-1)$ and the number of variables is $k(t+1) - r - w$. Thus we get the condition: $r + w \leq t + k$. The complexity of such an attack is $2^{\frac{n_I + n_X}{\varphi} n}$. This implies $\frac{n_I + n_X}{\varphi} \leq k$, i.e. $\frac{(k+t)\ell + r + w}{2\ell + 2} \leq k$.

5.3 R2 Attacks

Here we have horizontal conditions on the input variables and vertical conditions on the output variables. Again these attacks are more general than the attacks named R2 in [18] since we allow more horizontal conditions on the input variables and more vertical conditions on the output variables. We have:

	I_1	...	I_u	I_{u+1}	...	I_k		S_1	...	S_{k-v}	S_{k-v+1}	...	S_k
Round 0	Δ^0_1	...	Δ^0_u	Δ^0_{u+1}	...	Δ^0_k	Round d	Δ^d_1	...	Δ^d_{k-v}	0	...	0

Thus, $n_I = (k+u)\ell$, $n_X = t\ell + w$, $n_S = k\ell + v$. The number of horizontal conditions on the input variables is denoted by u. The number of rounds is given by $u + t + w$. The condition $n_X \leq n_S$ is equivalent to $(k-t)\ell \geq w - v$.

For $R2$ attacks, it is easy to check that the number of equations is given by $(t+1)(k-1)$ and the number of variables is $k(t+2)-w$. Thus we get the condition: $w \leq t+k+1$. The complexity of such an attack is $2^{\frac{n_I+n_X}{\varphi}n}$. This implies $\frac{n_I+n_X}{\varphi} \leq k$, i.e. $\frac{(k+t+u)\ell+w}{2\ell+2} \leq k$.

5.4 R3 and R4 Attacks

We describe briefly, R3 and R4 attacks. It is easy to get the number of rounds and the conditions on the number of equations and variables.

For R3 attacks, we have vertical conditions on the input variables and horizontal conditions on the output variables. This gives:

	I_1	$\ldots$	I_r	I_{r+1}	$\ldots$	I_k		S_1	$\ldots$	S_{k-s}	S_{k-s+1}	$\ldots$	S_k
Round 0	0	$\ldots$	0	Δ^0_{r+1}	$\ldots$	Δ^0_k	Round d	Δ^d_1	$\ldots$	Δ^d_{k-s}	$.\Delta^d_{k-s+1}$	$\ldots$	$.\Delta^d_k$

and $n_I = k\ell + r$, $n_X = t\ell + w$, $n_S = (k+s)\ell$.
For R4 attacks, we have horizontal conditions on the input and output variables. This gives:

	I_1	$\ldots$	I_u	I_{u+1}	$\ldots$	I_k		S_1	$\ldots$	S_{k-s}	S_{k-s+1}	$\ldots$	S_k
Round 0	$.\Delta^0_1$	$\ldots$	$.\Delta^0_u$	Δ^0_{u+1}	$\ldots$	Δ^0_k	Round d	Δ^d_1	$\ldots$	Δ^d_{k-s}	$.\Delta^d_{k-s+1}$	$\ldots$	$.\Delta^d_k$

and $n_I = (k+u)\ell$, $n_X = t\ell + w$, $n_S = (k+s)\ell$.

6 Best KPA Attacks: R_1, R_2

In this section we describe the best attacks we have found. As mentioned before, we know that for $k \leq 7$, they are the best possible attacks. We will mostly describe one example of R2 attacks since for any round there are many possible R2 attacks that give the best complexity. It can be noticed that in KPA, there is a symmetry between R2 and R3 attacks. Thus there always exist R2 and R3 attacks with the same complexity. Sometimes, it is also possible to have R1 attacks. Most of the time, R4 attacks are worse. We give attacks from $k+3$ rounds to $3k-1$ rounds since from 1 to $k+2$ rounds, TWO attacks are most of the time better and they are described in [18]. In all our attacks, it is easily checked that the conditions given in the previous section are satisfied. Moreover, we always look for attacks where the number of points is minimum. Our best R2 KPA attacks are summarized in Table 2:

Remarks

1. We have the following R1 attacks:
   (a) When $k+3 \leq d \leq 2k-2$ and $d = k+2q$, we set

$$n_I = k\ell + k - 1, \quad n_X = q\ell + q + 1, \quad n_S = k\ell + q + 1$$

   It is possible to choose $\ell = 1$ and the complexity is also $2^{\frac{k+d}{4}n}$

Table 2. Best known KPA on F_k^d, for any $k \geq 3$

d values	n_I	n_X	n_S	ℓ	Complexity
$k+2q \in [k{+}3, 2k{-}2]$	$(2k{-}1)\ell$	$q\ell+q+1$	$k\ell{+}q{+}1$	1	$2^{\frac{k+d}{4}n}$
$k+2q+1 \in [k{+}3, 2k{-}2]$	$(2k{-}1)\ell$	$q\ell+q+2$	$k\ell{+}q{+}2$	1	$2^{\frac{k+d}{4}n}$
$k+2q \in [2k{-}1, 3k{-}2]$	$(2k{-}1)\ell$	$(q-\lfloor\frac{k-1}{2}\rfloor)\ell+q+\lfloor\frac{k+1}{2}\rfloor$	$k\ell{+}k{-}1$	1	$2^{\frac{k+d}{4}n}$
$k+2q+1 \in [2k{-}1, 3k{-}2]$	$(2k{-}1)\ell$	$(q-\lfloor\frac{k-3}{2}\rfloor)\ell+q+\lfloor\frac{k+1}{2}\rfloor$	$k\ell{+}k{-}1$	1	$2^{\frac{k+d}{4}n}$
$3k-1$	$k\ell{+}\ell$	$k\ell+2k-1$	$k\ell{+}k{-}1$	k	$2^{(k-\frac{1}{2k+2})n}$

(b) When $2k-1 \leq d \leq 3k-2$ and $d = k+2q$, we set

$$n_I = k\ell+2, \quad n_X = q\ell+k+q-2, \quad n_S = k\ell+k-1$$

The complexity is still $2^{\frac{k+d}{4}n}$, but ℓ is greater than 1.

2. In [7], Jutla gave a R1 attack on $3k-3$ rounds but the complexity that we obtain with a R2 attack here is better. It is possible to perform a R1 attack on $3k-2$ rounds just by adding a vertical condition on the input variables to the attack on $3k-3$ rounds and the we obtain the same complexity as the one we get with a R2 attack. Due to the conditions between the number of equations and internal variables, it is not possible to use the same idea for $3k-1$ rounds. In this last case, we have R2 (and of course R3) attacks.

7 Way to Transform KPA Attacks into CPA-1 Attacks

We have analyzed all the possible situations and we are now able to present formulas that give us directly the CPA complexity depending on the initial conditions. We call u the number of horizontal conditions, r the number of vertical condition and $\delta = |u-r|$. So we can distinguish four cases:

Case 1 $u = 0$: $\underbrace{0\ 0\ \ldots\ 0}_{r\text{ “0”}}\ \Delta_1\ \ldots\ \Delta_{k-r}$

Case 2 $r = 0$: $\overbrace{.\Delta_1\ .\Delta_2\ \ldots\ .\Delta_u}^{u\text{ “.”}}\ \Delta_{u+1}\ldots\ \Delta_{k-r}$

Case 3 $u \leq r$: $\underbrace{\overbrace{.0\ .0\ \ldots\ .0}^{u\text{ “.”}}\ \overbrace{0\ \ldots\ 0}^{\delta=r-u}}_{r\text{ “0”}}\ \Delta_1\ \ldots\ \Delta_{k-r},$

Case 4 $u \geq r$: $\underbrace{\overbrace{.0\ .0\ \ldots\ .0}^{u\text{ “.”}}\ .\underbrace{\Delta_1\ \ldots\ .\Delta_\delta}_{\delta=u-r}}_{r\text{ “0”}}\ \Delta_{\delta+1}\ldots\ \Delta_{k-r}$

We can notice that the best CPA-1 attacks do not always come from the best KPA attacks. Nevertheless, if we want to express the CPA complexity with the KPA complexity, we can use the following formula: $\log_{2^n}(KPA) = \dfrac{r+(u+k)\ell+n_X}{2\ell+2}$.

Table 3. KPA to CPA

Conditions			$\log_{2^n}(CPA)$
$u = 0$		$\frac{n_X}{\ell+2} \leq k - r$	$\frac{n_X}{\ell+2}$
r vertical conditions		$\frac{n_X}{\ell+2} > k - r$	$\frac{n_X - k + r}{\ell+1}$
$r = 0$		$\frac{n_X}{\ell+2} \leq k - u$	$\frac{n_X}{\ell+2}$
u horizontal conditions		$\frac{n_X}{\ell+2} > k - u$	$\frac{n_X - \ell(k-u)}{2}$
$r \neq 0$ and $u \neq 0$	$u \leq r$	$(\ell+2)(k-r) > n_X$	$\frac{n_X}{\ell+2}$
		$(\ell+2)(k-r) + (\ell+1)\delta > n_X$ and $(\ell+2)(k-r) \leq n_X$	$\frac{n_X - k + r}{\ell+1}$
		$(\ell+2)(k-r) + (\ell+1)\delta \leq n_X$	$n_X - k + r - \ell(k-u)$
u horizontal conditions and r vertical conditions	$u > r$	$(\ell+2)(k-u) > n_X$	$\frac{n_X}{\ell+2}$
		$(\ell+2)(k-u) + 2\delta > n_X$ and $(\ell+2)(k-u) \leq n_X$	$\frac{n_X - \ell(k-u)}{2}$
		$(\ell+2)(k-u) + 2\delta \leq n_X$	$n_X - k + r - \ell(k-u)$

For all the CPA-1 Attacks we found, we prove that the best choice is to keep the first bits constant and generate all the possible messages with the same first bits.

Let's show how we prove it for Case 1. The best way to choose messages is to keep some of the bits constant (for example equal to zero) and consider all the possible combination for the other bits. We call b the number of varying bits among the first rn bits, and we call β the number of varying bits among the last $(k-r)n$ bits. So we have $0 \leq b \leq rn$ and $0 \leq \beta \leq (k-r)n$, and this allow us to generate $2^{b+\beta}$ points (plaintext/cyphertext pair). Now we count how many φ-tuples $M^0(1), \ldots, M^0(\varphi)$ of points will verify the input conditions. For $M^0(1)$ we have $2^{b+\beta}$ possibilities, for $M^0(2)$ only $2^\beta - 1 \approx 2^\beta$ possibilities, because the first rn bits are imposed by $M^0(1)$. For $M^0(3)$ we have again $2^{b+\beta} - 2 \approx 2^{b+\beta}$ possibilities. For $M^0(4)$ only one possibility : $M^0(4) = M^0(3) \oplus (M^0(1) \oplus M^0(2))$. We continue like this until we reach the last two points. For $M^0(\varphi - 1)$ we have again almost $2^{b+\beta}$ possibilities, and for $M^0(\varphi)$ only one possibility. So, the total number of φ-tuples is $\left(2^{b+\beta}\right)^{\varphi/2} \times 2^\beta = 2^{(b+\beta)(\ell+1)+\beta}$. The complexity of the CPA-1 is equal to $2^{b+\beta}$. We want this number to be as small as possible, and at the same time we want to generate a maximum of φ-tuples that satisfy the input conditions. So, we want to have β as large as possible. Each φ-tuple has a

probability equal to $1/2^{n_X \cdot n}$ to satisfy the internal conditions. In order to have a reasonable chance to realize these conditions, we must have $(b+\beta)(\ell+1)+\beta = n_X \cdot n$. If $b = 0$ we get $\beta = \frac{n_X}{\ell+2}n$. But this is possible only if $\beta \leq (k-r)n$, i.e. if $\frac{n_X}{\ell+2} \leq k-r$. If we have $\frac{n_X}{\ell+2} > k-r$ then we must take the maximum possible value for β: $\beta = (k-r)n$ and that gives us a CPA-1 complexity equal to $2^{b+\beta} = 2^{\frac{n_X-k+r}{\ell+1}n}$.

All the cases are summarized in Table 3.

8 Best CPA-1 Attacks: R_1, R_2, Simulation

8.1 CPA-1 Attacks

In this section, we describe the best CPA-1 that we have obtained. Again for $k \leq 7$ we know that we have the best possible attacks. Except for $3k-1$ rounds, we obtain a better complexity than in [18]. The best CPA-1 are generally R2 attacks. Sometimes R1 attacks exist with the same complexity. It is interesting to note that the best CPA-1 do not come from the best KPA. We will use the study of CPA-1 made in Section 7. We will describe CPA-1 for $k+3 \leq d \leq 3k-1$ since for $d \leq k+2$, the best attacks are the TWO attacks given in [18]. Again we will give an example of such an attack for each round. We notice that for the same conditions on the input and output variables, we can find several attacks: the horizontal and vertical conditions on the internal variables can be displayed differently inside the attack, but we must respect the conditions between the number of equations and variables at each step of the attack. An example is given at the end of this section. Our best R2 CPA-1 attacks are summarized in the following table:

Table 4. Best known CPA-1 on F_k^d, for any $k \geq 3$

d values	n_I	n_X	n_S	ℓ	Complexity
$k+3$	$k\ell+(k-1)\ell$	$\ell+3$	$k\ell+1$	1	$2^{\frac{3n}{2}}$
$k+4$	$k\ell+(k-2)\ell$	$2\ell+4$	$k\ell+2$	1	2^{2n}
$k+5$	$k\ell+(k-2)\ell$	$2\ell+5$	$k\ell+2$	1	$2^{5n/2}$
$k+2q \in [k+6, 3k-4]$	$k\ell+(k-q)\ell$	$(q-1)\ell+2q+1$	$k\ell+1$	$q-1$	$2^{\frac{q^2+2}{q+1}n}$
$k+2q+1 \in [k+7, 3k-5]$	$k\ell+(k-q)\ell$	$(q-1)\ell+2q+2$	$k\ell+1$	$q-1$	$2^{\frac{q^2+3}{q+1}n}$
$3k-3$	$k\ell+\ell$	$(k-2)\ell+2k-2$	$k\ell+1$	$k-1$	$2^{\frac{(k-1)k}{k+1}n}$
$3k-2$	$k\ell+(k-1)\ell$	$(k-\lfloor\frac{k}{2}\rfloor)\ell+2k-\lfloor\frac{k-1}{2}\rfloor$	$k\ell+k-1$	1	$2^{(k-1)n}$
$3k-1$	$k\ell+\ell$	$(k-1)\ell+2k-1$	$k\ell+k-1$	k	$2^{(k-\frac{1}{2})n}$

Remark. For $d = k+3, k+4, k+5$ and $3k-2$, there exist R1 attacks with the same complexity and the same number of points.

8.2 Overview of the R2 CPA-1 Attack on F_k^{3k-1}

We did a simulation of our best CPA-1 Attack. The input and output conditions were the following:

	I_1	I_2	...	I_k		S_1	S_2	...	S_k
Round 0	$.\Delta_1^0$	Δ_2^0	...	Δ_k^0	Round d	Δ_1^d	0	...	0

Several different differential paths match with these input and output conditions. For example let's see all the R2 path for the F_3^8 and F_4^{11}permutations. See Table 5 and Table 9 in Appendix A.

We counted the number of paths for $k \leq 7$:

k	3	4	5	6	7
# path	2	8	27	89	296

We will see that, the greater k is, the better the attacks work.

Table 5. All the paths for the R2 attack against F_3^8, $\varphi = 8$

	0	$.\Delta_1$	Δ_2	Δ_3		0	$.\Delta_1$	Δ_2	Δ_3
	1	0	0	$.\Delta_1$		1	0	Δ_4	$.\Delta_1$
	2	0	Δ_1	0		2	$.\Delta_4$	Δ_1	0
	3	$.\Delta_1$	0	0		3	0	0	$.\Delta_4$
Path 1:	4	0	Δ_4	$.\Delta_1$	Path 2:	4	0	Δ_4	0
	5	$.\Delta_4$	Δ_1	0		5	$.\Delta_4$	0	0
	6	0	0	$.\Delta_4$		6	0	0	$.\Delta_4$
	7	0	Δ_4	0		7	0	Δ_4	0
	8	Δ_4	0	0		8	Δ_4	0	0

Table 6. Experimental results for F_k^{3k-1}

n	k	kn	% of success	% of false alarm	# iteration
2	3	6	29,09%	0,35%	100000
2	4	8	61,6%	0,06%	10000
2	5	10	98,37%	0%	10000
2	6	12	99,99%	0%	10000
2	7	14	100%	0%	10000
2	8	16	100%	0%	1000
2	9	18	100%	0%	500
2	10	20	100%	0%	100
4	3	12	21,15%	1,12%	10000
4	4	16	42,5%	0%	1000
4	5	20	93%	0%	100
4	6	24	100%	0%	100
6	3	18	8%	1,2%	500
8	3	24	2%	0%	100

8.3 Experimental Results

We did simulations of these CPA-1 attacks. For each simulation, we generate a random Feistel scheme with 20 rounds, and a F_k^{3k-1} scheme. For both schemes, we compute $2^{(k-1/2)n}$ ciphertext/plaintext pairs, by varying only the last $(k-1/2)n$ bits. After this, we extract all the couples of points that satisfy both input and output conditions. We sort these couples of points in order to count how many φ-tuples of points match the input and output condition. If we found q couples of points that satisfy all these conditions with $q \geq \varphi/2$, we count as if we have found $\frac{q!}{(q-\varphi/2)!}$ φ-tuples, because this is the number of φ-tuples we can take out these points, by changing the position of the couple of points. Once this is finished, we compare the number found for each permutation. Most of the time, that enables us to distinguish between them. See Table 6.

9 Summary of the Attacks

In Tables 7 and 8, we give the complexity of the attacks we have found. For $k \leq 7$, since we have generated all the attacks, these are the best possible attacks. Then we have generalized the results for $k > 7$ and we believe that the attacks presented here are also the best possible attacks. For $d \leq k+2$, we have TWO attacks. For $d \geq k+3$, we have rectangle attacks. As mentioned before, in KPA, there are always R2 and R3 attacks that give the best complexity sometimes there is also a R1 attacks (for $3k-2$ rounds for example). In CPA-1, the best complexity is given by R2 attacks, and sometimes R1 attacks.

Table 7. Best known TWO and Rectangle attacks on F_3^d. Details about the parameters in this table: (new) means that we have found a better attack than previously known.

	KPA	CPA-1
F_3^1	1	1
F_3^2	$2^{\frac{n}{2}}$, TWO	2
F_3^3	2^n, TWO	2
F_3^4	$2^{\frac{3}{2}n}$, TWO	$2^{\frac{n}{2}}$, TWO
F_3^5	2^{2n}, TWO	2^n, TWO
F_3^6	$2^{\frac{9}{4}n}$, R2, R3	$2^{\frac{3}{2}n}$, R2 (new)
F_3^7	$2^{\frac{5}{2}n}$, R1, R2, R3	2^{2n}, R2
F_3^8	$2^{\frac{23}{8}n}$, R2, R3	$2^{\frac{5}{2}n}$, R2

In these tables, "new" means that the complexity that we obtain is better than the complexity given in [18]. $(*)$ means that for $3k-1$ rounds our complexity is worse than the complexity in [18]. This comes from the fact, as we mentioned earlier, that the conditions between the equations and the internal variables were not all considered in [18].

Table 8. Best known TWO and Rectangle attacks on F_k^d, for any $k \geq 3$. Details about the parameters in this table: (new) means that we have found a better attack than previously know.

	KPA	CPA-1
F_k^1	1	1
F_k^2	$2^{\frac{n}{2}}$, TWO	2
F_k^3	2^n, TWO	2
F_k^d, $2 \leq d \leq k$	$2^{\frac{d-1}{2}n}$,TWO	2
F_k^{k+1}	$2^{\frac{k}{2}n}$, TWO	$2^{\frac{n}{2}}$, TWO
F_k^{k+2}	$2^{\frac{k+1}{2}n}$, TWO	2^n, TWO
F_k^{k+3}	$2^{\frac{2k+3}{4}n}$, R2, R3	$2^{3n/2}$, R2 (new)
F_k^{k+4}	$2^{\frac{k+2}{2}n}$, R1, R2, R3	2^{2n}, R2 (new)
F_k^{k+5}	$2^{\frac{2k+5}{4}n}$, R2, R3	$2^{5n/2}$, R2 (new)
$\vdots$	$\vdots$	$\vdots$
F_k^d, d= $k+2q$, $3 \leq q \leq k-2$	$2^{\frac{d+k}{4}n}$, R1, R2, R3	$2^{\frac{q^2+2}{q+1}n}$, R2 (new)
F_k^d, d= $k+2q+1$, $3 \leq q \leq k-3$	$2^{\frac{d+k}{4}n}$, R2, R3	$2^{\frac{q^2+3}{q+1}n}$, R2 (new)
$\vdots$	$\vdots$	$\vdots$
F_k^{3k-3}	$2^{(k-\frac{3}{4})n}$, R2, R3	$2^{\frac{(k-1)k}{k+1}n}$, R2 (new)
F_k^{3k-2}	$2^{(k-\frac{1}{2})n}$, R1, R2, R3	$2^{(k-1)n}$, R2 (new)
F_k^{3k-1}	$2^{(k-\frac{1}{2k+2})n}$, R2, R3, (*)	$2^{(k-\frac{1}{2})n}$, R2

10 Conclusion

In this paper we make a systematic study of rectangle generic attacks on unbalanced Feistel schemes with expanding functions. Although these attacks were already analyzed in [7] and [18], this paper brings many improvements. Generation of all possible rectangle attacks for $k \leq 7$ was performed thanks to a computer program and the most efficient ones were selected. Then the generalization for any k was possible. This gives attacks for which conditions between equations and internal variables are satisfied. This was not detected in [18]. We also provide a complete description of the way to obtain CPA-1 from KPA. This

shows how to get the best CPA-1 and we improved the CPA-1 complexity of [18]. Also many simulations confirm our theoretical results.

There are still some open problems. It would be interesting to complete the program in order to generate all the attacks for any k. This seems to be a memory space problem. Also, in this paper, we did not study attacks with complexity greater than kn. In that case, we need to attack permutations generators and not only one single permutation. In [18], attacks called "multi-rectangle attacks" were introduced, but so far no significant results have been obtained on these attacks. It might give a new way to study generic attacks on unbalanced Feistel schemes with expanding functions. As we mentioned in Section 3, when we have exactly the same condition on the input and output variables, there are many possible CPA-1 attacks (for $k = 7$, there exist 286 attacks on F_7^{20}, with the same conditions on the input and output variables). An estimation for any k will strengthen the attack.

References

1. Aiello, W., Venkatesan, R.: Foiling Birthday Attacks in Length-Doubling Transformations - Benes: A Non-Reversible Alternative to Feistel. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 307–320. Springer, Heidelberg (1996)
2. Anderson, R.J., Biham, E.: Two Practical and Provably Secure Block Ciphers: BEARS and LION. In: Gollmann, D. (ed.) FSE 1996. LNCS, vol. 1039, pp. 113–120. Springer, Heidelberg (1996)
3. Coppersmith, D.: Another Birthday Attack. In: Williams, H.C. (ed.) CRYPTO 1985. LNCS, vol. 218, pp. 14–17. Springer, Heidelberg (1986)
4. Coppersmith, D.: Luby-Rackoff: Four rounds is not enough. Technical Report RC20674, IBM Research Report (December 1996)
5. Girault, M., Cohen, R., Campana, M.: A Generalized Birthday Attack. In: Guenther, C.G. (ed.) EUROCRYPT 1988. LNCS, vol. 330, pp. 129–156. Springer, Heidelberg (1988)
6. Goubin, L., Ivasco, M., Jalby, W., Ly, O., Nachef, V., Patarin, J., Treger, J., Volte, E.: CRUNCH. Technical report, Submission to NIST (October 2008)
7. Jutla, C.S.: Generalized Birthday Attacks on Unbalanced Feistel Networks. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 186–199. Springer, Heidelberg (1998)
8. Knudsen, L.R.: DEAL - A 128-bit Block Cipher. Technical Report 151, University of Bergen, Department of Informatics, Norway (February 1998)
9. Knudsen, L.R., Lai, X., Preneel, B.: Attacks on Fast Double Block Length Hash Functions. J. Cryptology 11(1), 59–72 (1998)
10. Knudsen, L.R., Rijmen, V.: On the Decorrelated Fast Cipher (DFC) and Its Theory. In: Knudsen, L.R. (ed.) FSE 1999. LNCS, vol. 1636, pp. 81–94. Springer, Heidelberg (1999)
11. Luby, M.: Pseudorandomness and Cryptographic Applications. Princeton University Press, Princeton (1996)
12. Luby, M., Rackoff, C.: How to Construct Pseudorandom Permutations from Pseudorandom Functions. SIAM J. Comput. 17(2), 373–386 (1988)
13. Naor, M., Reingold, O.: On the Construction of Pseudorandom Permutations: Luby-Rackoff Revisited. J. Cryptology 12(1), 29–66 (1999)

14. Patarin, J.: New Results on Pseudorandom Permutation Generators Based on the DES Scheme. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 301–312. Springer, Heidelberg (1992)
15. Patarin, J.: Generic Attacks on Feistel Schemes. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 222–238. Springer, Heidelberg (2001)
16. Patarin, J.: Security of Random Feistel Schemes with 5 or More Rounds. In: Franklin, M.K. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 106–122. Springer, Heidelberg (2004)
17. Patarin, J., Nachef, V., Berbain, C.: Generic Attacks on Unbalanced Feistel Schemes with Contracting Functions. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 396–411. Springer, Heidelberg (2006)
18. Patarin, J., Nachef, V., Berbain, C.: Generic Attacks on Unbalanced Feistel Schemes with Expanding Functions. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 325–341. Springer, Heidelberg (2007)
19. Schneier, B., Kelsey, J.: Unbalanced Feistel Networks and Block Cipher Design. In: Gollmann, D. (ed.) FSE 1996. LNCS, vol. 1039, pp. 121–144. Springer, Heidelberg (1996)
20. Yun, A., Park, J.H., Lee, J.: Lai-Massey Scheme and Quasi-Feistel Networks. Cryptology ePrint archive: 2007/347: Listing for (2007)

A All the Paths for the R2 Attack against F_4^{11}, $\varphi = 10$

Table 9. All the paths for the R2 attack against F_4^{11}, $\varphi = 10$

0	$.\Delta_1$	Δ_2	Δ_3	Δ_4	$.\Delta_1$	Δ_2	Δ_3	Δ_4	$.\Delta_1$	Δ_2	Δ_3	Δ_4	$.\Delta_1$	Δ_2	Δ_3	Δ_4
1	0	Δ_5	Δ_6	$.\Delta_1$	0	Δ_5	Δ_6	$.\Delta_1$	0	Δ_5	Δ_6	$.\Delta_1$	0	0	Δ_5	$.\Delta_1$
2	$.\Delta_5$	Δ_6	Δ_1	0	$.\Delta_5$	Δ_6	Δ_1	0	$.\Delta_5$	Δ_6	Δ_1	0	0	Δ_5	Δ_1	0
3	0	0	Δ_7	$.\Delta_5$	0	Δ_7	Δ_8	$.\Delta_5$	0	0	0	$.\Delta_5$	$.\Delta_5$	Δ_1	0	0
4	0	Δ_7	Δ_5	0	$.\Delta_7$	Δ_8	Δ_5	0	0	0	Δ_5	0	0	Δ_6	Δ_7	$.\Delta_5$
5	$.\Delta_7$	Δ_5	0	0	0	0	Δ_9	$.\Delta_7$	0	Δ_5	0	0	$.\Delta_6$	Δ_7	Δ_5	0
6	0	Δ_8	Δ_9	$.\Delta_7$	0	Δ_9	Δ_7	0	$.\Delta_5$	.0	0	0	0	Δ_8	Δ_9	$.\Delta_6$
7	$.\Delta_8$	Δ_9	Δ_7	0	$.\Delta_9$	Δ_7	0	0	$.\Delta_7$	Δ_8	Δ_9	$.\Delta_5$	$.\Delta_8$	Δ_9	Δ_6	0
8	0	0	0	$.\Delta_8$	0	0	0	$.\Delta_9$	0	0	.0	$.\Delta_7$	0	0	0	$.\Delta_8$
9	0	0	Δ_8	0	0	0	Δ_9	0	0	0	Δ_7	0	0	0	Δ_8	0
10	0	Δ_8	0	0	0	Δ_9	0	0	0	Δ_7	0	0	0	Δ_8	0	0
11	Δ_8	0	0	0	Δ_9	0	0	0	Δ_7	0	0	0	Δ_8	0	0	0

0	$.\Delta_1$	Δ_2	Δ_3	Δ_4	$.\Delta_1$	Δ_2	Δ_3	Δ_4	$.\Delta_1$	Δ_2	Δ_3	Δ_4	$.\Delta_1$	Δ_2	Δ_3	Δ_4
1	0	0	0	$.\Delta_1$	0	0	0	$.\Delta_1$	0	0	Δ_5	$.\Delta_1$	0	0	Δ_5	$.\Delta_1$
2	0	0	Δ_1	0	0	0	Δ_1	0	0	Δ_5	Δ_1	0	0	Δ_5	Δ_1	0
3	0	Δ_1	0	0	0	Δ_1	0	0	$.\Delta_5$	$.\Delta_1$	0	0	$.\Delta_5$	Δ_1	0	0
4	$.\Delta_1$	.0	0	0	$.\Delta_1$	0	0	0	$.\Delta_6$	Δ_7	Δ_8	$.\Delta_5$	0	0	Δ_6	$.\Delta_5$
5	$.\Delta_5$	Δ_6	Δ_7	$.\Delta_1$	0	Δ_5	Δ_6	$.\Delta_1$	0	0	$.\Delta_9$	$.\Delta_6$	0	Δ_6	Δ_5	0
6	0	Δ_8	$.\Delta_9$	$.\Delta_5$	$.\Delta_5$	$.\Delta_6$	Δ_1	0	0	Δ_9	Δ_6	0	$.\Delta_6$	$.\Delta_5$	0	0
7	$.\Delta_8$	Δ_9	Δ_5	0	$.\Delta_7$	Δ_8	Δ_9	$.\Delta_5$	$.\Delta_9$	Δ_6	0	0	$.\Delta_7$	Δ_8	Δ_9	$.\Delta_6$
8	0	0	0	$.\Delta_8$	0	0	.0	$.\Delta_7$	0	0	0	$.\Delta_9$	0	0	.0	$.\Delta_7$
9	0	0	Δ_8	0	0	0	Δ_7	0	0	0	Δ_9	0	0	0	Δ_7	0
10	0	Δ_8	0	0	0	Δ_7	0	0	0	Δ_9	0	0	0	Δ_7	0	0
11	Δ_8	0	0	0	Δ_7	0	0	0	Δ_9	0	0	0	Δ_7	0	0	0

The World Is Not Enough: Another Look on Second-Order DPA

François-Xavier Standaert[1], Nicolas Veyrat-Charvillon[1], Elisabeth Oswald[2], Benedikt Gierlichs[3], Marcel Medwed[4], Markus Kasper[5], Stefan Mangard[6]

[1] Université catholique de Louvain, Crypto Group, Belgium
[2] University of Bristol, Department of Computer Science, UK
[3] K.U. Leuven, ESAT/SCD-COSIC and IBBT, Belgium
[4] Graz University of Technology, IAIK, Austria
[5] Ruhr University Bochum, Horst Görtz Institute for IT Security, Germany
[6] Infineon Technologies AG, Security Innovation, Germany

Abstract. In a recent work, Mangard *et al.* showed that under certain assumptions, the (so-called) standard univariate side-channel attacks using a distance-of-means test, correlation analysis and Gaussian templates are essentially equivalent. In this paper, we show that in the context of multivariate attacks against masked implementations, this conclusion does not hold anymore. While a single distinguisher can be used to compare the susceptibility of different unprotected devices to first-order DPA, understanding second-order attacks requires to carefully investigate the information leakages and the adversaries exploiting these leakages, separately. Using a framework put forward by Standaert *et al.* at Eurocrypt 2009, we provide the first analysis that explores these two topics in the case of a masked implementation exhibiting a Hamming weight leakage model. Our results lead to refined intuitions regarding the efficiency of various practically-relevant distinguishers. Further, we also investigate the case of second- and third-order masking (*i.e.* using three and four shares to represent one value). This evaluation confirms that higher-order masking only leads to significant security improvements if the secret sharing is combined with a sufficient amount of noise. Eventually, we show that an information theoretic analysis allows determining this necessary noise level, for different masking schemes and target security levels, with high accuracy and smaller data complexity than previous methods.

1 Introduction

Masking (as described, *e.g.* in [2,7,19]) is a very frequently considered solution to thwart side-channel attacks. The basic idea is to randomize all the sensitive variables during a cryptographic computation by splitting them into d shares. The value $d-1$ is usually denoted as the order of the masking scheme. As most countermeasures against side-channel attacks, masking does not totally prevent the leakages but it is expected to increase the difficulty of performing a successful key-recovery. For example, masking can be defeated because of technological issues such as glitches [9]. Alternatively, an adversary can always perform a

M. Abe (Ed.): ASIACRYPT 2010, LNCS 6477, pp. 112–129, 2010.

higher-order DPA (*e.g.* [10,13,23]) in which he "combines" the leakages corresponding to the d shares in order to extract key-dependent information. From a performance point of view, masking a block cipher implies significant performance overheads, because it requires to compute the encryption of the different shares separately. As a result, an important problem is to determine the exact security level that it provides in function of the order of the scheme $d - 1$.

In order to solve this problem, Prouff *et al.* proposed a comprehensive study of first-order masking (*i.e.* second-order power analysis) in [17]. In their paper, the two leakage samples corresponding to the different shares are first mingled with a combination function. Next, a (key-dependent) leakage model is used to predict the output of this function. Eventually, the combined physical leakages are compared with the key-dependent predictions, thanks to Pearson's correlation coefficient [1]. Different combination functions are analyzed regarding the efficiency of the resulting attacks, leading to the following conclusions:

1. For every device and combination function, an optimal prediction function (or model) can be exhibited, that leads to the best attack efficiency.
2. Following an analysis based on Pearson's coefficient and assuming a "Hamming weight leakage model", the "normalized product combining function" (both to be detailed in this paper) is the best available in the literature.

The first observation is in fact quite natural. Since every device is characterized by its leakage function, there is one optimal model to predict these leakages that perfectly captures their probability density function (pdf for short). And for every optimal model, there is one way to combine the leakage samples that leads to the best possible correlation. But the idea of *optimal combination function* also leads to a number of issues. On the one hand, as acknowledged by the authors of [17], their analysis is carried out for a fixed (Hamming weight) leakage function. Therefore, how the observations made in this context would be affected by a different leakage function is an open question. On the other hand, their analysis is also performed for a given statistical tool, *i.e.* Pearson's correlation coefficient. Hence, one can wonder about the extent to which this statistical tool is generic enough for evaluating second-order DPA.

This second question is particularly interesting in view of the recent results of [12]. This reference shows that in the context of (so-called) standard first-order DPA and when provided with the same leakage model, the most popular distinguishers such as using distance-of-means tests [8], correlation analysis and Gaussian templates [3] require approximately the same number of traces to extract keys. Differences observed in practice are only due to statistical artifacts. In addition, it is shown that the correlation coefficient can be related to the concept of conditional entropy which has been established as a measure for side-channel leakage in [20]. Therefore, a natural question is to ask if these observations still hold in the second-order case. For example, can the correlation coefficient be used to evaluate the information leakage of a masked implementation?

In this paper, we answer this question negatively. We show that second-order DPA attacks are a typical context in which the two parts of the framework for the analysis of side-channel key-recovery of Eurocrypt 2009 lead to different

intuitions. First, an information theoretic analysis measures the amount of leakage provided by the masked implementation. It quantifies its security limits and relates to the success rate of an adversary who can perfectly profile the leakage pdf. Second, a security analysis measures the efficiency of one particular distinguisher. By applying this framework, we exhibit refined intuitions regarding the behavior of different second-order DPA attacks and combination functions. We then discuss the impact of these observations in profiled and non-profiled attack scenarios and confirm our theoretical investigations with practical experiments. We note that our results do not contradict [17] but rather emphasize that a single distinguisher cannot capture all the specificities of a leakage function. Eventually, we extend our analysis towards higher-order masking. This allows us to confirm that, from an information theoretic point of view, increasing the number of shares in a masking scheme only leads to an improved physical security if a sufficient amount of noise is limiting the quality of the adversary's measurements [2]. Higher-order masking also provides a case for the information theoretic metric introduced in [20]. We show that this metric can be used to determine the exact amount of shares and noise required to reach a certain security level (against worst-case template attacks, exploiting intensively profiled leakage models), with smaller data complexity than previous methods.

Summarizing, first-order side-channel attacks are a quite simple context in which (under certain conditions) most popular distinguishers behave similarly, if they are fed with the same leakage models. As a consequence, it can be sound to use "one distinguisher for all" in this context. By contrast, second-order (or higher-order) DPA can be confronted with leakage probability distributions that can take very different forms (mixtures, typically). Hence, given a certain amount of information leaked by a masked implementation, and even if fed with the same leakage models (and combination functions), different statistical tools will take advantage of the key-dependencies in very different manners. In other words, depending on the devices and countermeasures, one or another attack may perform better, hence suggesting our title "the world is not enough".

2 Boolean Masking and Second-Order Attacks

Many different masking schemes have been proposed in the literature. Although they can result in significantly different performances, the application of second-order attacks generally relies on the same principles, independent of the type of masking. In the following, we decided to focus on the Generalized Look Up Table (GLUT for short) that is described, *e.g.* in [16]. Such a scheme is represented in the lower left part of Figure 1, using the key addition and S-box layer of a block cipher as a concrete example. It can be explained as follows. For an input plaintext x_i, a random mask a_i is first generated within the device. The value $x_i \oplus a_i$ is generally denoted as the masked variable. Then, the encryption algorithm (here, the key addition and S-box) are applied to the masked variables, where s denotes a secret key byte (we will use the term subkey in the following). Concurrently, some correction terms are also computed such that anytime during

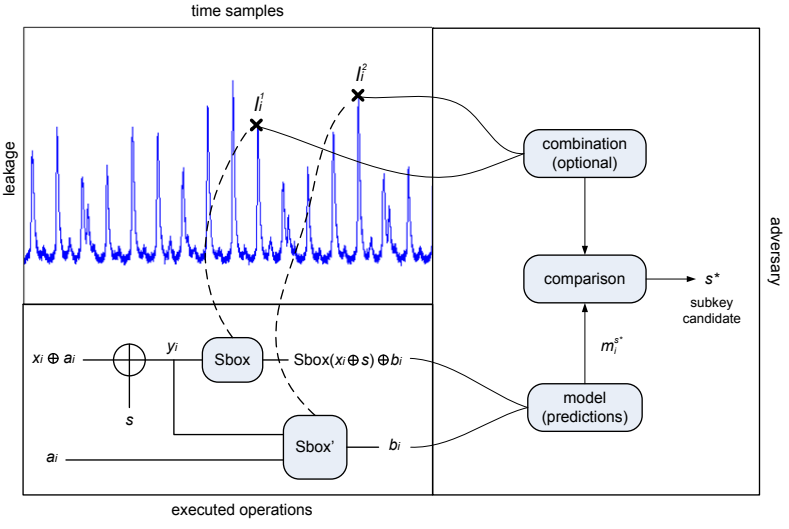


Fig. 1. Illustrative second-order DPA

the cryptographic computation, the XOR between a masked variable and its corresponding mask produces the original variable. In the case of the GLUT proposal, a precomputed function Sbox' is used for this purpose. For example in Figure 1, the masked S-box output $\mathsf{Sbox}(x_i \oplus a_i \oplus s)$ can be written as $\mathsf{Sbox}(x_i \oplus s) \oplus b_i$, where b_i denotes an output mask produced by Sbox'.

In practice, the GLUT countermeasure can be implemented in different manners. Mainly, the two S-box computations can be performed sequentially (as typical for software implementations) or in parallel (as typical for hardware implementations). In order to describe the second-order DPA that we investigate in this paper, we first use the sequential approach (the parallel one will be discussed in the next section). Also, we rely on the terminology introduced in [20]. Essentially, the idea of second-order DPA is to take advantage of the joint leakage of two intermediate computations during the encryption process (*i.e.* the masked value and its mask). In the software approach, the computation of these intermediate variables will typically be performed in two different clock cycles. Hence, two leakage samples l_i^1 and l_i^2 corresponding to these computations can be found in the leakage traces, as in the top of Figure 1. Following the standard DPA described in [12], the adversary will then work in three (plus one optional) steps:

1. For different plaintexts x_i and subkey candidates s^*, the adversary predicts some intermediate values in the target implementation. For example, one could predict the S-box outputs $\mathsf{Sbox}(x_i \oplus s)$ in Figure 1.
2. For each of these predicted values, the adversary models the leakages. Because of the presence of a mask in the implementation, this prediction can use a pdf (where the probability is taken over the masks and leakage noise) or some simpler function *e.g.* capturing only certain moments of this pdf.
3. Optionally, the adversary combines the leakage samples into a single variable.

4. For each subkey candidate s^*, the adversary finally compares the modeled leakages with actual measurements, produced with the same plaintexts x_i and a secret subkey s. In a second-order DPA, each model is compared with two samples in the traces. This comparison is independent of all other points. Consequently, these attacks are referred to as *bivariate*. In practice, this comparison is applied to many pairs of points in the leakage traces and the subkey candidate that performs best is selected by the adversary.

As for the analysis of first-order attacks, comparing different distinguishers requires to provide them with the same leakage samples. However, contrary to the first-order case and as will be discussed in the following sections, the best pair of leakage samples is not necessarily the same for all distinguishers. This is because different distinguishers can take advantage of different leakage pdf with different efficiencies in this case. In practice, this requires to test all pairs of samples in the traces (but this means $N(N-1)/2$ statistical tests to perform if the traces have N samples). In this paper, we will generally assume that this best pair of samples is provided to the attacks we perform (which can be done easily when simulating experiments and requires significant - but tractable - computational power when performing attacks based on real measurements).

Finally, we will use the following notations:

- $\mathbf{x}_q = [x_1, x_2, \ldots, x_q]$: a vector of plaintext bytes.
- $\mathbf{a}_q = [a_1, a_2, \ldots, a_q]$: a vector of random input mask bytes.
- $\mathbf{b}_q = [b_1, b_2, \ldots, b_q]$: a vector of random output mask bytes.
- $v_i^1 = \mathsf{Sbox}(x_i \oplus s) \oplus b_i$: an intermediate value in the encryption of x_i.
- $v_i^2 = b_i$: another intermediate value in the encryption of x_i.
- $\mathbf{l}_q^1 = [l_1^1, l_2^1, \ldots, l_q^1]$: a vector of leakage samples corresponding to the first intermediate values v_i^1 during the encryption process.
- $\mathbf{l}_q^2 = [l_1^2, l_2^2, \ldots, l_q^2]$: a vector of leakage samples corresponding to the second intermediate values v_i^2 during the encryption process.
- $\mathbf{m}_q^{s^*} = [m_1^{s^*}, m_2^{s^*}, \ldots, m_q^{s^*}]$: a vector containing leakage models (*i.e.* predictions) corresponding to a subkey candidate s^* and the plaintexts $\mathbf{x}_q$.

In the rest of the paper, these notations (in small caps) will represent sampled values, while their counterpart in capital letters will represent random variables.

3 Second-Order Attacks with Pearson's Coefficient

In theory, second-order DPA is possible if the joint probability distributions $\Pr[\mathbf{L}_q^1, \mathbf{L}_q^2 | \mathbf{X}_q, s]$ are different for different subkey values s. This can be illustrated, *e.g.* for a Hamming weight leakage function which is frequently considered in the practice of side-channel attacks [11] and has been the running example in [17]. It means assuming that the leakage samples l_i^1 and l_i^2 can be written as:

$$l_i^1 = \mathrm{W_H}(v_i^1) + n_i^1, \tag{1}$$

$$l_i^2 = \mathrm{W_H}(v_i^2) + n_i^2, \tag{2}$$

where $\mathrm{W_H}$ is the Hamming weight function and n_i^1, n_i^2 are normally distributed noise values with mean 0 and standard deviation σ_n. In the context of an 8-bit S-box (*e.g.* the AES one), it leads to 9 possible leakage distributions, corresponding to the 9 Hamming weight values of a secret state $\Sigma_i = \mathsf{Sbox}(x_i \oplus s)$, as observed in [21]. The left parts of Figures 11, 12 and 13 in Appendix A show the joint leakage distributions in this setting and clearly illustrate that they are key-dependent. As detailed in the previous section, taking advantage of these dependencies requires a comparison tool. In their statistical evaluation of second-order DPA, Prouff *et al.* use Pearson's correlation coefficient. In the context of first-order attacks exploiting a single leakage sample l_i, it implies computing:

$$\hat{\rho}(\mathbf{M}_q^{s^*}, \mathbf{L}_q) = \frac{\hat{\mathbf{E}}\Big(\big(l_i - \hat{\mathbf{E}}(\mathbf{L}_q)\big) \cdot \big(m_i^{s^*} - \hat{\mathbf{E}}(\mathbf{M}_q^{s^*})\big)\Big)}{\hat{\sigma}(\mathbf{L}_q) \cdot \hat{\sigma}(\mathbf{M}_q^{s^*})},$$

where $\hat{\mathbf{E}}$ and $\hat{\sigma}$ denote the sample means and standard deviations of a random variable, respectively. In order to extend this tool towards the second-order case, the classical approach is to first combine the two leakage samples l_i^1 and l_i^2 with a *combination function* C. For example, Chari *et al.* proposed to take the product of two centered samples [2]: $\mathsf{C}(l_i^1, l_i^2) = (l_i^1 - \hat{\mathbf{E}}(\mathbf{L}_q^1)) \cdot (l_i^2 - \hat{\mathbf{E}}(\mathbf{L}_q^2))$ and Messerges used the absolute difference between them [13]: $\mathsf{C}(l_i^1, l_i^2) = |l_i^1 - l_i^2|$. As illustrated in the right parts of Figures 11, 12 and 13, those combining functions also lead to key-dependencies. In addition to these standard examples, we finally plotted the distribution of the sum combining function $\mathsf{C}(l_i^1, l_i^2) = l_i^1 + l_i^2$ because it can be used to emulate the behavior of the GLUT masking in a hardware setting, where the two S-boxes of Figure 1 are computed in parallel.

3.1 Choice of a Model and Leakage-Dependency of C

Given the above descriptions and assuming that the adversary knows a good leakage model for the samples l_i^1 and l_i^2, it remains to determine which model to use when computing $\hat{\rho}(\mathbf{M}_q^{s^*}, \mathsf{C}(\mathbf{L}_q^1, \mathbf{L}_q^2))$. That is, we do not need to predict the leakage samples separately, but their combination. In addition and contrary to the first-order case, there is an additional variable (*i.e.* the mask) that is unknown to the adversary. But given a model for the separate samples, it is possible to derive one for their combination. For example, assuming a Hamming weight model that perfectly corresponds to the leakages of Equations (1) and (2), we can use the mean of the combination function, taken over the masks. For each subkey candidate s^*, the model is is then given by:

$$m_i^{s^*} = \mathop{\mathbf{E}}_{b_i}\Big(\mathsf{C}\big(\mathrm{W_H}(\Sigma_i^* \oplus b_i), \mathrm{W_H}(b_i)\big)\Big).$$

This is in fact similar to what is proposed in [17], where the mean is additionally taken over the leakage noise (which is more general, but implies additional profiling, *i.e.* a sufficiently precise knowledge of the noise distribution). As an illustration, Figure 2 shows the leakage models corresponding to the absolute

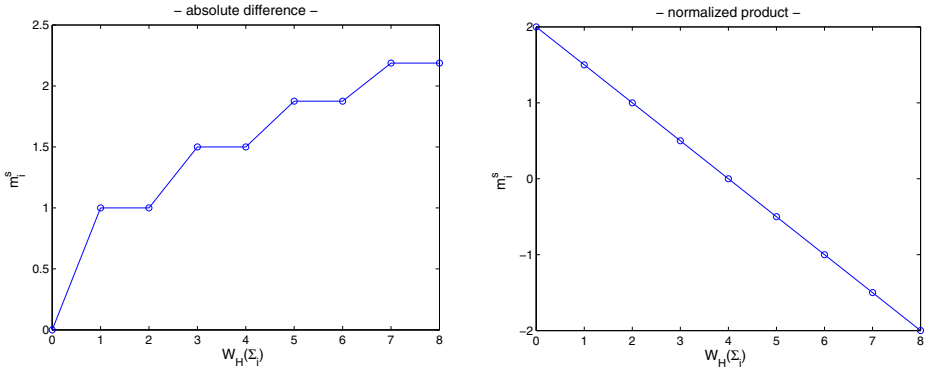


Fig. 2. Leakage models for second-order DPA using the correlation coefficient

difference and normalized product combination functions. They again only depend on the 9 Hamming weight values of the secret state, as opposed to the ones of a sum combining function for which the mean value (over the masks) is constant for all secret states. Hence, as already observed in [11], this sum combining function will not lead to successful second-order correlation attacks.

The figure intuitively confirms the previous theoretical analysis of Prouff *et al.* where it is demonstrated that the normalized product combining function leads to the most efficient second-order side-channel attacks when using Pearson's coefficient and assuming a Hamming weight leakage model for the separate samples. Indeed, this particular setting gives rise to nicely linear dependencies of the models m_i^s in the Hamming weight of the secret states $\mathrm{W_H}(\Sigma_i)$. Also, and contrary to the absolute difference combining function, all the 9 possible Hamming weights correspond to a different model m_i^s in this particular case.

Interestingly, the efficiency of the normalized product combining function can be simply explained when looking at the equations since it computes:

$$\hat{\rho}(\mathbf{M}_q^{s^*}, \mathsf{C}(\mathbf{L}_q^1, \mathbf{L}_q^2)) = \frac{\hat{\mathbf{E}}\Big(\big(\mathsf{C}(l_i^1, l_i^2) - \hat{\mathbf{E}}(\mathsf{C}(\mathbf{L}_q^1, \mathbf{L}_q^2))\big) \cdot \big(m_i^{s^*} - \hat{\mathbf{E}}(\mathbf{M}_q^{s^*})\big)\Big)}{\hat{\sigma}(\mathsf{C}(\mathbf{L}_q^1, \mathbf{L}_q^2)) \cdot \hat{\sigma}(\mathbf{M}_q^{s^*})}.$$

As the product is normalized, we have that $\hat{\mathbf{E}}(\mathsf{C}(\mathbf{L}_q^1, \mathbf{L}_q^2)) = 0$, which leads to:

$$\hat{\rho}(\mathbf{M}_q^{s^*}, \mathsf{C}(\mathbf{L}_q^1, \mathbf{L}_q^2)) = \frac{\hat{\mathbf{E}}\Big(\big(l_i^1 - \hat{\mathbf{E}}(\mathbf{L}_q^1)\big) \cdot \big(l_i^2 - \hat{\mathbf{E}}(\mathbf{L}_q^2)\big) \cdot \big(m_i^{s^*} - \hat{\mathbf{E}}(\mathbf{M}_q^{s^*})\big)\Big)}{\hat{\sigma}(\mathsf{C}(\mathbf{L}_q^1, \mathbf{L}_q^2)) \cdot \hat{\sigma}(\mathbf{M}_q^{s^*})}. \quad (3)$$

And this formula is in fact very close to the straightforward generalization of Pearson's correlation coefficient to the case of three random variables:

$$\hat{\rho}(\mathbf{M}_q^{s^*}, \mathbf{L}_q^1, \mathbf{L}_q^2) = \frac{\hat{\mathbf{E}}\Big(\big(l_i^1 - \hat{\mathbf{E}}(\mathbf{L}_q^1)\big) \cdot \big(l_i^2 - \hat{\mathbf{E}}(\mathbf{L}_q^2)\big) \cdot \big(m_i^{s^*} - \hat{\mathbf{E}}(\mathbf{M}_q^{s^*})\big)\Big)}{\hat{\sigma}(\mathbf{L}_q^1) \cdot \hat{\sigma}(\mathbf{L}_q^2) \cdot \hat{\sigma}(\mathbf{M}_q^{s^*})}. \quad (4)$$

The only difference between Equations (3) and (4) is in the leakage samples' standard deviation terms, which are key-independent. Hence, when applied to

the same pair of samples, attacks using Equations (3) or (4) are equivalent. Intuitively, these equations provide a simple explanation of the normalized product combining function. That is, such a combining function will efficiently take advantage of pairs of leakage samples that are linearly correlated conditioned on the key. As illustrated in Figures 11, 12 and 13, this is nicely achieved in the case of a Hamming weight leakage function for the two samples l_i^1 and l_i^2.

4 Evaluating Second-Order Leakage: IT Analysis

In general, the evaluation of second-order side-channel attacks is not straightforward to capture. More precisely, it is easy to see that an analysis based only on the correlation coefficient may suffer from certain limitations. For example:

- Given Pearson's correlation coefficient as a distinguisher and a Hamming weight leakage function, there exist (trivial) combination functions for the samples (*e.g.* the sum) that do not lead to successful key recoveries.
- Given Pearson's coefficient as a distinguisher and the normalized product combination function, there exist leakage functions (*e.g.* with no linear dependencies between the samples) that don't lead to successful key recoveries.

These observations suggest that the simple situation in the first-order context, where the correlation coefficient could (under certain physical assumptions detailed in [12]) be used both as a distinguisher and as a measure of side-channel leakage, does not hold here. In second-order side-channel attacks, this correlation is only a distinguisher. Hence, it is a typical context in which the evaluation framework of Eurocrypt 2009 is interesting to put into practice:

1. First, an information theoretic analysis is performed, in order to evaluate the physical leakages, independently of the adversary who exploits them. When applied to a countermeasure (*e.g.* masking), this step allows to quantify how much the security of the device has been improved against an adversary who can perfectly profile the leakage pdf. In other words, it can be used as an objective measure of the quality of the countermeasure, in a worst case scenario (*i.e.* best adversary, large number of queries - see [20] for the details).
2. Second, a security analysis is performed, in order to evaluate how efficiently a particular distinguisher (*e.g.* Pearson's correlation coefficient with a given combining function) can exploit the available leakage. This step is useful to translate the previous information theoretic analysis into a "number of measurements required to extract the key", in a given scenario.

In this section, we tackle the first part of the analysis. For this purpose, and in order to compare our conclusions with previous works, we use exactly the same assumptions as [17], *i.e.* a Hamming weight leakage function for the two samples, just as described in Section 3. Following the definitions in [20], we compute:

$$\mathrm{H}[S|\mathbf{L}_1^1, \mathbf{L}_1^2, \mathbf{X}_1] = -\sum_s \Pr[s] \sum_{x_1} \Pr[x_1] \int_{l_1^1} \int_{l_1^2} \Pr[l_1^1, l_1^2|s, x_1] \log_2 \Pr[s|l_1^1, l_1^2, x_1] \; dl_1^1 dl_1^2.$$

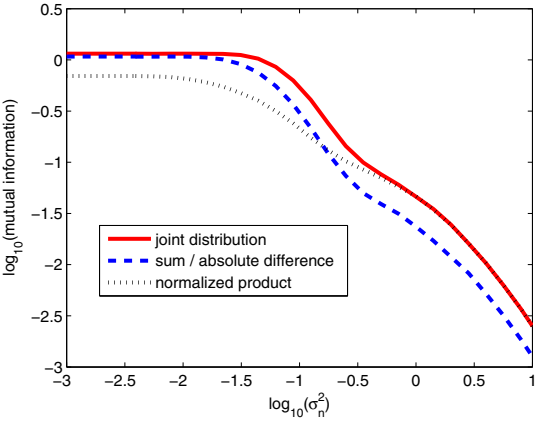


Fig. 3. Information leakage for different combination functions

Since the leakage samples are assumed to be normally distributed, this can be quite easily done in function of the noise standard deviation σ_n. Some simplifications allow to speed up the computations, *e.g.* by observing that only nine distributions are possible, corresponding to the nine Hamming weights of the secret states Σ_i. Also, in order to evaluate the information loss caused by the different combination functions, we similarly evaluated $\mathrm{H}[S|\mathsf{C}(\mathbf{L}_1^1, \mathbf{L}_1^2), \mathbf{X}_1]$. This implies slightly more complex integrals since, *e.g.* the product combining gives rise to mixtures of normal product distributions. Figure 9 in Appendix A illustrates these distributions for two secret states and two σ_n's. The mutual information values corresponding to these different information leakages (*i.e.* $\mathrm{I}(S; (\mathbf{L}_1^1, \mathbf{L}_1^2, \mathbf{X}_1)) = \mathrm{H}[S] - \mathrm{H}[S|\mathbf{L}_1^1, \mathbf{L}_1^2, \mathbf{X}_1]$) are then plotted in Figure 3, in function of the noise variance σ_n^2 (in log scale). From this figure, we can observe:

1. All combination functions imply a loss of information that can be avoided by dealing directly with the 2-dimensional joint leakage distribution.
2. The sum and absolute difference combining functions give rise to exactly the same information leakage. This can be understood from the shape of their distributions: the distribution of the absolute difference combining can be seen as the one of the sum combining that has been folded up.
3. For small σ_n^2, the normalized product is the least informative combining function. By contrast, when increasing the noise, the information leakage of the normalized product combining gets close to the one of the joint distribution.
4. The respective efficiency of different combining functions varies with the amount of noise. In particular, after a certain noise threshold, the product combining carries more information on S than the sum/absolute difference.

Note that the leakage of the sum combining's output clearly relates to the previous evaluation of [21] in which masking is analyzed in the hardware setting.

5 Implications for Profiled Attacks: Security Analysis (I)

The previous information theoretic analysis provides a new perspective to understand the relation between a masking scheme, its physical leakages and the exploitation of this information by a side-channel attack. For example, it exhibits that the sum combining function leads to significant information leakages (as can also be seen from the different pdf in appendix), although they cannot be directly exploited with Pearson's correlation coefficient. Previous works such as the one of Waddle and Wagner [23] showed how to overcome this limitation of the correlation coefficient, by squaring the combined samples. But our analysis raises the question whether these information leakages can be directly exploited (*i.e.* without squaring) by other distinguishers. In order to tackle this question, we apply the second part of the framework in [20], *i.e.* security analysis. This section starts with the evaluation of profiled (template) attacks, for which a strong relation with the previous information theoretic analysis should hold.

The results of various template attacks performed against the same masked AES S-box as in the previous sections are given in Figure 4, for two different noise standard deviations. We mention that these attacks do *not* use Gaussian templates as in [3] but the exact leakage distributions as in the previous information theoretic analysis (*e.g.* attacks using the joint distributions exploit Gaussian mixtures; attacks using the normalized product combining function exploit normal product distribution mixtures, *etc.* as plotted in appendix A). The different success rates are computed over 1000 independent experiments and nicely confirm the theoretical predictions of Theorem 2 in [20].

First, we see that the sum and absolute difference combining functions lead to the same attack efficiency in this profiled case (since their outputs lead to the same information leakages). Second, we see that the point in Figure 3 where the sum / absolute difference and the normalized product curves intersect is meaningful. Left of the intersection (*e.g.* for $\sigma_n = 0.25$), the sum / absolute difference combining functions allow more efficient attacks than the normalized product one. Right of the intersection (*e.g.* for $\sigma_n = 0.75$), the opposite conclusion holds.

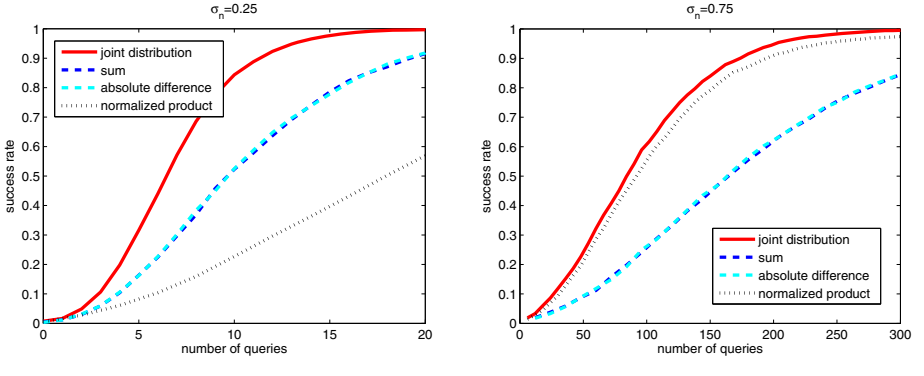


Fig. 4. Success rate of (simulated) profiled attacks against a masked AES S-box

And as shown in Appendix A, Figure 10, these attacks have a similar efficiency at the intersection that falls around $\sigma_n = 0.4$ (that is, $\log_{10}(\sigma_n^2) \approx -0.8$).

Of course, these experiments are partially artificial since in practice, an adversary who can profile the leakages will generally use the templates based on the joint distribution only. At least, this is the best strategy if the adversary has enough data and time to profile the multivariate leakage pdf. However, our results confirm that an information theoretic analysis provides an objective evaluation of the quality of a countermeasure against the "best-available" template adversaries in the DPA setting. Hence, they emphasize that such an analysis is an important part in the evaluation of side-channel countermeasures. Also, these results lead to the same conclusions as [14], and show that resistance against sufficiently profiled template attacks cannot be achieved by masking only.

6 Implications for Non-profiled Attacks: Security Analysis (II)

The previous section showed that for carefully profiled template attacks, there is a strong connection between the information leakage of a device and the success rate of the adversary. By contrast, we know that in the non-profiled context of correlation attacks, this observation does not hold in general. For example, Pearson's coefficient cannot be used to exploit the leakages corresponding to the sum combining of Section 3.1. Hence, it is natural to check whether there exist other non-profiled distinguishers that can be successful in this case. We answer this question positively, using the Mutual Information Analysis (MIA) introduced in [5]. It can be seen as the counterpart of template attacks, in which the leakage distributions are estimated "on-the-fly" rather than prior to the attacks.

The success rates of correlation and MIA attacks (here, and in the rest of the paper, computed over 500 independent experiments), using different combining functions, are given in Figure 5, again using the (simulated) setting described in the previous section. In our experiments, MIA estimates the pdf using histograms with N_b linearly-spaced bins, and N_b corresponding to the number of possible values for the models, as proposed in [5]. That is, we use 9 bins per leakage sample and we partition the leakage samples according to the 9 Hamming weights of the secret state Σ_i. The following observations can be emphasized:

1. In the low noise scenario, MIA with the sum and absolute difference combining functions works best, as similarly observed for template attacks.
2. By contrast, and contrary to template attacks, MIA without combining function (*i.e.* using the joint distribution directly, as in [6,18]), is not the most efficient solution in our simulations. This is caused by the need to estimate two-dimensional distributions, which turns out to require more data.
3. For similar reasons (*i.e.* also related to the different efficiency of the "on-the-fly" pdf estimation), when increasing the noise, MIA with the sum and absolute difference combining functions are not equivalent anymore.
4. Finally, attacks using Pearson's correlation coefficient perform well, specially when combined with the normalized product (which is natural since our simulated leakages perfectly fulfill the requirements of Section 3.1).

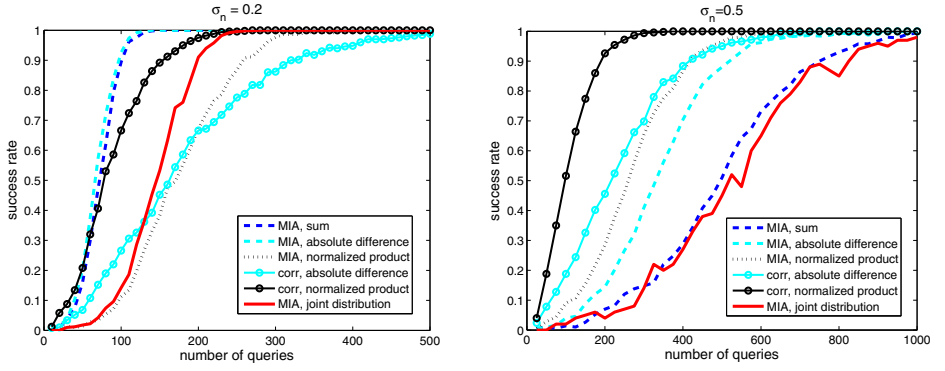


Fig. 5. Success rate of (simulated) non-profiled attacks - masked AES S-box

Importantly, we note that all these non-profiled distinguishers lead to significantly lower efficiencies than the profiled ones in the previous section.

7 Experimental Results

The previous sections evaluated the impact of masking an S-box with respect to various side-channel distinguishers, based on simulations. But as for most investigations in physically observable cryptography, it is important to verify that our conclusions are reasonably confirmed by practical measurements performed against a real chip. For this purpose, we also carried out a set of attacks against a masked implementation of the DES in an 8-bit RISC microcontroller from the Atmel AVR family. Considering the DES (rather than the AES) was motivated by practical facilities. Since the output of the DES S-box is 4-bit wide, it allows considering different contexts: in a first (low noise) scenario, the 4 remaining bits on the bus are kept constant; in a second scenario, these 4 bits are used in order to produce some additional algorithmic noise, by concatenating (secret) random strings to the two target values of Figure 1. This is interesting since the noise level was an important parameter, *e.g.* in our simulations of Figure 5. Hence, the different scenarios can be used to adapt the noise level in our experimental setting as well. The results in Figure 6 bring an interesting complement to our previous simulations and lead to the following observations:

1. The excellent efficiency of template attacks[1] and the good behavior of correlation attacks using the normalized product combining function are again exhibited. Interestingly, their respective efficiency gets closer when increasing the amount of algorithmic noise in the measurements, as it is suggested by the information theoretic analysis of Section 4.
2. MIA using the joint distribution is much more efficient than in the AES case. This is in fact related to the reduced number of bins that the 4-bit DES S-box allows in the pdf estimations (*i.e.* 25 rather than 81).

[1] We profiled our templates as described in the template-based DPA of [14].

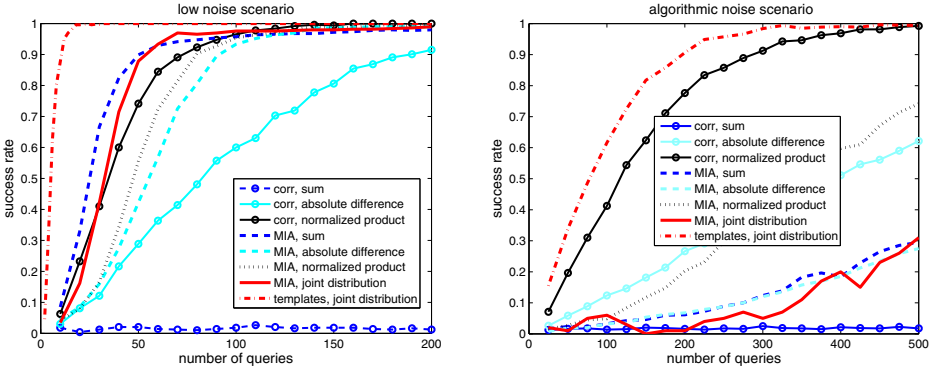


Fig. 6. Success rate of various experimental attacks against a masked DES

3. The presence of algorithmic noise (in the right part of Figure 6), affects the different distinguishers in a very different manner. To give a single example, MIA with the absolute difference combining function is strongly affected by this noise addition, compared to its counterpart using Pearson's coefficient.

Summarizing, these experiments confirm the "world is not enough" nature of second-order DPA that was already underlined in the previous simulations. The only strong statement that can be made in this context is that an information theoretic metric estimated with perfect templates captures the security against the best possible profiled adversary. As for all the other distinguishers, their efficiency highly depends on the actual shape of the leakage pdf and the engineering knowledge that can be exploited when mounting an attack. And contrary to the first-order case discussed in [12], the Gaussian assumption for the leakage samples does not hold anymore from the adversary's point of view (*e.g.* masking typically imply mixtures of Gaussians - or other - distributions).

8 Generalization to Higher-Orders

In order to improve the security of masking schemes further, one approach is to increase their order. For this purpose, this final section analyzes the cost *vs.* security tradeoff that can be obtained by generalizing the GLUT countermeasure in such a way, and details the second- and third-order cases for illustration. That is, rather than using one input mask per S-box, we now use two or three masks per S-box. In terms of cost, this implies using one or two additional tables Sbox'' and Sbox''', as described, *e.g.* in [15]. Conveniently, all the tools used in second-order DPA can be easily generalized to these third- and fourth-order attack cases. In particular, the information theoretic analysis of Section 4 just requires to integrate over three or four leakage samples l_i^1, l_i^2, l_i^3 and l_i^4.

The information leakage of these different masking schemes is represented in Figure 7, in function of the noise variance. On the same plot, we represented

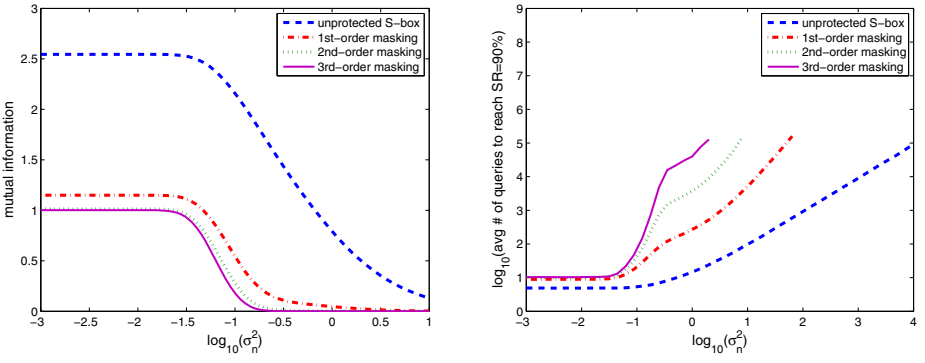


Fig. 7. Information leakage and success rates for 1^{st}, 2^{nd} and 3^{rd}-order masking

the average number of queries to the target device required for a perfectly profiled attack (similar to the ones in Section 5) to reach a success rate of 90%. These figures provide a quantitative insight to the observations in [2], where it is demonstrated that, given a large enough noise variance, the data complexity of a side-channel attack increases exponentially with the amount of shares in the masking scheme. That is, given a noise variance σ_n^2 in the leakage samples and k shares, the data complexity required to attack a masking scheme successfully is proportional to $(\sigma_n^2)^{k/2}$. The linear regions of the (log scale) curves that are observed in the right part of the figure suggest that this expectation is fulfilled in our experiments. Importantly, it also shows that the impact of (higher-order) masking can be extremely small in terms of security increases, for small σ_n^2's.

Note finally that these results give a practical counterpart to the recent theoretical analysis of [4], where it is shown that masking schemes based on secret sharing techniques lead to secure implementations if the number of shares is adjusted to be large enough with respect to the noise in the measurements.

8.1 A Case for the Information Theoretic Metric

Looking at Figure 7, the main question for a designer (or evaluation laboratory) is to best trade the amount of shares and the amount of noise that he has to add to his implementation, in order to reach a certain security level. This is essential since increasing these parameters has a strong impact on the performance of the implementation. Unfortunately, for high security levels, the proper estimation of the number of traces required to reach a certain success rate becomes intensive (because of statistical sampling issues). Already in simulations, running 1000 attacks, each of them using 10^5 queries, is time consuming. And when moving to the analysis of real traces (taking much more time to be generated and space to be stored), this limitation becomes even more critical. Interestingly, this is exactly the context where an information theoretic analysis becomes useful. Given a leakage model, the mutual information $\mathrm{I}(S; \mathbf{L}_1^1, \mathbf{L}_1^2, \ldots)$ can be estimated with less data than the success rate of the corresponding template attack. And

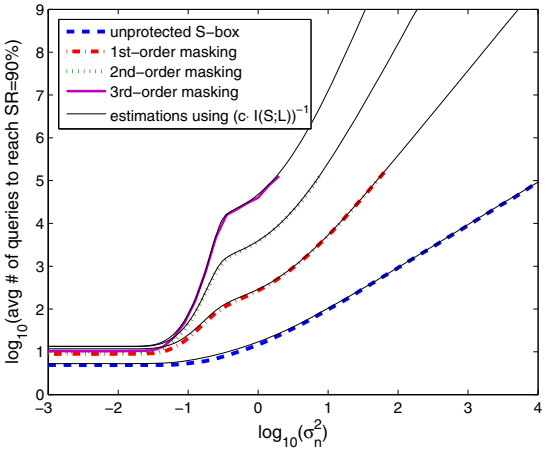


Fig. 8. Information leakage and success rates for 1^{st}, 2^{nd} and 3^{rd}-order masking

following [20], Theorem 2, it should hold that this mutual information is reasonably correlated with the number of traces required to reach a certain success rate. In order to confirm this expectation, we plotted an estimation of this number, based on the inverse of the mutual information multiplied with a constant factor c. As illustrated in Figure 8, this approximation holds nearly perfectly, with the same constant c for all attacks, essentially depending on the success rate to reach (here 90%). Summarizing, these simulations confirm the relevance of an information theoretic analysis when designing countermeasures against side-channel attacks.

Before to conclude, we note again that such an information theoretic analysis only captures the most powerful adversaries for which the profiling of the leakage distributions is perfect. But in practice, the reduction of the information leakage is not the only effect that increases the security in masked implementations. Namely, the pdf estimation of multidimensional distributions may become too complex for allowing the exploitation of all the information in the traces. And the number of pairs, triples, *etc.* of samples to test in the attacks also increases their time complexity considerably (up to N^2, N^3, *etc.*). However, we believe that the formal analysis of a worst-case scenario as in this paper is an important step towards a better understanding of the masking countermeasure.

9 Conclusions

The results in this paper provide a first complete and unifying treatment of higher-order power analysis. They allow putting forward the strengths and weaknesses of various approaches to second-order DPA and provide a sound explanation for them. Our analysis illustrates that in the context of cryptographic devices protected with masking, it is not sufficient to run a single arbitrary distinguisher to quantify the security of an implementation. Evaluations should

hold in two steps. First, an information theoretic analysis determines the actual information leakage (*i.e.* the impact of the countermeasure, independently of the adversary). Second, a security analysis determines the efficiency of various distinguishers in exploiting this leakage. By applying such a methodology to simulations and practical experiments, we consequently obtain a fair and comprehensive evaluation of the security level that a masking scheme can ensure.

While not in contradiction with previous results in the field, these investigations reshape the understanding of certain assumptions and allow refined intuitions. First, theoretical analysis and empirical attacks sometimes show a large gap between the efficiency of profiled attacks that best exploit the information from two or more leakage samples and the one of non-profiled attacks that are most frequently used in practice. This relates to the observation that the statistics in side-channel attacks are only used to discriminate secret data (while their natural objective is to allow a good estimation). Hence, the study of advanced pdf estimation techniques in the context of side-channel attacks is an interesting direction for further research, as initiated with MIA in [5].

Second, the security improvement obtained when increasing the order of a masking scheme beyond one is negligible if it is not combined with a sufficient amount of noise in the leakages. This observation relates to the generally accepted intuition that side-channel resistance requires the combination of several countermeasures in order to be effective. We additionally show in this paper that an information theoretic analysis has very convenient features for evaluating this noise threshold precisely. As a result, the best combination of masking with other countermeasures (*e.g.* dual rail logic styles, time randomization, *etc.*) is a second interesting scope for further research. Finally, the relationship between the mutual information and the success rate of a profiled attack, that is experimentally exhibited in this paper in the context of second- (and higher-) order DPA, could be analyzed in order to obtain a more formal justification of it, *e.g.* under the assumption of Gaussian noise in the leakages.

Acknowledgements. This work has been funded in part by the European Commission's ECRYPT-II NoE (ICT-2007-216676), by the Belgian State's IAP program P6/26 BCRYPT, by the Walloon region's SCEPTIC project, by FWO project G.0300.07, by the K.U. Leuven-BOF (OT/06/40), and by the Austrian Science Fund (FWF) under grant number P22241-N23: "Investigation of Implementation Attacks". E. Oswald has been supported in part by the EPSRC grant number EP/F039638/1. F.-X. Standaert is a Research Associate of the Belgian Fund for Scientific Research (FNRS-F.R.S).

References

1. Brier, E., Clavier, C., Olivier, F.: Correlation Power Analysis with a Leakage Model. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 16–29. Springer, Heidelberg (2004)
2. Chari, S., Jutla, C.S., Rao, J.R., 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)

3. 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)
4. 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)
5. Gierlichs, B., Batina, L., Tuyls, P., Preneel, B.: Mutual Information Analysis - A Generic Side-Channel Distinguisher. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 426–442. Springer, Heidelberg (2008)
6. Gierlichs, B., Batina, L., Preneel, B., Verbauwhede, I.: Revisiting Higher-Order DPA Attacks: Multivariate Mutual Information Analysis. In: Pieprzyk, J. (ed.) CT-RSA 2010. LNCS, vol. 5985, pp. 221–234. Springer, Heidelberg (2010)
7. Goubin, L., Patarin, J.: DES and Differential Power Analysis. In: Koç, Ç.K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, pp. 158–172. Springer, Heidelberg (1999)
8. Kocher, P.C., Jaffe, J., Jun, B.: Differential Power Analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
9. Mangard, S., Popp, T., Gammel, B.M.: Side-Channel Leakage of Masked CMOS Gates. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 351–365. Springer, Heidelberg (2005)
10. Joye, M., Paillier, P., Schoenmakers, B.: On Second-Order Differential Power Analysis. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 293–308. Springer, Heidelberg (2005)
11. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks. Springer, Heidelberg (2007)
12. Mangard, S., Oswald, E., Standaert, F.-X.: One for All, All for One: Unifying Standard DPA Attacks, Cryptology ePrint Archive, Report 2009/449
13. Messerges, T.S.: Using Second-Order Power Analysis to Attack DPA Resistant Software. In: Paar, C., Koç, Ç.K. (eds.) CHES 2000. LNCS, vol. 1965, pp. 238–251. Springer, Heidelberg (2000)
14. Oswald, E., Mangard, S.: Template Attacks on Masking - Resistance Is Futile. In: Abe, M. (ed.) CT-RSA 2007. LNCS, vol. 4377, pp. 243–256. Springer, Heidelberg (2006)
15. Piret, G., Standaert, F.-X.: Security Analysis of Higher-Order Boolean Masking Schemes for Block Ciphers (with Conditions of Perfect Masking). IET Information Security 2(1), 1–11 (2008)
16. Prouff, E., Rivain, M.: A Generic Method for Secure S-box Implementation. In: Kim, S., Yung, M., Lee, H.-W. (eds.) WISA 2007. LNCS, vol. 4867, pp. 227–244. Springer, Heidelberg (2008)
17. Prouff, E., Rivain, M., Bévan, R.: Statistical Analysis of Second Order DPA. IEEE Transactions on Computers 58(6), 799–811 (2009)
18. Prouff, E., Rivain, M.: Theoretical and Practical Aspects of Mutual Information Based Side-Channel Analysis. In: Abdalla, M., Pointcheval, D., Fouque, P.-A., Vergnaud, D. (eds.) ACNS 2009. LNCS, vol. 5536, pp. 499–518. Springer, Heidelberg (2009)
19. Schramm, K., Paar, C.: Higher Order Masking of the AES. In: Pointcheval, D. (ed.) CT-RSA 2006. LNCS, vol. 3860, pp. 208–225. Springer, Heidelberg (2006)
20. 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); extended version available on the Cryptology ePrint Archive, Report 2006/139, http://eprint.iacr.org/2006/139

21. Standaert, F.-X., Peeters, E., Archambeau, C., Quisquater, J.-J.: Towards Security Limits in Side-Channel Attacks. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 30–45. Springer, Heidelberg (2006); latest version available on the Cryptology ePrint Archive, Report 2007/222, http://eprint.iacr.org/2007/222
22. Veyrat-Charvillon, N., Standaert, F.-X.: Mutual Information Analysis: How, When and Why? In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 429–443. Springer, Heidelberg (2009)
23. Waddle, J., Wagner, D.: Towards Efficient Second-Order DPA. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 1–15. Springer, Heidelberg (2004)

A Additional Figures

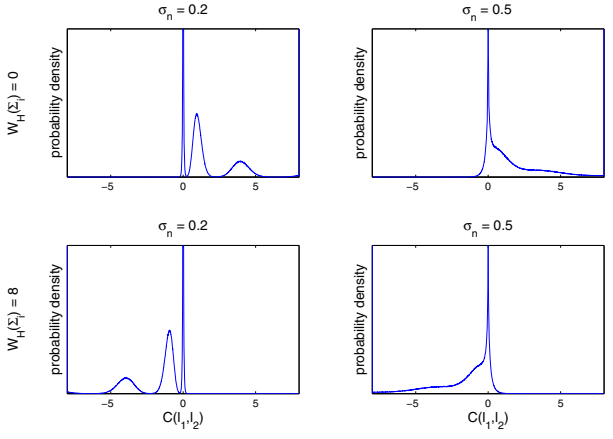


Fig. 9. Leakage probability distributions for the product combining function

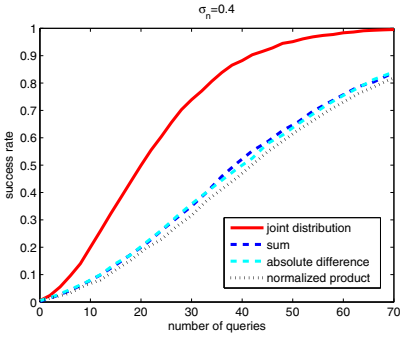


Fig. 10. Success rate of (simulated) profiled attacks against a masked AES S-box

Fig. 11. Available from: http://eprint.iacr.org/2010/180
Fig. 12. Available from: http://eprint.iacr.org/2010/180
Fig. 13. Available from: http://eprint.iacr.org/2010/180

Conditional Differential Cryptanalysis of NLFSR-Based Cryptosystems

Simon Knellwolf*, Willi Meier, and María Naya-Plasencia**

FHNW, Switzerland

Abstract. Non-linear feedback shift registers are widely used in lightweight cryptographic primitives. For such constructions we propose a general analysis technique based on differential cryptanalysis. The essential idea is to identify conditions on the internal state to obtain a deterministic differential characteristic for a large number of rounds. Depending on whether these conditions involve public variables only, or also key variables, we derive distinguishing and partial key recovery attacks. We apply these methods to analyse the security of the eSTREAM finalist Grain v1 as well as the block cipher family KATAN/KTANTAN. This allows us to distinguish Grain v1 reduced to 104 of its 160 rounds and to recover some information on the key. The technique naturally extends to higher order differentials and enables us to distinguish Grain-128 up to 215 of its 256 rounds and to recover parts of the key up to 213 rounds. All results are the best known thus far and are achieved by experiments in practical time.

Keywords: differential cryptanalysis, NLFSR, distinguishing attack, key recovery, Grain, KATAN/KTANTAN.

1 Introduction

For constrained environments like RFID tags or sensor networks a number of cryptographic primitives, such as stream ciphers and lightweight block ciphers have been developed, to provide security and privacy. Well known such cryptographic algorithms are the stream ciphers Trivium [5] and Grain [12,13] that have been selected in the eSTREAM portfolio of promising stream ciphers for small hardware [9], and the block cipher family KATAN/KTANTAN [6]. All these constructions build essentially on non-linear feedback shift registers (NLFSRs). These facilitate an efficient hardware implementation and at the same time enable to counter algebraic attacks.

Stream ciphers and block ciphers both mix a secret key a and public parameter (the initial value for stream ciphers and the plaintext for block ciphers) in an involved way to produce the keystream or the ciphertext, respectively. In cryptanalysis, such systems are often analysed in terms of boolean functions

* Supported by the Hasler Foundation `www.haslerfoundation.ch` under project number 08065.

** Supported by an ERCIM "Alain Bensoussan" Fellowship Programme.

M. Abe (Ed.): ASIACRYPT 2010, LNCS 6477, pp. 130–145, 2010.

that to each key k and public parameter x assign an output bit $f(k, x)$. Several cryptanalytic methods analyse derived functions from f. They can be roughly divided into algebraic and statistical methods. The cube attack presented in [8] is an algebraic method. It consists in finding many derivatives of f that are linear in the key bits such that the key can be found by solving a system of linear equations. The d-monomial test introduced in [10] provides a statistical framework to analyse the distribution of degree d monomials in the algebraic normal form of f. Another statistical approach is presented in [11,14], where the concept of probabilistc neutral key bits is applied to derivatives of f. The notion of cube testers introduced in [2] covers many of these methods. All of them have in common that they interact with f mainly in a black box manner, exploiting the structure of the underlying primitive only indirectly.

In this paper we propose a general analysis principle that we call *conditional differential cryptanalysis*. It consists in analysing the output frequency of derivatives of f on specifically chosen plaintexts (or initial values). Differential cryptanalyis, introduced in [4] for the analysis of block ciphers, studies the propagation of an input difference through an iterated construction and has become a common tool in the analysis of initialization mechanisms of stream ciphers, see [3,7,18]. In the case of NLFSR-based constructions, only few state bits are updated at each iteration, and the remaining bits are merely shifted. This results in a relatively slow diffusion. Inspired by message modification techniques introduced in [17] for hash function cryptanalysis, we trace the differences round by round and identify conditions on the internal state bits that control the propagation of the difference through the initial iterations. From these conditions we derive plaintexts (or initial values) that follow the same characteristic at the initial rounds and allow us to detect a bias in the output difference. In some cases the conditions also involve specific key bits which enables us to recover these bits in a key recovery attack.

The general idea of conditional differential cryptanalysis has to be elaborated and adapted with respect to each specific primitive. This is effected for the block cipher family KATAN and its hardware optimized variant KTANTAN as well as for the stream ciphers Grain v1 and Grain-128. The analysis of the block cipher family KATAN/KTANTAN is based on first order derivatives and nicely illustrates our analysis principle. For a variant of KATAN32 reduced to 78 of the 254 rounds we can recover at least two key bits with probability almost one and complexity 2^{22}. Comparable results are obtained for the other members of the family. We are not aware of previous cryptanalytic results on the KATAN/KTANTAN family. The analysis of Grain v1 is similar to that of KATAN, however the involved conditions are more sophisticated. We obtain a practical distinguisher for up to 104 of the 160 rounds. The same attack can be used to recover one key bit and four linear relations in key bits with high probability. Grain v1 was previously analysed in [7], where a sliding property is used to speed up exhaustive search by a factor two, and in [1], where a non-randomness property for 81 rounds could be detected.

Conditional differential cryptanalysis naturally extends to higher order derivatives. This is demonstrated by our analysis of Grain-128, which, compared to

Grain v1, is surprisingly more vulnerable to higher order derivatives. We get a practical distinguisher for up to 215 of the 256 rounds and various partial key recovery attacks for only slightly less rounds. For a 197 round variant we recover eight key bits with probability up to 0.87, for a 213 round variant two key bits with probability up to 0.59. The previously best known cryptanalytic result was a theoretical key recovery attack on 180 rounds, and was able to speed up exhaustive key search by a factor 2^4, but without the feasibility to predict the value of single key bits, see [11]. Moreover, a result in [7] mentions key recovery for up to 192 rounds and in [1] a non-randomness property was detected in a chosen key scenario.

The paper is organised as follows. Section 2 recalls the definition of higher order derivatives of boolean functions and discusses the application of frequency tests to such derivatives. Section 3 provides the general idea of conditional differential cryptanalysis of NLFSR-based cryptosystems. In the Sections 4, 5 and 6 this idea is refined and adapted to a specific analysis of the KATAN/KTANTAN family, Grain v1 and Grain-128.

2 Notation and Preliminaries

In this paper $\mathbb{F}_2$ denotes the binary field and $\mathbb{F}_2^n$ the n-dimensional vector space over $\mathbb{F}_2$. Addition in $\mathbb{F}_2$ is denoted by $+$, whereas addition in $\mathbb{F}_2^n$ is denoted by $\oplus$ to avoid ambiguity. For $0 \leq i \leq n-1$ we denote $e_i \in \mathbb{F}_2^n$ the vector with a one at position i and zero otherwise.

We now recall the definition of the i-th derivative of a boolean function introduced in [15,16] and we discuss the application of a frequency test to such derivatives.

2.1 Derivatives of Boolean Functions

Let $f : \mathbb{F}_2^n \to \mathbb{F}_2$ be a boolean function. The *derivative of f with respect to* $a \in \mathbb{F}_2^n$ is defined as

$$\Delta_a f(x) = f(x \oplus a) + f(x).$$

The derivative of f is itself a boolean function. If $\sigma = \{a_1, \ldots, a_i\}$ is a set of vectors in $\mathbb{F}_2^n$, let $L(\sigma)$ denote the set of all 2^i linear combinations of elements in σ. The *i-th derivative of f with respect to σ* is defined as

$$\Delta_\sigma^{(i)} f(x) = \sum_{c \in L(\sigma)} f(x \oplus c).$$

We note that the i-th derivative of f can be evaluated by summing up 2^i evaluations of f. We always assume that $a_1, \ldots, a_i$ are linearly independent, since otherwise $\Delta_\sigma^{(i)} f(x) = 0$ trivially holds. If we consider a keyed boolean function $f(k, \cdot)$ we always assume that the differences are applied to the second argument and not to the key.

2.2 Random Boolean Functions and Frequency Test

Let D be a non-empty subgroup of $\mathbb{F}_2^n$. A *random boolean function on* D is a function $D \to \mathbb{F}_2$ whose output is an independent uniformly distributed random variable. If f is a random boolean function on D, the law of large numbers says that for sufficiently many inputs $x_1, \ldots, x_s \in D$ the value

$$t = \frac{\sum_{k=1}^{s} f(x_k) - s/2}{\sqrt{s/4}}$$

approximately follows a standard normal distribution. Denoting

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{1}{2}u^2} du$$

the standard normal distribution function, a boolean function is said to *pass the frequency test* on $x_1, \ldots, x_s$ at a significance level α if

$$\Phi(t) < 1 - \frac{\alpha}{2}$$

A random boolean function passes the frequency test with probability $1 - \alpha$. If the frequency test is used to distinguish a keyed boolean function $f(k, \cdot)$ from a random boolean function, we denote by β the probability that $f(k, \cdot)$ passes the frequency test for a random key k. The distinguishing advantage is then given by $1 - \alpha - \beta$.

2.3 Frequency Test on Derivatives

If $\sigma = \{a_1, \ldots, a_i\}$ is a set of linearly independent differences, the i-th derivative of a boolean random function is again a boolean random function. Its output is the sum of 2^i independent uniformly distributed random variables. But for any two inputs x, x' with $x \oplus x' \in L(\sigma)$ the output values are computed by the same sum and thus $\Delta_\sigma^{(i)} f(x) = \Delta_\sigma^{(i)} f(x')$. Hence, the i-th derivative is not a random function on D, but on the quotient group $D/L(\sigma)$. A frequency test of $\Delta_\sigma^{(i)} f$ on s inputs needs $s2^i$ queries to f.

3 Conditional Differential Cryptanalysis of NLFSR

This section provides the general idea of our analysis. It is inspired by message modification techniques as they were introduced in [17] to speed up the collision search for hash functions. We trace differences through NLFSR-based cryptosystems and exploit the non-linear update to prevent their propagation whenever possible. This is achieved by identifying conditions on the internal state variables of the NLFSR. Depending on whether these conditions involve the public parameter or also the secret key, they have to be treated differently in a chosen plaintext attack scenario. The goal is to obtain many inputs that satisfy the conditions, i.e. that follow the same differential characteristic at the initial rounds.

In more abstract terms, we analyse derivatives of keyed boolean functions and exploit that their output values are iteratively computed.

We briefly explain NLFSR-based cryptosystems and why our analysis principle applies to them. Then we define three types of conditions that control the difference propagation in NLFSR-based cryptosystems and we explain how to deal with each of these types in a chosen plaintext (chosen initial value) attack scenario. The basic strategy is refined and adapted in the later sections to derive specific attacks on KATAN/KTANTAN, Grain v1 and Grain-128.

3.1 NLFSR-Based Cryptosystems

An NLFSR of length l consists of an initial state $s_0, \ldots, s_{l-1} \in \mathbb{F}_2$ and a recursive update formula $s_{l+i} = g(s_i, \ldots, s_{l+i-1})$ for $i \geq 0$, where g is a non-linear boolean function. The bit s_{l+i} is called the *bit generated at round* i and $s_i, \ldots, s_{l+i-1}$ is called the *state of round* $i-1$. Our analysis principle applies to any cryptographic construction that uses an NLFSR as a main building block. These constructions perform a certain number of rounds, generating at each round one or more bits that non-linearly depend on the state of the previous round. It is this non-linear dependency that we exploit in conditional differential cryptanalysis.

Let $f : \mathbb{F}_2^m \times \mathbb{F}_2^n \to \mathbb{F}_2$ denote the keyed boolean function that to every key k and public parameter x assigns one output bit $f(k, x)$ of an NLFSR-based construction. If we consider a first order derivative of the function f, we apply a difference $a \in \mathbb{F}_2^n$ to the public parameter. The value $\Delta_a f(k, x)$ then denotes the output difference $f(k, x) + f(k, x \oplus a)$. If s_i is a state bit of our construction, we denote $\Delta_a s_i(k, x)$ the difference in this state bit for the key k, the public parameter x and the difference a.

3.2 Conditions and Classification

We now introduce the concepts of our analysis principle. In general, the difference of a newly generated state bit depends on the differences and the values of previously generated state bits. Each time that $\Delta_a s_i(k, x)$ non-linearly depends on a bit that contains a difference, we can identify conditions on previously generated state bits that control the value of $\Delta_a s_i(k, x)$. In most cases, the conditions are imposed to prevent the propagation of the difference to the newly generated state bits. In particular it is important to prevent the propagation at the initial rounds. Since we want to statistically test the frequency of $\Delta_a f(k, \cdot)$ on inputs that satisfy the conditions, there is an important tradeoff between the number of imposed conditions and the number of inputs that we can derive. The conditions can not only involve bits of x, but also bits of k. We classify them into three types:

- Type 0 conditions only involve bits of x.
- Type 1 conditions involve bits of x and bits of k.
- Type 2 conditions only involve bits of k.

In a chosen plaintext (chosen initial value) scenario, type 0 conditions can easily be satisfied by the attacker, whereas he cannot control type 2 conditions at all. In most cases, type 2 conditions consist of simple equations and the probability that they are satisfied for a uniformly random key can easily be determined. Since we do not assume that our attacks can be repeated for more than one key, type 2 conditions generally decrease the advantage of distinguishing attacks and define classes of weak keys for this kind of attacks. On the other hand we specifically exploit type 2 conditions to derive key recovery attacks based on hypothesis tests. This is explained in Section 6 where we analyse Grain-128.

In a different way, also type 1 conditions can be used to recover parts of the key. To deal with the type 1 conditions, we introduce the concept of free bits. Suppose that the state bit s_i depends on x as well as on some bits of k, and suppose that we want to satisfy the type 1 condition $s_i = 0$. In a chosen plaintext scenario, we cannot control this condition in a simple way. We call those bits of x that do not influence the value of s_i for any key k, the *free bits* for the condition. The remaining bits of x are called *non-free*. Together with k the non-free bits determine whether the condition is satisfied or not. We call x a *valid input* if, for a given key k, it satisfies the imposed condition. If we define the set φ as $\varphi = \{e_i \in \mathbb{F}_2^n | x_i \text{ is a free bit}\}$ then we can generate $2^{|\varphi|}$ valid inputs from a single valid input x: these are the elements of the coset $x \oplus L(\varphi)$. In general, more than one type 1 condition are imposed. In that case, the free bits are those that are free for all of these conditions. In some cases it may be possible to give a finite number of configurations for the non-free bits such that at least one configuration determines a valid input. Otherwise, if t type 1 conditions are imposed, we expect that about one of 2^t different inputs is valid and we just repeat the attack several times with different random inputs.

In some cases we can not obtain enough inputs only by the method of free bits. We then try to find non-free bits that only must satisfy a given equation but otherwise can be freely chosen. This provides us with more degrees of freedom to generate a sample of valid inputs. We refer to the analysis of KATAN and Grain v1 for concrete examples of this method.

3.3 Choosing the Differences

The choice of a suitable difference for conditional differential cryptanalysis is not easy and strongly depends on the specific construction. In particular this holds for higher order derivatives, but also for first order ones. In general, the difference propagation should be controllable for as many rounds as possible with a small number of conditions. In particular, there should not be too many type 1 and type 2 conditions at the initial rounds. Differences which can be controlled by isolated conditions of type 1 or type 2 are favorable for key recovery attacks.

The set of differences for higher order derivatives can be determined by combining first order differences whose characteristics do not influence each other at the initial rounds. In a non-conditional setting, [1] describes a genetic algorithm for finding good sets of differences. This black-box approach did not yield particularly good sets for our conditional analysis.

4 Analysis of KATAN/KTANTAN

KATAN/KTANTAN is a family of lightweight block ciphers proposed in [6]. The family consists of six ciphers denoted by KATANn and KTANTANn for $n = 32, 48, 64$ indicating the block size of the cipher. All instances accept an 80-bit key and use the same building blocks, namely two NLFSRs and a small LFSR acting as a counter. The only difference between KATANn and KTANTANn is the key scheduling.

In the following we describe KATAN32 and provide the details of our analysis for this particular instance of the family. Our analysis of the other instances is very similar. We only sketch the differences and provide the empirical results.

We emphasize that our analysis does not reveal a weakness of any of the original KATAN/KTANTAN ciphers. In contrary, with respect to our method, it seems that the number of rounds is sufficiently large to provide a confident security margin.

4.1 Description of KATAN32

The two NLFSRs of KATAN32 have length 13 and 19 and we denote their states by $l_i, \ldots, l_{i+12}$ and $r_i, \ldots, r_{i+18}$, respectively. A 32-bit plaintext block x is loaded to the registers by $l_i = x_{31-i}$ for $0 \leq i \leq 12$ and $r_i = x_{18-i}$ for $0 \leq i \leq 18$. The LFSR has length 8 and we denote its state by $c_i, \ldots, c_{i+7}$. Initialization is done by $c_i = 1$ for $0 \leq i \leq 6$ and $c_7 = 0$. The full encryption process takes 254 rounds defined by

$$\begin{aligned} c_{i+8} &= c_i + c_{i+1} + c_{i+3} + c_{i+8}, \\ l_{i+13} &= r_i + r_{i+11} + r_{i+6}r_{i+8} + r_{i+10}r_{i+15} + k_{2i+1}, \\ r_{i+19} &= l_i + l_{i+5} + l_{i+4}l_{i+7} + l_{i+9}c_i + k_{2i}, \end{aligned}$$

where $k_0, \ldots, k_{79}$ are the bits of the key and k_i is recursively computed by

$$k_{j+80} = k_j + k_{j+19} + k_{j+30} + k_{j+67}$$

for $i \geq 80$. Finally, the states of the two NLFSRs are output as the ciphertext. If we consider a round-reduced variant of KATAN32 with r rounds, the bits l_{r+i} for $0 \leq i \leq 12$ and r_{r+i} for $0 \leq i \leq 18$ will be the ciphertext.

4.2 Key Recovery for KATAN32 Reduced to 78 Rounds

Our analysis is based on a first order derivative and uses the concept of free bits to satisfy type 1 conditions. Here, to obtain enough inputs, we will identify non-free bits that only must satisfy an underdefined system of linear equations, which gives us more freedom degrees generate the samples.

We consider a difference of weight five at the positions 1,7,12,22 and 27 of the plaintext block. Let $a = e_1 \oplus e_7 \oplus e_{12} \oplus e_{22} \oplus e_{27}$ denote the initial difference. At round 0 we have

$$\begin{aligned} \Delta_a l_{13}(k, x) &= 1 + x_{10}, \\ \Delta_a r_{19}(k, x) &= x_{24} + 1 \end{aligned}$$

and impose the conditions $x_{10} = 1$ and $x_{24} = 1$ to prevent the difference propagation. Similarly at the rounds 1, 2, 3 and 5, we impose the bits $x_2, x_6, x_5, x_9, x_{19}$, x_{25} to be zero. At round 7 we have

$$\Delta_a l_{20}(k, x) = r_{22}$$

and we impose the first type 1 condition

$$r_{22} = x_{28} + x_{23} + x_{21} + k_6 = 0. \tag{1}$$

At round 9 we impose $x_3 = 0$. Then three additional type 1 conditions

$$r_{19} = x_{31} + x_{26} + x_{27} + x_{22} + k_0 = 1, \tag{2}$$
$$r_{23} = x_{27} + x_{22} + x_{23}x_{20} + x_{18} + x_7 + x_{12} + k_1 + k_8 = 0, \tag{3}$$
$$r_{26} = 1 + x_{20}(x_{17} + k_3) + k_{14} = 0 \tag{4}$$

are imposed at the rounds 11, 13 and 20.

The free bits for these conditions can be directly read from the equations. They are:

$$x_0, x_4, x_8, x_{11}, x_{13}, x_{14}, x_{15}, x_{16}, x_{29} \text{ and } x_{30}.$$

So far, for any valid plaintext we can derive a sample of 2^{10} valid plaintexts. Since, in this case, this is not enough to perform a significant frequency test, we try to obtain larger samples by better analysing the non-free bits. Looking at the equations (1) to (4), we note that the non-free bits $x_7, x_{12}, x_{18}, x_{21}, x_{22}, x_{26}, x_{27}, x_{28}$ and x_{31} only occur linearly. They can be freely chosen as long as they satisfy the system of linear equations

$$\begin{cases} x_{28} + x_{21} = A \\ x_{31} + x_{26} + x_{27} + x_{22} = B \\ x_{27} + x_{22} + x_{18} + x_7 + x_{12} = C \end{cases}$$

for constants A, B, C. This system has 2^6 different solutions that can be added to each valid plaintext. In total this gives a sample of size 2^{16} that we can generate from a valid plaintext. Since we imposed 9 type 0 conditions we are left with 2^5 different samples of plaintexts for a given key. The conditions are satisfied for at least one of these samples. On this sample the difference in bit 18 of the ciphertext after 78 rounds (this is bit r_{78}) is strongly biased. We perfom a frequency test of $\Delta_a r_{78}(k, \cdot)$ on each of the 2^5 generated samples. At significance level $\alpha = 10^{-4}$ the frequency test fails on at least one of them with probability almost one, and if it fails, all four type 1 conditions are satisfied with probability almost one. This allows us to recover k_0, k_6, the relation $k_1 + k_8$ and either k_{14} (if $x_{20} = 0$) or the relation $k_3 + k_{14}$ with high probability. The complexity of this attack is 2^{22}.

4.3 Analysis of KATAN48 and KATAN64

All the three members of the KATAN family perform 254 rounds, they use the same LFSR and the algebraic structure of the non-linear update functions is

the same. The differences between the KATANn ciphers are the block size n, the length of the NLFSRs, the tap positions for the non-linear update and the number of times the NLFSRs are updated per round.

For KATAN48 the NLFSRs have length 19 and 29 and each register is updated twice per round. We obtained our best result with a difference of weight four at the positions 1, 10, 19 and 28 in the plaintext block. Imposing four type 0 conditions and two type 1 conditions we are able to derive a sample of size 2^{31} from a valid plaintext. This allows us to recover the key bit k_{12} and the relation $k_1 + k_{14}$ after 70 rounds (this corresponds to 140 updates of the NLFSRs) with a complexity of 2^{34}.

For KATAN64 the NFLSRs have length 25 and 39 and each register is updated three times per round. We obtained our best result with a difference of weight three at the positions 0, 13 and 26. Imposing six type 0 conditions and two type 1 conditions we are able to derive a sample of size at least 2^{32} from a valid plaintext. This allows us to recover k_2 and $k_1 + k_6$ after 68 rounds (204 updates of the NLFSRs) with a complexity of 2^{35}.

4.4 Analysis of the KTANTAN Family

KTANTANn is very similar to KATANn. They only differ in the key scheduling part. In KATAN the key is loaded into a register and linearly expanded to the round keys after round 40. Until round 40 the original key bits are used as the round keys. In KTANTAN, from the first round, the round keys are a linear combination of key bits (depending on the state of the counter LFSR, which is entirely known). Hence, our analysis of KATANn directly translates to KTANTANn, but instead of recovering a single key bit, we recover a linear relation of key bits. For instance in KATAN32 we recover the relation $k_7 + k_{71}$ instead of bit k_0.

5 Analysis of Grain v1

Grain v1 is a stream cipher proposed in [13] and has been selected for the final eSTREAM portfolio [9]. It accepts an 80-bit key k and a 64-bit initial value x. The cipher consists of three building blocks, namely an 80-bit LFSR, an 80-bit NLFSR and a non-linear output function. The state of the LFSR is denoted by $s_i, \ldots, s_{i+79}$ and the state of the NLFSR by $b_i, \ldots, b_{i+79}$. The registers are initialized by $b_i = k_i$ for $0 \leq i \leq 79$, $s_i = x_i$ for $0 \leq i \leq 63$ and $s_i = 1$ for $64 \leq i \leq 79$ and updated according to

$$\begin{aligned} s_{i+80} &= f(s_i, \ldots, s_{i+79}), \\ b_{i+80} &= g(b_i, \ldots, b_{i+79}) + s_i, \end{aligned}$$

where f is linear and g has degree 6. The output function is taken as

$$z_i = \sum_{k \in \mathcal{A}} b_{i+k} + h(s_{i+3}, s_{i+25}, s_{i+46}, s_{i+64}, b_{i+63}),$$

where $\mathcal{A} = \{1, 2, 4, 10, 31, 43, 56\}$ and h is defined as

$$\begin{aligned} h(s_{i+3}, s_{i+25}, s_{i+46}, s_{i+64}, b_{i+63}) &= s_{i+25} + b_{i+63} \\ &+ s_{i+3}s_{i+64} + s_{i+46}s_{i+64} + s_{i+64}b_{i+63} \\ &+ s_{i+3}s_{i+25}s_{i+46} + s_{i+3}s_{i+46}s_{i+64} + s_{i+3}s_{i+46}b_{i+63} \\ &+ s_{i+25}s_{i+46}b_{i+63} + s_{i+46}s_{i+64}b_{i+63} \end{aligned}$$

The cipher is clocked 160 times without producing any keystream. Instead the output function is fed back to the LFSR and to the NLFSR.

If we consider round-reduced variants of Grain v1 with r initialization rounds, the feedback of the output stops after r rounds and the first keystream bit is z_r.

Our analysis is similar to the one of KATAN32, but the equations for the conditions are more complex. We first present an attack on 97 rounds and then extend it to 104 rounds.

5.1 Distinguishing Attack and Key Recovery for 97 Rounds

Our analysis is based on the first order derivative with respect to a single difference in bit 37 of the initial value. Let $a = e_{37}$ denote the difference. The first conditions are defined at round 12, where the difference in s_{37} eventually propagates to the state bits s_{92} and b_{92} via the feedback of z_{12}. We have

$$\Delta_a z_{12}(k, x) = 1 + x_{15}x_{58} + x_{58}k_{75}.$$

We impose the type 0 condition $x_{58} = 1$ and we define the type 1 condition $x_{15} + k_{75} = 0$ to prevent the propagation. The next conditions are determined at round 34, where we have

$$\Delta_a z_{34}(k, x) = s_{98} + x_{59}s_{80} + s_{80}s_{98} + s_{80}b_{97}.$$

We define the conditions $s_{80} = 0$ and $s_{98} = 0$. Similarly we determine $s_{86} = 0$ and $s_{92} = 0$ at the rounds 40 and 46, respectively. So far, we imposed one type 0 condition at round 12 and we have five type 1 conditions at the rounds 12, 34, 40 and 46. The type 1 conditions jointly have 25 free bits:

$$\begin{gathered} x_7, x_8, x_{10}, x_{11}, x_{14}, x_{16}, x_{17}, x_{20}, x_{22}, x_{24}, x_{28}, x_{30}, x_{32}, x_{33}, \\ x_{34}, x_{36}, x_{39}, x_{42}, x_{45}, x_{49}, x_{54}, x_{55}, x_{59}, x_{60} \text{ and } x_{61}. \end{gathered}$$

In average we expect that one out of 2^5 randomly chosen initial values satisfies the conditions. We define a distinguisher that chooses 2^5 random initial values and for each performs a frequency test of $\Delta_a z_{97}(k, \cdot)$ on the sample of 2^{25} inputs generated by the free bits. Instead of randomly choosing 2^5 initial values we can choose 2^4 and test each of them for $x_{15} = 0$ and $x_{15} = 1$. This guarantees that the condition from round 12 is satisfied for at least one of them. Experiments with 2^{10} keys at a significance level $\alpha = 0.005$ show that at least one of the 2^5 tests fails with probability 0.99. This gives a distinguisher with complexity 2^{31} and advantage of about 0.83 for Grain v1 reduced to 97 rounds.

The two conditions $x_{15}+k_{75}=0$ and $s_{86}=0$ are crucial to obtain a significant bias after 97 rounds. In a key recovery scenario this reveals information about the key. Experiments show that both conditions hold with probability almost one if the frequency test fails. This recovers the key bit k_{75} and the value of $k_7+k_8+k_{10}+k_{37}+k_{49}+k_{62}+k_{69}$ (coming from $s_{86}=0$).

5.2 Extension to 104 Rounds

Using the same conditions as before, we extend the attack to 104 rounds. We use the same idea as for KATAN32 to increase the size of the sample that can be generated from one initial value. We gain four additional degrees of freedom by noting that the non-free bits x_6, x_{19}, x_{29}, x_{44} and x_{57} influence only the condition imposed at round 40 and must only satisfy the linear equation

$$x_6+x_{19}+x_{29}+x_{44}+x_{57}=A$$

for a constant A. In total, we can now derive a sample of size 2^{29} from one initial value.

The distinguisher defined above has now a complexity of 2^{35} and advantage of about 0.45. When the frequency test fails, the conditions $x_{15}+k_{75}=0$ and $s_{92}=0$ are satisfied with a probability almost one, which gives us k_{75} and the value of $k_{13}+k_{14}+k_{16}+k_{22}+k_{43}+k_{55}+k_{68}$ (coming from $s_{92}=0$). The remaining three conditions are satisfied with a probability about 0.70 and give us similar relations in the key bits.

The sample size can be further increased, because also the non-free bits $x_{13}, x_{23}, x_{38}, x_{51}$ and x_{62} only must satisfy a linear equation. This gives a distinguisher with complexity 2^{39} and advantage of about 0.58.

6 Analysis of Grain-128

Grain-128 was proposed in [12] as a bigger version of Grain v1. It accepts a 128-bit key k and a 96-bit initial value x. The general construction of the cipher is the same as for Grain v1, but the LFSR and the NLFSR both contain 128-bits. The content of the LFSR is denoted by $s_i,\ldots,s_{i+127}$ and the content of the NLFSR is denoted by $b_i,\ldots,b_{i+127}$. The initialization with the key and the initial value is analogous to Grain v1 and the update is performed according to

$$\begin{aligned} s_{i+128} &= f(s_i,\ldots,s_{i+127}),\\ b_{i+128} &= g(b_i,\ldots,b_{i+127})+s_i, \end{aligned}$$

where f is linear and g has degree 2. The output function is taken as

$$z_i=\sum_{k\in\mathcal{A}} b_{i+k}+h(b_{i+12},s_{i+8},s_{i+13},s_{i+20},b_{i+95},s_{i+42},s_{i+60},s_{i+79},s_{i+95}),$$

where $\mathcal{A}=\{2,15,36,45,64,73,89\}$ and h is defined as

$$h(x)=b_{i+12}s_{i+8}+s_{i+13}s_{i+20}+b_{i+95}s_{i+42}+s_{i+60}s_{i+79}+b_{i+12}b_{i+95}s_{i+95}$$

The cipher is clocked 256 times without producing any keystream. Instead the output function is fed back to the LFSR and to the NLFSR.

If we consider round-reduced variants of Grain-128 with r initialization rounds, the feedback of the output stops after r rounds and the first keystream bit is z_r.

For the analysis of Grain-128 we use higher order derivatives. The general idea of conditional differential cryptanalysis naturally extends. As in the case of first order derivatives we always assume that the differences are applied to the initial value and not to the key.

6.1 Distinguishing Attack up to 215 Rounds

Our attack is based on a derivative of order thirteen with respect to the set of differences

$$\sigma = \{e_0, e_1, e_2, e_{34}, e_{35}, e_{36}, e_{37}, e_{65}, e_{66}, e_{67}, e_{68}, e_{69}, e_{95}\}.$$

These differences are chosen because they do not influence each other in the initial rounds. As a consequence the corresponding differential characteristic (of order thirteen) is zero for as many as 170 rounds. This can be extended to 190 rounds by imposing simple type 0 conditions that control the propagation of each single difference. As an example we derive the conditions for the difference e_{65}. The first condition is derived from round 5, where we have

$$\Delta_{e_{65}} z_5(k, x) = x_{84}.$$

We impose $x_{84} = 0$. In the same way the conditions $x_{58} = 0$ and $x_{72} = 0$ prevent difference propagation at rounds 45 and 52. At round 23 we have

$$\Delta_{e_{65}} z_{23}(k, x) = k_{118}.$$

As we will see below, the type 2 condition $k_{118} = 0$ determines a class of weak keys for the distinguishing attack.

Proceeding the same way for the other differences we derive 24 type 0 conditions that consist in setting the following bits to zero: x_{27}, x_{28}, x_{29}, x_{30}, x_{41}, x_{42}, x_{43}, x_{44}, x_{58}, x_{59}, x_{60}, x_{61}, x_{62}, x_{72}, x_{73}, x_{74}, x_{75}, x_{76}, x_{77}, x_{84}, x_{85}, x_{86}, x_{87}, x_{88}. In addition to k_{118} the key bits k_{39}, k_{119}, k_{120} and k_{122} can be identified to define classes of weak keys.

There are $2^{96-13-24} = 2^{59}$ initial values that are different in $\mathbb{F}_2^n/L(\sigma)$ and satisfy all type 0 conditions. We define a distinguisher that performs a frequency test of $\Delta_\sigma^{(13)} z_r(k, \cdot)$ on 2^{12} of these inputs. Table 1 summarizes the empirical results obtained for 2^{12} different keys tested at a significance level $\alpha = 0.005$. The indicated values denote the probabilty $1-\beta$, where β denotes the probability that $\Delta_\sigma^{(13)} z_r(k, \cdot)$ passes the frequency test. Our distinguisher has complexity 2^{25} and advantage $1-\alpha-\beta$. The values in the first row are obtained without any condition on the key. They show that we can distinguish Grain-128 reduced to 215 rounds with an advantage of about 0.008. The other rows indicate the probabilities for the classes of weak keys defined by the indicated type 2 conditions.

Table 1. Distinguishing attack on Grain-128 reduced to r rounds: Probability $1 - \beta$ for $\alpha = 0.005$ and complexity 2^{25}. Type 2 conditions define classes of weak keys.

type 2 condition	$r = 203$	$r = 207$	$r = 211$	$r = 213$	$r = 215$
–	1.000	0.587	0.117	0.173	0.013
$k_{39} = 0$	1.000	0.630	0.128	0.275	0.017
$k_{118} = 0$	1.000	0.653	0.177	0.231	0.024
$k_{119} = 0$	1.000	0.732	0.151	0.267	0.025
$k_{120} = 0$	1.000	0.876	0.234	0.249	0.026
$k_{122} = 0$	1.000	0.668	0.160	0.285	0.015

6.2 Key Recovery up to 213 Rounds

In this section we specifically exploit type 2 conditions to recover single key bits with high probability. The attack is explained by a prototypical example that recovers three bits of Grain-128 reduced to 197 rounds with a probability up to 0.87. It is based on a derivative of order five and can easily be extended to recover more bits by using slightly other derivatives. This is demonstrated by an attack that recovers eight bits using two additional derivatives (both of order five). A second attack uses the derivative of order thirteen from the previous section and recovers three bits for Grain-128 reduced to 213 rounds with a probability up to 0.59.

Prototypical Example. We use a derivative of order five with respect to the differences $\sigma = \{e_1, e_{36}, e_{66}, e_{67}, e_{68}\}$. In the same way as in the distinguishing attack, we impose conditions on the initial value to control the propagation of each difference. Altogether we impose 12 type 0 conditions and denote by W the set of initial values satisfying all of them. The crucial observation is the following. The key bit k_{121} controls the characteristic of e_{68} in the very early phase of initialization, namely at round 26. If $k_{121} = 1$ the difference propagates, otherwise it does not. This strongly influences the frequency of $\Delta_\sigma^{(5)} z_r(k, \cdot)$ after $r = 197$ rounds. Similar strong influences can be found for k_{40} after $r = 199$ rounds and for k_{119} after $r = 200$ rounds. This allows to recover these bits by a binary hypothesis tests.

Key Recovery by Hypothesis Test. Let X be a uniformly distributed random variable taking values in $W/L(\sigma)$ and define

$$p_r(k) = \Pr[\Delta_\sigma^{(5)} z_r(k, X) = 1].$$

If the key is considered as a uniformly distributed random variable K, $p_r(K)$ is a random variable in the interval $[0, 1]$. Our attack is based on the observation that the conditional distributions of $p_r(K)$ conditioned on $K_i = 0$ and $K_i = 1$, for well chosen i, strongly differ even for a large number of rounds. This can be exploited to perform a binary hypothesis test on the value of K_i. An attacker can estimate a single observation $\hat{p}_r$ of $p_r(K)$ to take her decision. Since in all

our attacks the expectation of $p_r(K)$ conditioned on $K_i = 0$ is significantly smaller than the conditional expectation conditioned on $K_i = 1$, we determine a parameter $\pi \in [0, 1]$ and take our decision according to the rule defined as

$$K_i = \begin{cases} 0 & \text{if } \hat{p}_r < \pi \\ 1 & \text{otherwise.} \end{cases}$$

The success probability of the attack essentially depends on the choice of π. If we denote $\alpha = \Pr[p_r(K) \geq \pi | K_i = 0]$ the probability that we falsely guess $K_i = 1$ and $\beta = \Pr[p_r(K) < \pi | K_i = 1]$ the corresponding probability that we falsely guess $K_i = 0$, then the *probability of a correct decision*, denoted P_c, is given as

$$\mathrm{P}_c = 1 - (\alpha + \beta)/2.$$

An optimal π maximizes P_c. Since the conditional distributions of $p_r(K)$ are not known explicitely, we empirically determine π in a precomputation phase of the attack.

Back to the Example. The first row of Table 2 shows the precomputed parameters π and the resulting probability P_c for our prototypical example. The precomputation of each π was done for 2^{14} key pairs and 2^{14} initial values for each key. This gives an overall precomputation complexity of $6 \cdot 2^{33}$ since we have to compute two histograms for each key bit. The attack itself consists in estimating $\hat{p}_r$ for $r = 197, 199$ and 200. Note that all three estimates can be obtained by the same computation which has complexity 2^{19} when estimating over 2^{14} initial values. The probabilities P_c are not completely independent and the probability of correctly guessing all three bits together is about 0.463.

Recovering 8 Bits after 197 Rounds. The prototypical example can be extended by using two other sets of differences which are obtained by shifting all differences by one position to the left and to the right, respectively. This allows to recover five additional bits of the key, namely $k_{39}, k_{40}, k_{118}, k_{120}$ and k_{122}. The complexities of this extended attack are $9 \cdot 2^{34}$ for the precomputation and $3 \cdot 2^{19}$ for the attack

Table 2. Key recovery for reduced Grain-128: P_c is the probability of correctly guessing key bit k_i. The attack complexity is 2^{19} for $|\sigma| = 5$ and 2^{25} for $|\sigma| = 13$.

Difference set	k_i	r	π	P_c
$\sigma = \{e_1, e_{36}, e_{66}, e_{67}, e_{68}\}$	k_{40}	199	0.494	0.801
	k_{119}	200	0.492	0.682
	k_{121}	197	0.486	0.867
$\sigma = \{e_0, e_1, e_2, e_{34}, e_{35}, e_{36}, e_{37},$	k_{39}	213	0.490	0.591
$e_{65}, e_{66}, e_{67}, e_{68}, e_{69}, e_{95}\}$	k_{72}	213	0.488	0.566
	k_{119}	206	0.356	0.830
	k_{120}	207	0.486	0.807
	k_{120}	211	0.484	0.592
	k_{122}	213	0.478	0.581

itself. We recover all eight bits correctly with a probability of 0.123. This can be improved up to 0.236 by first determining k_{121} and k_{122} and then recovering the remaining bits conditioned on the values of k_{121} and k_{122}.

Recovering Bits up to 213 Rounds. If we use the derivative of order thirteen that we already used in the distinguishing attack, after 213 rounds we can recover two key bits with probability of almost 0.6. The last row of Table 2 summarizes the results. Here, the precomputation was done for 2^{12} key pairs and 2^{12} initial values for each key which gives a precomputation complexity of 2^{38}. The complexity of the attack itself is 2^{25}.

7 Conclusion

We presented a first analysis of the KATAN/KTANTAN family as well as the best known cryptanalytic results on Grain v1 and Grain-128. This was obtained by conditional differential cryptanalysis which also applies to other NLFSR-based contructions and provides further hints for choosing an appropriate number of rounds with regard to the security/efficiency tradeoff in future designs of such constructions.

Acknowledgements

This work was partially supported by the European Commission through the ICT programme under contract ICT-2007-216676 ECRYPT II.

References

1. Aumasson, J.P., Dinur, I., Henzen, L., Meier, W., Shamir, A.: Efficient FPGA Implementations of High-Dimensional Cube Testers on the Stream Cipher Grain-128. In: SHARCS (2009)
2. Aumasson, J.P., Dinur, I., Meier, W., Shamir, A.: Cube Testers and Key Recovery Attacks on Reduced-Round MD6 and Trivium. In: Dunkelman, O. (ed.) Fast Software Encryption. LNCS, vol. 5665, pp. 1–22. Springer, Heidelberg (2009)
3. Biham, E., Dunkelman, O.: Differential Cryptanalysis in Stream Ciphers. Cryptology ePrint Archive, Report 2007/218 (2007), http://eprint.iacr.org/
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. Cannière, C.D.: Trivium: A Stream Cipher Construction Inspired by Block Cipher Design Principles. In: Katsikas, S.K., López, J., Backes, M., Gritzalis, S., Preneel, B. (eds.) ISC 2006. LNCS, vol. 4176, pp. 171–186. Springer, Heidelberg (2006)
6. Cannière, C.D., Dunkelman, O., Knezevic, M.: KATAN and KTANTAN - A Family of Small and Efficient Hardware-Oriented Block Ciphers. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 272–288. Springer, Heidelberg (2009)

7. Cannière, C.D., Küçük, Ö., Preneel, B.: Analysis of Grain's Initialization Algorithm. In: Vaudenay, S. (ed.) AFRICACRYPT 2008. LNCS, vol. 5023, pp. 276–289. Springer, Heidelberg (2008)
8. Dinur, I., Shamir, A.: Cube Attacks on Tweakable Black Box Polynomials. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 278–299. Springer, Heidelberg (2010)
9. ECRYPT: The eSTREAM project, http://www.ecrypt.eu.org/stream/
10. Englund, H., Johansson, T., Turan, M.S.: A Framework for Chosen IV Statistical Analysis of Stream Ciphers. In: Srinathan, K., Rangan, C.P., Yung, M. (eds.) INDOCRYPT 2007. LNCS, vol. 4859, pp. 268–281. Springer, Heidelberg (2007)
11. Fischer, S., Khazaei, S., Meier, W.: Chosen IV Statistical Analysis for Key Recovery Attacks on Stream Ciphers. In: Vaudenay, S. (ed.) AFRICACRYPT 2008. LNCS, vol. 5023, pp. 236–245. Springer, Heidelberg (2008)
12. Hell, M., Johansson, T., Maximov, A., Meier, W.: A Stream Cipher Proposal: Grain-128. In: ISIT, pp. 1614–1618 (2006)
13. Hell, M., Johansson, T., Meier, W.: Grain: A Stream Cipher for Constrained Environments. IJWMC 2(1), 86–93 (2007)
14. Khazaei, S., Meier, W.: New Directions in Cryptanalysis of Self-Synchronizing Stream Ciphers. In: Chowdhury, D.R., Rijmen, V., Das, A. (eds.) INDOCRYPT 2008. LNCS, vol. 5365, pp. 15–26. Springer, Heidelberg (2008)
15. Knudsen, L.R.: Truncated and Higher Order Differentials. In: Preneel, B. (ed.) FSE 1994. LNCS, vol. 1008, pp. 196–211. Springer, Heidelberg (1995)
16. Lai, X.: Higher order derivatives and differential cryptanalysis. In: Blahut, R.E., Costello, D.J., Maurer, U., Mittelholzer, T. (eds.) Communicationis and Cryptography: Two Sides of one Tapestry, pp. 227–233. Kluwer Academic Publishers, Dordrecht (1994)
17. 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)
18. Wu, H., Preneel, B.: Resynchronization Attacks on WG and LEX. In: Robshaw, M.J.B. (ed.) FSE 2006. LNCS, vol. 4047, pp. 422–432. Springer, Heidelberg (2006)

A Byte-Based Guess and Determine Attack on SOSEMANUK*

Xiutao Feng, Jun Liu, Zhaocun Zhou, Chuankun Wu, and Dengguo Feng

State Key Laboratory of Information Security, Institute of Software,
Chinese Academy of Sciences, Beijing, 100190, China
{fengxt,liuj,zhouzc,ckwu,feng}@is.iscas.ac.cn

Abstract. SOSEMANUK is a software-oriented stream cipher proposed by C. Berbain et al for the eSTREAM project and has been selected into the final portfolio. It is noticed that most components of SOSEMANUK can be calculated byte-oriented. Hence an attacker can observe SOSEMANUK from the view of byte units instead of the original 32-bit word units. Based on the above idea, in this work we present a new byte-based guess and determine attack on SOSEMANUK, where we view a byte as a basic data unit and guess some certain bytes of the internal states instead of the whole 32-bit words during the execution of the attack. Surprisingly, our attack only needs a few words of known key stream to recover all the internal states of SOSEMANUK, and the time complexity can be dramatically reduced to $O(2^{176})$. Since SOSEMANUK has a key with the length varying from 128 to 256 bits, our results show that when the length of an encryption key is larger than 176 bits, our guess and determine attack is more efficient than an exhaustive key search.

Keywords: eSTREAM, SOSEMANUK, Guess and Determine Attack.

1 Introduction

The European eSTREAM project [1] was launched in 2004 to call for stream ciphers and was ended in 2008. At first about 34 stream cipher candidates were submitted to the eSTREAM project, and after the challenge of three rounds, 7 of them were selected into the final portfolio. SOSEMANUK proposed by C. Berbain et al [2] is one of the above seven algorithms. SOSEMANUK is a software-oriented stream cipher and has a key with the length varying from 128 to 256 bits. The design of SOSEMANUK adopted the ideas of both the stream cipher SNOW 2.0 [3] and the block cipher SERPENT [4], and aimed at improving SNOW 2.0 both from the security and from the efficiency points of view.

The guess and determine attack is a common attack on stream ciphers [5,6,7,8]. Its main idea is that: an attacker first guesses the values of a portion of the internal states of the target algorithm, then it takes a little cost to deduce the

* This work was supported by the National Natural Science Foundation (Grant No. 60833008 and 60902024).

M. Abe (Ed.): ASIACRYPT 2010, LNCS 6477, pp. 146–157, 2010.

values of all the rest of the internal states of the algorithm by making use of the values of the guessed portion of internal states and a few known key stream. When the values of all the internal states of the algorithm are recovered, the attacker tests the correctness of these values by producing a key stream using the above recovered values and comparing it with the known key stream. If the key streams agree, it shows that the recovered states are correct. If the key streams don't agree, then the attacker repeats the above process until the correct internal states are found. As for SOSEMANUK, the designers of SOSEMANUK [2] presented a guess and determine attack method, whose time complexity is $O(2^{256})$. In 2006 H. Ahmadi et al [9] revised the attack and reduced the time complexity to $O(2^{226})$, and this result was further reduced to $O(2^{224})$ by Y. Tsunoo et al [10]. Recently Lin and Jie [11] gave a new result that they could recover all internal states of SOSEMANUK with time complexity $O(2^{192})$. Unfortunately, a mistake was made in their work. In step 1 of their attack, $f_{t-2}, f_{t-1}, f_t, f_{t+1}$ and $s_{t-2}, s_{t-1}, s_t, s_{t+1}$ were used to output key words $z_{t-2}, z_{t-1}, z_t, z_{t+1}$, and in step 14, $f_t, f_{t+1}, f_{t+1}, f_{t+2}$ and $s_t, s_{t+1}, s_{t+1}, s_{t+2}$ were used to output key words $z_t, z_{t+1}, z_{t+1}, z_{t+2}$. However, according to the description of SOSEMANUK, the output key words in the next pad should be $z_{t+2}, z_{t+3}, z_{t+4}, z_{t+5}$, which should be produced by $f_{t+2}, f_{t+3}, f_{t+4}, f_{t+5}$ and $s_{t+2}, s_{t+3}, s_{t+4}, s_{t+5}$. Therefore the time complexity they gave is incorrect.

It is known that most word-oriented stream ciphers make a trade-off between security and efficiency. From the view of a designer, for pursuit of more efficient software implementation of the algorithm, some certain operators, for example, the exclusive OR, S-boxes, the modulo 2^n addition, the multiplication or the division by a primitive element in the finite field F_{2^n}, where n may be equal to 8, 16 or 32, are often used. We notice that most of these operations can be done based on the smaller units, for example, 16-bit words or bytes. Therefore from the view of an attacker, he can observe the algorithm from the viewpoints of smaller units instead of the original word units. Based on the above idea, in this work we present a byte-based guess and determine attack on SOSEMANUK, where we view a byte as a basic data unit and guess some certain bytes of the internal states instead of the whole 32-bit words during the execution of the attack. Surprisingly, our attack only needs a few known key stream to recover all the internal states of SOSEMANUK, and the time complexity can be dramatically reduced to $O(2^{176})$. It shows that when the length of an encryption key is larger than 176 bits, the guess and determine attack is more efficient than an exhaustive key search. What's more, our results also show that during the design of stream cipher algorithms, it is necessary to break the bound between different operands.

The rest of this paper is organized as follows: in section 2 we recall the SOSEMANUK algorithm briefly, and in section 3 we give some basic properties of SOSEMANUK. In section 4 we describe all the detail of our attack on SOSEMANUK. In section 5 we give a estimate on the time and data complexity of our attack. Section 6 gives a further discussion, and Section 7 concludes the paper.

2 Description of SOMEMANUK

In this section we recall the SOSEMANUK algorithm briefly and all the details can be found in [2].

SOSEMANUK is a 32-bit word-oriented stream cipher, and logically composed of three parts: a linear feedback shift register (LFSR), a finite state machine (FSM) and a round function Serpent1, see Figure 1.

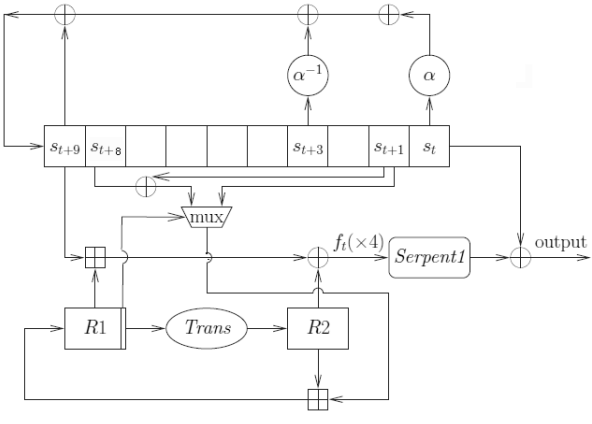


Fig. 1. The structure of SOSEMANUK

2.1 The LFSR

The LFSR of SOSEMANUK is defined over the finite field $F_{2^{32}}$, and contains 10 of 32-bit registers s_i, $1 \leq i \leq 10$. The feedback polynomial $\pi(x)$ of LFSR is defined as follows:

$$\pi(x) = \alpha x^{10} + \alpha^{-1} x^7 + x + 1, \tag{1}$$

where α is a root of the polynomial

$$P(x) = x^4 + \beta^{23} x^3 + \beta^{245} x^2 + \beta^{48} x + \beta^{239}$$

over the finite field F_{2^8}, and β is a root of the binary polynomial

$$Q(x) = x^8 + x^7 + x^5 + x^3 + 1.$$

Let $\{ s_t \}_{t \geq 1}$ be a sequence generated by the LFSR. Then it satisfies

$$s_{t+10} = s_{t+9} \oplus \alpha^{-1} s_{t+3} \oplus \alpha s_t, \forall t \geq 1. \tag{2}$$

2.2 The FSM

The nonlinear filtering part of SOSEMANUK is a finite state machine (FSM), which contains two 32-bit memory units $R1$ and $R2$. At time t, the FSM takes

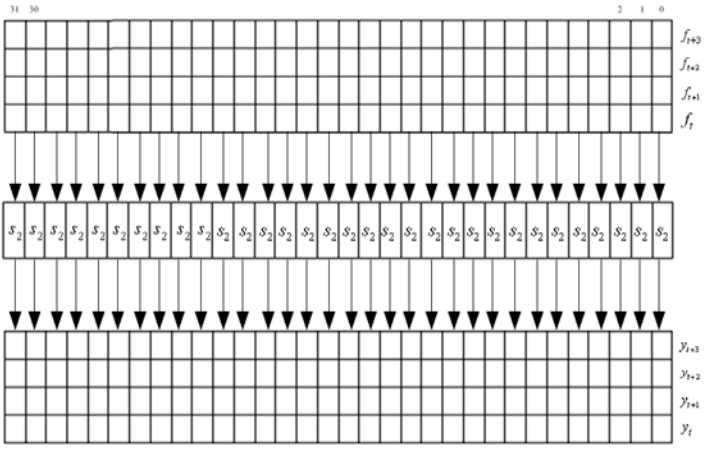


Fig. 2. The round function Serpent1 in the bit-slice mode

the values s_{t+1}, s_{t+8} and s_{t+9} of registers s_1, s_8 and s_9 of the LFSR as inputs, and outputs a 32-bit word f_t. The execution of the FSM is as follows:

$$R1_t = R2_{t-1} \boxplus \mathrm{mux}(\mathrm{lsb}(R1_{t-1}), s_{t+1}, s_{t+1} \oplus s_{t+8}), \tag{3}$$

$$R2_t = \mathrm{Trans}(R1_{t-1}), \tag{4}$$

$$f_t = (s_{t+9} \boxplus R1_t) \oplus R2_t, \tag{5}$$

where $\boxplus$ is the modulo 2^{32} addition; $\mathrm{lsb}(x)$ is the least significant bit of x; $\mathrm{mux}(c, x, y)$ is equal to x if $c = 0$, or equal to y if $c = 1$; and the internal transition function Trans on 32-bit integers is defined by

$$\mathrm{Trans}(z) = (0\mathrm{x}54655307 \cdot z \bmod 2^{32}) \lll 7,$$

where $\lll$ is the left cyclic shift operator on 32-bit strings.

2.3 The Round Function Serpent1

In the block cipher SERPENT a raw SERPENT round consists of, in that order:

- a subkey addition;
- S-boxes transformations;
- a linear transformation.

Here the function Serpent1 is one round of SERPENT without the subkey addition and the linear transformation. The S-box used in Serpent1 is the S-box S_2 of SERPENT and runs in the bit-slice mode. Serpent1 takes outputs f_{t+i} of the FSM at four successive times as inputs and outputs four 32-bit words y_{t+i}, where $i = 0, 1, 2, 3$, see Figure 2.

2.4 Generation of Key Stream

Let $s_t, s_{t+1}, s_{t+2}, s_{t+3}$ and $f_t, f_{t+1}, f_{t+2}, f_{t+3}$ be the outputs of the LFSR and that of the FSM respectively at the successive times starting from time t, and $z_t, z_{t+1}, z_{t+2}, z_{t+3}$ be the key words generated by SOSEMANUK at those four successive times. Then we have

$$(z_{t+3}, z_{t+2}, z_{t+1}, z_t) = \text{Serpent1}(f_{t+3}, f_{t+2}, f_{t+1}, f_t) \oplus (s_{t+3}, s_{t+2}, s_{t+1}, s_t). \quad (6)$$

3 Some Properties of SOSEMANUK

In this section we view a byte as a basic data unit and give some basic properties of SOSEMANUK from the view of byte units. First we introduce some notations.

Let x be a 32-bit word. We denote by $x^{(i)}$ the i-th byte component of x, $0 \le i \le 3$, that is,

$$x = x^{(3)} \parallel x^{(2)} \parallel x^{(1)} \parallel x^{(0)},$$

where each $x^{(i)}$ is a byte, and $\parallel$ is the concatenation of two bit strings. For simplicity we write $x^{(1)} \parallel x^{(0)}$ as $x^{(0,1)}$ and $x^{(2)} \parallel x^{(1)} \parallel x^{(0)}$ as $x^{(0,1,2)}$.

For any given 32-bit word x, the word x may have the different meanings in different contexts as follows:

1. As an operand of the operator $\oplus$. Here x is a 32-bit string, and $\oplus$ is the bitwise exclusive OR.
2. As an operand of the integer addition $+$ or the modulo 2^{32} addition $\boxplus$. Here x denotes the integer $\sum_{i=0}^{3} x^{(i)}(2^8)^i$.
3. As an element of the finite field $F_{2^{32}}$. Here x denotes the element $x^{(3)}\alpha^3 + x^{(2)}\alpha^2 + x^{(1)}\alpha + x^{(0)}$ in $F_{2^{32}}$, where α is defined as in equation (1).

Now we consider the SOSEMANUK algorithm from the view of byte units. First we notice that the feedback calculation of the LFSR (see equation (2)) can be represented in the byte form.

Lemma 1. *Equation (2) can be written in the byte form as follows:*

$$s_{t+10}^{(0)} = s_{t+9}^{(0)} \oplus s_{t+3}^{(1)} \oplus \beta^{64} s_{t+3}^{(0)} \oplus \beta^{239} s_t^{(3)}, \quad (2a)$$

$$s_{t+10}^{(1)} = s_{t+9}^{(1)} \oplus s_{t+3}^{(2)} \oplus \beta^{6} s_{t+3}^{(0)} \oplus \beta^{48} s_t^{(3)} \oplus s_t^{(0)}, \quad (2b)$$

$$s_{t+10}^{(2)} = s_{t+9}^{(2)} \oplus s_{t+3}^{(3)} \oplus \beta^{39} s_{t+3}^{(0)} \oplus \beta^{245} s_t^{(3)} \oplus s_t^{(1)}, \quad (2c)$$

$$s_{t+10}^{(3)} = s_{t+9}^{(3)} \oplus \beta^{16} s_{t+3}^{(0)} \oplus \beta^{23} s_t^{(3)} \oplus s_t^{(2)}. \quad (2d)$$

Proof. By the definition of α, we have,

$$\alpha^4 + \beta^{23}\alpha^3 + \beta^{245}\alpha^2 + \beta^{48}\alpha + \beta^{239} = 0.$$

It follows that

$$\alpha^{-1} = \beta^{16}\alpha^3 + \beta^{39}\alpha^2 + \beta^{6}\alpha + \beta^{64}.$$

Let $s_t = \sum_{i=0}^{3} s_t^{(i)} \alpha^i$ and $s_{t+3} = \sum_{i=0}^{3} s_{t+3}^{(i)} \alpha^i$. Then we have

$$\begin{aligned}\alpha s_t &= s_t^{(3)}\alpha^4 + s_t^{(2)}\alpha^3 + s_t^{(1)}\alpha^2 + s_t^{(0)}\alpha \\ &= s_t^{(3)}(\beta^{23}\alpha^3 + \beta^{245}\alpha^2 + \beta^{48}\alpha + \beta^{239}) + s_t^{(2)}\alpha^3 + s_t^{(1)}\alpha^2 + s_t^{(0)}\alpha \\ &= (\beta^{23}s_t^{(3)} + s_t^{(2)})\alpha^3 + (\beta^{245}s_t^{(3)} + s_t^{(1)})\alpha^2 + (\beta^{48}x^{(3)} + s_t^{(0)})\alpha + \beta^{239}s_t^{(3)}\end{aligned}$$

and

$$\begin{aligned}\alpha^{-1}s_{t+3} &= s_{t+3}^{(3)}\alpha^2 + s_{t+3}^{(2)}\alpha^1 + s_{t+3}^{(1)} + s_{t+3}^{(0)}\alpha^{-1} \\ &= s_{t+3}^{(3)}\alpha^2 + s_{t+3}^{(2)}\alpha^1 + s_{t+3}^{(1)} + x^{(0)}(\beta^{16}\alpha^3 + \beta^{39}\alpha^2 + \beta^6\alpha + \beta^{64}) \\ &= \beta^{16}s_{t+3}^{(0)}\alpha^3 + (s_{t+3}^{(3)} + \beta^{39}s_{t+3}^{(0)})\alpha^2 + (s_{t+3}^{(2)} + \beta^6 x^{(0)})\alpha + (s_{t+3}^{(1)} + \beta^{64}s_{t+3}^{(0)}).\end{aligned}$$

Combine the above equations and equation (2), and we immediately get the desired conclusion. ■

Second we observe the update of $R1$ and the output of the FSM and have the following conclusions:

Lemma 2. *Equations (3) and (5) also hold in the sense of modulo 2^k for all $1 \le k < 32$, that is,*

$$R1_t^{[k]} = R2_{t-1}^{[k]} \boxplus \mathrm{mux}(\mathrm{lsb}(R1_{t-1}), s_{t+1}^{[k]}, s_{t+1}^{[k]} \oplus s_{t+8}^{[k]}), \tag{3'}$$

$$f_t^{[k]} = (s_{t+9}^{[k]} \boxplus R1_t^{[k]}) \oplus R2_t^{[k]}, \tag{5'}$$

where $x^{[k]}$ denotes the lowest k bits of x, and the operator $\boxplus$ still denotes the modulo 2^k addition without confusion. In particular, the cases $k = 8, 16$ and 24 are considered in this paper.

Finally we observe the round function Serpent1 and have the following conclusion:

Lemma 3. *For any $1 \le k \le 32$, if the values of the k-th bit of each s_{t+i} ($i = 0, 1, 2, 3$) are known, then the values of the k-th bit of each f_{t+i} can be calculated by the definition of* Serpent1 *given some known key stream, that is,*

$$f'_k = S_2^{-1}(z'_k \oplus s'_k), \tag{7}$$

where

$$\begin{aligned}f'_k &= f_{t+3,k} \parallel f_{t+2,k} \parallel f_{t+1,k} \parallel f_{t,k}, \\ s'_k &= s_{t+3,k} \parallel s_{t+2,k} \parallel s_{t+1,k} \parallel s_{t,k}, \\ z'_k &= z_{t+3,k} \parallel z_{t+2,k} \parallel z_{t+1,k} \parallel z_{t,k},\end{aligned}$$

and $f_{t+i,k}$, $s_{t+i,k}$ and $z_{t+i,k}$ are the k-th bits of f_{t+i}, s_{t+i} and z_{t+i} respectively, $i = 0, 1, 2, 3$. Similarly, if the i-th bytes of each s_{t+i} are known, then we can calculate the i-th bytes of each f_{t+i}, $i = 0, 1, 2, 3$.

4 Execution of the Attack

In this section we always assume that a portion of key stream words $\{ z_t \}$ have been observed, where $t = 1, 2, \cdots, N$, and N is large enough for the attack to work. For convenience, we denote by

$$\mathcal{A} \stackrel{(*)}{\Longrightarrow} \mathcal{B}$$

the deduction of $\mathcal{B}$ from $\mathcal{A}$ by equation (*).

Before the description of the attack, we make the following assumption:

Assumption 1. *The least significant bit of* $R1_1$ *is one, that is,* $\mathrm{lsb}(R1_1) = 1$.

The whole description of the attack on SOSEMANUK can be divided into five phases as follows.

Phase 1. We first guess the total 159-bit values of s_1, s_2, s_3, $s_4^{(0)}$, $R2_1^{(0,1,2)}$ and the rest 31-bit values of $R1_1$.

Step 1.1 We first deduce $s_{10}^{(0)}$, $R1_2^{(0)}$, $R2_2$, $s_{11}^{(0)}$ and $s_4^{(1)}$ as follows:

$$\begin{aligned}
\{ s_1^{(0)}, s_2^{(0)}, s_3^{(0)}, s_4^{(0)} \} &\stackrel{(7)}{\Longrightarrow} \{ f_1^{(0)}, f_2^{(0)}, f_3^{(0)}, f_4^{(0)} \}, \\
\{ f_1^{(0)}, R1_1^{(0)}, R2_1^{(0)} \} &\stackrel{(51)}{\Longrightarrow} s_{10}^{(0)}, \\
\{ R1_1^{(0)}, R2_1^{(0)}, s_3^{(0)}, s_{10}^{(0)} \} &\stackrel{(31)}{\Longrightarrow} R1_2^{(0)}, \\
R1_1 &\stackrel{(4)}{\Longrightarrow} R2_2, \\
\{ f_2^{(0)}, R1_2^{(0)}, R2_2^{(0)} \} &\stackrel{(51)}{\Longrightarrow} s_{11}^{(0)}, \\
\{ s_1^{(3)}, s_4^{(0)}, s_{10}^{(0)}, s_{11}^{(0)} \} &\stackrel{(25)}{\Longrightarrow} s_4^{(1)}.
\end{aligned}$$

Step 1.2 Similar to Step 1.1, we further deduce $s_{10}^{(1)}$, $R1_2^{(1)}$, $s_{11}^{(1)}$ and $s_4^{(2)}$ as follows:

$$\begin{aligned}
\{ s_1^{(1)}, s_2^{(1)}, s_3^{(1)}, s_4^{(1)} \} &\stackrel{(7)}{\Longrightarrow} \{ f_1^{(1)}, f_2^{(1)}, f_3^{(1)}, f_4^{(1)} \}, \\
\{ f_1^{(0,1)}, R1_1^{(0,1)}, R2_1^{(0,1)} \} &\stackrel{(51)}{\Longrightarrow} s_{10}^{(0,1)}, \\
\{ R1_1^{(0,1)}, R2_1^{(0,1)}, s_3^{(0,1)}, s_{10}^{(0,1)} \} &\stackrel{(31)}{\Longrightarrow} R1_2^{(0,1)}, \\
\{ f_2^{(0,1)}, R1_2^{(0,1)}, R2_2^{(0,1)} \} &\stackrel{(51)}{\Longrightarrow} s_{11}^{(0,1)}, \\
\{ s_1^{(0)}, s_1^{(3)}, s_4^{(0)}, s_{10}^{(1)}, s_{11}^{(1)} \} &\stackrel{(25)}{\Longrightarrow} s_4^{(2)}.
\end{aligned}$$

Step 1.3 Similar to Step 1.2, we further deduce $s_{10}^{(2)}$, $R1_2^{(2)}$, $s_{11}^{(2)}$ and $s_4^{(3)}$ as follows:

$$\begin{aligned}
\{ s_1^{(2)}, s_2^{(2)}, s_3^{(2)}, s_4^{(2)} \} &\stackrel{(7)}{\Longrightarrow} \{ f_1^{(2)}, f_2^{(2)}, f_3^{(2)}, f_4^{(2)} \}, \\
\{ f_1^{(0,1,2)}, R1_1^{(0,1,2)}, R2_1^{(0,1,2)} \} &\stackrel{(51)}{\Longrightarrow} s_{10}^{(0,1,2)},
\end{aligned}$$

$$\{ R1_1^{(0,1,2)}, R2_1^{(0,1,2)}, s_3^{(0,1,2)}, s_{10}^{(0,1,2)} \} \stackrel{(3l)}{\Longrightarrow} R1_2^{(0,1,2)},$$
$$\{ f_2^{(0,1,2)}, R1_2^{(0,1,2)}, R2_2^{(0,1,2)} \} \stackrel{(5l)}{\Longrightarrow} s_{11}^{(0,1,2)},$$
$$\{ s_1^{(1)}, s_1^{(3)}, s_4^{(0)}, s_{10}^{(2)}, s_{11}^{(2)} \} \stackrel{(2s)}{\Longrightarrow} s_4^{(3)}.$$

In this phase we have obtained s_1, s_2, s_3, s_4, $s_{10}^{(0,1,2)}$, $s_{11}^{(0,1,2)}$, $R1_1$, $R1_2^{(0,1,2)}$, $R2_1^{(0,1,2)}$ and $R2_2$.

Phase 2. Since we have obtained $s_1^{(3)}$, $s_2^{(3)}$, $s_3^{(3)}$ and $s_4^{(3)}$ in phase 1, thus by equation (7), we can calculate $f_1^{(3)}$, $f_2^{(3)}$, $f_3^{(3)}$ and $f_4^{(3)}$, that is,

$$\{ s_1^{(3)}, s_2^{(3)}, s_3^{(3)}, s_4^{(3)} \} \stackrel{(7)}{\Longrightarrow} \{ f_1^{(3)}, f_2^{(3)}, f_3^{(3)}, f_4^{(3)} \}.$$

Furthermore, by equations (5l) and (2d), we have

$$f_1^{(3)} = (s_{10}^{(3)} + R1_1^{(3)} + c_1 \bmod 2^8) \oplus R2_1^{(3)}, \tag{8}$$
$$f_2^{(3)} = (s_{11}^{(3)} + R1_2^{(3)} + c_2 \bmod 2^8) \oplus R2_2^{(3)}, \tag{9}$$
$$s_{11}^{(3)} = s_{10}^{(3)} \oplus \beta^{16} s_4^{(0)} \oplus \beta^{23} s_1^{(3)} \oplus s_1^{(2)}. \tag{10}$$

where $c_1 = 1$ if $s_{10}^{(0,1,2)} + R1_1^{(0,1,2)} \geq 2^{24}$, or $c_1 = 0$, otherwise; and $c_2 = 1$ if $s_{11}^{(0,1,2)} + R1_2^{(0,1,2)} \geq 2^{24}$, or $c_2 = 0$, otherwise.

By the assumption $\mathrm{lsb}(R1_1) = 1$, we have $R1_2 = R2_1 \boxplus (S_3 \oplus S_{10})$. It follows that

$$R1_2^{(3)} = R2_1^{(3)} + (s_3^{(3)} \oplus s_{10}^{(3)}) + c_3 \bmod 2^8, \tag{11}$$

where $c_3 = 1$ if $R2_1^{(0,1,2)} + (s_3^{(0,1,2)} \oplus s_{10}^{(0,1,2)}) \geq 2^{24}$, or $c_3 = 0$, otherwise.

Combine equations (8), (9), (10) and (11), and then we have the equation on the variable $s_{10}^{(3)}$:

$$d = (s_{10}^{(3)} \oplus a) + (s_{10}^{(3)} \oplus s_3^{(3)}) + (f_1^{(3)} \oplus (s_{10}^{(3)} + b \bmod 2^8)) + c \bmod 2^8, \tag{12}$$

where $a = \beta^{16} s_4^{(0)} \oplus \beta^{23} s_1^{(3)} \oplus s_1^{(2)}$, $b = R1_1^{(3)} + c_1 \bmod 2^8$, $c = c_2 + c_3$ and $d = f_2^{(3)} \oplus R2_2^{(3)}$.

In equation (12), all variables except $s_{10}^{(3)}$ are known, since $s_{10}^{(3)}$ occurs three times in the above equation, it is easy to verify that equation (12) has exactly one solution. Denote its solution by $s_{10}^{(3)}$. When $s_{10}^{(3)}$ has been obtained, we deduce $R2_1^{(3)}$, $s_{11}^{(3)}$ and $R1_2^{(3)}$ by equations (8), (10) and (9) respectively.

Up to now we have obtained s_1, s_2, s_3, s_4, s_{10}, s_{11}, $R1_1$, $R2_1$, $R1_2$ and $R2_2$.

Phase 3. In this phase we further deduce $R1_3$, $R2_3$, $R1_4$, $R2_4$, $R1_5$, $R2_5$, $R2_6$, s_5, s_6, s_{12} and s_{13} as follows:

$$\begin{aligned}
\{ R1_2, R2_2, s_4, s_{11} \} &\xRightarrow{(3)} R1_3,\\
R1_2 &\xRightarrow{(4)} R2_3,\\
\{ f_3, R1_3, R2_3 \} &\xRightarrow{(5)} s_{12},\\
\{ s_2, s_{11}, s_{12} \} &\xRightarrow{(2)} s_5,\\
\{ R1_3, R2_3, s_5, s_{12} \} &\xRightarrow{(3)} R1_4,\\
R1_3 &\xRightarrow{(4)} R2_4,\\
\{ f_4, R1_4, R2_4 \} &\xRightarrow{(5)} s_{13},\\
\{ s_3, s_{12}, s_{13} \} &\xRightarrow{(2)} s_6,\\
\{ R1_4, R2_4, s_6, s_{13} \} &\xRightarrow{(3)} R1_5,\\
R1_4 &\xRightarrow{(4)} R2_5,\\
R1_5 &\xRightarrow{(4)} R2_6.
\end{aligned}$$

Phase 4. We guess both $s_7^{(0)}$ and $s_8^{(0)}$. The following deductions are entirely similar to phase 1, and we can recover both $s_7^{(1,2,3)}$ and $s_8^{(1,2,3)}$.

$$\begin{aligned}
\{ s_5^{(0)}, s_6^{(0)}, s_7^{(0)}, s_8^{(0)} \} &\xRightarrow{(7)} \{ f_5^{(0)}, f_6^{(0)}, f_7^{(0)}, f_8^{(0)} \},\\
\{ f_5^{(0)}, R1_5^{(0)}, R2_5^{(0)} \} &\xRightarrow{(5')} s_{14}^{(0)},\\
\{ s_4^{(3)}, s_7^{(0)}, s_{13}^{(0)}, s_{14}^{(0)} \} &\xRightarrow{(2')} s_7^{(1)},\\
\{ R1_5^{(0)}, R2_5^{(0)}, s_7^{(0)}, s_{14}^{(0)} \} &\xRightarrow{(3')} R1_6^{(0)},\\
\{ f_6^{(0)}, R1_6^{(0)}, R2_6^{(0)} \} &\xRightarrow{(5')} s_{15}^{(0)},\\
\{ s_5^{(3)}, s_8^{(0)}, s_{14}^{(0)}, s_{15}^{(0)} \} &\xRightarrow{(2')} s_8^{(1)},\\
\{ s_5^{(1)}, s_6^{(1)}, s_7^{(1)}, s_8^{(1)} \} &\xRightarrow{(7)} \{ f_5^{(1)}, f_6^{(1)}, f_7^{(1)}, f_8^{(1)} \},\\
\{ f_5^{(0,1)}, R1_5^{(0,1)}, R2_5^{(0,1)} \} &\xRightarrow{(5')} s_{14}^{(0,1)},\\
\{ s_4^{(0)}, s_4^{(3)}, s_7^{(0)}, s_{13}^{(1)}, s_{14}^{(1)} \} &\xRightarrow{(2b)} s_7^{(2)},\\
\{ R1_5^{(0,1)}, R2_5^{(0,1)}, s_7^{(0,1)}, s_{14}^{(0,1)} \} &\xRightarrow{(3')} R1_6^{(0,1)},\\
\{ f_6^{(0,1)}, R1_6^{(0,1)}, R2_6^{(0,1)} \} &\xRightarrow{(5')} s_{15}^{(0,1)},\\
\{ s_5^{(0)}, s_5^{(3)}, s_8^{(0)}, s_{14}^{(1)}, s_{15}^{(1)} \} &\xRightarrow{(2b)} s_8^{(2)},\\
\{ s_5^{(2)}, s_6^{(2)}, s_7^{(2)}, s_8^{(2)} \} &\xRightarrow{(7)} \{ f_5^{(2)}, f_6^{(2)}, f_7^{(2)}, f_8^{(2)} \},\\
\{ f_5^{(0,1,2)}, R1_5^{(0,1,2)}, R2_5^{(0,1,2)} \} &\xRightarrow{(5')} s_{14}^{(0,1,2)},
\end{aligned}$$

$$\{ s_4^{(1)}, s_4^{(3)}, s_7^{(0)}, s_{13}^{(2)}, s_{14}^{(2)} \} \stackrel{(26)}{\Longrightarrow} s_7^{(3)},$$
$$\{ R1_5^{(0,1,2)}, R2_5^{(0,1,2)}, s_7^{(0,1,2)}, s_{14}^{(0,1,2)} \} \stackrel{(31)}{\Longrightarrow} R1_6^{(0,1,2)},$$
$$\{ f_6^{(0,1,2)}, R1_6^{(0,1,2)}, R2_6^{(0,1,2)} \} \stackrel{(51)}{\Longrightarrow} s_{15}^{(0,1,2)},$$
$$\{ s_5^{(1)}, s_5^{(3)}, s_8^{(0)}, s_{14}^{(2)}, s_{15}^{(2)} \} \stackrel{(22)}{\Longrightarrow} s_8^{(3)}.$$

Phase 5. Finally we deduce s_9 as follows:

$$\{ s_5, s_6, s_7, s_8 \} \stackrel{(7)}{\Longrightarrow} \{ f_5, f_6, f_7, f_8 \},$$
$$\{ f_5, R1_5, R2_5 \} \stackrel{(5)}{\Longrightarrow} s_{14}^{(3)},$$
$$\{ R1_5, R2_5, s_7, s_{14} \} \stackrel{(3)}{\Longrightarrow} R1_6^{(3)},$$
$$\{ f_6, R1_6, R2_6 \} \stackrel{(5)}{\Longrightarrow} s_{15}^{(3)},$$
$$\{ R1_6, R2_6, s_8, s_{15} \} \stackrel{(3)}{\Longrightarrow} R1_7,$$
$$R1_6 \stackrel{(4)}{\Longrightarrow} R2_7,$$
$$\{ f_7, R1_7, R2_7 \} \stackrel{(5)}{\Longrightarrow} s_{16},$$
$$\{ s_6, s_{15}, s_{16} \} \stackrel{(2)}{\Longrightarrow} s_9.$$

Up to now we have recovered all internal states $s_1, s_2, \cdots, s_{10}$, $R1_1$ and $R2_1$ of the SOSEMANUK algorithm. And then we test the correctness of those values by producing a key stream using the above recovered values and comparing it with the known key stream. If the key streams agree, it shows that the recovered states are correct. If the key streams don't agree, then we will repeat the above process until the correct internal states are found.

The process of the above attack is demonstrated in Table 1.

5 On Time and Data Complexities of Our Attack

The execution of the above attack needs to guess a total of 175 bits of the internal states, including 159 bits of the internal states at phase 1 and 16 bits at phase 4, and then all the rest of the internal states can be deduced under the assumption $\mathrm{lsb}(R1_1) = 1$. Since the probability for $\mathrm{lsb}(R1_1) = 1$ to hold is $\frac{1}{2}$, thus the time complexity of the above attack on SOSEMANUK is $O(2^{176})$. In the attack we only make use of 8 words of the known key stream, and during the verification we need about another 8 words of the known key stream to verify whether the guessed internal states are correct or not. Since by shifting the keystream by 4 words we can test two cases, thus the total data complexity is about 20 words of the known key stream.

6 Further Discussion on the Assumption $\mathrm{lsb}(R1_1) = 1$

In the above attack, we make the assumption $\mathrm{lsb}(R1_1) = 1$, which will guarantee that equation (12) in phase 2 has exactly one solution. However it should be

Table 1. The process of our attack

Assume that $\mathrm{lsb}(R1_1) = 1$				
Steps		Guessed internal states	Determined internal states	Num. of States
Phase 1		$s_1, s_2, s_3, s_4^{(1)}, R1_1^*, R2_1^{(0,1,2)}$		2^{159}
	Step 1.1		$s_{10}^{(0)}, R1_2^{(0)}, R2_2, s_{11}^{(0)}, s_4^{(1)}$	2^{159}
	Step 1.2		$s_{10}^{(1)}, R1_2^{(1)}, s_{11}^{(1)}, s_4^{(2)}$	2^{159}
	Step 1.3		$s_{10}^{(2)}, R1_2^{(2)}, s_{11}^{(2)}, s_4^{(3)}$	2^{159}
Phase 2			$s_{10}^{(3)}, R2_1^{(3)}, R1_2^{(3)}, s_{11}^{(3)}$	2^{159}
Phase 3			$R1_3, R2_3, s_{12}, s_5, R1_4, R2_4$ $s_{13}, s_6, R1_5, R2_5, R2_6$	2^{159}
Phase 4		$s_7^{(0)}, s_8^{(0)}$	$R1_6^{(0,1,2)}, s_{14}^{(0,1,2)}, s_{15}^{(0,1,2)}, s_7, s_8$	2^{175}
Phase 5			$R1_6^{(3)}, s_{14}^{(3)}, s_{15}^{(3)}, R1_7, R2_7, s_{16}, s_9$	2^{175}

Note: $R1_1^*$ denotes the 31 bits of $R1_1$ from the second significant bit to the most significant bit.

pointed out that this assumption is not necessary for our attack to work. In fact, we directly guess the 160-bit values of s_1, s_2, s_3, $s_4^{(0)}$, $R1_1$ and $R2_1^{(0,1,2)}$ in phase 1. When $\mathrm{lsb}(R1_1) = 0$, we have $R1_2 = R2_1 \boxplus s_3$ in phase 2. It follows that

$$R1_2^{(3)} = R2_1^{(3)} + s_3^{(3)} + c_4 \bmod 2^8, \tag{11'}$$

where $c_4 = 1$ if $R2_1^{(0,1,2)} + s_3^{(0,1,2)} \geq 2^{24}$, or $c_4 = 0$, otherwise.

Similar to equation (12), combine equations (8), (9), (10) and (11′), and then we have the equation on the variable $s_{10}^{(3)}$:

$$d' = (s_{10}^{(3)} \oplus a') + (f_1^{(3)} \oplus (s_{10}^{(3)} + b' \bmod 2^8)) + c' \bmod 2^8, \tag{12'}$$

where $a' = \beta^{16} s_4^{(0)} \oplus \beta^{23} s_1^{(3)} \oplus s_1^{(2)}$, $b' = R1_1^{(3)} + c_1 \bmod 2^8$, $c' = s_3^{(3)} + c_2 + c_4 \bmod 2^8$ and $d' = f_2^{(3)} \oplus R2_2^{(3)}$. Since $s_{10}^{(3)}$ occurs two times in equation (12′), it is easy to verify that equation (12′) has either no solution, or 2^k solutions for some non-negative integer k. When equation (12′) has no solution, we will come back to phase 1 and repeat guessing new values of those internal states. When equation (12′) has 2^k solutions for some integer k, we write down all solutions, and then for each solution we go on the deductions according to phases 3, 4 and 5. Finally we obtain 2^k different values of the internal states of SOSEMANUK and verify their correctness respectively.

Now we estimate the time and data complexity of the above method. In phase 1 we guess total 160-bit values of the internal states instead of 159-bit values. 2^{159} of those values satisfy $\mathrm{lsb}(R1_1) = 1$ and another 2^{159} values satisfy $\mathrm{lsb}(R1_1) = 0$. As for 2^{159} values satisfying $\mathrm{lsb}(R1_1) = 1$, we have 2^{159} possible values of s_1, s_2, s_3, s_4, s_{10}, s_{11}, $R1_1$, $R2_1$, $R1_2$ and $R2_2$ after phase 2. As for 2^{159} values satisfying $\mathrm{lsb}(R1_1) = 0$, since equation (12′) has the same number of solutions as that of the possible values of the variables when all the variables except $s_{10}^{(3)}$ go through

all possible values, thus we have also 2^{159} possible values of s_1, s_2, s_3, s_4, s_{10}, s_{11}, $R1_1$, $R2_1$, $R1_2$ and $R2_2$ after phase 2. Therefore we have total 2^{160} possible values. For each possible values, we go on deducing according to phases 3, 4 and 5, hence the total time complexity is still $O(2^{176})$. But without the assumption, the data complexity reduces to 16 words of the known key stream.

7 Conclusion

In this paper, we presented a byte-based guess and determine attack on SOSEMANUK, which only needs a few words of key stream to recover all internal states of SOSEMANUK with time complexity $O(2^{176})$. The results show that when the length of an encryption key is larger than 176 bits, the guess and determine attack is more efficient than an exhaustive key search.

Acknowledgement

The authors gratefully acknowledge the anonymous referees, whose comments helped to improve the presentation.

References

1. The eSTREAM project, http://www.ecrypt.eu.org/stream/
2. Berbain, C., Billet, O., Canteaut, A., Courtois, N., Gilbert, H., Goubin, L., Gouget, A., Granboulan, L., Lauradoux, C., Minier, M., Pornin, T., Sibert, H.: SOSEMANUK, a fast software-oriented stream cipher, eSTREAM, the ECRYPT Stream Cipher Project, Report 2005/027 (2005)
3. Ekdahl, P., Johansson, T.: A new version of the stream cipher SNOW. In: Nyberg, K., Heys, H.M. (eds.) SAC 2002. LNCS, vol. 2595, pp. 47–61. Springer, Heidelberg (2003)
4. Anderson, R., Biham, E., Knudsen, L.R.: Serpent: A proposal for the advanced encryption standard, NIST AES Proposal (1998)
5. Bleichenbacher, D., Patel, S.: SOBER cryptanalysis, Fast Software Encryption. In: Knudsen, L.R. (ed.) FSE 1999. LNCS, vol. 1636, pp. 305–316. Springer, Heidelberg (1999)
6. Hawkes, P., Rose, G.: Exploiting multiplies of the connection polynomial in word-oriented stream ciphers. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 302–316. Springer, Heidelberg (2000)
7. Hawkes, P., Rose, G.: Guess and determine attacks on SNOW. In: Nyberg, K., Heys, H.M. (eds.) SAC 2002. LNCS, vol. 2595, pp. 37–46. Springer, Heidelberg (2003)
8. Canniere, C.D.: Guess and determine attacks on SNOW, NESSIE Public Document, NES/DOC/KUL/WP5/011/a (2001)
9. Ahmadi, H., Eghlidos, T., Khazaei, S.: Improved guess and determine Attack on SOSEMANUK, Tehran, Iran (2006), http://www.ecrypt.eu.org/stream/sosemanukp3.html
10. Tsunoo, Y., Saito, T., Shigeri, M., Suzaki, T., Ahmadi, H., Eghlidos, T., Khazaei, S.: Evaluation of SOSEMANUK with regard to guess-and-determine attacks (2006), http://www.ecrypt.eu.org/stream/sosemanukp3.html
11. Lin, D., Jie, G.: Guess and Determine Attack on SOSEMANUK. In: 2009 Fifth International Conference on Information Assurance and Security, vol. 1, pp. 658–661 (2009)

Improved Single-Key Attacks on 8-Round AES-192 and AES-256

Orr Dunkelman, Nathan Keller*, and Adi Shamir

Faculty of Mathematics and Computer Science
Weizmann Institute of Science
P.O. Box 26, Rehovot 76100, Israel
{orr.dunkelman,nathan.keller,adi.shamir}@weizmann.ac.il

Abstract. AES is the most widely used block cipher today, and its security is one of the most important issues in cryptanalysis. After 13 years of analysis, related-key attacks were recently found against two of its flavors (AES-192 and AES-256). However, such a strong type of attack is not universally accepted as a valid attack model, and in the more standard single-key attack model at most 8 rounds of these two versions can be currently attacked. In the case of 8-round AES-192, the only known attack (found 10 years ago) is extremely marginal, requiring the evaluation of essentially all the 2^{128} possible plaintext/ciphertext pairs in order to speed up exhaustive key search by a factor of 16. In this paper we introduce three new cryptanalytic techniques, and use them to get the first non-marginal attack on 8-round AES-192 (making its time complexity about a million times faster than exhaustive search, and reducing its data complexity to about $1/32,000$ of the full codebook). In addition, our new techniques can reduce the best known time complexities for all the other combinations of 7-round and 8-round AES-192 and AES-256.

1 Introduction

The Rijndael block cipher [6] was developed in the late 1990's by Joan Daemen and Vincent Rijmen, and was selected as the Advanced Encryption Standard (AES) in 2001. Over the last ten years it replaced the Data Encryption Standard (DES) in most applications, and had become the block cipher of choice for any new security application. It has three possible key sizes (128, 192, and 256 bits), and in 2003 the US government had publicly announced that AES-128 can be used to protect classified data up to the level of "secret", and that AES-192 and AES-256 can be used to protect classified data up to the level of "top secret".

Due to its importance and popularity, the security of AES had attracted a lot of attention, and is considered one of the hottest areas of research in cryptanalysis. A major breakthrough was the recent discovery of related-key attacks on the full versions of AES-192 and AES-256 [3,4] which are faster than exhaustive search, but have impractical complexities. In another line of research [2],

* The second author was partially supported by the Koshland center for basic research.

M. Abe (Ed.): ASIACRYPT 2010, LNCS 6477, pp. 158–176, 2010.

related-key attacks requiring practical time complexity of 2^{45} were found on AES-256 with up to 10 rounds, and related key attacks requiring semipractical time complexity of 2^{70} were found on AES-256 with 11 rounds.

The main weakness of AES-192 and AES-256 exploited in these attacks was their extremely simple key schedule. In a related-key attack model, this made it possible to cancel data differences with corresponding key differences over many rounds of AES. This created a very high probability differential characteristic, which led to a greatly improved time complexity. However, such attacks make a very strong assumption that the adversary can ask the encryption box to modify the unknown key in a known way. Some of these attacks even assume that the adversary can obtain a large number of related keys, or that he can obtain related intermediate subkeys — see [3] for a discussion of these possibilities. Consequently, related-key attacks are important considerations during the design and certification stage of new ciphers, but are not considered a realistic threat in practical security protocols which use the block cipher in a standard way.

In this paper we consider the classical attack model of a single key and multiple known or chosen plaintext/ciphertext pairs. In this model the adversary has to deal with the very well designed data path of AES, and cannot directly benefit from its weak key schedule. Consequently, there are no known attacks on the full cipher on any one of the three flavors of AES, and the best we can do is to attack reduced round versions of AES. In the case of AES-256, the largest number of rounds that can be attacked is 8. In the case of AES-192 there is one attack on 8-round AES-192 which was published in [10], it is extremely marginal: It requires the evaluation of essentially all the possible plaintext/ciphertext pairs under the unknown key, and even then the time required to derive the key is only 16 times faster than exhaustive search (one can argue that given the complete codebook of size 2^{128}, there is no need to find the actual key in order to easily decrypt any given ciphertext ...). In the case of AES-128, there are no known attacks on its 8-round version, but there are a few on 7-round variants.

In order to improve all these known attacks, and especially the marginal attack on 8-round AES-192 which no one was able to improve upon in the last ten years, we develop three new cryptanalytic techniques. Our starting point is the attack on 7-round AES developed by Gilbert and Minier [11], which constructs a large table of 2^{72} entries, where each entry contains a sequence of 256 byte values. This idea was extended to 8-round AES by Demirci and Selçuk [7], who constructed an even larger table of 2^{192} entries (again containing sequences of 256 byte values, which are constructed in a slightly modified way). Due to the 2^{200} time required just to construct this table, this attack is worse than exhaustive search for 8-round AES-192, and can only be applied to 8-round AES-256.

Our first new idea (called *multiset tabulation*) is to replace the sequence of 256-byte values in each table entry by the multiset of its values. Even though we lose some information, we show that it is still possible to use such a table in order to discard with very high probability incorrect key guesses. This modification makes it possible to reduce the number of table entries (and thus also the time required to prepare the table) by a factor of 2^{16}. An even bigger saving (by a factor of

2^{57}) in the size of the table is obtained by another new technique which we call *differential enumeration*. It uses some truncated differential (which need not have particularly high or low probability, as required in standard or impossible differential attacks) in order to enumerate the entries of such a table in a much more efficient way: Instead of directly enumerating state values, the adversary derives them indirectly by enumerating the input and output differential values of certain internal S-boxes. By reducing the space complexity in such a major way, we can now trade it off with the high time complexity of the Demirci and Selçuk attack in order to greatly improve it. Finally, we develop a new *key bridging* technique which exploits the weak key schedule of AES by using the following surprising observation: In the particular case of 8-round AES-192, it is possible to compute one byte of the whitening subkey (used before the first round) directly from four bytes of the last subkey (used at the end of the eighth round), despite their distance. Since our attack requires guessing these five subkey bytes in the first and last rounds, we get an extra savings of 2^8 in our time complexity.[1] By combining these three techniques, we can now break 8-round AES-192 in about one millionth of the complexity of exhaustive search.

Our new results are summarized and compared with previously known single-key attacks in Table 1. As can be seen, our time complexities for 8-round AES are considerably better than the best previous results for both AES-192 and AES-256.

This paper is organized as follows: In Section 2 we describe the AES block cipher and introduce our notations. In Section 3 we describe the techniques used in previous attacks, and analyze their complexity. In Section 4 we introduce our new cryptanalytic techniques. We use them in Section 5 to improve the best known attacks on 7-round AES, and in Section 6 to improve the best known attacks on 8-round AES. Finally, in Section 7 we summarize our results.

2 A Short Description of AES

The advanced encryption standard (AES) [6] is an SP-network that supports key sizes of 128, 192, and 256 bits. A 128-bit plaintext is treated as a byte matrix of size 4x4, where each byte represents a value in $GF(2^8)$. An AES round applies four operations to the state matrix:

- SubBytes (SB) — applying the same 8-bit to 8-bit invertible S-box 16 times in parallel on each byte of the state,
- ShiftRows (SR) — cyclically shifting the i'th row by i bytes to the left,
- MixColumns (MC) — multiplication of each column by a constant 4x4 matrix over the field $GF(2^8)$, and
- AddRoundKey (ARK) — XORing the state with a 128-bit subkey.

We outline an AES round in Figure 1.

[1] The same idea can be used to improve the time complexity of several other attacks such as [10,14] by the same factor of 2^8.

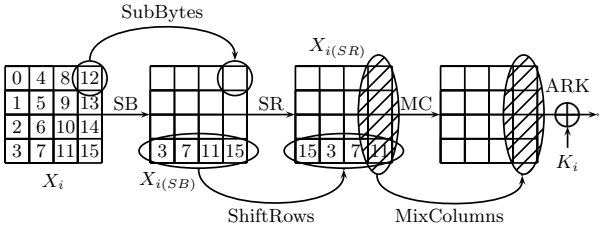


Fig. 1. An AES round

In the first round, an additional AddRoundKey operation (using a whitening subkey) is applied, and in the last round the MixColumns operation is omitted. Rounds which include the MixColumns operation are called *full rounds*.

The number of rounds depends on the key length: 10 rounds for 128-bit keys, 12 rounds for 192-bit keys, and 14 rounds for 256-bit keys. The rounds are numbered $0, \ldots, Nr-1$, where Nr is the number of rounds. For the sake of simplicity we shall denote AES with n-bit keys by AES-n, e.g., AES with 128-bit keys (and thus with 10 rounds) is denoted by AES-128. We use AES to mean all three variants of AES.

The key schedule of AES takes the user key and transforms it into $Nr+1$ subkeys of 128 bits each. The subkey array is denoted by $W[0, \ldots, 4 \cdot Nr+3]$, where each word of $W[\cdot]$ consists of 32 bits. Let the length of the key be Nk 32-bit words, then the first Nk words of $W[\cdot]$ are loaded with the user supplied key. The remaining words of $W[\cdot]$ are updated according to the following rule:

- For $i = Nk, \ldots, 4 \cdot Nr+3$, do
  - If $i \equiv 0 \bmod Nk$ then $W[i] = W[i-Nk] \oplus SB(W[i-1] \lll 8) \oplus RCON[i/Nk]$,
  - else if $Nk = 8$ and $i \equiv 4 \bmod 8$ then $W[i] = W[i-8] \oplus SB(W[i-1])$,
  - Otherwise $W[i] = W[i-1] \oplus W[i-Nk]$,

where $RCON[\cdot]$ is an array of predetermined constants, and $\lll$ denotes rotation of the word by 8 bits to the left.

2.1 The Notations Used in the Paper

In the sequel we use the following definitions and notations: The state matrix at the beginning of round i is denoted by X_i, and its bytes are denoted by $0, 1, 2, \ldots, 15$, as described in Figure 1. Similarly, the state matrix after the SubBytes and the ShiftRows operations of round i are denoted by $X_{i(SB)}$ and $X_{i(SR)}$, respectively.

We denote the subkey of round i by k_i, and the first (whitening) key by k_{-1}, i.e., $k_i = W[4 \cdot (i+1)]||W[4 \cdot (i+1)+1]||W[4 \cdot (i+1)+2]||W[4 \cdot (i+1)+3]$. In some cases, we are interested in interchanging the order of the MixColumns operation and the subkey addition. As these operations are linear they can be interchanged, by first XORing the data with an equivalent subkey and only

then applying the MixColumns operation. We denote the equivalent subkey for the altered version by u_i, i.e., $u_i = MC^{-1}(k_i)$. The bytes of the subkeys are numbered by $0, 1, \ldots, 15$, in accordance with the corresponding state bytes.

We use the following notations for intermediate encryption values: The intermediate state at the beginning of round i in the encryption of P^j is denoted by X_i^j, and its bytes are denoted by $X_{i,l}^j$, for $0 \leq l \leq 15$. Similarly, the intermediate values after the SubBytes and the ShiftRows operations of round i are denoted by $X_{i(SB),l}^j$ and $X_{i(SR),l}^j$, respectively.

In our attacks we mostly consider the encryption of *δ-sets*, which are structured sets of 256 plaintexts $\{P^0, P^1, \ldots, P^{255}\}$ in which one *active* byte assumes each one of the 256 possible values exactly once, and each one of the other 15 bytes is a (possibly different) constant. A state byte is called *balanced* if the XOR of its 256 values during the encryption of a δ-set is zero.

In all the observations considering reduced-round versions of AES, the numbering of the rounds starts with round 0. When we analyze the behavior of some consecutive inner rounds of AES, we shift the round numbering accordingly, depending on the number of rounds we add at the beginning.

Finally, we measure the time complexity of all the attacks in units which are equivalent to a single encryption operation of the relevant reduced round variant of AES. We measure the space complexity in units which are equivalent to the storage of a single plaintext (namely, 128 bits). To be completely fair, we count all operations carried out during our attacks, and in particular we do not ignore the time and space required to prepare the various tables we use.

3 Previous Work

The first attack developed against AES was the SQUARE attack, which was found by its designers [5]. The SQUARE attack is based on:

Observation 1. *Consider the encryption of a δ-set through three full AES rounds. The set of 256 corresponding ciphertexts is balanced, i.e., the XOR of the 256 values in each one of its* 16 *bytes is zero.*

The observation follows easily from the structure of AES, as demonstrated in Figure 2. This property is the basis of many attacks on reduced round variants of AES. The original submission [5] offers a 6-round attack with time complexity of 2^{72}, which was later improved in [10] using the partial sums technique to offer the best known attack on 6-round AES (with time 2^{42}).

In [11], Gilbert and Minier proposed to refine the information on the intermediate encryption values of the δ-sets exploited in the SQUARE attack. Their attack is based on the following observation:

Observation 2. *Consider the encryption of a δ-set through three full AES rounds. For each one of the 16 bytes of the ciphertext, we can define a sequence of 256 values for this byte by ordering the plaintexts according to the value of their active byte. Then any such sequence is fully determined by just 9 bytes, which are complex functions of the constants in the δ-set and the key bytes. Consequently, for*

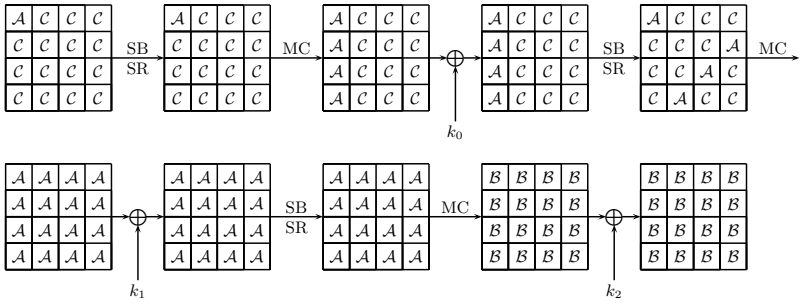


Fig. 2. The development of a δ-set through 3 rounds of AES, where $\mathcal{A}$ stands for an active byte, $\mathcal{B}$ stands for a balanced byte, and $\mathcal{C}$ stands for a constant byte

any fixed byte position, there are at most 2^{72} *possible sequences when we consider all the possible choices of keys and* δ*-sets (out of the* $(2^8)^{256} = 2^{2048}$ *of the "theoretically possible" 256-byte sequences, and out of the* $2^{256+15\times 8} = 2^{376}$ *sequences which could be potentially defined by the choice of* 15 *constant bytes and 256 key bits.)*

This observation was used in [11] to mount an attack on 7-round AES-128 with time complexity slightly smaller than that of exhaustive key search. Since the attack algorithm is a bit complex and not used in our paper, we omit it here.

In [7], Demirci and Selçuk extended the observation of [11] by another round. They showed the following:

Observation 3. *Consider the encryption of a* δ*-set through four full AES rounds. For each of the 16 bytes of the state, the ordered sequence of 256 values of that byte in the corresponding ciphertexts is fully determined by just 25 byte parameters. Consequently, for any fixed byte position, there are at most* 2^{200} *possible sequences when we consider all the possible choices of keys and* δ*-sets.*[2]

This observation was used in [7] to mount attacks on 7-round and 8-round variants of AES-256. The attack on 7-round AES-256 is roughly as follows:

1. **Preprocessing phase:** Compute all the 2^{192} possible values of the 255-byte sequence given in Observation 3, and store them in a hash table.
2. **Online phase:**
   (a) Guess the value of four bytes in the whitening key k_{-1} and of one byte in k_0, and for each guess, construct a δ-set from the data. (For example, if the active byte of the δ-set is byte 0, then the guessed bytes are bytes

[2] In [7] the authors note that the function $f_{c_1,\ldots,c_{25}}(x)$ can be written as $f_{c_1,\ldots,c_{25}}(x) = g_{c_1,\ldots,c_{24}}(x) \oplus c_{25}$, and thus one can reduce the number of possible sequences by picking some x_0, and considering the augmented function $f'_{c_1,\ldots,c_{24}}(x) = f_{c_1,\ldots,c_{25}}(x) - f_{c_1,\ldots,c_{25}}(x_0) = g_{c_1,\ldots,c_{24}}(x) - g_{c_1,\ldots,c_{24}}(x_0)$. In this case, the number of parameters is reduced to 24, the number of "interesting" entries in each sequence is reduced to 255 (as $f'(x_0) = 0$, independently of the choice of x_0 and $c_1, \ldots, c_{24}$), and the number of possible sequences is reduced to 2^{192}.

$0, 5, 10, 15$ of k_{-1} and byte 0 of k_0. Note that byte 0 of k_0 is used only to compute the *order* of the values in the δ-set).

(b) Guess four bytes of the equivalent subkey u_6 and one byte of the equivalent subkey u_5 and partially decrypt the ciphertexts of the δ-set to obtain the sequence of 256 intermediate values of one byte of the state X_5. (For example, if the byte to be checked is byte 0, then the subkey bytes the adversary should guess are byte 0 of u_5 and bytes $0, 7, 10, 13$ of u_6).

(c) Check whether the sequence exists in the hash table. If not, discard the key guess.

The data complexity of the attack is 2^{32} chosen plaintexts. The time complexity of the online phase is relatively modest at 2^{80}, but the space complexity and the time complexity in encryption operations required to prepare this large table are about 2^{200}. These complexities are worse than exhaustive search for both AES-192 and AES-128. However, Demirci and Selçuk presented a tradeoff, which makes it possible to decrease the memory complexity at the expense of increasing both the data and the online time complexities. This results in an attack on 7-round AES-192 with data complexity of 2^{96} chosen plaintexts, and time and space complexities of 2^{144}.

The attack in [7] can be extended to 8-round AES-256 by guessing the full subkey of the last round. This increases the time complexity of the online phase from 2^{80} to 2^{208} encryptions, and makes it impossible to rebalance the parameters in order to attack 8-round AES-192[3].

4 Our New Techniques

In this section we present three new techniques. First, we present a new variant of Observation 3 which is stronger and simpler to analyze. Then we show how a combination of the δ-set analysis with a 4-round differential allows to reduce the memory complexity of the attack by a factor of 2^{57}. Finally, we show that for AES-192 and AES-256, the time complexity of the 8-round attack can be reduced using key schedule considerations by a factor of 2^{32} and 2^8, respectively.

4.1 The Multiset Variant of the Demirci-Selçuk Observation

We start with our new variant of Observation 3.

Observation 4. *Consider the encryption of a δ-set $\{P^0, P^1, \ldots, P^{255}\}$ through four full AES rounds.*

[3] We note that in a more recent paper, Demirci et al. [8] claim that by optimizing their technique they can also attack 7-round AES-128 faster than exhaustive search. However, we note that the analysis of [8] is flawed, and the correct running time of the attack is about 2^{32} times more than claimed, and in particular more than the complexity of exhaustive key search for the 128-bit key version.

For each $0 \leq l \leq 15$*, the (un-ordered) multiset*[4] $\left[X^0_{4,l} \oplus X^0_{4,l}, X^1_{4,l} \oplus X^0_{4,l}, \ldots, X^{255}_{4,l} \oplus X^0_{4,l}\right]$ *is fully determined by the following 24 byte parameters:*

- *The full 16-byte state* X^0_2*.*
- *Four bytes of the state* X^0_1*. (For example, if the active byte of the* δ*-set is byte* 0 *then these are bytes* $0, 1, 2, 3$*).*
- *Four bytes of the subkey* k_2*. (For example, if* $l = 0$ *then these are bytes* $0, 5, 10, 15$*).*

Moreover, this multiset can assume only 2^{184} *values (out of the* $\binom{511}{256} \approx 2^{507.6}$ *"theoretically possible" values).*

Our variant has several advantages over Observation 3:

- The parameters upon which the sequence depends are specified explicitly. This is crucial for the major reduction in the number of parameters presented in the next section.
- The smaller number of possible configurations in our variant (2^{184} instead of 2^{192}) allows to reduce the memory requirements of the attack and the time complexity of the preprocessing phase by a factor of 2^8.
- Since we consider a multiset instead of an ordered sequence, the adversary does not need to know the *order* of the values in the δ-set at the beginning of the four rounds. This allows to skip the guess of one byte in the subkey k_0 (reducing the time complexity of the online phase by 2^8).

Proof. The proof emphasizes the meet-in-the-middle nature of the observation.

We start with the "bottom side" of the four rounds. First, we observe that if $\{X^0_2, X^1_2, \ldots, X^{255}_2\}$ are known, then the knowledge of bytes $0, 5, 10, 15$ of k_2 yields the knowledge of the entire first column before the AddRoundKey of round 3 in all the 256 encryptions. Since the AddRoundKey preserves differences, this yields the desired values of the vector of differences $\left(X^0_{4,l} \oplus X^0_{4,l}, X^1_{4,l} \oplus X^0_{4,l}, \ldots, X^{255}_{4,l} \oplus X^0_{4,l}\right)$.

Secondly, to know the values $\{X^0_2, X^1_2, \ldots, X^{255}_2\}$, it is sufficient to know the value X^0_2 which is given as part of the parameters, and the differences $\left(X^0_2 \oplus X^0_2, X^1_2 \oplus X^0_2, \ldots, X^{255}_2 \oplus X^0_2\right)$. Since the ShiftRows, the MixColumns and the AddRoundKey operations are linear, it is sufficient to know the differences $\left(X^0_{1(SB)} \oplus X^0_{1(SB)}, X^1_{1(SB)} \oplus X^0_{1(SB)}, \ldots, X^{255}_{1(SB)} \oplus X^0_{1(SB)}\right)$.

Now we turn to the "top side" of the four rounds. In round 0, the differences $\left(X^0_{0(SB)} \oplus X^0_{0(SB)},\ X^1_{0(SB)} \oplus X^0_{0(SB)}, \ldots, X^{255}_{0(SB)} \oplus X^0_{0(SB)}\right)$ are known — these are exactly the 256 possible differences in byte 0 (the rest of the bytes are equal). Note that the *order* of the differences is not known, but this does not disturb the adversary since in our attack she is interested only in the multiset and not

[4] Unlike sets, elements can occur multiple times, and the multiset retains this multiplicity along with the values.

in the sequence. Since the ShiftRows, the MixColumns, and the AddRoundKey operations are linear, the differences $\left(X_1^0 \oplus X_1^0, X_1^1 \oplus X_1^0, \ldots, X_1^{255} \oplus X_1^0\right)$ are also known. By the structure of the δ-set, these differences are active in bytes $0, 1, 2, 3$ and passive in the rest of the bytes. Since bytes $0, 1, 2, 3$ of X_1^0 are given as part of the parameters, bytes $0, 1, 2, 3$ of the values $\{X_1^1, \ldots, X_1^{255}\}$ are thus also known, and so are bytes $0, 1, 2, 3$ of $\{X_{1(SB)}^0, X_{1(SB)}^1, \ldots, X_{1(SB)}^{255}\}$. Since the differences $X_{1(SB)}^j \oplus X_{1(SB)}^0$ in all the bytes except for $0, 1, 2, 3$ are zero for all $j = 1, 2, \ldots, 255$, this implies that the full vector of differences $\left(X_{1(SB)}^0 \oplus X_{1(SB)}^0, X_{1(SB)}^1 \oplus X_{1(SB)}^0, \ldots, X_{1(SB)}^{255} \oplus X_{1(SB)}^0\right)$ is known, as required above.

Finally, since the multiset depends on 24 byte parameters, it can assume at most 2^{192} possible values. However, in this count, each δ-set is represented by 2^8 multisets, according to the 256 possible choices of P^0. We can then reduce the number of parameters by one by choosing P^0 such that $X_{1,0}^0 = 0$ (this is possible since byte 0 in state X_1 is active). This reduces the number of possible multisets to 2^{184}, concluding the proof. □

4.2 The Differential Enumeration Technique

Observation 4 shows that the possible multisets depend on 24 explicitly stated parameters. 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. At first glance, it seems that the bytes in the intermediate states X_1^0 and X_2^0 also cannot be made equal to pre-determined constants by choosing the plaintexts appropriately, since they are separated from the plaintexts by operations involving an unknown key. However, we show that by using an expected-probability differential (i.e., a differential whose probability is not assumed to be especially high or especially low) for 4-round AES, the plaintext P^0 can be chosen such that the full 128-bit state X_2^0 will assume one of at most 2^{64} particular values (which can be computed in advance and are independent of the choice of key) instead of 2^{128} possible values.

Consider a truncated differential for four full AES rounds, in which both the input and the output differences are non-zero in a single byte (e.g., byte 0 both in the input and in the output). The probability of this differential is expected to be about 2^{-120},[5] and thus it is expected that 2^{120} randomly chosen pairs with difference only in byte 0 would contain one pair that satisfies the differential. Moreover, since each δ-set contains 2^{15} pairs with difference in a single byte, a collection of 2^{105} randomly chosen δ-sets in which byte 0 is active is expected to contain a right pair with respect to the differential. For right pairs, we show the following:

[5] The probability of 2^{-120} is based on the assumption that 4-round AES behaves like a random permutation with respect to this differential, and thus forcing 120 bits to be equal has this probability. If this is not the case, it is expected that other more powerful attacks on AES may exist.

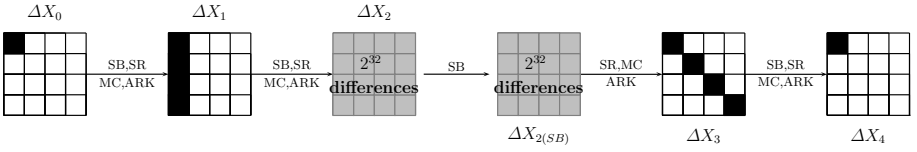


Fig. 3. The 4-Round Differential Characteristic Used in Our Attack

Observation 5. *Let (P^1, P^2) be a right pair with respect to the differential (i.e., the difference $P^1 \oplus P^2$ is non-zero only in byte 0, and the difference between the corresponding ciphertexts, $C^1 \oplus C^2$, is also non-zero only in byte 0). Then the intermediate state X_2^1 assumes one of at most 2^{64} prescribed values.*

Proof

The proof is a meet-in-the-middle argument. We start with the "top side" of the four rounds. Due to the structure of AES, the difference between the states $X^1_{1(SB)}$ and $X^2_{1(SB)}$ (i.e., the intermediate values after SubBytes of round 1) is non-zero only in bytes $0, 1, 2, 3$. Thus, this difference can assume at most 2^{32} distinct values. Since the ShiftRows, the MixColumns, and the AddRoundKey operations are linear, this implies that the difference $X_2^1 \oplus X_2^2$ can assume at most 2^{32} different values.

On the other hand, from the "bottom side" we see that the difference $X_3^1 \oplus X_3^2$ is non-zero only in bytes $0, 5, 10, 15$. Since the ShiftRows, the MixColumns, and the AddRoundKey operations are linear, this implies that the difference $X^1_{2(SB)} \oplus X^2_{2(SB)}$ can assume at most 2^{32} different values.

It is well-known that given the input and output differences of the SubBytes operation, there is one possibility on average for the actual pair of input/output values.[6] Moreover, this pair of actual values does not depend on the key, and can be easily found by precomputing the full difference distribution table of the SubBytes operation. Since for the right pair we consider, there are at most $2^{32} \cdot 2^{32} = 2^{64}$ possible pairs of input/output differences of the SubBytes operation in round 2, there are at most 2^{64} possible values of the full state X_2^1, as asserted.□

It follows from the observation that if we choose the δ-set such that P^0 is a member of a right pair with respect to this expected-probability differential, we are assured that the state X_2^0 can assume at most 2^{64} possible values. Moreover, since these values do not depend on the key and can be computed in advance, this allows to construct the "table of possible multisets" only for these 2^{64} values, which reduces the size of the table and the time complexity of the preprocessing phase by a huge factor of 2^{57} as shown below.

Three additional remarks are due:

– Firstly, in order to exploit the expected-probability differential we have to consider as many as 2^{113} chosen plain texts, which increases the data

[6] Actually, given the input/output differences, with probability of about 1/2 there are no such pairs, with probability of about 1/2 there are two pairs, and with probability of about 1/256 there are four pairs.

complexity of the attack. However, the resultant tradeoff is advantageous since the data complexity was smaller than the other complexities.
- Secondly, in order to detect the right pair with respect to the differential, the adversary has to guess several key bytes in the rounds before and after the differential. However, it turns out that if the differential is chosen such that the non-zero differences are in the bytes which are active in the δ-set, these key bytes coincide with the key bytes that should be guessed in the original Demirci-Selçuk attack. Hence, this does not increase the time complexity of the online phase of the attack.
- Finally, the total number of possible multisets after the combination with the differential is not $2^{184} \cdot 2^{-64} = 2^{120}$, but rather 2^{127}. The reason for this increase is that in the original attack, the number of multisets is reduced by a factor of 2^8 since each δ-set corresponds to 2^8 different multisets, according to the possible choices of P^0 (see proof of Observation 4). In the new version of the attack, we are forced to choose P^0 to be one of the members of the right pair w.r.t. the differential, and thus each δ-set corresponds to only two "special" multisets.[7] Therefore, the memory complexity and the time complexity of the preprocessing phase are reduced by a factor of 2^{57} rather than 2^{64}, compared to Observation 4.

4.3 The Key Bridging Technique

In this section we show that the time complexity of the online phase in the attacks on 8-round AES-192 and AES-256 can be reduced significantly by using key schedule considerations. While most of these considerations are simple, one of them is a novel observation that allows the adversary to deduce some subkey bytes from some other subkey bytes, even though they are separated by many key mixing steps.

We start with the attack on 8-round AES-192. Recall that in the online phase of this attack, the adversary has to guess four bytes of the subkey k_{-1}, one byte of the equivalent subkey u_5, four bytes of the equivalent subkey u_6, and the full k_7. The exact number of bytes that should be guessed depends on the choice of the active byte of the δ-set and of the byte in which the multiset is constructed. It turns out that if the byte to be examined at the end of round 4 is one of the bytes $1, 6, 11, 12$, then the number of guessed key bytes is reduced by three. Indeed, by the key schedule of AES-192, the knowledge of k_7 yields the

[7] We note that while the table of possible multisets is constructed according to one member of the right pair, it may occur that in the actual attack, the other member is chosen as P^0, and thus the multiset does not match the table (even for the right key guess). A simple solution is to repeat the attack for both members of the right pair. A more advanced solution, which allows to save the extra factor two in the time complexity of the attack, is to store the multisets only up to XOR with a constant value. This can be achieved by a small modification to the preprocessing phase, consisting of XORing each multiset with the 256 possible byte values and storing in the table the resulting multiset which is the least in the lexicographic order amongst the 256 possibilities.

knowledge of the first two columns of k_6 (and thus also of u_6) and of the last column of k_5 (and thus also of u_5).

If the byte to be checked at the end of round 4 is byte 1, then the bytes to guess are byte 13 of u_5, bytes $3, 6, 9, 12$ of u_6, and the full subkey k_7. However, once k_7 is guessed, bytes $3, 6$ of u_6 and byte 13 of u_5 can be computed from the key schedule, thus reducing the time complexity of the online phase of the attack by a factor of 2^{24}.

The complexity can be further reduced by another factor of 2^8 using the following novel observation:

Observation 6. *By the key schedule of AES-192, knowledge of columns* $0, 1, 3$ *of the subkey* k_7 *allows to deduce column* 3 *of the whitening key* k_{-1} *(which is actually Column* 3 *of the master key).*

The main novelty in this observation is that it exploits the weak key schedule of AES-192 in order to provide a surprisingly long "bridge" between two subkeys which are separated by 8 key mixing steps (applied in the reverse direction). In particular, it makes it possible to compute one byte in the whitening subkey k_{-1} directly from four bytes in the last subkey k_7,[8] which saves a factor of 2^8 in the time complexity of any attack which has to guess these five subkey bytes. Since guessing key material in the first and last round is a very common in attack, this observation can be widely applicable (e.g., it can reduce the time complexity of the related-key attack on 8-round AES-192 presented in [14] from 2^{180} to 2^{172}).

Proof. For the detailed proof and reasoning, we refer the reader to the full version of the paper. Given $W[32], W[33], W[35]$, it is possible to compute $W[27] = W[32] \oplus W[33]$ and $W[23] = W[33] \oplus W[35]$. From these two values, it is possible to compute $W[3] = W[27] \oplus SB(W[23] \lll 8) \oplus RCON[4]$. □

Since in the 8-round attack, one of the subkey bytes guessed by the adversary is included in the column $W[3]$ (regardless of the active byte in the δ-set), this reduces the time complexity by another factor of 2^8. In total, the key schedule considerations reduce the time complexity of the online phase of the attack on AES-192 by a factor of 2^{32}.

In the attack on 8-round AES-256, key schedule considerations can help the adversary only a little. By the key schedule, the subkey u_6 is independent of the subkey k_7, and thus the only subkey byte the adversary can retrieve is the single byte of u_5. As the novel observation does not hold for AES-256, key schedule arguments can reduce the time complexity only by a factor of 2^8.

5 Our New Attack on 7-Round AES

In this section we present our new attack on 7-round AES. For the sake of simplicity, we present here only the basic variant of the attack, which is used later

[8] The four bytes of k_7 are 0 and 4 (for obtaining byte 0 of $W[27]$) and bytes 7 and 15 (for obtaining byte 3 of $W[23]$).

as part of the 8-round attack. In Appendix A we show how to improve the attack using alteration of the expected-probability differential and time/memory/data tradeoffs, such that the resulting time complexity will be lower than the complexity of all previously known attacks on 7-round AES (in all its three flavors).

5.1 The Basic Attack

In this attack, the byte with non-zero difference in the expected-probability differential is byte 0, both in the input and in the output differences. The active byte of the δ-set and the byte that is checked in the state X_5 are taken to be byte 0 as well. The attack works similarly if these bytes are replaced by any other pair of bytes, as long as the correspondence between the differential and the δ-set is preserved.

The algorithm of the basic attack is as follows:

1. **Preprocessing phase:** Compute the 2^{127} possible values of the "special" multisets defined by Observations 4 and 5, and store them in a hash table.
2. **Online phase:**
   (a) **Phase A – Detecting the right pair**
      i. Ask for the encryption of 2^{81} structures of 2^{32} plaintexts, such that in each structure, bytes $0, 5, 10, 15$ assume the 2^{32} possible values and the rest of the bytes are constant.
      ii. For each structure, store the ciphertexts in a hash table and look for pairs in with no difference in bytes $1, 2, 3, 4, 5, 6, 8, 9, 11, 12, 14, 15$.[9] Since this is a 96-bit filtering, about 2^{48} pairs remain.
      iii. For each remaining pair, guess bytes $0, 5, 10, 15$ of k_{-1} and check whether the difference in the state X_1 is non-zero only in byte 0. For each key guess, about 2^{24} pairs are expected to remain.
      iv. For each remaining pair, guess bytes $0, 7, 10, 13$ of u_6 and check whether the difference in the state X_5 is non-zero only in byte 0. For each key guess, only one pair is expected to remain.
   (b) **Phase B – Checking the δ-set**
      i. For each guess of the eight subkey bytes made in Phase A and for the corresponding pair, take one of the members of the pair, denote it by P^0, and find its δ-set using the knowledge of bytes $0, 5, 10, 15$ of k_{-1}. (This is done by taking X_1^0, XORing it with the 255 possible values which are non-zero only in byte 0, and decrypting the 255 obtained values through round 0 using the known subkey bytes. The resulting plaintexts are the other members of the δ-set.)
      ii. Guess byte 0 of u_5, and using the knowledge of bytes $0, 7, 10, 13$ of u_6, partially decrypt the ciphertexts of the δ-set to obtain the multiset $\left[X_{5,0}^0 \oplus X_{5,0}^0, X_{5,0}^1 \oplus X_{5,0}^0, \ldots, X_{5,0}^{255} \oplus X_{5,0}^0\right]$.

[9] In the description of our attack we assume that the last round does not contain the MixColumns operation. If it does contain it, one can swap the order of the last round's MixColumns and AddRoundKey and apply the attack with the respective changes.

iii. Check whether the multiset exists in the hash table. If not, discard the key guess (possibly using auxiliary techniques such as repetition of the attack with a different output byte).

(c) **Exhaustively search the rest of the key:** For each remaining key guess, find the remaining key bytes by exhaustive search.

It is clear that the time complexity of the online phase of the attack is dominated by encrypting 2^{113} plaintexts, and hence, the data and time complexity of this part of the attack is 2^{113}. The memory complexity is 2^{129} 128-bit blocks, since each multiset contains about 512 bits of information and its representation can be easily compressed into 512 bits of space. The time complexity of the preprocessing phase of the attack is approximately $2^{127} \cdot 2^8 \cdot 2^{-3} = 2^{132}$ encryptions.

In Appendix A we show that the attack can be improved by altering the expected-probability differential, using several differentials in parallel, and applying time/memory/data tradeoffs. The resulting complexities lie on the following tradeoff curve: Data complexity – 2^{103+n} chosen plaintexts, Time complexity – 2^{103+n} encryptions, Memory requirement – 2^{129-n} AES blocks, for any $n \geq 0$. Choosing $n = 13$, all the three complexities are equalized at 2^{116}, which is lower than the time complexities of all known attacks on 7-round AES, in all its three flavors (see Table 1).

6 Extension to Attacks on 8-Round AES-192 and AES-256

In this section we present the first non-marginal attack on 8-round AES-192. The data complexity of the attack is 2^{113} chosen plaintexts, the memory requirement is 2^{129} 128-bit blocks, and the time complexity is 2^{172} encryptions. A variant of the attack can be applied to 8-round AES-256. The data and memory requirements remain unchanged, but the time complexity is increased to 2^{196} encryptions. We present the attack on AES-192; the attack on AES-256 is similar.

In the attack presented below, we choose the non-zero byte in the output difference of the expected-probability differential to be byte 1. Accordingly, the byte to be checked in the δ-set is also chosen as byte 1. This change is required in order to apply the key schedule considerations presented in Section 4.3. The only non-zero byte in the input difference of the differential and the only active byte of the δ-set can be still chosen arbitrarily, as long as they are the same. Without lose of generality, in the sequel we assume that this byte is byte 0.

A trivial generalization of the 7-round attack presented in Section 5 to eight rounds is to guess the full k_7, and for each guess, decrypt all the ciphertexts through the last round and apply the 7-round attack. In our attack this approach leads to an extremely high time complexity. Specifically, the detection of the right pair would require $2^{113} \cdot 2^{128} = 2^{241}$ encryptions. Instead, we use the *early abort* technique that was described in [12]. We present here the technique only briefly, and refer the reader to [12] for the full details.

Table 1. A Comparison of Previous Results with Our New Attacks

Rounds	Key Size	Complexity				Attack Type & Source
		Data (CP)	Memory	Time	MinMax*	
7	128	$2^{112.2}$	$2^{112.2}$	$2^{117.2}$ MA	$2^{117.2}$	Impossible Differential [12]
		$\mathbf{2}^{103+n}$	$\mathbf{2}^{129-n}$	$\mathbf{2}^{103-n}$	$\mathbf{2}^{116}$	**Our Results (Sect. 5)**
	192	$19 \cdot 2^{32}$	$19 \cdot 2^{32}$	2^{155}	2^{155}	SQUARE [10]
		2^{32}	2^{80}	2^{140}	2^{140}	Collision [11]
		2^{46+n}	2^{192-n}	2^{94+n}	2^{143}	Meet in the Middle [7]
		$2^{113.8}$	$2^{113.8}$	$2^{118.8}$ MA	$2^{118.8}$	Impossible Differential [12]
		$\mathbf{2}^{103+n}$	$\mathbf{2}^{129-n}$	$\mathbf{2}^{103+n}$	$\mathbf{2}^{116}$	**Our Results (Sect 5)**
	256	$21 \cdot 2^{32}$	$21 \cdot 2^{32}$	2^{172}	2^{172}	SQUARE [10]
		2^{32}	2^{80}	2^{140}	2^{140}	Collision [11]
		2^{34+n}	2^{204-n}	2^{82+n}	2^{143}	Meet in the Middle [7]
		$2^{113.8}$	$2^{113.8}$	$2^{118.8}$ MA	$2^{118.8}$	Impossible Differential [12]
		$\mathbf{2}^{103+n}$	$\mathbf{2}^{129-n}$	$\mathbf{2}^{103+n}$	$\mathbf{2}^{116}$	**Our Results (Sect 5)**
8	192	$2^{127.997}$	2^{128}	2^{188}	2^{188}	SQUARE [10]
		$\mathbf{2}^{113+n}$	$\mathbf{2}^{129-n}$	$\mathbf{2}^{172+n}$	$\mathbf{2}^{172}$	**Our Results (Sect. 6)**
	256	2^{34+n}	2^{206-n}	$2^{205.6+n}$	$2^{205.8}$	Meet in the Middle [7]†
		$2^{34+\max(n-24,0)}$	2^{208-n}	2^{206+n} MA	2^{208}	Meet in the Middle [8]‡
		$2^{89.1}$	2^{97}	$2^{229.7}$ MA	$2^{229.7}$	Impossible Differential [12]
		$2^{127.997}$	2^{128}	2^{204}	2^{204}	SQUARE [10]
		$\mathbf{2}^{113+n}$	$\mathbf{2}^{129-n}$	$\mathbf{2}^{196+n}$	$\mathbf{2}^{196}$	**Our Results (Sect. 6)**

* — the lowest time complexity which exceeds the other complexities via the tradeoff option (if available).
† — [7] estimates the cost of partial encryption as 2^{-8} of an encryption. As there are at least six columns which take part in the partial encryption/decryption, we believe that $2^{-2.4}$ is a more accurate estimate.
‡ — The complexity is higher than claimed in [8] due to a flaw in the analysis.
Time complexity measures the time in encryption units unless mentioned otherwise.
Memory complexity is measured in AES blocks.
CP — Chosen plaintext, MA — Memory Accesses.

In the following, the adversary examines each of the $2^{113} \cdot 2^{31} = 2^{144}$ pairs separately, and her goal is to detect the subkey candidates for which that pair satisfies the expected-probability differential.

Note that if (P^1, P^2) is a right pair, then the corresponding intermediate states $(X^1_{6(SR)}, X^2_{6(SR)})$ have non-zero difference only in bytes $3, 6, 9, 12$. Hence, in each column of $X_{6(SR)}$ there are only 2^8 possible differences. Since the MixColumns and AddRoundKey operations are linear, this implies that in each column of X_7 there are only 2^8 possible differences, and thus only $2^{32} \cdot 2^8 = 2^{40}$ possible pairs of actual values. In the technique presented in [12], the adversary considers these 2^{40} pairs in advance, encrypts them through round 7, and stores the actual values before the last AddRoundKey operation in a hash table, sorted by the output difference. In the online phase of the attack, for each examined pair, the adversary considers each shifted column (e.g., bytes $0, 7, 10, 13$) independently, and accesses the hash table in the row corresponding to the ciphertext difference. It is expected that $2^{40} \cdot 2^{-32} = 2^8$ values appear in each row. Since the table gives the actual values before the AddRoundKey operation, and the ciphertexts

are the values after that operation, each of the pairs in the table suggests one value for the 32-bit subkey corresponding to that shifted column.

Therefore, for each examined pair, and for each shifted column, the adversary obtains a list of 2^8 candidates for the 32-bit subkey corresponding to that column. In a basic variant of the attack, the adversary aggregates these suggestions to 2^{32} suggestions for the full k_7, and for each suggestion, she decrypts the ciphertext pair through round 7. Then she uses a similar precomputed table for round 6 to get a list of 2^8 possible values of bytes $3, 6, 9, 12$ of u_6. For each such value, the adversary checks whether the relations between bytes $3, 6$ of u_6 and the subkey k_7 described in Section 4.3 hold. If not, the subkey guess is discarded. Since this is a 16-bit filtering, the adversary is left with 2^{24} candidates for the full k_7 and bytes $3, 6, 9, 12$ of u_6. Finally, using a precomputed table also in round 0, the adversary obtains a list of 2^8 possible values of bytes $0, 5, 10, 15$ of k_{-1}. For each such value, the adversary checks whether the relation between byte 15 of k_{-1} and the subkey k_7 described in Section 4.3 holds. If not, the subkey guess is discarded. Since this is an 8-bit filtering, the adversary is left with 2^{24} candidates for the full k_7, bytes $3, 6, 9, 12$ of u_6, and bytes $0, 5, 10, 15$ of k_{-1}. For each of these candidates, (P^1, P^2) is a right pair w.r.t. the expected-probability differential, and the second-phase of the attack can be applied.

The time complexity of this procedure is 2^{40} simple operations for each examined pair, or $2^{144} \cdot 2^{40} \cdot 2^{-8} = 2^{176}$ encryptions in total.

The time complexity can be slightly reduced by using a more sophisticated precomputed table in order to check the consistency between bytes $3, 6$ of u_6 and the subkey k_7. The table takes bytes 3,6 of $MC^{-1}(X_6)$ in both pairs, along with bytes 2,3,5,6 of u_7, and returns the consistent values for bytes 3,6 of u_6, if there are any. The precomputation is done by trying all possible candidates for the pair of bytes for $MC^{-1}(X_6)$ along with the corresponding bytes of u_6, to see if the decrypted values satisfy the linear relation on the differences before the SubBytes operation of round 5. If this is the case, the entry corresponding to the $MC^{-1}(X_6)$ values and all subkeys of u_7 which satisfy the key relation is stored with the respective u_6 bytes. We note that for each key and each pair, there is probability of 2^{-8} that the condition is satisfied, and thus, only 2^{56} of the entries in the table are nonempty.

At the second part of the online phase of the attack, performed for each of the 2^{144} pairs (P^1, P^2) and each of the 2^{24} subkeys corresponding to the pair, the adversary constructs a δ-set and checks whether the corresponding multiset appears in the table. Note that while in the 7-round attack this phase requires guessing an additional subkey byte (which is byte 13 of u_5), in this attack that subkey byte can be derived from the subkey k_7. The time complexity of the second part is $2^{168} \cdot 2^8 \cdot 2^{-4} = 2^{172}$ encryptions.

Therefore, the overall memory requirement of the attack is 2^{129} 128-bit blocks (as in the basic version of the 7-round attack), the data complexity is 2^{113} chosen plaintexts, and the time complexity is 2^{172} encryptions. These complexities improve significantly over the only previously known attack on AES-192, which is a

SQUARE attack [10] requiring almost the entire codebook and time complexity of 2^{188} encryptions.

7 Summary

In this paper we introduced three new cryptanalytic techniques which can be used to improve the best known complexities of all the known attacks on 7 and 8 round versions of AES, as detailed in Table 1. In particular, we describe the first real attack on 8-round AES-192 which does not use the full codebook in order to marginally improve the time complexity of exhaustive search. However, all our attacks have impractical complexities, and thus they do not endanger the security of any fielded system.

References

1. Bahrak, B., Aref, M.R.: Impossible Differential Attack on 7-round AES-128. IET (IEE). J. on Information Security 2(2), 28–32 (2008)
2. 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, H. (ed.) Advances in Cryptology – EUROCRYPT 2010. LNCS, vol. 6110, pp. 299–319. Springer, Heidelberg (2010)
3. 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)
4. Biryukov, A., Khovratovich, D., Nikolic, I.: Distinguisher and Related-Key Attack on the Full AES-256. In: Halevi, S. (ed.) Advances in Cryptology - CRYPTO 2009. LNCS, vol. 5677, pp. 231–249. Springer, Heidelberg (2009)
5. Daemen, J., Rijmen, V.: AES Proposal: Rijndael, AES proposal (1998)
6. Daemen, J., Rijmen, V.: The design of Rijndael: AES — the Advanced Encryption Standard. Springer, Heidelberg (2002)
7. 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)
8. Demirci, H., Taskin, I., 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)
9. Dunkelman, O., Keller, N.: A New Attack on the LEX Stream Cipher. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 539–556. Springer, Heidelberg (2008)
10. Ferguson, N., Kelsey, J., Lucks, S., Schneier, B., Stay, M., Wagner, D., Whiting, D.: Improved Cryptanalysis of Rijndael. In: Schneier, B. (ed.) FSE 2000. LNCS, vol. 1978, pp. 213–230. Springer, Heidelberg (2001)
11. Gilbert, H., Minier, M.: A collision attack on 7 rounds of Rijndael. In: Proceedings of the Third AES Candidate Conference (AES3), New York, USA, pp. 230–241 (2000)
12. 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)

13. US National Institute of Standards and Technology, *Advanced Encryption Standard*, Federal Information Processing Standards Publications No. 197 (2001)
14. Zhang, W., Wu, W., Zhang, L., Feng, D.: Improved Related-Key Impossible Differential Attacks on Reduced-Round AES-192. In: Biham, E., Youssef, A.M. (eds.) SAC 2006. LNCS, vol. 4356, pp. 15–27. Springer, Heidelberg (2007)

A Improvements of the Attack on 7-Round AES

A.1 Altering the Expected-Probability Differential

Our first improvement reduces the data and time complexities of the attack by a factor of 2^8 without affecting the memory requirements.

We observe that the time complexity of most components of the attack is significantly lower than the time required to encrypt the plaintexts. Therefore, a tradeoff that would decrease the data complexity, even at the price of increasing the time complexity of the other parts of the attack, may reduce its overall complexity.

Such tradeoff is achieved by slightly modifying the expected-probability differential used in the attack. Instead of requiring the input difference to be non-zero only in byte 0, we can allow the difference to be non-zero also in one of the bytes $5, 10, 15$. These bytes are chosen such that the number of possible differences in the state X_2 is not increased, and thus the memory complexity is preserved.

This change reduces the data complexity of the attack to 2^{105}, since it allows the adversary to use structures of size 2^{16} that contain 2^{31} pairs with the input difference of the differential. On the other hand, the change requires to guess four additional bytes of k_{-1} in order to detect the right pair (if the additional byte is byte 5, then the additional guessed bytes are $3, 4, 9, 14$). As a result, the number of pairs remaining after the first filtering step of the attack is increased to 2^{72} (instead of 2^{48}). For each such pair, there are 2^{24} possible values of 12 subkey bytes (8 bytes of k_{-1} and 4 bytes of u_6) for which that pair satisfies the expected-probability differential. As in the 8-round attack, these values can be found with time complexity of 2^{24} table look-ups for each pair, using the early abort technique. Thus, the time complexity of Phase A of the modified attack is 2^{96} table look-ups.

In Phase B, we observe that since the value of bytes $3, 4, 9, 14$ of k_{-1} is irrelevant to the examination of the δ-set, the phase has to be performed only 2^{16} times for each of the 2^{72} pairs (instead of 2^{24} times). Thus, its time complexity is $2^{72} \cdot 2^{16} \cdot 2^8 \cdot 2^8 \cdot 2^{-3} = 2^{101}$ encryptions. Therefore, the overall time complexity of the attack is still dominated by the encryption of the plaintexts, and thus both the data and the time complexity of the attack are reduced to 2^{105}.

A.2 Using Several Differentials in Parallel

Our second improvement further reduces the data and time complexities by a factor of 5 without affecting the memory requirements.

We observe that the data complexity can be reduced by using several differentials in parallel. Since there is no specialty in the choice of the active byte at the input and the output of the original differential, there are 256 possible differentials that can be used in parallel. In the basic 7-round attack this improvement leads to a data/memory tradeoff: The attack requires the "active" bytes of the δ-set to correspond to the non-zero difference bytes of the differential, and altering the active bytes of the δ-set requires preparing a different precomputed table for each choice of the bytes. As a result, the data complexity can be reduced by factor of up to 256, but the memory requirement is increased by the same factor. Since the memory complexity is the dominant one in the 7-round attack, this tradeoff is not profitable.

However, in the modified attack the data complexity can be reduced by a small factor without affecting the memory complexity. We observe that since the additional "active" byte in the expected-probability differential is not used in the analysis of the δ-set, it can be chosen without affecting the memory complexity. There are six possible ways to choose this byte (bytes $5, 10, 15$ in the input and bytes $1, 2, 3$ in the output), and five of them can be used in parallel with the same set of chosen plaintexts.[10] This reduces the data complexity of the attack by a factor of 5 without affecting the memory complexity. Since the time complexity is dominated by encrypting the plaintexts, it is also reduced by a factor of 5. Therefore, the data and time complexities of the modified attack are smaller than 2^{103}. In the sequel, we assume for the sake of simplicity that these complexities are equal to 2^{103}.

A.3 Time/Memory/Data Tradeoffs

We conclude with a fine tuning of the complexities using a simple tradeoff between data, time, and memory as proposed in [7]. In the preprocessing phase, we precompute the table only for some of the values, and then for each key guess, we perform the attack for several δ-sets in order to compensate for the missing part of the table. For each $n \geq 0$, this tradeoff decreases the memory complexity and the time complexity of the preprocessing phase by a factor of 2^n, and increases the data complexity and the online time complexity by the same factor 2^n. The resulting complexities lie on the following tradeoff curve: Data complexity – 2^{103+n} chosen plaintexts, Time complexity – 2^{103+n} encryptions, Memory requirement – 2^{129-n} AES blocks, for any $n \geq 0$. Choosing $n = 13$, all the three complexities are equalized at 2^{116}, which is lower than the time complexities of all known attacks on 7-round AES, in all its three flavors.

[10] In order to do this, the adversary considers structures of size 2^{96} plaintexts each, in which bytes $1, 6, 11, 12$ are constant and the other bytes take all the 2^{96} possible values. This allows to use bytes 5 and 10 as the additional active byte in the input of the differential.

Constant-Size Commitments to Polynomials and Their Applications*

Aniket Kate[1], Gregory M. Zaverucha[2], and Ian Goldberg[3]

[1] Max Planck Institute for Software Systems (MPI-SWS)
[2] Certicom Research
[3] University of Waterloo

Abstract. We introduce and formally define polynomial commitment schemes, and provide two efficient constructions. A polynomial commitment scheme allows a committer to commit to a polynomial with a short string that can be used by a verifier to confirm claimed evaluations of the committed polynomial. Although the homomorphic commitment schemes in the literature can be used to achieve this goal, the sizes of their commitments are linear in the degree of the committed polynomial. On the other hand, polynomial commitments in our schemes are of constant size (single elements). The overhead of opening a commitment is also constant; even opening multiple evaluations requires only a constant amount of communication overhead. Therefore, our schemes are useful tools to reduce the communication cost in cryptographic protocols. On that front, we apply our polynomial commitment schemes to four problems in cryptography: verifiable secret sharing, zero-knowledge sets, credentials and content extraction signatures.

Keywords: Polynomial Commitments, Verifiable Secret Sharing, Zero-Knowledge Sets, Credentials.

1 Introduction

Commitment schemes are fundamental components of many cryptographic protocols. A commitment scheme allows a committer to publish a value, called the *commitment*, which binds her to a message (*binding*) without revealing it (*hiding*). Later, she may *open* the commitment and reveal the committed message to a verifier, who can check that the message is consistent with the commitment.

We review three well-known ways a committer can commit to a message. Let g and h be two random generators of a group $\mathbb{G}$ of prime order p. The committer can commit to a message $m \in_R \mathbb{Z}_p$ simply as $\mathcal{C}_{\langle g \rangle}(m) = g^m$. This scheme is unconditionally binding, and computationally hiding under the assumption that the discrete logarithm (DL) problem is hard in $\mathbb{G}$. The second scheme, known as a Pedersen commitment [31], is of the form $\mathcal{C}_{\langle g,h \rangle}(m, r) = g^m h^r$, where $r \in_R \mathbb{Z}_p$. Pedersen commitments are unconditionally hiding, and computationally binding

* An extended version of this paper is available [24]. This research was completed at the University of Waterloo.

M. Abe (Ed.): ASIACRYPT 2010, LNCS 6477, pp. 177–194, 2010.

under the DL assumption. Third, the committer may publish $H(m)$ or $H(m||r)$ for any one-way function H. In practice a collision-resistant hash function is often used. A survey by Damgård [16] covers commitment schemes in detail.

Now consider committing to a polynomial $\phi(x) \in_R \mathbb{Z}_p[x]$, a problem motivated by verifiable secret sharing. Suppose $\phi(x)$ has degree t and coefficients $\phi_0, \ldots, \phi_t$. We could commit to the string $(\phi_0|\phi_1|\ldots|\phi_t)$, or to some other unambiguous string representation of $\phi(x)$. Based on the commitment function used, this option may have a constant size commitment which uniquely determines $\phi(x)$. However, it limits the options for opening the commitment; opening must reveal the entire polynomial. This is not always suitable for cryptographic applications, most notably secret sharing, that require evaluations of the polynomial (*i.e.*, $\phi(i)$ for $i \in \mathbb{Z}_p$) be revealed to different parties or at different points in the protocol *without* revealing the entire polynomial. One solution is to commit to the coefficients, *e.g.*, $C = (g^{\phi_0}, \ldots, g^{\phi_t})$, which allows one to easily confirm that an opening $\phi(i)$ for index i is consistent with C. However, this has the drawback that the size of the commitment is now $t + 1$ elements of $\mathbb{G}$.

Our Contributions. The main contribution of this paper is an efficient scheme to commit to polynomials $\phi(x) \in \mathbb{Z}_p[x]$ over a bilinear pairing group, called $\mathsf{PolyCommit}_{\mathrm{DL}}$, with the following features. The size of the commitment is constant, a single group element. The committer can efficiently open the commitment to any correct evaluation $\phi(i)$ along with an element called the *witness*, which allows a verifier to confirm that $\phi(i)$ is indeed the evaluation at i of the polynomial $\phi(x)$. The construction is based on an algebraic property of polynomials $\phi(x) \in \mathbb{Z}_p[x]$ that $(x-i)$ *perfectly* divides the polynomial $\phi(x)-\phi(i)$ for any $i \in \mathbb{Z}_p$. The hiding property of the scheme is based on the DL assumption. The binding property of the main scheme is proven under the SDH assumption [6]. Using a technique similar to Pedersen commitments, we also define a stronger commitment scheme $\mathsf{PolyCommit}_{\mathsf{Ped}}$, which achieves unconditional hiding and computational binding under the SDH assumption.

When a set of evaluations $\{\phi(i) : i \in S\}$ is opened at the same time, what we term *batch opening*, the overhead still remains a *single* witness element. Security of batch opening assumes that the bilinear version of the SDH (BSDH) problem [21] is hard. Further, our schemes are homomorphic and easily randomizable. As in other work on reducing communication costs (*e.g.*, [8]) the global system parameters are somewhat large ($O(t)$ in our case). Reducing communication complexity (*i.e.*, the number of bits transferred) is our goal, and to this end we apply the PolyCommit schemes to the following four applications.

First we apply the PolyCommit schemes to the Feldman verifiable secret sharing (VSS) protocol [18]. The new VSS protocol requires a broadcast with size $O(1)$ as compared to $O(n)$ required in the best known protocols in the literature (where n is the number of participants) [18, 31].

Second, we define and use the PolyCommit schemes to construct a relaxed type of zero-knowledge set (ZKS) [27]. A ZKS is a commitment to a set S, such that the committer may prove that $i \in S$, or $i \notin S$ without revealing additional information about S, not even $|S|$. We define *nearly zero-knowledge sets* as ZKS

that do not attempt to hide the size of the committed set. This is sufficient for most applications of zero-knowledge sets, and our construction has constant size proofs of (non)membership as compared to the best known constructions of ZKS that require non-constant communication [12, 25]. We apply the same relaxation to elementary zero-knowledge databases (ZK-EDB), and achieve constant communication there as well.

In the next application we leverage the efficiency of batch opening, by using the PolyCommit schemes in an efficient general construction of a *content extraction signature* (CES) scheme [35]. A CES scheme allows a signature holder to extract signatures for subsections of the signed message. The general construction, when instantiated with our commitment scheme and a general secure signature scheme, is as efficient as the best known CES scheme, which relies on specific properties of the RSA signature scheme.

In the special case when the CES scheme is used to authenticate a list of attributes, the result is a digital credential with an efficient *selective show* operation. A selective show allows the credential holder to reveal only a subset of the attributes, with proof that the revealed attributes are signed. More precisely, the communication cost of revealing k attributes in a credential with t attributes is $O(k)$, while known credential systems must communicate $O(t)$ bits. We also show how to efficiently prove knowledge of committed values, allowing predicates on attributes to be proven in zero-knowledge (also with complexity $O(k)$).

Outline. In the rest of this section, we compare our contributions with related work (work related to each application is included in the respective subsection). In §2, we cover some preliminary material and describe our cryptographic assumptions. §3 defines polynomial commitments and presents our constructions. §4 is devoted to applications while §5 presents some open problems. Due to space constraints, all security proofs are included in the extended version [24].

Related Work. Similar to our scheme, a Merkle hash tree [26] allows many values to be committed to with a single element. Here, the leaves of a binary tree are the messages. Each non-leaf node has the value $H(L||R)$ where L and R are its children, and H is a collision-resistant hash function. One can open the commitment to an individual message by revealing the message, and a path up the tree to the root. The opening has size $O(\log n)$ as compared to $O(1)$ in our scheme, where n is the total number of (leaf) elements.

Chase *et al.* [13] introduce mercurial commitments to construct ZKS, which eventually led to the commitment schemes for committing to a vector of messages [12, 25]. Catalano *et al.* [12], and Libert and Yung [25] construct vector commitment schemes under the name *trapdoor t-mercurial commitments*. The security of both of these commitment schemes is based on SDH-like assumptions and their system parameters have size $O(t)$, as in our scheme. In [12], all messages must be revealed when opening, while in [25], the committer may open a commitment to a single message. However, in [25], it is not possible to have arbitrary indices for committed messages since each of the t committed messages

is associated with a value in the system parameters g^{α^j} for $j \in [1, t]$. Our scheme have no such restriction on the domain for the indices, offering greater flexibility.

Another related primitive is an *accumulator* [3], which aggregates a large set of input elements into a single element and can provide a witness as evidence that an element is included in the accumulator. Further, it is possible to use a witness to prove (in zero-knowledge) that the element is included in the accumulator. Camenisch and Lysyanskaya [10] extend the concept to *dynamic accumulators*, which support efficient updates. Au *et al.* [1] observe that a paring-based accumulator by Nguyen [29] is a *bounded* accumulator, *i.e.*, only a fixed number of elements can be accumulated. They then go on to use bounded accumulators to construct a compact e-cash scheme [2]. However, the accumulated elements in this scheme are not ordered, which makes it inappropriate for accumulating polynomials. While our PolyCommit schemes provide the same features as non-dynamic accumulators, they have additional features (hiding and batch opening) and are more general since we can commit to a polynomial instead of a set.

2 Preliminaries

In what follows, all adversaries are probabilistic polynomial time (PPT) algorithms with respect to a security parameter κ expect if stated otherwise. Further, they are *static* and they have to choose their nodes before protocol instances start. A function $\epsilon(\cdot) : \mathbb{N} \to \mathbb{R}^+$ is called *negligible* if for all $c > 0$ there exists a k_0 such that $\epsilon(k) < 1/k^c$ for all $k > k_0$. In the rest of the paper, $\epsilon(\cdot)$ will always denote a negligible function. We use the notation $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ to denote a symmetric (type 1) bilinear pairing in groups of prime order $p \geq 2^{2\kappa}$. The choice of type 1 pairings was made to simplify presentation, however, our constructions can easily be modified to work with pairings of types 2 and 3 as well. For details of bilinear pairings, see the extended version of the paper.

We use the *discrete logarithm* (DL) assumption [26, Chap. 3], and the *t-strong Diffie-Hellman* (t-SDH) assumption [6] to prove the security of the $\mathsf{PolyCommit}_{\mathrm{DL}}$ and $\mathsf{PolyCommit}_{\mathsf{Ped}}$ schemes. Security of two additional properties of the schemes require a generalization of the *t-Diffie-Hellman inversion* (t-DHI) assumption [28, 5], and the bilinear version of t-SDH, the t-BSDH assumption [21].

Definition 1. Discrete logarithm (DL) Assumption. *Given a generator g of $\mathbb{G}^*$, where $\mathbb{G}^* = \mathbb{G}$ or $\mathbb{G}_T$, and a $\in_R \mathbb{Z}_p^*$, for every adversary $\mathcal{A}_{DL}$,* $\Pr[\mathcal{A}_{DL}(g, g^a) = a] = \epsilon(\kappa)$.

Mitsunari, Sakai and Kasahara [28] introduced the *weak Diffie-Hellman assumption*, which was renamed the t-DHI assumption by Boneh and Boyen [5] as this assumption is stronger than the Diffie-Hellman assumption, especially for large values of t. See Cheon [14] for a security analysis.

The t-DHI problem is, on input $\langle g, g^{\alpha}, \ldots, g^{\alpha^t} \rangle \in \mathbb{G}^{t+1}$ to output $g^{1/\alpha}$, or equivalently (see [7]), $g^{\alpha^{t+1}}$. In this paper, we use a generalization of the t-DHI assumption, where $\mathcal{A}$ has to output a pair $\langle \phi(x), g^{\phi(\alpha)} \rangle \in \mathbb{Z}_p[x] \times \mathbb{G}$ such that

$2^\kappa > \deg(\phi) > t$. We call this assumption the *t-polynomial Diffie-Hellman* (t-polyDH) assumption. This assumption was implicitly made by [1, 2] to support their claim that the accumulator of [29] is bounded.[1]

Definition 2. ***t*-polynomial Diffie-Hellman (*t*-polyDH) Assumption.** *Let $\alpha \in_R \mathbb{Z}_p^*$. Given as input a $(t+1)$-tuple $\langle g, g^\alpha, \ldots, g^{\alpha^t}\rangle \in \mathbb{G}^{t+1}$, for every adversary $\mathcal{A}_{t\text{-}polyDH}$, the probability $\Pr[\mathcal{A}_{t\text{-}polyDH}(g, g^\alpha, \ldots, g^{\alpha^t}) = \langle \phi(x), g^{\phi(\alpha)}\rangle] = \epsilon(\kappa)$ for any freely chosen $\phi(x) \in \mathbb{Z}_p[x]$ such that $2^\kappa > \deg(\phi) > t$.*

Boneh and Boyen [6] defined the t-SDH assumption that is related to but stronger than the t-DHI assumption and has exponentially many non-trivially different solutions, all of which are acceptable.

Definition 3. ***t*-Strong Diffie-Hellman (*t*-SDH) Assumption.** *Let $\alpha \in_R \mathbb{Z}_p^*$. Given as input a $(t+1)$-tuple $\langle g, g^\alpha, \ldots, g^{\alpha^t}\rangle \in \mathbb{G}^{t+1}$, for every adversary $\mathcal{A}_{t\text{-}SDH}$, the probability $\Pr[\mathcal{A}_{t\text{-}SDH}(g, g^\alpha, \ldots, g^{\alpha^t}) = \langle c, g^{\frac{1}{\alpha+c}}\rangle] = \epsilon(\kappa)$ for any value of $c \in \mathbb{Z}_p \backslash \{-\alpha\}$.*

Security of the batch opening extension of our commitment schemes requires the bilinear version of the t-SDH assumption, the t-BSDH assumption [21].

Definition 4. ***t*-Bilinear Strong Diffie-Hellman (*t*-BSDH) Assumption.** *Let $\alpha \in_R \mathbb{Z}_p^*$. Given as input a $(t+1)$-tuple $\langle g, g^\alpha, \ldots, g^{\alpha^t}\rangle \in \mathbb{G}^{t+1}$, for every adversary $\mathcal{A}_{t\text{-}BSDH}$, the probability $\Pr[\mathcal{A}_{t\text{-}BSDH}(g, g^\alpha, \ldots, g^{\alpha^t}) = \langle c, e(g,g)^{\frac{1}{\alpha+c}}\rangle] = \epsilon(\kappa)$ for any value of $c \in \mathbb{Z}_p \backslash \{-\alpha\}$.*

A similar assumption was also made in [22], but with a different solution: $\langle c, e(g,h)^{1/(\alpha+c)}\rangle$, where $h \in_R G$ is an additional system parameter.

3 Polynomial Commitments

In this section we provide a formal definition of a polynomial commitment scheme, followed by two constructions. In the first construction the commitments are computationally hiding, while in the second they are unconditionally hiding. We also discuss some useful features of our constructions.

3.1 Definition

A polynomial commitment scheme consists of six algorithms: Setup, Commit, Open, VerifyPoly, CreateWitness, and VerifyEval.

[1] Note that we bound $\deg(\phi)$ by 2^κ as evaluations can be found for polynomials with higher degrees in PPT using number theoretic techniques (*e.g.*, for $\phi(x) = x^{p-1}$, $g^{\phi(\alpha)} = g$ for any $\alpha \in \mathbb{Z}_p^*$). In practice, $\deg(\phi) \ll 2^\kappa$.

Setup$(1^\kappa, t)$ generates an appropriate algebraic structure $\mathcal{G}$ and a commitment public-private key pair $\langle \mathsf{PK}, \mathsf{SK} \rangle$ to commit to a polynomial of degree $\leq t$. For simplicity, we add $\mathcal{G}$ to the public key PK. Setup is run by a trusted or distributed authority. Note that SK is not required in the rest of the scheme.

Commit$(\mathsf{PK}, \phi(x))$ outputs a commitment $\mathcal{C}$ to a polynomial $\phi(x)$ for the public key PK, and some associated decommitment information d. (In some constructions, d is null.)

Open$(\mathsf{PK}, \mathcal{C}, \phi(x), d)$ outputs the polynomial $\phi(x)$ used while creating the commitment, with decommitment information d.

VerifyPoly$(\mathsf{PK}, \mathcal{C}, \phi(x), d)$ verifies that $\mathcal{C}$ is a commitment to $\phi(x)$, created with decommitment information d. If so it outputs 1, otherwise it outputs 0.

CreateWitness$(\mathsf{PK}, \phi(x), i, d)$ outputs $\langle i, \phi(i), w_i \rangle$, where w_i is a witness for the evaluation $\phi(i)$ of $\phi(x)$ at the index i and d is the decommitment information.

VerifyEval$(\mathsf{PK}, \mathcal{C}, i, \phi(i), w_i)$ verifies that $\phi(i)$ is indeed the evaluation at the index i of the polynomial committed in $\mathcal{C}$. If so it outputs 1, otherwise it outputs 0.

Note that it is possible to commit to a list of messages $(m_1, \ldots, m_{t+1})$ by associating each to a unique key (index) $k_1, \ldots, k_{t+1}$ in $\mathbb{Z}_p$, and interpolating to find $\phi(x) \in \mathbb{Z}_p[x]$, such that $\deg(\phi) \leq t$ and $\phi(k_j) = m_j$.

In terms of security, a malicious committer should not be able to convincingly present two different values as $\phi(i)$ with respect to $\mathcal{C}$. Further, until more than $\deg(\phi)$ points are revealed, the polynomial should remain hidden. Next, we formally define the security and correctness of a polynomial commitment.

Definition 5. *(Setup, Commit, Open, VerifyPoly, CreateWitness, and VerifyEval) is a* secure polynomial commitment scheme *if it satisfies the following properties.*

Correctness. *Let* $\mathsf{PK} \leftarrow \mathsf{Setup}(1^\kappa)$ *and* $\mathcal{C} \leftarrow \mathsf{Commit}(\mathsf{PK}, \phi(x))$. *For a commitment* $\mathcal{C}$ *output by* $\mathsf{Commit}(\mathsf{PK}, \phi(x))$, *and all* $\phi(x) \in \mathbb{Z}_p[x]$,

- *the output of* $\mathsf{Open}(\mathsf{PK}, \mathcal{C}, \phi(x))$ *is successfully verified by* $\mathsf{VerifyPoly}(\mathsf{PK}, \mathcal{C}, \phi(x))$, *and,*
- *any* $\langle i, \phi(i), w_i \rangle$ *output by* $\mathsf{CreateWitness}(\mathsf{PK}, \phi(x), i)$ *is successfully verified by* $\mathsf{VerifyEval}(\mathsf{PK}, \mathcal{C}, i, \phi(i), w_i)$.

Polynomial Binding. *For all adversaries* $\mathcal{A}$*:*

$$\Pr\left(\begin{array}{c}\mathsf{PK} \leftarrow \mathsf{Setup}(1^\kappa),\ (\mathcal{C}, \langle \phi(x), \phi'(x) \rangle) \leftarrow \mathcal{A}(\mathsf{PK}) : \\ \mathsf{VerifyPoly}(\mathsf{PK}, \mathcal{C}, \phi(x)) = 1 \wedge \\ \mathsf{VerifyPoly}(\mathsf{PK}, \mathcal{C}, \phi'(x)) = 1 \ \wedge \phi(x) \neq \phi'(x)\end{array}\right) = \epsilon(\kappa).$$

Evaluation Binding. *For all adversaries* $\mathcal{A}$*:*

$$\Pr\left(\begin{array}{c}\mathsf{PK} \leftarrow \mathsf{Setup}(1^\kappa),\ (\mathcal{C}, \langle i, \phi(i), w_i \rangle, \langle i, \phi(i)', w_i' \rangle) \leftarrow \mathcal{A}(\mathsf{PK}) : \\ \mathsf{VerifyEval}(\mathsf{PK}, \mathcal{C}, i, \phi(i), w_i) = 1 \wedge \\ \mathsf{VerifyEval}(\mathsf{PK}, \mathcal{C}, i, \phi(i)', w_i') = 1 \ \wedge \phi(i) \neq \phi(i)'\end{array}\right) = \epsilon(\kappa).$$

Hiding. *Given* $\langle \mathsf{PK}, \mathcal{C} \rangle$ *and* $\{\langle i_j, \phi(i_j), w_{\phi_{i_j}} \rangle : j \in [1, \deg(\phi)]\}$ *for a polynomial* $\phi(x) \in_R \mathbb{Z}_p[x]$ *such that* $\mathsf{VerifyEval}(\mathsf{PK}, \mathcal{C}, i_j, \phi(i_j), w_{\phi_{i_j}}) = 1$ *for each* j,

- *no adversary $\mathcal{A}$ can determine $\phi(\hat{i})$ with non-negligible probability for any unqueried index $\hat{i}$ (computational hiding) or*
- *no computationally unbounded adversary $\hat{\mathcal{A}}$ has any information about $\phi(\hat{i})$ for any unqueried index $\hat{i}$ (unconditional hiding).*

3.2 Construction: $\mathsf{PolyCommit}_{\mathrm{DL}}$

We now provide an efficient construction of a polynomial commitment scheme. $\mathsf{PolyCommit}_{\mathrm{DL}}$ is based on an algebraic property of polynomials $\phi(x) \in \mathbb{Z}_p[x]$: $(x-i)$ perfectly divides the polynomial $\phi(x) - \phi(i)$ for $i \in \mathbb{Z}_p$. In the literature, Herzberg *et al.* [23] have used this technique in their share recovery scheme.

Setup$(1^\kappa, t)$ computes two groups $\mathbb{G}$, and $\mathbb{G}_T$ of prime order p (providing κ-bit security) such that there exists a symmetric bilinear pairing $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ and for which the t-SDH assumption holds. We denote the generated bilinear pairing group as $\mathcal{G} = \langle e, \mathbb{G}, \mathbb{G}_t \rangle$. Choose a generator $g \in_R \mathbb{G}$. Let $\alpha \in_R \mathbb{Z}_p^*$ be SK, generated by a (possibly distributed) trusted authority. Setup also generates a $(t+1)$-tuple $\langle g, g^\alpha, \ldots, g^{\alpha^t} \rangle \in \mathbb{G}^{t+1}$ and outputs $\mathsf{PK} = \langle \mathcal{G}, g, g^\alpha, \ldots, g^{\alpha^t} \rangle$. SK is not required in the rest of the construction.

Commit$(\mathsf{PK}, \phi(x))$ computes the commitment $\mathcal{C} = g^{\phi(\alpha)} \in \mathbb{G}$ for polynomial $\phi(x) \in \mathbb{Z}_p[X]$ of degree t or less. For $\phi(x) = \sum_{j=0}^{\deg(\phi)} \phi_j x^j$, it outputs $\mathcal{C} = \prod_{j=0}^{\deg(\phi)} (g^{\alpha^j})^{\phi_j}$ as the commitment to $\phi(x)$.

Open$(\mathsf{PK}, \mathcal{C}, \phi(x))$ outputs the committed polynomial $\phi(x)$.

VerifyPoly$(\mathsf{PK}, \mathcal{C}, \phi(x))$ verifies that $\mathcal{C} \stackrel{?}{=} g^{\phi(\alpha)}$. If $\mathcal{C} = \prod_{j=0}^{\deg(\phi)} (g^{\alpha^j})^{\phi_j}$ for $\phi(x) = \sum_{j=0}^{\deg(\phi)} \phi_j x^j$ the algorithm outputs 1, else it outputs 0. Note that this only works when $\deg(\phi) \leq t$.

CreateWitness$(\mathsf{PK}, \phi(x), i)$ computes $\psi_i(x) = \frac{\phi(x) - \phi(i)}{(x-i)}$ and outputs $\langle i, \phi(i), w_i \rangle$, where the witness $w_i = g^{\psi_i(\alpha)}$ is computed in a manner similar to $\mathcal{C}$, above.

VerifyEval$(\mathsf{PK}, \mathcal{C}, i, \phi(i), w_i)$ verifies that $\phi(i)$ is the evaluation at the index i of the polynomial committed to by $\mathcal{C}$. If $e(\mathcal{C}, g) \stackrel{?}{=} e(w_i, g^\alpha / g^i) e(g, g)^{\phi(i)}$, the algorithm outputs 1, else it outputs 0.

$\mathsf{VerifyEval}$ is correct because

$$\begin{aligned} e(w_i, g^\alpha / g^i) e(g, g)^{\phi(i)} &= e(g^{\psi_i(\alpha)}, g^{(\alpha - i)}) e(g, g)^{\phi(i)} = e(g, g)^{\psi_i(\alpha)(\alpha - i) + \phi(i)} \\ &= e(g, g)^{\phi(\alpha)} = e(\mathcal{C}, g) \text{ as } \phi(x) = \psi_i(x)(x - i) + \phi(i) \end{aligned}$$

Theorem 1. *$\mathsf{PolyCommit}_{DL}$ is a secure polynomial commitment scheme (as defined in Definition 5) provided the DL and t-SDH assumptions hold in $\mathcal{G}$.*

A proof is provided in the extended version. The proof of the binding property uses the t-SDH assumption, while the proof of the hiding property uses the DL assumption.

3.3 Construction: $\mathsf{PolyCommit}_{\mathsf{Ped}}$

$\mathsf{PolyCommit}_{\mathsf{Ped}}$ is also based on the same algebraic property of $\phi(x) \in \mathbb{Z}_p[x]$: $(x-i)$ perfectly divides the polynomial $\phi(x) - \phi(i)$ for $i \in \mathbb{Z}_p$; however, it uses an additional random polynomial $\hat{\phi}(x)$ to achieve unconditional hiding.

The $\mathsf{PolyCommit}_{\mathrm{DL}}$ scheme is homomorphic in nature. Given $\mathsf{PolyCommit}_{\mathrm{DL}}$ commitments $\mathcal{C}_{\phi_1}$ and $\mathcal{C}_{\phi_2}$ associated with polynomials $\phi_1(x)$ and $\phi_2(x)$ respectively, one can compute the commitment $\mathcal{C}_\phi$ for $\phi(x) = \phi_1(x) + \phi_2(x)$ as $\mathcal{C}_\phi = \mathcal{C}_{\phi_1}\mathcal{C}_{\phi_2}$. Further, given two witness-tuples $\langle i, \phi_1(i), w_{\phi_1 i}\rangle$ and $\langle i, \phi_2(i), w_{\phi_2 i}\rangle$ at index i associated with polynomials $\phi_1(x)$ and $\phi_2(x)$ respectively, the corresponding tuple for polynomial $\phi(x)$ can be given as $\langle i, \phi_1(i) + \phi_2(i), w_{\phi_1 i} w_{\phi_2 i}\rangle$. The $\mathsf{PolyCommit}_{\mathsf{Ped}}$ construction uses the homomorphic property to combine two commitments (one to $\phi(x)$, one to $\hat{\phi}(x)$), although each commitment uses a different generator. Next, we define our $\mathsf{PolyCommit}_{\mathsf{Ped}}$ construction.

Setup$(1^\kappa, t)$ computes two groups $\mathbb{G}$ and $\mathbb{G}_T$ of prime order p (providing κ-bit security) such that there exists a symmetric bilinear pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ and for which the t-SDH assumption holds. We denote the generated bilinear pairing group as $\mathcal{G} = \langle e, \mathbb{G}, \mathbb{G}_t\rangle$. Choose two generators $g, h \in_R \mathbb{G}$. Let $\alpha \in_R \mathbb{Z}_p^*$ be SK, generated by a (possibly distributed) trusted authority. Setup also generates a $(2t+2)$-tuple $\langle g, g^\alpha, \ldots, g^{\alpha^t}, h, h^\alpha, \ldots, h^{\alpha^t}\rangle \in \mathbb{G}^{2t+2}$ and outputs $\mathsf{PK} = \langle \mathcal{G}, g, g^\alpha, \ldots, g^{\alpha^t}, h, h^\alpha, \ldots, h^{\alpha^t}\rangle$. Similar to $\mathsf{PolyCommit}_{\mathrm{DL}}$, SK is not required by the other algorithms of the commitment scheme.

Commit$(\mathsf{PK}, \phi(x))$ chooses $\hat{\phi}(x) \in_R \mathbb{Z}_p[x]$ of degree t and computes the commitment $\mathcal{C} = g^{\phi(\alpha)} h^{\hat{\phi}(\alpha)} \in \mathbb{G}$ for the polynomial $\phi(x) \in \mathbb{Z}_p[X]$ of degree t or less. For $\phi(x) = \sum_{j=0}^{\deg(\phi)} \phi_j x^j$ and $\hat{\phi}(x) = \sum_{j=0}^{\deg(\hat{\phi})} \hat{\phi}_j x^j$, it outputs $\mathcal{C} = \prod_{j=0}^{\deg(\phi)} (g^{\alpha^j})^{\phi_j} \prod_{j=0}^{\deg(\hat{\phi})} (h^{\alpha^j})^{\hat{\phi}_j}$ as the commitment to $\phi(x)$.

Open$(\mathsf{PK}, \mathcal{C}, \phi(x), \hat{\phi}(x))$ outputs the committed polynomials $\phi(x)$ and $\hat{\phi}(x)$.

VerifyPoly$(\mathsf{PK}, \mathcal{C}, \phi(x), \hat{\phi}(x))$ verifies that $\mathcal{C} \stackrel{?}{=} g^{\phi(\alpha)} h^{\hat{\phi}(\alpha)}$. If $\mathcal{C} = \prod_{j=0}^{\deg(\phi)} (g^{\alpha^j})^{\phi_j} \prod_{j=0}^{\deg(\hat{\phi})} (h^{\alpha^j})^{\hat{\phi}_j}$ for $\phi(x) = \sum_{j=0}^{\deg(\phi)} \phi_j x^j$ and $\hat{\phi}(x) = \sum_{j=0}^{\deg(\hat{\phi})} \hat{\phi}_j x^j$, it outputs 1, else it outputs 0. This only works when both $\deg(\phi)$ and $\deg(\hat{\phi}) \leq t$.

CreateWitness$(\mathsf{PK}, \phi(x), \hat{\phi}(x), i)$ calculates $\psi_i(x) = \frac{\phi(x) - \phi(i)}{(x-i)}$ and $\hat{\psi}_i(x) = \frac{\hat{\phi}(x) - \hat{\phi}(i)}{(x-i)}$, and outputs $\langle i, \phi(i), \hat{\phi}(i), w_i\rangle$. Here, the witness $w_i = g^{\psi_i(\alpha)} h^{\hat{\psi}_i(\alpha)}$.

VerifyEval$(\mathsf{PK}, \mathcal{C}, i, \phi(i), \hat{\phi}(i), w_i)$ verifies that $\phi(i)$ is the evaluation at the index i of the polynomial committed to by $\mathcal{C}$. If $e(\mathcal{C}, g) \stackrel{?}{=} e(w_i, g^\alpha / g^i) e(g^{\phi(i)} h^{\hat{\phi}(i)}, g)$, the algorithm outputs 1, else it outputs 0.

In the extended version we show $\mathsf{PolyCommit}_{\mathsf{Ped}}$ is correct and prove the following security theorem.

Theorem 2. *$\mathsf{PolyCommit}_{\mathsf{Ped}}$ is a secure polynomial commitment scheme (as defined in Definition 5) provided the t-SDH assumption holds in $\mathcal{G}$.*

The proof of the binding property is based on the t-SDH assumption, while the hiding property is unconditional.

3.4 Features

We next discuss some important features of $\mathsf{PolyCommit}_{\mathrm{DL}}$ and $\mathsf{PolyCommit}_{\mathsf{Ped}}$.

Homomorphism. In §3.3, we describe that the $\mathsf{PolyCommit}_{\mathrm{DL}}$ scheme is (additive) homomorphic in nature. In the full version we show that $\mathsf{PolyCommit}_{\mathsf{Ped}}$ is also homomorphic.

Unconditional Hiding for $\mathsf{PolyCommit}_{DL}$. When $t' < \deg(\phi)$ evaluations have been revealed, $\mathsf{PolyCommit}_{\mathrm{DL}}$ unconditionally hides any unrevealed evaluation, since the $t'+1$ evaluations $\langle \alpha, \phi(\alpha)\rangle, \langle i_1, \phi(i_1)\rangle, \ldots, \langle i_{t'}, \phi(i_{t'})\rangle$ are insufficient to interpolate a polynomial of degree $> t'$. Note that the evaluation pair $\langle \alpha, \phi(\alpha)\rangle$ is available in an exponentiated form $\langle g^{\alpha}, g^{\phi(\alpha)}\rangle$.

Indistinguishability of Commitments. When a polynomial commitment scheme is randomized, an unbounded adversary cannot distinguish commitments to chosen sets of key-value pairs. When committing to a set of key-value pairs $(\langle k_1, m_1\rangle, \ldots, \langle k_{t+1}, m_{t+1}\rangle)$, if indistinguishable $\mathsf{PolyCommit}_{\mathrm{DL}}$ commitments are required, it is sufficient to set one m_i to a random value. On the other hand, the $\mathsf{PolyCommit}_{\mathsf{Ped}}$ scheme is inherently randomized and can be used directly.

Trapdoor Commitment. The constructions are also trapdoor commitment schemes, where $\mathsf{SK} = \alpha$ is the trapdoor. Refer to the extended version for details.

Batch Opening. In the case when multiple evaluations in a $\mathsf{PolyCommit}_{\mathrm{DL}}$ commitment must be opened, the opening may be batched to reduce the computation and the communication of both the committer and the verifier; *i.e.*, opening multiple evaluations at the same time is cheaper than opening each of those evaluations individually using $\mathsf{CreateWitness}$ and $\mathsf{VerifyEval}$. Let $B \subset \mathbb{Z}_p$, $|B| < t$ be a set of indices to be opened, for a commitment $\mathcal{C} = g^{\phi(\alpha)}$ created using $\mathsf{PolyCommit}_{\mathrm{DL}}$. The witness for the values $\phi(i)$, for all $i \in B$, is computed as $w_B = g^{\psi_B(\alpha)}$ for the polynomial $\psi_B(x) = \frac{\phi(x) - r(x)}{\prod_{i \in B}(x-i)}$ where $r(x)$ is the remainder of the polynomial division $\phi(x)/(\prod_{i\in B}(x-i))$; *i.e.*, $\phi(x) = \psi_B(x)\left(\prod_{i\in B}(x-i)\right) + r(x)$ and for $i \in B$, $\phi(i) = r(i)$. We define two algorithms for batch opening. The algorithm $\mathsf{CreateWitnessBatch}(\mathsf{PK}, \phi(x), B)$ outputs $\langle B, r(x), w_B\rangle$ and the algorithm $\mathsf{VerifyEvalBatch}(\mathsf{PK}, \mathcal{C}, B, r(x), w_B)$ outputs 1 if $e(\mathcal{C}, g) \stackrel{?}{=} e(g^{\prod_{i\in B}(\alpha - i)}, w_B)e(g, g^{r(\alpha)})$ holds, $\deg r(x) = |B|$, and $r(i) = \phi(i)$ for all $i \in B$.

In terms of security, since commitments are formed in the same way as the Commit algorithm of $\mathsf{PolyCommit}_{\mathrm{DL}}$ and $\mathsf{CreateWitnessBatch}$ reveals no more information than running the $\mathsf{CreateWitness}$ algorithm of $\mathsf{PolyCommit}_{\mathrm{DL}}$ for all batch elements individually, the hiding property (Theorem 1) still holds. For binding, an adversary should not be able to open a batch B containing an index i, in a manner that conflicts with the value $\phi(i)$ output in an individual opening of index i. Formally, we say that batch opening is binding if the following holds:

$$\Pr\left(\begin{array}{c}\mathsf{PK} \leftarrow \mathsf{Setup}(1^{\kappa}, t), (\mathcal{C}, \langle B, w_B, r(x)\rangle, \langle i \in B, w_i, \phi(i)\rangle) \leftarrow \mathcal{A}(\mathsf{PK}) : \\ \mathsf{VerifyEvalBatch}(\mathsf{PK}, \mathcal{C}, B, w_B, r(x)) = 1 \wedge \\ \mathsf{VerifyEval}(\mathsf{PK}, \mathcal{C}, i, w_i, \phi(i)) = 1 \ \wedge \phi(i) \neq r(i)\end{array}\right) = \epsilon(\kappa).$$

Theorem 3. *The construction of* CreateWitnessBatch, VerifyEvalBatch *in §3.4 is binding provided the t-BSDH assumption holds in* $\mathbb{G}$.

This theorem is proven in the full version. The batch construction can be modified for $\mathsf{PolyCommit}_{\mathsf{Ped}}$ due to homomorphic nature of $\mathsf{PolyCommit}_{\mathrm{DL}}$. In the full version we also compare the overhead of various commitment schemes, when Alice commits to t values, and then must reveal k of them. *Overhead* excludes the communication cost of sending the committed values. Notably, the communication overhead of $\mathsf{PolyCommit}_{\mathrm{DL}}$ is constant when batch opening is used.

Strong Correctness. VSS schemes will require an additional property of the PolyCommit scheme: it should not be possible to commit to a polynomial of degree greater than t. This is called the *strong correctness* property.

To define strong correctness for the PolyCommit schemes is not easy, *e.g.*, there are many polynomials ϕ' of degree greater than t such that $\phi(\alpha) = z \in_R \mathbb{Z}_p$ and so g^z is trivially a $\mathsf{PolyCommit}_{\mathrm{DL}}$ commitment to some polynomial of degree t' such that $2^\kappa > t' > t$. To avoid this triviality, we require that an adversary $\mathcal{A}$ must convince a challenger $\mathcal{B}$ that he knows ϕ with the following game. $\mathcal{A}$ creates a commitment to a claimed polynomial ϕ' of degree t'. $\mathcal{B}$ challenges $\mathcal{A}$ with $t'+1$ indices $I \subset \mathbb{Z}_p$. $\mathcal{A}$ wins if he is able to produce $\{\langle i, \phi(i), w_i\rangle\}_{i\in I}$ accepted by VerifyEval and that the interpolation (in exponents) of any $t'+1$ witnesses generates a degree $t-1$ polynomial. Similarly for $\mathsf{PolyCommit}_{\mathsf{Ped}}$. Refer to the extended version of the paper for proof of the following theorem.

Theorem 4. $\mathit{PolyCommit}_{DL}$ *and* $\mathsf{PolyCommit}_{\mathsf{Ped}}$ *have the strong correctness property if the t-polyDH assumption holds in* $\mathcal{G}$.

Practicality and Efficiency Improvements. In absence of a single trusted party, computing Setup can be distributed. Here, $\mathsf{SK} = \alpha$ is computed in a distributed form (*i.e.*, shared by multiple parties forming a distributed authority) using the concept of distributed key generation [31]. PK is computed using a distributed multiplication protocol [20]. As we do not require SK during our protocols and as anybody can verify the correctness of PK using pairings, it is possible to consider PK as a global reusable set, shared in many systems.

Further, the exponentiations required when committing and creating witnesses can be trivially parallelized. Also, since $\mathcal{C} = g^{\phi(\alpha)}$ is computed as a product of powers (sometimes called a *multi-exponentiation*), we suggest using fast exponentiation techniques [32] instead of a naïve implementation. It is also possible to precompute $e(\mathcal{C}, g)$ and $e(g, g)$ for use during verification.

4 Applications

In this section, we describe applications of our commitment schemes to verifiable secret sharing (§4.1), zero-knowledge sets and elementary databases (§4.2), and selective disclosure of signed data and credentials (§4.3).

4.1 Verifiable Secret Sharing (VSS)

For integers n and t such that $n > t \geq 0$, an (n, t)-*secret sharing* scheme [34, 4] is a method used by a *dealer* P_d to share a secret s among a set of n participants (the *sharing* Sh phase) in such a way that in the *reconstruction* Rec phase any subset of $t+1$ or more honest participants can compute the secret s, but subsets of size t or fewer cannot. Furthermore, in secret sharing, nodes may need a procedure to verify the correctness of the dealt values in order to prevent malicious behaviour by the dealer. To solve this problem, Chor *et al.* [15] introduced verifiability in secret sharing, which led to the concept of *verifiable secret sharing* (VSS).

VSS schemes have two security requirements: *secrecy* and *correctness*.

Secrecy (VSS-S). A *t-limited* adversary who compromises t nodes cannot compute s during the Sh phase.

Correctness (VSS-C). The reconstructed value should be equal to the shared secret s or every honest node concludes that the dealer is malicious by outputting $\perp$.

In the computational complexity setting, any malicious behaviour by P_d is caught by the honest nodes in the Sh phase itself and the VSS-C property simplifies to the following: the reconstructed value should be equal to the shared secret s.

Many VSS applications require that broadcasts from any $t+1$ honest nodes or any $2t+1$ nodes is sufficient to reconstruct s. Therefore, along with VSS-S and VSS-C, we mandate the correctness property that we refer as the *strong correctness* property. Further, some VSS schemes achieve a stronger secrecy guarantee.

Strong Correctness (VSS-SC). The same unique value s is reconstructed regardless of the subset of nodes (of size greater than $2t$) chosen by the adversary in the Rec algorithm.

Strong Secrecy (VSS-SS). The adversary who compromises t nodes have no more information about s except what is implied by the public parameters.

Feldman [18] developed the first efficient VSS protocol, which forms the basis of all VSS schemes defined in the literature. In the literature, the discrete logarithm commitment or Pedersen commitment is used in the Feldman VSS achieve the binding (correctness) and the hiding (secrecy) properties. Both of these commitment schemes trivially satisfy the strong correctness (VSS-SC) property of VSS by the fact that the size of a commitment to a polynomial $\phi(x) \in \mathbb{Z}_p[x]$ is equal to $\deg(\phi) + 1$, which is $O(n)$ (since for optimal resiliency, $\deg(\phi) = t = \lfloor \frac{n-1}{2} \rfloor$). However, the commitment to a polynomial has to be broadcast to all nodes, which results in a linear-size broadcast for Feldman VSS and their variants and a linear complexity gap between the message and the bit complexities. Our goal is to close this gap using any of the PolyCommit schemes. Next, we apply $\mathsf{PolyCommit}_{\mathrm{DL}}$ to existing polynomial-based VSS schemes and reduce the broadcast message size of VSS by a linear factor, making it equal to the message complexity. Although $\mathsf{PolyCommit}_{\mathrm{DL}}$ can be used in any univariate polynomial-based scheme, we choose the Feldman VSS for ease of exposition.

Sh Phase

1. To share a secret $s \in \mathbb{Z}_p^*$, the dealer P_d chooses a random degree t polynomial $\phi(x) = \sum_{j=0}^{t} \phi_j x^j \in \mathbb{Z}_p[x]$ such that $\phi(0) = \phi_0 = s$. It then broadcasts $\mathcal{C} = \mathsf{Commit}(\mathsf{PK}, \phi(x))$.
2. For $\ell \in [1, n]$, P_d computes a share $s_\ell = \phi(\ell)$, a witness $w_\ell = \mathsf{CreateWitness}(\mathsf{PK}, \phi(x), \ell)$ and sends $\langle \ell, \phi(\ell), w_\ell \rangle$ to node P_ℓ over a secure and authenticated channel.
3. After receiving $\langle i, \phi(i), w_i \rangle$ from P_d, node P_i runs $\mathsf{VerifyEval}(\mathsf{PK}, \mathcal{C}, i, \phi(i), w_i)$. If the verification fails, P_i broadcasts an accusation message against P_d.
4. If more than t nodes accuse P_d, then it is clearly faulty and is disqualified. If not, for each accusing party P_ℓ, P_d broadcasts the corresponding share and witness $\langle \ell, \phi(\ell), w_\ell \rangle$ such that $\mathsf{VerifyEval}$ holds.
5. If any of the revealed shares fails $\mathsf{VerifyEval}$, P_d is disqualified and the protocol stops. If there is no disqualification, each node P_ℓ accepts $s_\ell = \phi(\ell)$.

Rec Phase

Any $t+1$ or more nodes P_i publish their accepted shares and witnesses $\langle i, s_i, w_i \rangle$. All $t+1$ (or more) nodes verify each of the broadcast shares $\langle i, \phi(i), w_i \rangle$ using $\mathsf{VerifyEval}$ and then interpolate the pairs $\langle i, \phi(i) \rangle$ to determine the secret $s = \phi(0)$.

Fig. 1. eVSS: An efficient Feldman VSS using $\mathsf{PolyCommit}_{\mathrm{DL}}$

Our *efficient Feldman VSS* (eVSS) scheme runs $\mathsf{Setup}(1^\kappa, t)$ of $\mathsf{PolyCommit}_{\mathrm{DL}}$ once, which outputs $\mathsf{PK} = \langle \mathcal{G}, g, g^\alpha, \ldots, g^{\alpha^t} \rangle$. Further, as we are working in the synchronous communication model, a *resiliency bound* of $n \geq 2t+1$ is required for VSS to provide correctness against a t-limited Byzantine adversary as the $n - t$ honest nodes available during the Sh and Rec phases should at least be equal to $t+1$ (the required threshold). In Figure 1, we present eVSS that uses the $\mathsf{PolyCommit}_{\mathrm{DL}}$ scheme in the Feldman VSS. In the Sh and the Rec phases of the eVSS scheme, the VSS methodology remains exactly the same as that of Feldman VSS except here $t+1$ commitments of the form g^{ϕ_j} for $\phi(x) = \sum_{j=0}^{t} \phi_j x^j$ are replaced by a single polynomial commitment $\mathcal{C} = g^{\phi(\alpha)}$. In addition, along with a share s_i, P_d now sends a witness w_i to node P_i. Overall, the eVSS protocol needs $O(1)$ broadcast instead of $O(n)$ required by the Feldman VSS. In case of multiple accusations, dealer P_d can use the batch opening feature described in §3.4 to provide a single witness for the complete batch. Furthermore, due to the homomorphic nature of $\mathsf{PolyCommit}$, the eVSS scheme can easily converted to a distributed key generation protocol [31].

Theorem 5. *The eVSS protocol implements a synchronous VSS scheme with the VSS-S and VSS-SC properties for $n \geq 2t+1$ provided the DL, t-SDH and t-polyDH assumptions hold in $\mathcal{G}$.*

We need to prove the secrecy, correctness and strong correctness properties of a synchronous VSS scheme. Secrecy and correctness result directly from Theorem 1, while Theorem 4 provides the strong correctness property. The secrecy

provided by eVSS is computational against a t-bounded adversary, and unconditional against a $t-1$ bounded adversary. Share correctness is computational.

$\mathsf{PolyCommit}_{\mathrm{DL}}$ can easily be replaced by $\mathsf{PolyCommit}_{\mathsf{Ped}}$ in the above eVSS scheme. In that case, we achieve the strong secrecy (VSS-SS) property due to the unconditional hiding property (Theorem 2) of $\mathsf{PolyCommit}_{\mathsf{Ped}}$.

4.2 Nearly ZKSs and Nearly ZK-EDBs

Micali *et al.* [27] define zero-knowledge sets (ZKSs). Basically a ZKS allows a committer to create a short commitment to a set of values S, such that he may later efficiently prove statements of the form $k_j \in S$ or $k_j \notin S$ in zero-knowledge. No additional information about S is revealed. Perhaps the most challenging aspect in ZKS construction is that not even an upper bound on $|S|$ may be revealed. The closely related notion of *zero-knowledge elementary databases* (ZK-EDB) is also defined in [27]. Loosely speaking, an EDB is a list of key-value pairs, and a ZK-EDB allows a committer to prove that a given value is associated with a given key with respect to a short commitment.

We argue that relaxing the requirements of a ZKS is sufficient for known applications, and show this leads to a significantly more efficient primitive. In particular, by not hiding $|S|$, the size of the proof that an element is (or is not) in a committed set is reduced by a factor of sixteen or more, when compared to the best known ZKS construction.

Motivation. Much of the literature on ZKSs does not consider applications [12, 13, 19, 25, 33]. In the applications of ZKSs (and ZK-EDBs) suggested in [27] the size of the set (or DB) is not crucial to the intended security or privacy of the application. The applications given are to improve privacy and access control when the records of an EDB contain sensitive information about people, *e.g.*, medical records. In such cases, revealing a bound on the number of records in the database clearly does not affect the privacy of an individual whose information is kept in the database.

Another use of ZKSs and ZK-EDBs is for committed databases [30]. In this application, a database owner commits to the database and then proves for every query that the response is consistent with the commitment. For many applications the *contents* of the committed database must be hidden, but the size may be revealed. An example is given in Buldas *et al.* [9]. Here ZK-EDBs are used to increase the accountability of a certification authority by preventing it from providing inconsistent responses to queries about the validity of a certificate. Clearly, keeping the number of certificates hidden is not required. Therefore, a weaker type of ZKS primitive that does not hide the size of the set will suffice for most practical applications of ZKSs. We call a ZKS that may leak information about the size of the set a *nearly ZKS*. Similarly, a *nearly ZK-EDB* is a ZK-EDB that may leak information about the number of records it contains.

Note that an accumulator also represents a set of values with proofs of membership, and some even allow proofs of non-membership (*e.g.*, see [17]). They do

SetupZKS$(1^\kappa, t)$ outputs $\mathsf{PK} = \mathsf{Setup}(1^\kappa, t)$. t is an upper bound on the size of the set which may be committed.

CommitZKS$(\mathbf{PK}, S)$ requires $|S| \leq t$. Define $\phi(x) = \prod_{k_j \in S}(x - k_j) \in \mathbb{Z}_p[x]$. Output $\mathcal{C} = \mathsf{Commit}(\mathsf{PK}, \phi(x))$. Let $\hat{\phi}(x) \in \mathbb{Z}_p[x]$ be the random degree t polynomial chosen in $\mathsf{PolyCommit}_{\mathsf{Ped}}$.

QueryZKS$(\mathbf{PK}, \mathcal{C}, k_j)$ allows the committer to create a proof that either $k_j \in S$ or $k_j \notin S$. Compute $\langle k_j, \phi(k_j), \hat{\phi}(k_j), w_j \rangle = \mathsf{CreateWitness}(\mathsf{PK}, \phi(x), \hat{\phi}(x), k_j)$.

(i) If $k_j \in S$, output $\pi_{Sj} = (k_j, w_j, \hat{\phi}(k_j), \perp)$.

(ii) If $k_j \notin S$, create $z_j = g^{\phi(k_j)} h^{\hat{\phi}(k_j)}$ and a ZK proof of knowledge of $\phi(k_j)$ and $\hat{\phi}(k_j)$ in $z_j = g^{\phi(k_j)} h^{\hat{\phi}(k_j)}$. Let $\gamma_j = \langle z_j, \text{ZK proof} \rangle$. Output $\pi_{Sj} = (k_j, w_j, \perp, \gamma_j)$.

VerifyZKS$(\mathbf{PK}, \mathcal{C}, \pi_{Sj})$ parses π_{Sj} as $(k_j, w_j, \hat{\phi}(k_j), \gamma_j)$.

(i) If $\hat{\phi}(k_j) \neq \perp$, then $k_j \in S$. Output 1 if $\mathsf{VerifyEval}(\mathsf{PK}, \mathcal{C}, k_j, 0, \hat{\phi}(k_j), w_j) = 1$.

(ii) If $\gamma_j \neq \perp$, then $k_j \notin S$. Parse γ_j as $\langle z_j, \text{ZK proof} \rangle$. If $e(\mathcal{C}, g) \stackrel{?}{=} e(w_j, g^{\alpha - k_j}) e(z_j, g)$, and the ZK proof of knowledge of z_j is valid, output 1. Output 0 if either check fails.

Fig. 2. A nearly ZKS scheme based on $\mathsf{PolyCommit}_{\mathsf{Ped}}$

not however, guarantee hiding (the ZK property), in [17] after seeing responses to t non-membership queries we may recover the entire accumulated set.

Construction of a Nearly ZKS. This construction (Figure 2) will use $\mathsf{PolyCommit}_{\mathsf{Ped}}$, and allows us to commit to $S \subset \mathbb{Z}_p$ such that $|S| \leq t$. The basic idea is to commit to a polynomial ϕ, such that $\phi(k_j) = 0$ for $k_j \in S$, and $\phi(k_j) \neq 0$ for $k_j \notin S$. Our construction relies on a ZK proof that proves $\phi(k_j) \neq 0$ without revealing $\phi(k_j)$ to maintain privacy for queries when $k_j \notin S$. Although a construction based on $\mathsf{PolyCommit}_{\mathrm{DL}}$ is also possible, we choose $\mathsf{PolyCommit}_{\mathsf{Ped}}$ as the required ZK proof is simpler in the latter case. For convenience we describe our protocols assuming the ZK proof is non-interactive, however, an interactive ZK proof may be used as well.

A security definition and proof are provided in the full version. The ZK proof of knowledge may be implemented using any secure ZK proof system allowing proof of knowledge of a discrete logarithm (see [11] for examples).

Construction of a Nearly ZK-EDB. This construction (Figure 3) makes use of the above nearly ZKS construction and $\mathsf{PolyCommit}_{\mathrm{DL}}$. Let $D = (K, V) \subset \mathbb{Z}_p^t \times \mathbb{Z}_p^t$ be a list of key-value pairs that will define the database (K and V are ordered lists of equal length such that the value $m_j \in V$ corresponds to the key $k_j \in K$). The values may repeat, but the keys must be distinct. We write $D(k_j)$ to denote the value associated to key k_j (if $k_j \notin K$, then $D(k_j) = \perp$). The underlying idea of our construction is to commit to the keys using our nearly ZKS, and also commit to ϕ, such that $\phi(k_j) = m_j$, using $\mathsf{PolyCommit}_{\mathrm{DL}}$, since it is sufficient for security and more efficient. The reason for using the nearly ZKS is to respond to queries when $k \notin D$ without revealing any additional information.

SetupEDB$(1^\kappa, t)$ runs **SetupZKS**$(1^\kappa, t)$, and outputs **PK**.

CommitEDB$(\mathsf{PK}, D = (K, V))$ sets $\mathcal{C}_1 = \mathsf{CommitZKS}(\mathsf{PK}, K)$. It then interpolates the t (or fewer) points $(k_j, m_j) \in D$, and one or more random points $(k_r, m_r) \in_R \mathbb{Z}_p \times \mathbb{Z}_p$ to get a polynomial $\phi_2(x) \in \mathbb{Z}_p[x]$, assured to be of degree t. Set $\mathcal{C}_2 = \mathsf{Commit}(\mathsf{PK}, \phi_2(x))$ and output $\mathcal{E} = (\mathcal{C}_1, \mathcal{C}_2)$.

QueryEDB$(\mathsf{PK}, \mathcal{E}, k_j)$ parses $\mathcal{E}$ as $(\mathcal{C}_1, \mathcal{C}_2)$.

(i) If $k_j \in K$, compute $\pi_{Sj} = \mathsf{QueryZKS}(\mathsf{PK}, \mathcal{C}_1, k_j)$ to show that $k_j \in K$ and $\langle k_j, m_j, w_{m_j} \rangle = \mathsf{CreateWitness}(\mathsf{PK}, \phi_2(x), k_j)$ to show that $D(k_j) = m_j$. Output $\pi_{Dj} = (\pi_{Si}, m_j, w_{m_j})$.

(ii) If $k_j \notin K$, we show that $k_j \notin K$, set $\pi_{Sj} = \mathsf{QueryZKS}(\mathsf{PK}, \mathcal{C}_1, k_j)$. Output $\pi_{Dj} = (\pi_{Sj}, \perp, \perp)$.

VerifyEDB$(\mathsf{PK}, \mathcal{E}, \pi_{Dj})$ parses π_{Dj} as (π_{Sj}, m_j, w_{m_j}) and $\mathcal{E}$ as $(\mathcal{C}_1, \mathcal{C}_2)$.

(i) If $m_j \neq \perp$, then $k_j \in K$, output (k_j, m_j) if $\mathsf{VerifyZKS}(\mathsf{PK}, \mathcal{C}_1, \pi_{Sj}) = 1$ and $\mathsf{VerifyEval}(\mathsf{PK}, \mathcal{C}_2, k_j, m_j, w_{m_j}) = 1$.

(ii) If $m_j = \perp$, then $k_j \notin K$, output 1 if $\mathsf{VerifyZKS}(\mathsf{PK}, \mathcal{C}_1, \pi_{Sj}) = 1$.

Fig. 3. A nearly ZK-EDB scheme constructed using our nearly ZKS construction (Figure 2) and $\mathsf{PolyCommit}_{\mathrm{DL}}$

Efficiency of our nearly ZKS and ZK-EDBs. The size of the commitment is a single group element for a nearly ZKS, or two elements for a nearly ZK-EDB. Proof that $k_j \in S$, consists of two group elements, while proof that $k_j \notin S$ consists of about five group elements (when ZK proofs are implemented using a standard three-move ZK protocol, made non-interactive with the Fiat-Shamir heuristic). The proof sizes for our nearly ZK-EDB construction are three and about five group elements (respectively).

The ZK-EDB in the literature with the shortest proofs is that of Libert and Yung [25] (based on their ZKS construction). Asymptotically, (non)membership proofs are $O(\kappa / \log(t))$ bits, where κ is a security parameter, and t is the size of the system parameters. For the parameter choices given [25], proof sizes range from 80–220 group elements. The computation of their scheme and ours is nearly equal. Therefore, using nearly ZK-EDBs in the place of ZK-EDBs reduces communication costs by at least a factor of sixteen.

4.3 Credentials and Selective Disclosure of Signed Data

In this section we briefly describe two applications of the PolyCommit schemes, and we will show how polynomial commitments can reduce communication costs. Both applications are based on the following idea. Suppose Alice has a list of values $(m_1, \ldots, m_t)$, which will be signed by Trent. If Trent signs the concatenation, then Alice must reveal all m_i to allow Bob to verify the signature. However, if Trent signs $\mathcal{C} = \mathsf{Commit}(\mathsf{PK}, \phi(x))$ where $\phi(i) = m_i$, then Alice may allow Bob to verify that Trent has signed m_i without revealing the other m_j. Bob verifies the signature on $\mathcal{C}$, and Alice produces a witness to prove that $\mathcal{C}$ opens to m_i at position i, allowing Alice to convince Bob that Trent signed m_i.

Content Extraction Signatures. If $(m_1, \ldots, m_t)$ are parts of a document, then signing a polynomial commitment is a *content extraction signature* (CES) scheme. Steinfeld *et al.* [35] introduce CES and give a generic construction of CES signatures. The scheme requires a standard commitment scheme, which is then used to commit to each of the t sub-messages $(m_1, \ldots, m_t)$ individually, forming a vector of commitments, which is signed. The scheme is secure, provided the signature scheme is secure, and the commitment scheme is hiding and binding.

Since both PolyCommit schemes are hiding and binding, and allow a list of messages to be committed, they can be used in the general construction of Steinfeld *et al.* Along with the commitment, Trent should also sign t so that Bob knows that only indices $\{1, \ldots, t\}$ correspond to valid sub-messages. The new scheme is nearly as communication efficient as a specific scheme in [35] which has the lowest communication cost. The latter, however, depends on specific properties of the RSA signature scheme and is secure in the random oracle model. Using a polynomial commitment scheme gives an efficient generic construction. Therefore, efficient standard model CES schemes are possible by combining any of the PolyCommit schemes with a signature scheme secure in the standard model.

Pseudonymous Credentials. If $(m_1, \ldots, m_t)$ are attributes about Alice, and Trent is an identity provider, then the signature Alice holds on $\mathcal{C}$ is a digital credential that allows Alice to reveal only as much information as is necessary to complete an online transaction. Here, we create $\mathcal{C}$ using $\mathsf{PolyCommit}_{\mathrm{DL}}$, as batched openings are efficient for $\mathsf{PolyCommit}_{\mathrm{DL}}$. Disclosing a single m_i requires Alice to transmit $(\mathcal{C}, \mathsf{Sign}_{\mathrm{Trent}}(\mathcal{C}), \langle i, m_i, w_i \rangle)$, the size of which is independent of t. If Alice reveals a subset of the attributes, a single witness may be used to reduce communication even further using batch opening (described in §3.4). Further, if Trent signs multiple commitments to the same attributes (but includes an extra randomized attribute), Alice may present a different commitment to the same verifier unlinkably.

For many interesting applications of credentials, selective show is insufficient because Alice would like to prove something about m_i (*e.g.*, $m_i < 1990$) without revealing m_i. Alice may prove knowledge of a nonzero committed value $\phi(i)$ without revealing it, and compose this proof with other proofs about m_i using standard ZK proof techniques for proving knowledge of, relations between or the length of discrete logarithms [11]. Since the communication costs per attribute of proving knowledge of a committed value are constant, if k attributes are involved in showing a credential, the complexity of the show will be $O(k)$. In existing schemes the communication is $O(t)$ where t is the total number of attributes in the credential. Further details of this application are given in the full version of this paper.

5 Open Problems

Finally, we list a few open problems related to polynomial commitment schemes.
1. Is it possible to construct efficient polynomial commitment schemes under

weaker assumptions? 2. What other protocols does PolyCommit improve? (For example, can PolyCommit reduce the communication of asynchronous VSS protocols or verifiable shuffles?) 3. We have mainly focused on the communication costs, but our construction asks for nontrivial computation. Is it possible to reduce computation cost as well?

Acknowledgements. We thank the anonymous reviewers for providing many helpful comments, and thank Benoit Libert for sending us a preprint of [25]. We gratefully acknowledge the financial support of NSERC, MITACS, and the David R. Cheriton Graduate Scholarship program.

References

1. Au, M.H., Wu, Q., Susilo, W., Mu, Y.: Compact E-Cash from Bounded Accumulator. In: Abe, M. (ed.) CT-RSA 2007. LNCS, vol. 4377, pp. 178–195. Springer, Heidelberg (2006)
2. Au, M.H., Susilo, W., Mu, Y.: Practical anonymous divisible e-cash from bounded accumulators. In: Tsudik, G. (ed.) FC 2008. LNCS, vol. 5143, pp. 287–301. Springer, Heidelberg (2008)
3. Benaloh, J.C., de Mare, M.: One-way accumulators: A decentralized alternative to digital signatures (extended abstract). In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 274–285. Springer, Heidelberg (1994)
4. Blakley, G.R.: Safeguarding cryptographic keys. In: Proceedings of the National Computer Conference, pp. 313–317 (1979)
5. Boneh, D., Boyen, X.: Efficient Selective-ID Secure Identity-Based Encryption Without Random Oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)
6. Boneh, D., Boyen, X.: Short Signatures Without Random Oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004)
7. 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)
8. 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)
9. Buldas, A., Laud, P., Lipmaa, H.: Eliminating Counterevidence with Applications to Accountable Certificate Management. Journal of Computer Security 10(3), 273–296 (2002)
10. 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)
11. Camenisch, J., Stadler, M.: Proof systems for general statements about discrete logarithms. Technical report, 260, Dept. of Computer Science, ETH Zurich (1997)
12. Catalano, D., Fiore, D., Messina, M.: Zero-knowledge sets with short proofs. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 433–450. Springer, Heidelberg (2008)
13. Chase, M., Healy, A., Lysyanskaya, A., Malkin, T., Reyzin, L.: Mercurial commitments with applications to zero-knowledge sets. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 422–439. Springer, Heidelberg (2005)
14. Cheon, J.H.: Security analysis of the strong Diffie-Hellman problem. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 1–13. Springer, Heidelberg (2006)

15. Chor, B., Goldwasser, S., Micali, S., Awerbuch, B.: Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults (Extended Abstract). In: Proceedings of FOCS 1985, pp. 383–395 (1985)
16. Damgård, I.: Commitment schemes and zero-knowledge protocols. In: Damgård, I.B. (ed.) EEF School 1998. LNCS, vol. 1561, pp. 63–86. Springer, Heidelberg (1999)
17. Damgård, I., Triandopoulos, N.: Supporting non-membership proofs with bilinear-map accumulators. Cryptology ePrint Archive: Report 2008/538 (2008)
18. Feldman, P.: A Practical Scheme for Non-interactive Verifiable Secret Sharing. In: Proceedings of FOCS 1987, pp. 427–437 (1987)
19. Gennaro, R., Micali, S.: Independent zero-knowledge sets. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4052, pp. 34–45. Springer, Heidelberg (2006)
20. Gennaro, R., Rabin, M.O., Rabin, T.: Simplified VSS and Fast-Track Multiparty Computations with Applications to Threshold Cryptography. In: Proceedings of PODC 1998, pp. 101–111. ACM Press, New York (1998)
21. Goyal, V.: Reducing Trust in the PKG in Identity Based Cryptosystems. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 430–447. Springer, Heidelberg (2007)
22. Guo, F., Mu, Y., Chen, Z.: Identity-Based Encryption: How to Decrypt Multiple Ciphertexts Using a Single Decryption Key. In: Takagi, T., Okamoto, T., Okamoto, E., Okamoto, T. (eds.) Pairing 2007. LNCS, vol. 4575, pp. 392–406. Springer, Heidelberg (2007)
23. Herzberg, A., Jarecki, S., Krawczyk, H., Yung, M.: Proactive Secret Sharing Or: How to Cope With Perpetual Leakage. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 339–352. Springer, Heidelberg (1995)
24. Kate, A., Zaverucha, G., Goldberg, I.: Polynomial commitments. Technical report, CACR 2010-10, Centre for Applied Cryptographic Research, University of Waterloo (2010)
25. Libert, B., Yung, M.: Concise Mercurial Vector Commitments and Independent Zero-Knowledge Sets with Short Proofs. In: Micciancio, D. (ed.) Theory of Cryptography. LNCS, vol. 5978, pp. 499–517. Springer, Heidelberg (2010)
26. Menezes, A., Van Oorschot, P., Vanstone, S.: Handbook of Applied Cryptography, 1st edn. CRC Press, Boca Raton (1997)
27. Micali, S., Rabin, M., Kilian, J.: Zero-knowledge sets. In: Proceedings of FOCS 2003, pp. 80–91. IEEE, Los Alamitos (2003)
28. Mitsunari, S., Sakai, R., Kasahara, M.: A New Traitor Tracing. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences E85-A(2), 481–484 (2002)
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. Ostrovsky, R., Rackoff, C., Smith, A.: Efficient Consistency Proofs for Generalized Queries on a Committed Database. In: Díaz, J., Karhumäki, J., Lepistö, A., Sannella, D. (eds.) ICALP 2004. LNCS, vol. 3142, pp. 1041–1053. Springer, Heidelberg (2004)
31. 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)
32. Pippenger, N.: On the evaluation of powers and related problems. In: IEEE SFCS FOCS 1976, pp. 258–263 (1976)
33. Prabhakaran, M., Xue, R.: Statistically Hiding Sets. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 100–116. Springer, Heidelberg (2009)
34. Shamir, A.: How to Share a Secret. ACM Commun. 22(11), 612–613 (1979)
35. Steinfeld, R., Bull, L., Zheng, Y.: Content extraction signatures. In: Kim, K.-C. (ed.) ICISC 2001. LNCS, vol. 2288, pp. 285–304. Springer, Heidelberg (2002)

Computationally Secure Pattern Matching in the Presence of Malicious Adversaries

Carmit Hazay and Tomas Toft

Aarhus University
{carmit,ttoft}@cs.au.dk

Abstract. We propose a dedicated protocol for the highly motivated problem of secure two-party pattern matching: Alice holds a text $t \in \{0,1\}^*$ of length n, while Bob has a pattern $p \in \{0,1\}^*$ of length m. The goal is for Bob to learn where his pattern occurs in Alice's text. Our construction guarantees full simulation in the presence of malicious, polynomial-time adversaries (assuming that ElGamal encryption is semantically secure) and exhibits computation and communication costs of $O(n+m)$ in a constant round complexity.

In addition to the above, we propose a collection of protocols for variations of the secure pattern matching problem: The pattern may contain wildcards ($O(nm)$ communication in $O(1)$ rounds). The matches may be approximated, i.e., Hamming distance less than some threshold ($O(nm)$ communication in $O(1)$ rounds). The length, m, of Bob's pattern is secret ($O(nm)$ communication in $O(1)$ rounds). The length, n, of Alice's text is secret ($O(n+m)$ communication in $O(1)$ rounds).

keywords: Pattern matching, secure two-party computation, full simulation, malicious adversary.

1 Introduction

In the setting of secure two-party computation, two parties with private inputs wish to jointly compute some function of their inputs while preserving certain security properties like privacy, correctness and more. The standard definition [GL91, Bea92, MR91, Can00] formalizes security by comparing the execution of such protocol to an "ideal execution" where a trusted third party computes the function for the parties. Specifically, in the ideal world the parties just send their inputs over perfectly secure communication lines to a trusted party, who then computes the function honestly and sends the output to the designated party. Then, a real protocol is said to be secure if no adversary can do more harm in a real protocol execution than in an ideal one (where by definition no harm can be done).

Secure two-party computation has been extensively studied, and it has been demonstrated that any polynomial-time two-party computation can be generically compiled into a secure function evaluation protocol with polynomial complexity [Yao86, GMW87, Gol04]. These results apply in various settings,

M. Abe (Ed.): ASIACRYPT 2010, LNCS 6477, pp. 195–212, 2010.

(considering semi-honest and malicious adversaries). However, more often than not, the resulting protocols are inefficient for practical uses (in part because they are general and so do not utilize any specific properties of the protocol problem at hand) and hence attention has been given to constructing efficient protocols for specific functions. This approach has proved quite successful for the semi-honest setting (see, e.g., [LP02, AMP04, FNP04, KS05, TPKC07]), while the malicious setting remained impractical (a notable exception is [AMP04]).

In this paper we consider the following classic search problem: Alice holds a text $t \in \{0,1\}^*$ of length n and Bob is given a pattern (i.e., search word) $p \in \{0,1\}^*$ of length m, where the sizes of t and p are mutually known. The goal is for Bob to learn all the locations in the text that match the pattern, while Alice learns nothing about the pattern. This problem has been widely studied for decades due to its potential applications for text retrieval, music retrieval, computational biology, data mining, network security, and many more. The most known application in the context of privacy is in compering two DNA strings; our example is taken from [GHS10]. Consider the case of a hospital holding a DNA database of all the participants in a research study, and a researcher wanting to determine the frequency of the occurrence of a specific gene. This is a classical pattern matching application, which is however complicated by privacy considerations. The hospital may be forbidden from releasing the DNA records to a third party. Likewise, the researcher may not want to reveal what specific gene he is working on, nor trust the hospital to perform the search correctly.

Although most of the existing solutions are highly practical they fail to achieve any level of security (if at all); see [Blo70, KMP77, BM77, ACR99, NM07] for just a few examples. In this work, we focus our attention on the secure computation of the basic pattern matching problem and several important variants of it.

Our Contribution. We achieve efficiency that is a significant improvement on the current state of the art for the following problems:

- Secure Pattern Matching. We develop an efficient, constant rounds protocol for this problem that requires $O(n+m)$ exponentiations and bandwidth of $O(n+m)$ group elements. Our protocol lays the foundations for the following constructions.
- Secure Pattern Matching with Wildcards. This problem is a known variant of the classic problem where Bob (who holds the pattern) introduces a new "don't care" character to its alphabet, denoted by $\star$ (wildcard). The goal is for Bob to learn all the locations in the text that match the pattern, where $\star$ matches any character in the text. This problem has been widely looked at by researchers with the aim of generalizing the basic searching model to searching with errors. This variant is known as pattern matching with don't cares and can be solved in $O(n+m)$ time [IR07]. In this paper, we develop a protocol that computes this functionality with $O(nm)$ costs.
- Secure Approximate Pattern Matching. In this problem the goal is for Bob to find the locations where the Hamming distance of the (text) substrings and the pattern is less than some threshold $\tau \leq m$. We design a protocol for this problem with $O(mn)$ costs.

– Secure Pattern Matching where the length of the pattern or the text remains hidden. Finally, we consider two variants with an additional security requirement of hiding the input length. Solutions for these problems can be achieved in $O(nm)$ time.

Our protocols are based on ElGamal encryption and are proven secure in the plain model under the standard DDH assumption and achieve full simulation in the presence of malicious adversaries.

Prior Work. To the best of our knowledge, the first who considered pattern matching in the context of secure computation were [TPKC07] who considered a secure version of oblivious automata evaluation to achieve secure pattern matching. Their protocol implements the KMP algorithm [KMP77] in the semi honest setting. Loosely speaking, the KMP algorithm works in $O(n)$ time and searches for occurrences of the pattern within the text by employing the observation that when a mismatch occurs, the pattern embodies sufficient information to determine where the next match could begin. Their costs are linear in the input length.

This problem was also studied by Hazay and Lindell in [HL08] who used oblivious pseudorandom function (PRF) evaluation. However, their protocol achieves only a weaker notion of security called one-sided simulatability which does not guarantee full simulation for both corruption cases. The only construction to achieve full simulation in the malicious setting was developed by Gennaro et al. [GHS10]. They took a different approach to implement the KMP algorithm and described a protocol that runs in $O(m)$ rounds and requires $O(nm)$ exponentiations and bandwidth.

Finally, a recent paper by Katz and Malka [KM10] presents a secure solution for a generalized pattern matching problem, denoted text processing. Namely, the party who holds the pattern has some additional information y and his goal is to learn a function of the text and y, for the text locations where the pattern matches. They show how to modify Yao's garbled circuit approach to obtain a protocol where the size of the garbled circuit is linear in the number of occurrences of p in t (rather than linear in $|t|$). Their costs are dominated by the size of the circuit times the number of occurrences u (as P_1 sends u such circuits). Nevertheless, they assume a common input of some threshold on the number of occurrences.

To the best of our knowledge, the only work which addresses one of the above variants is the work by Jarrous and Pinkas [JP09]. In this work, the authors solve the hamming distance problem for two equal length strings against malicious adversaries. Their protocol requires a committed oblivious transfer for each bit. Moreover, the costs of their protocol are inflated by a statistical parameter s for running a subprotocol for the oblivious polynomial evaluation functionality (namely, the protocol requires $O(d \cdot s)$ exponentiations, where d is the degree of the polynomial, i.e., the input length). Finally, their protocol utilizes the Paillier encryption scheme and thus requires an RSA modulus with unknown factorization. Our protocol, on the other hand, takes a different approach and requires linear costs, for the case of equal length strings.

Efficiency. In addition to prior work, we compare our protocols to the generic garbling-technique by Yao (formally proved by Lindell and Pinkas) [LP07] for secure computation of any functionality in the two-party setting. Recall that Yao's protocol uses a Boolean circuit that computes the function, and its computational complexity is linear in the size of the circuit. Note that computing the pattern matching functionality would require a circuit of size $O(nm)$, as the circuit will compare every pattern against every text location (As noted by [GHS10], a circuit that implements the functionality for oblivious automata evaluation would require $O(mn \log m)$ gates, thus the KMP technique does not contribute to efficiency here). Consequently, our protocol for the basic pattern matching functionality is more efficient than Yao's construction even in the presence of semi-honest adversaries; this is also the case for other circuit based approaches.

Organization of this paper. We first present the underlying primitives in Section 2. The following sections then contain our protocols. The basic protocol is presented in Section 3. This is then extended, first with wildcards in the pattern (Section 4) followed by approximate matching (Section 5). Finally, the paper concludes with the protocols which hide the pattern and texts lengths (Sections 6 and 7).

2 Preliminaries and Tools

Throughout the paper, we denote the security parameter by κ. A function $\mu(\cdot)$ is *negligible* in κ (or simply *negligible*) if for every polynomial $p(\cdot)$ there exists a value K such that $\mu(\kappa) < \frac{1}{p(\kappa)}$ for all $\kappa > K$; i.e., $\mu(\kappa) = \kappa^{-\omega(1)}$. Let $X = \{X(\kappa, a)\}_{\kappa \in \mathbb{N}, a \in \{0,1\}^*}$ and $Y = \{Y(\kappa, a)\}_{\kappa \in \mathbb{N}, a \in \{0,1\}^*}$ be distribution ensembles. We say that X and Y are *computationally indistinguishable*, denoted $X \stackrel{c}{\equiv} Y$, if for every polynomial non-uniform distinguisher D there exists a negligible $\mu(\cdot)$ such that for every $\kappa \in \mathbb{N}$ and $a \in \{0,1\}^*$

$$\Big|\Pr[D(X(\kappa, a)) = 1] - \Pr[D(Y(\kappa, a)) = 1]\Big| < \mu(\kappa).$$

2.1 The ElGamal Encryption Scheme

At the core of the proposed protocols lies the additively homomorphic variation of ElGamal encryption – $E_{pk}(m, r) = \langle g^r, h^r g^m \rangle$ with distributed decryption over a group $\mathbb{G}_q$ in which DDH is hard, [ElG85]. Essentially, we use the framework of Brandt [Bra05] with minor variations. We present the computation of the parties with respect to the ciphertext space, in particular, we write C^r meaning $\langle \alpha^r, \beta^r \rangle$ and C/C' meaning $\langle \alpha/\alpha', \beta/\beta' \rangle$ for ciphertexts $C = \langle \alpha, \beta \rangle$ and $\mathbb{C}' = \langle \alpha', \beta' \rangle$, and $r \in \mathbb{Z}_q$.

2.2 Zero-knowledge Proofs for $\mathbb{G}_q$ and ElGamal Encryption

To prevent malicious behaviour, the parties must demonstrate that they are well-behaved. To achieve this, our protocols utilize zero-knowledge proofs of

knowledge. All of them are Σ-protocols (with constant communication complexity) which show knowledge of a witness that some statement is true (belong to a relation, $\mathcal{R}$) about one or more elements of $\mathbb{G}_q$. The Σ-protocols can be made secure against malicious verifiers using standard techniques; we denote the associated ideal functionalities for these protocols, $\mathcal{F}_{\text{ZK}}^{\mathcal{R}_{\text{DL}}}$, $\mathcal{F}_{\text{ZK}}^{\mathcal{R}_{\text{EqDL}}}$, $\mathcal{F}_{\text{ZK}}^{\mathcal{R}_{\text{isBit}}}$, $\mathcal{F}_{\text{ZK}}^{\mathcal{R}_{\text{mult}}}$, $\mathcal{F}_{\text{ZK}}^{\mathcal{R}_{\text{perm}}}$, and $\mathcal{F}_{\text{ZK}}^{\mathcal{R}_{\text{nze}}}$.

π_{DL}, due to Schnorr, allows the prover to demonstrate knowledge of the solution, x, to a discrete logarithm problem, [Sch89].

$$\mathcal{R}_{\text{DL}} = \{((\mathbb{G}_q, q, g, h), x) \mid h = g^x\}$$

π_{EqDL}, due to Chaum and Pedersen, demonstrates equality of two discrete logarithm problems (as well as knowledge of the solution), [CP93].

$$\mathcal{R}_{\text{EqDL}} = \{((\mathbb{G}_q, q, g_1, g_2, h_1, h_2), x) \mid h_1 = g_1^x \wedge h_2 = g_2^x\}$$

Phrased differently, π_{EqDL} demonstrates that a quadruple forms a Diffie-Hellman tuple or, equivalently, that a ciphertext is an encryption of 0.

π_{isBit} demonstrates that for ciphertext C, either C or $C\left\langle 1, g^{-1}\right\rangle$ is an encryption of 0, i.e. that it is an encryption of either 0 or 1. This can be obtained directly from π_{EqDL} using the compound proof of Cramer et al. [CGS97].

$$\mathcal{R}_{\text{isBit}} = \left\{((\mathbb{G}_q, q, g, h, \alpha, \beta), (b, r)) \mid (\alpha, \beta) = (g^r, h^r \cdot g^b) \wedge b \in \{0, 1\}\right\}$$

π_{mult}, due to Abe et al., demonstrates that a party, the prover P has performed a multiplication under the encryption correctly [ACF02]. I.e. given ciphertext C, P, knowing f, has computed $C_f = E_{pk}(f, r_f)$ and $C_\pi = C^f \cdot E_{pk}(0, r_\pi)$; clearly the plaintext of C_π is the product of the other plaintexts.

$$\mathcal{R}_{\text{mult}} = \left\{((sk, C, C_f, C_\pi), (f, r_f, r_\pi)) \text{ s.t. } \begin{array}{l} C_f = \left\langle g^{r_f}, h^{r_f} \cdot g^f \right\rangle \wedge \\ C_\pi = C^f \cdot \left\langle g^{r_\pi}, h^{r_\pi} \right\rangle \end{array}\right\}$$

π_{perm} allows a prover to demonstrate that a set of encryptions, $\{C_i\}_i$, is a permutation and rerandomization of the another, $\{C_i'\}_i$ – i.e. that their plaintexts are equal. Any protocol will do, Groth's solution [Gro03] is one possibility.

$$\mathcal{R}_{\text{perm}} = \left\{((sk, \{C_i\}_i, \{C_i'\}_i), (\pi, \{r_i\}_i)) \text{ s.t. } \langle \alpha_i', \beta_i' \rangle = \left\langle \alpha_{\pi(i)} g^{r_i}, \beta_{\pi(i)} h^{r_i} \right\rangle\right\}$$

π_{nze} demonstrates that the prover has obtained ciphertext C' from C, by raising C to a non-zero exponent and rerandomizing, i.e. $C' = C^R \cdot E_{pk}(0, r)$. The tricky part when constructing a proof of knowledge for the relation,

$$\mathcal{R}_{\text{nze}} = \left\{((sk, \alpha, \beta, \alpha', \beta'), (R, r)) \text{ s.t. } \langle \alpha', \beta' \rangle = \left\langle \alpha^R g^r, \beta^R h^r \right\rangle \wedge R \neq 0\right\},$$

is to show that $R \neq 0$. To do this, the prover, P, picks $R' \in_R \mathbb{Z}_q^*$, supplies the verifier with additional ciphertexts, $C_R = E_{pk}(R, r_R)$, $C_{R'} = E_{pk}(R', r_{R'})$, and $C_\pi = E_{pk}(RR', r_\pi)$, and executes π_{mult} twice: on (C, C_R, C') and $(C_R, C_{R'}, C_\pi)$. The prover then sends RR' to the verifier and demonstrates it is the plaintext of C_π using π_{EqDL}. Finally, the verifier checks that the RR' is non-zero.

The executions of π_{mult} demonstrate that C' has been obtained from C through exponentiation, and that the plaintext of C_π depends on R. π_{EqDL} and the final check ensures that $RR' \neq 0$ implying that so is R. Hence the protocol demonstrates that C' has been obtained correctly. Further, since the verifier receives only ciphertexts along with RR' – which is uniformly random due to R' – π_{nze} is zero-knowledge.

2.3 Distributed ElGamal Encryption

In a distributed scheme, the parties hold shares of the secret key so that the combined key remains a secret. In order to decrypt, each party uses its share to generate an intermediate computation which are eventually combined into the decryption.

Note that the Diffie-Hellman key exchange [DH76] can be used for generating a public key and an additive sharing of the corresponding secret key [Ped91]. The parties first agree on $\mathbb{G}_q$ and g. Then, each party P_i picks $s_i \in_R \mathbb{Z}_q$ and sends $h_i = g^{s_i}$ to the other. Finally, the parties compute $h = h_1 \cdot h_2$ and set $pk = \langle \mathbb{G}_q, q, g, h \rangle$. Clearly the secret key associated with this public key is $s = s_1 + s_2$. In order to ensure correct behavior, the parties must prove knowledge of their s_i by running π_{DL} on (g, h_i). We denote this protocol by π_{KeyGen} which is correlated with the functionality $\mathcal{F}_{\text{KeyGen}}(1^\kappa, 1^\kappa) = ((pk, sk_1), (pk, sk_2))$.

To decrypt a ciphertext $C = \langle \alpha, \beta \rangle$, the parties raise α to the power of their shares, send these to each other, and prove this was done correctly. Both then output $\beta/(\alpha_1\alpha_2)$. We denote this protocol by π_{Dec}. Note that this protocol allows variation where only one party obtains the decrypted result.

Our final primitive is a variation of π_{Dec} where P_1 learns whether the ciphertext m of the input $C = \langle \alpha, \beta \rangle$ is zero, but nothing more. P_2 first raises C to a random, non-zero power, rerandomizes the result, and sends it to P_1. The parties then execute π_{nze} to let P_1 verify P_2's behavior. They then decrypt the final ciphertext towards P_1, who concludes that $m = 0$ iff the masked plaintext was 0. Simulation is trivial given access to $\mathcal{F}_{\text{ZK}}^{\mathcal{R}_{\text{nze}}}$. We denote this protocol π_{dec0} and the associated ideal functionality $\mathcal{F}_{\text{dec0}}$.

3 The Basic, Linear Solution

In this section we present our solution for the classic pattern matching problem. Initially, Alice holds an n-bit string t, while Bob holds an m-bit pattern, p and the parties wish the compute the functionality $\mathcal{F}_{\text{PM}}$ defined by,

$$((p, n), (t, m)) \mapsto \begin{cases} (\{j \mid \bar{t}_j = p\}_{j=1}^{n-m+1}, \lambda) & \text{if } |p| = m \text{ and } |t| = n \\ (\lambda, \lambda) & \text{otherwise} \end{cases}$$

where λ is an empty string and $\bar{t}_j$ is the substring of length m that begins at the jth position in t. This problem has been widely studied for decades due to its potential applications and can be solved in linear time complexity [KMP77, BM77],

when no level of security is required. We examine a secure version for this problem where Alice does not gain any information about the pattern from the protocol execution, whereas Bob does not learn anything but the matched text locations. In our setting, the parties share no information (except for the input length), though it is assumed that they are connected by an authenticated communication channel, and that the inputs are over the binary alphabet. Extending this to larger alphabets is discussed below. Our protocol exhibits overall linear communication and computation costs and achieves full simulation in the presence of malicious adversaries.

Here and below, we have the parties jointly (and securely) transform their input from binary representation into elements of $\mathbb{Z}_q$ (we assume that $m < \log_2 q$; larger pattern-lengths can be accommodated, e.g. by increasing the plaintext space.), while exploiting the fact that every two consecutive substrings of the text are closely related. Informally, both parties break their inputs into bits and encrypt each bit separately. Next, the parties map every m consecutive encryptions of bits into a single encryption that denotes an m-character for which its binary representation is assembled from these m bits. Thus, the problem is reduced to comparing two elements of $\mathbb{Z}_m$ (embedded into $\mathbb{Z}_q$). The crux of our protocol is to efficiently compute this mapping.

We are now ready to give a detailed description of our construction.

Protocol π_{PM}

- **Inputs:** The input of *Alice* is a binary string t of length n and an integer m, whereas the input of *Bob* is a binary string p of length m and an integer n. The parties share a security parameter 1^κ as well.
- **The protocol:**
  1. Alice and Bob run protocol $\pi_{\mathrm{KeyGen}}(1^\kappa, 1^\kappa)$ to generate a public key $pk = \langle \mathbb{G}_q, q, g, h \rangle$, and the respective shares s_A and s_B of the secret key sk of Alice and Bob.
  2. Bob sends encryptions $P_i = E_{pk}(p_i; r_{p_i})$, $i = 1, \ldots, m$, of his m-bit pattern, p, to Alice. Further, for each encryption the parties run the zero-knowledge proof of knowledge π_{isBit}, allowing Alice to verify that the plaintext of P_i is a bit known to Bob, i.e. that he has provided a bit-string of length m. Both parties then compute an encryption of Bob's pattern,
  $$P \leftarrow \prod_{i=1}^{m} P_i^{2^{i-1}} \tag{1}$$
  using the homomorphic property of ElGamal encryption.
  3. Alice sends encryptions, $T_j = E_{pk}(t_j; r_{t_j})$ $j = 1, \ldots, n$, of the bits t_j of her n-bit text, t, to Bob. Further, for each encryption the parties run π_{isBit}, allowing Bob to verify that the plaintext of T_j is a bit known to Alice, i.e. that she has indeed provided the encryption of a bit-string of length n that she knows.
  4. Let $\bar{t}_j$ be the m-bit substring of Alice's text t, starting at position $j = 1, \ldots, n - m + 1$. For each such string both parties compute an encryption of that string,
  $$\bar{T}_j \leftarrow \prod_{i=j}^{j+m-1} T_i^{2^{i-j}}. \tag{2}$$

5. For every $\bar{T}_j$, $j = 1, \ldots, n-m+1$, both parties compute
$$\Delta_j \leftarrow \bar{T}_j \cdot P^{-1}. \tag{3}$$
6. For every Δ_j $j = 1, \ldots, n-m+1$, Alice and Bob reveal to Bob whether its plaintext δ_j is zero by running π_{dec0}. Bob then outputs j if this is the case.

CORRECTNESS OF π_{PM}. Before turning to our proof, we explain the intuition and demonstrate that protocol π_{PM} correctly determines which substrings of the text t match the pattern p. Recall that the value P that is computed in Eq. (1) (Step 2) is an encryption of Bob's pattern, $p = \sum_{i=1}^{m} 2^{i-1} p_i$. This follows from the homomorphic property of ElGamal encryption,

$$P = \prod_{i=1}^{m} P_i^{2^{i-1}} = E_{pk}\left(\sum_{i=1}^{m} 2^{i-1} p_i, \sum_{i=1}^{m} 2^{i-1} r_{p_i}\right). \tag{4}$$

Note that P is obtained deterministically from the P_i, hence both Alice and Bob hold the same fixed encryption. Similarly, in Eq. (2) computed in Step 4, the parties compute encryptions of the substrings of length m of Alice's text,

$$\bar{t}_j = \sum_{i=j}^{j+m-1} 2^{i-j} t_i,$$

see a detailed discussion in the complexity paragraph regarding the efficiency of this step. As with P, the parties hold the same, fixed encryptions (with randomness $r_{\bar{t}_j} = \sum_{i=j}^{j+m-1} 2^{i-j} r_{t_i}$). The encryption Δ_j computed by Eq. (3) is an encryption of $\delta_j = \bar{t}_j - p$, i.e. the $(\mathbb{Z}_q)$ difference between the substring of the text starting at position j and the pattern.

$$\begin{aligned} \Delta_j &= \bar{T}_j \cdot P^{-1} \\ &= E_{pk}\left(\bar{t}_j - p, r_{\bar{t}_j} - r_p\right) \end{aligned}$$

At this point, it simply remains for Bob to securely determine which of the Δ_j are encryptions of zero, as

$$\delta_j = 0 \Leftrightarrow \bar{t}_j = p.$$

SECURITY OF π_{PM}. We are now ready to prove the following theorem,

Theorem 1 (linear pattern matching): *Assume that π_{KeyGen}, π_{dec0} and π_{isBit} are as described in Section 2 and that (G, E, D) is the ElGamal scheme. Then π_{PM} securely computes $\mathcal{F}_{\text{PM}}$ in the presence of malicious adversaries.*

Proof. We separately prove security in the case that Alice is corrupted and the case that Bob is corrupted. Our proof is in a hybrid model where a trusted party computes the ideal functionalities $\mathcal{F}_{\text{KeyGen}}$, $\mathcal{F}_{\text{dec0}}$ and $\mathcal{F}_{\text{ZK}}^{\mathcal{R}_{\text{isBit}}}$.

Bob is corrupted. Let $\mathcal{A}$ denote an adversary controlling Bob. In this case we need to prove that Bob does not learn anything but the matching text locations. We construct a simulator $\mathcal{S}$ as follows,

1. $\mathcal{S}$ is given a pattern p of length m, an integer n and $\mathcal{A}$'s auxiliary input and invokes $\mathcal{A}$ on these values.
2. $\mathcal{S}$ emulates the trusted party for π_{KeyGen} as follows. It first chooses two random elements $s_A, s_B \in \mathbb{G}_q$ and hands $\mathcal{A}$, its share s_B and the public key $\langle \mathbb{G}_q, q, g, h = g^{s_A \cdot s_B} \rangle$.
3. $\mathcal{S}$ receives from $\mathcal{A}$, m encryptions and $\mathcal{A}$'s input for the trusted party for $\mathcal{F}_{\text{ZK}}^{\mathcal{R}_{\text{isBit}}}$. If the conditions for which the functionality outputs 1 are not met, $\mathcal{S}$ aborts by sending $\perp$ to the trusted party for $\mathcal{F}_{\text{PM}}$ and outputs whatever $\mathcal{A}$ outputs.
4. Otherwise, $\mathcal{S}$ defines P according to the witness for π_{isBit} and sends it to its trusted party. Let Z be the set of returned indices.
5. $\mathcal{S}$ defines a text t' that is consistent with Z. That is, for every $j \in Z$, $\mathcal{S}$ defines the substring $t'_j = p_1, \ldots, t'_{j+m-1} = p_m$. For the remaining indices $\mathcal{S}$ uses the bit one. ($\mathcal{S}$ verifies that the only matches in t' indeed correspond to the indices from set Z). $\mathcal{S}$ completes the execution as the honest Alice would on input t'.
6. If at any point $\mathcal{A}$ sends an invalid message $\mathcal{S}$ aborts, sending $\perp$ to the trusted party for $\mathcal{F}_{\text{PM}}$. Otherwise, it outputs whatever $\mathcal{A}$ does.

It is immediate to see that $\mathcal{S}$ runs in probabilistic polynomial time. We prove next that the adversary's views are computational indistinguishable via a reduction to the security of ElGamal. Recalling that the only difference within these views is with respect to the text locations that do not match the pattern, (as $\mathcal{S}$ uses the bit one instead of the actual bit value from t) we reduce the ability to distinguish these views to the ability to distinguish the encryptions of the real text against the simulated one for these locations.

Assume there exists a distinguisher D for these executions, we construct a distinguisher D_E breaking the semantic security of ElGamal encryption as follows. Upon receiving a public key pk and auxiliary input t, D_E engages in an execution of π_{KeyGen} with $\mathcal{A}$ and sends it (s_B, pk) where $s_B \in_R \mathbb{Z}_q$. D_E continues emulating the role of Alice as $\mathcal{S}$ does except for Step 3 where it needs to send the encryptions of $t_1, \ldots, t_n$. In this step D_E outputs two sets of plaintexts: (i) $t_1, \ldots, t_n$ and, (ii) $t'_1, \ldots, t'_n$. We denote by $\tilde{T}_1, \ldots, \tilde{T}_n$ the set of encryptions it receives back. D_E hands $\mathcal{A}$ this set and completes the run as follows. In Step 6 D_E replaces Δ_j with an encryption of zero if and only if $j \in Z$. Otherwise, D_E sends an encryption of a random value in $\mathbb{Z}_q^*$. Clearly, this step is computed differently than in both the hybrid and simulated executions. Nevertheless, we claim that the distributions on the encryptions are identical. This is due to the fact that for every matched text location the masking result equals zero, and for every non-matching text location the masking result equals a random element of $\mathbb{Z}_q^*$. Hence, the adversary's views are identical.

Finally, D_E invokes D on $\mathcal{A}$'s output and outputs whatever D outputs. Note that if D_E is given the encryptions of t then the adversary's view is distributed

as in the hybrid execution. Moreover, if it receives an encryption of t', then the adversary's view is as in the simulation with $\mathcal{S}$.

Alice is corrupted. Since Alice does not receive any output from the execution, we only need to prove that privacy is preserved, and that Bob's output cannot be affected (except with negligible probability). The proof follows the outlines of the former case. Therefore, due to space considerations we omit the details here.

Complexity of π_{PM}. The round complexity is constant, as the key generation process and the zero knowledge proofs run in constant rounds. Further, the number of group elements exchanged is bounded by $O(n+m)$, as there are $n-m+1$ substrings of length m and each zero-knowledge proof requires constant number of group elements.

Regarding computational complexity, it is clear that except for Step 4 at most $O(m+n)$ exponentiations are required. Note first that Eq. (2) can be implemented using the *square and multiply* technique. Namely, for every $j = 1,\ldots,n-m+1$, $\bar{T}_j$ is computed by $(\ldots((T_j)^2 \cdot T_{j+1})^2 \cdot T_{j+2}\ldots)^2 \cdot T_{j+m-1}$.

This requires $O(m)$ multiplications for each text location, which amounts to total $O(nm)$ multiplications for the entire text. Reducing the number of multiplications into $O(n)$ (on the expense of increasing the number of exponentiations) can be easily shown. Loosely speaking, in addition to sending an encryption of 0 or 1 for each text location, Alice sends an encryption of 0 or 2^m, respectively, and proves consistency. From practical point of view, it may be much more efficient to compute $O(m)$ multiplications for each location, than proving this consistency (even though it only requires a constant number of exponentiations.)

Finally, note that our protocols utilize ElGamal encryption which can be implemented over an elliptic curve group. This may reduce the modulus value dramatically, as now only 160 bits are typically needed for the size of the key.

3.1 Variations

Non-binary alphabets. Alphabets of larger size, s, can be handled by encoding the characters as elements of $\mathbb{Z}_s$ and using s-ary rather than binary notation for the $\bar{T}_j$ and P. Proving in ZK that an encryption contains a valid character is straightforward, e.g. it can be provided in binary (which of course requires $O(\log s)$ encryptions).

Long patterns. When the pattern length, m, (or the alphabet size, s) is large, requiring $q > s^m$ may not be acceptable. This can be avoided by encoding the pattern p and substrings $\bar{t}_j$ into multiple $\mathbb{Z}_q$ values, $\{p^{(i)}\}_i, \{\bar{t}_j^{(i)}\}_i$. Having computed encryptions $\{\Delta_i\}_i$ of the differences $\{\delta_i = p^{(i)} - \bar{t}_j^{(i)}\}_i$, Alice raises each encryption to a random, non-zero exponents r_i, rerandomizes them and sends them to Bob (and proves that everything was done correctly). The parties then executes π_{dec0} on the product of these encryptions and Bob reports a match if a 0 is found. Note that the plaintext of this product is $\sum_i r_i \cdot \delta_i$. Thus, if the pattern matches, all $\delta_i = 0$ implying that this is an encryption of 0. If one or more $\delta_i \neq 0$, then the probability of this being an encryption of 0 is negligible.

HIDING MATCH LOCATIONS. It may be required that Bob only learns the number of matches and not the actual locations of the hits. One example is determining *how frequently* some gene occurs rather than *where* it occurs in some DNA sequence. This is easily achieved by simply having Alice pick a uniformly random permutation and permute (and rerandomize) the Δ_j of Eq. (3). The encryptions are sent to Bob, and π_{perm} is executed, allowing him to verify Alice's behavior. Finally, π_{dec0} is run and Bob outputs the number of encryptions of 0 received.

Correctness is immediate: An encryption of 0 still signals that a match occurred. However, due to the random permutation that Alice applies, the locations are shuffled, implying that Bob does not learn the actual matches.

4 Secure Pattern Matching with Wildcards

The first variant of the classical pattern matching problem allows Bob to place wildcards, denoted by $\star$, in his pattern; these should match both 0 and 1. More formally, the parties wish the compute the functionality $\mathcal{F}_{\text{PM}-\star}$ defined by,

$$((p,n),(t,m)) \mapsto \begin{cases} (\{j \mid \bar{t}_j \overset{\star}{\equiv} p\}_{j=1}^{n-m+1}, \lambda) & \text{if } |p| = m \text{ and } |t| = n \\ (\lambda, \lambda) & \text{otherwise} \end{cases}$$

where $\bar{t}_j$ is the substring of length m that begins at the jth position of t and $\overset{\star}{\equiv}$ is defined as "equal except with respect to $\star$-positions." This problem has been widely looked at by researchers with the aim to generalize the basic searching model to searching with errors. This variant is known as *pattern matching with don't cares* and can be solved in $O(n+m)$ time [IR07]. The secure version of this problem guarantees that Alice will not be able to trace the locations of the don't cares in addition to the security requirement introduced for the basic problem.

The core idea of the solution is to proceed as in the standard one with two exceptions: Bob must supply the wildcard positions in encrypted form, and the substrings of Alice's text must be modified to ensure that they will match (i.e. equal) the pattern at those positions. Achieving correctness and ensuring correct behavior requires substantial modification of the protocol. Intuitively, for every m-bit substring $\bar{t}_j$ of t, Bob replaces Alice's value by 0 at the wildcard positions resulting in a string $\bar{t}'_j$, see Step 6 below. Similarly, a pattern p' is obtained from p by replacing the wildcards by 0. Clearly this ensures that the bits of $\bar{t}'_j$ and p' are equal at all wildcard positions. Thus, $\bar{t}'_j = p'$ precisely when $\bar{t}_j$ equals p at all non-wildcard positions.

Protocol $\pi_{\text{PM}-\star}$

- **Inputs:** The input of *Alice* is a binary string t of length n and an integer m, whereas the input of *Bob* is a string p over the alphabet $\{0,1,\star\}$ of length m and an integer n. The parties share a security parameter 1^κ as well.
- **The protocol:**
  1. Alice and Bob run protocol $\pi_{\text{KeyGen}}(1^\kappa, 1^\kappa)$ to generate a public key $pk = \langle \mathbb{G}_q, q, g, h \rangle$, and the respective shares s_A and s_B of the secret key sk.

2. For each position $i = 1, \ldots, m$, Bob first replaces $\star$ by 0

$$p'_i \leftarrow \begin{cases} 1 & \text{if } p_i = 1 \\ 0 & \text{otherwise} \end{cases}.$$

He then sends encryptions $P'_i = E_{pk}(p'_i; r_{p'_i})$ for $i = 1, \ldots, m$ to Alice, and for each one they execute π_{isBit}. Finally, both parties compute an encryption of Bob's "pattern" in binary,

$$P' \leftarrow \prod_{i=1}^{m} {P'_i}^{2^{i-1}}.$$

3. For each position $i = 1, \ldots, m$ of Bob's pattern, he computes a bit denoting the occurrences of a $\star$,

$$w_i \leftarrow \begin{cases} 0 & \text{if } p_i = \star \\ 1 & \text{otherwise} \end{cases}.$$

He then encrypts these and sends the result to Alice,

$$W_i \leftarrow E_{pk}\left(w_i, r_{w_i}\right),$$

and the two run π_{isBit} for each one.

4. For each $i = 1, \ldots, m$, Bob and Alice run π_{isBit} on W_i / P'_i. This demonstrates to Alice that if p'_i is set, then so is w_i, i.e. that only 0's occur at wildcard position.
5. Alice supplies her input as in Step 3 of Protocol π_{PM} in Section 3. She sends encryptions, $T_j = E_{pk}(t_j; r_{t_j})$ $j = 1, \ldots, n$, of the bits of t to Bob. Then the parties run π_{isBit} for each of the encryptions.
6. For every entry $i = 1, \ldots, m$ of every m-bit substring of t starting at position $j = 1, \ldots, n - m + 1$, Bob computes an encryption

$$\hat{T}_{j,i} \leftarrow (T_{j+i-1})^{w_i} \cdot E_{pk}\left(0, r_{j,i}\right).$$

He sends these to Alice, and they run π_{mult} on each triple $\left(T_{j+i-1}, W_i, \hat{T}_{j,i}\right)$, allowing Alice to verify that Bob has correctly multiplied the plaintexts of the W_i onto the T_{j+i-1}. Both parties then compute encryptions of the modified substrings of Alice's text

$$\bar{T}'_j \leftarrow \prod_{i=1}^{m} \left(\hat{T}_{j,i}\right)^{2^{i-1}}.$$

7. The protocol concludes as Protocol π_{PM} does. For each of the $\bar{T}'_j$ where $j = 1, \ldots, n - m + 1$, the parties compute

$$\Delta_j \leftarrow \bar{T}'_j \cdot {P'}^{-1},$$

and run π_{dec0}. This reveals to Bob which of plaintexts δ_j are 0. For each $\delta_j = 0$ he concludes that the pattern matched and outputs j.

To see that the protocol does not introduce new opportunities for malicious behavior, first note that Alice specification is essentially as in the basic protocol

π_{PM}. Regarding Bob, the proofs of correct behavior limit him to supplying an input that an honest Bob could have supplied as well. Bob's input, p'_i $i = 1, \ldots, m$, is first shown to be a bit string, Step 2. The invocations of π_{isBit} of Step 3 then ensure that so is the "wildcard string." Finally, in Step 4 it is verified that for each wildcard p_i of p, $p'_i = 0$. In other words, there is a valid input where the honest Bob would send encryptions of the values that the malicious Bob can use. The only remaining option for a malicious Bob is in Step 6, however, the invocations of π_{mult} ensure his correct behavior. Formal simulation is analogous to that in Section 3. We state the following theorem:

Theorem 2 (wildcards): *Assume that* π_{KeyGen}, π_{dec0}, π_{isBit}, *and* π_{mult} *are as described in Section 2 and that* (G, E, D) *is the ElGamal scheme. Then* $\pi_{\text{PM}-\star}$ *securely computes* $\mathcal{F}_{\text{PM}-\star}$ *in the presence of malicious adversaries.*

Regarding complexity, clearly the most costly part of the protocol is Step 6 which requires Bob to sends $\Theta(nm)$ encryptions, $T_{j,i}$ to Alice, as well as an invocation of π_{mult} for each of them. Hence, communication and computation complexity is increased to $O(nm)$, while round complexity remains constant.

5 Secure Approximate Matching

The second variation considered is approximate pattern matching: Alice holds an n-bit string t, while Bob holds an m-bit pattern p. The parties wish to determine approximate matches – strings with Hamming distance less than some threshold $\tau \leq m$. This is captured by the functionality $\mathcal{F}_{\text{APM}}$ defined by,

$$((p, n, \tau), (t, m, \tau')) \mapsto \begin{cases} (\{j \mid \delta_H(\bar{t}_j, p) < \tau\}_{j=1}^{n-m+1}, \lambda) & \text{if } |p| = m \geq \tau = \tau' \\ & \text{and } |t| = n \\ (\lambda, \lambda) & \text{otherwise} \end{cases}$$

where δ_H denotes Hamming distance and $\bar{t}_j$ is the substring of length m that begins at the jth position in t. We assume that the parties share some threshold $\tau \in \mathbb{N}$. Note that this problem is an extension of pattern matching with don't cares problem introduced in Section 4. Bob is able to learn *all* the matches within some error bound instead of learning the matches for specified error locations.

Two of the most important applications of approximate pattern matching are spell checking and matching DNA sequences. The most recent algorithm for solving this problem without considering privacy is by Amir et al. [ALP00], which introduced a solution in time $O(n\sqrt{\tau \log \tau})$. Our solution achieves $O(nm)$ computation and communication complexity.

The main idea behind the construction is to have the parties securely supply their inputs in binary as above. Then, to determine the matches, the parties first compute the (encrypted) Hamming distances h_j using the homomorphic properties of ElGamal encryption (Steps 5 and 6). They then check whether $h_j = k$ for each $k < \tau$. To avoid leaking information, these results are permuted before the final decryption.

Protocol π_{APM}

- **Inputs:** The input of *Alice* is a binary string t of length n, an integer m and a threshold τ', whereas the input of *Bob* is a binary string p of length m, an integer n and a threshold τ. The parties share a security parameter 1^κ as well.
- **The protocol:**
  1. Alice and Bob run protocol $\pi_{\text{KeyGen}}(1^\kappa, 1^\kappa)$ to generate a public key $pk = \langle \mathbb{G}_q, q, g, h \rangle$, and the respective shares s_A and s_B of the secret key sk.
  2. Alice sends Bob τ' and the parties continue if $\tau = \tau'$.
  3. As in the basic solution, Bob first sends encryptions $P_i = E_{pk}(p_i; r_{p_i})$ $i = 1, \ldots, m$, of the bits of his m-bit pattern, p, to Alice. They then run π_{isBit} for each one.
  4. Alice similarly provides encryptions, $T_j = E_{pk}(t_j; r_{t_j})$ $j = 1, \ldots, n$ of her input as in π_{PM}; for each one the parties execute π_{isBit}.
  5. For every entry $i = 1, \ldots, m$ of every m-bit substring of t starting at position $j = 1, \ldots, n - m + 1$, Bob computes an encryption
  $$\Pi_{j,i} \leftarrow T_{j+i-1}^{p_i} \cdot E_{pk}(0, r_{j,i}). \tag{5}$$
  He sends these to Alice, and for each triple $(T_{j+i-1}, P_i, \Pi_{j,i})$ the parties run π_{mult}. This allows Alice to verify that Bob has correctly multiplied the plaintexts of the P_i onto the T_{j+i-1}.
  6. For every entry $i = 1, \ldots, m$ of every m-bit substring of t starting at position $j = 1, \ldots, n - m + 1$, both parties compute encryptions $X_{j,i}$,
  $$X_{j,i} \leftarrow T_{j+i-1} \cdot P_i \cdot \Pi_{j,i}^{-2}.$$
  Note that as the plaintext of $\Pi_{j,i}$ is $p_i \cdot t_{j+i-1}$, the plaintext of $X_{j,i}$ is $p_i \oplus t_{j+i-1}$. For every $j = 1, \ldots, n - m + 1$ – i.e. for every substring – both parties compute
  $$H_j \leftarrow \prod_{i=1}^{m} X_{j,i}.$$
  7. For every $k = 0, \ldots, \tau - 1$ (i.e. for every Hamming distance which would be considered a match) and for every substring of length m starting at $j = 1, \ldots, n - m + 1$, both parties compute
  $$\Delta_{j,k} \leftarrow H_j \cdot \left\langle 1, g^{-k} \right\rangle. \tag{6}$$
  8. For every $j = 1, \ldots, n - m + 1$, Alice picks a uniformly random permutation $\pi_j : \mathbb{Z}_\tau \to \mathbb{Z}_\tau$ and applies π_j to the set $\{\Delta_{j,k}\}_k$,
  $$\left(\Delta'_{j,0}, \ldots, \Delta'_{j,\tau-1}\right) \leftarrow \pi_j \left(\Delta_{j,0}, \ldots, \Delta_{j,\tau-1}\right),$$
  rerandomizes all encryptions,
  $$\Delta''_{j,k} \leftarrow \Delta'_{j,k} \cdot E_{pk}\left(0, r'_{j,k}\right)$$
  for $j = 1, \ldots, n - m + 1$ and $k = 0, \ldots, \tau - 1$, and sends the $\Delta''_{j,k}$ to Bob. For every permutation, $j = 1, \ldots, n - m + 1$, the parties execute π_{perm} on $\left(\left(\Delta_{j,0}, \ldots, \Delta_{j,\tau-1}\right), \left(\Delta''_{j,0}, \ldots, \Delta''_{j,\tau-1}\right)\right)$ allowing Bob to verify that the plaintexts of the $\Delta''_{j,k}$ correspond to those of the $\Delta_{j,k}$ for all (fixed) j.

9. Finally, Alice and Bob execute π_{dec0} on each $\Delta''_{j,k}$ for $j = 1, \ldots, n-m+1$ and $k = 0, \ldots, \tau - 1$. This reveals to Bob which plaintexts $\delta_{j,k}$ are 0. He then outputs j iff this is the case for one of $\delta''_{j,0}, \ldots, \delta''_{j,\tau-1}$.

Correctness follows from the intuition: The plaintexts of the H_j from Equation (5) are the sum of the ones of the $X_{j,i}$ $i = 1, \ldots, m$. I.e. it is the number of differing bits of p and $\bar{t}_j$ – the Hamming distance – as the plaintext of $X_{j,i}$ is $t_{j+i-1} + p_i - 2 \cdot t_{j+i-1} \cdot p_i = t_{j+i-1} \oplus p_i$.

Each threshold test is performed using τ tests of equality, one for each possible value $k < \tau$, where each test simply subtracts the associated k from H_j under the encryption, Eq. (6), at which point the parties may mask and decrypt towards Bob. Note that the standard masking combined with the permutation of Step 8 ensures that for every potential match, Bob either receives τ uniformly random encryptions of random, non-zero values, or $\tau - 1$ such encryptions and a single encryption of zero. Hence we state the following theorem:

Theorem 3 (approximate): *Assume that π_{KeyGen}, π_{dec0} and π_{isBit}, and π_{mult} are as described in Section 2 and that (G, E, D) is the ElGamal scheme. Then π_{APM} securely computes $\mathcal{F}_{\text{APM}}$ in the presence of malicious adversaries.*

Regarding complexity, the most expensive steps are those associated with computing the Hamming distances, Steps 5 and 6, as there are $\Theta(nm)$ $\Pi_{j,i}$ and $X_{j,i}$. The concluding steps – computing, randomizing (permuting), and decrypting the $\Delta_{j,k}$ – require $\Theta(n\tau)$ work, however, as $\tau \leq m$ this is no more expensive. Hence overall communication and computation is $O(mn)$, while round complexity is constant as in the previous solutions.

6 Hiding the Pattern Length

Here Alice is not required to know the length m of Bob's pattern, only an upper bound $M \geq m$. Moreover, she will not learn any information about m. More formally, the parties wish to compute the functionality $\mathcal{F}_{\text{PM-hpl}}$ defined by,

$$((p, n), (t, M)) \mapsto \begin{cases} (\{j \mid \bar{t}_j = p\}_{j=1}^{n-m+1}, \lambda) & \text{if } |p| \leq M \text{ and } |t| = n \\ (\lambda, \lambda) & \text{otherwise} \end{cases}$$

where $\bar{t}_j$ is the substring of length m that begins at the jth position in t. A protocol $\pi_{\text{PM-hpl}}$ that realizes $\mathcal{F}_{\text{PM-hpl}}$ can be obtained through minor alterations of $\pi_{\text{PM-}\star}$. Due to space constraints we only sketch these, and postpone the detailed description and simulator proof to the full version of the paper.

The main idea is to have Bob construct a pattern $\hat{p}$ of length M by padding p with $M - m$ wildcards. Though not completely correct, intuitively, executing $\pi_{\text{PM-}\star}$ on input $((\hat{p}, n), (t, M))$ provides the desired result, as the wildcards ensure that the irrelevant postfixes of the $\bar{t}_j$ are "ignored." There are two reasons why this does not suffice. Firstly, the wildcards of $\pi_{\text{PM-}\star}$ mean *match any character*, however, matches must also be found when the wildcards occur *after* the

end of the text (where there are no characters). Secondly, a malicious Bob must not have full access to wildcard-usage – i.e. he must not be able to arbitrarily place wildcards, they must occur only at the end of $\hat{p}$.

– **Matching $\bar{t}_j$ when $j > n - M + 1$:** The solution to the former problem is completely straightforward: extend (pad) t with symbols that only match wildcards. Going into more detail, first let $N = n + M - 1$. The parties pad Alice's encrypted text, $T_1, \ldots, T_n$ with $M - 1$ default encryptions of 2,

$$T_{n+1} = \ldots = T_N = \langle 1, g^2 \rangle .$$

Then, rather than use a binary representation for the encryptions P' and $\bar{T}'_j$ (Steps 2 and 6 of $\pi_{\text{PM}-\star}$), we use a ternary representation

$$\hat{P} \leftarrow \prod_{i=1}^{M} (P'_i)^{3^{i-1}}, \qquad \bar{T}'_j \leftarrow \prod_{i=j}^{j+M-1} \hat{T}_{j,i}^{3^{i-j}} .$$

Intuitively, this works as we have simply extended our alphabet with an additional character, 2.

– **Ensuring a proper $\hat{p}$:** To prevent malicious behavior, Bob should demonstrate to Alice that $\hat{p}$ has been properly constructed, i.e. that all wildcards occur at the end of the pattern. This can be done by showing that $w_1, \ldots, w_M$ is monotonically non-increasing, i.e. that a 1 (non-wildcard) never follows a 0 (wildcard). Bob can demonstrate this fact by executing π_{isBit} on W_i/W_{i+1} for $i = 1, \ldots, M - 1$.

Complexity is equivalent to $\pi_{\text{PM}-\star}$. We conclude with the following theorem,

Theorem 4 (pattern length hiding): *Assume that π_{KeyGen}, π_{Dec}, π_{isBit}, and π_{mult} are as described in Section 2 and that (G, E, D) is the ElGamal scheme. Then $\pi_{\text{PM}-\text{hpl}}$ securely computes $\mathcal{F}_{\text{PM}-\text{hpl}}$ in the presence of malicious adversaries.*

7 Hiding the Text Length

The final variant does not require Bob to know the actual text length n, only an upper bound $N \geq n$. Moreover, he learns no information about n other than what can be inferred from the output. This property is desirable in applications where it is crucial to hide the size of the database as it gives away sensitive information. More formally, the parties wish the compute the functionality $\mathcal{F}_{\text{PM}-\text{htl}}$,

$$((p, N), (t, m)) \mapsto \begin{cases} (\{j \mid \bar{t}_j = p\}_{j=1}^{n-m+1}, \lambda) & \text{if } |p| = m \text{ and } |t| \leq N \\ (\lambda, \lambda) & \text{otherwise} \end{cases}$$

where $\bar{t}_j$ is the substring of length m that begins at the jth position in t.

Due to space constraints, we only sketch the solution. The core idea is to have Alice pad her text with $N - n$ 2s, and then demonstrate that any 2s occur at the end. The details of the solution are similar to those of $\pi_{\text{PM}-\text{hpl}}$ above.

Regarding complexity, it can be shown that only $O(N+m)$ encryptions change hands, hence only this many zero-knowledge proofs of knowledge are needed as well; i.e. communication and computation complexity are linear. The required number of rounds is constant.

Theorem 5 (text length hiding): *Assume that* π_{KeyGen}, π_{dec0} *and* π_{isBit} *are as described in Section 2 and that* (G, E, D) *is the ElGamal scheme. Then* $\pi_{\text{PM-htl}}$ *securely computes* $\mathcal{F}_{\text{PM-htl}}$ *in the presence of malicious adversaries.*

References

[ACF02] Abe, M., Cramer, R., Fehr, S.: Non-interactive distributed-verifier proofs and proving relations among commitments. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 206–223. Springer, Heidelberg (2002)

[ACR99] Allauzen, C., Crochemore, M., Raffinot, M.: Factor oracle: A new structure for pattern matching. In: Bartosek, M., Tel, G., Pavelka, J. (eds.) SOFSEM 1999. LNCS, vol. 1725, pp. 295–310. Springer, Heidelberg (1999)

[ALP00] Amir, A., Lewenstein, M., Porat, E.: Faster algorithms for string matching with mismatches. In: SODA, San Francisco, California, USA, pp. 794–803 (2000)

[AMP04] Aggarwal, G., Mishra, N., Pinkas, B.: Secure computation of the k'th-ranked element. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 40–55. Springer, Heidelberg (2004)

[Bea92] Beaver, D.: Foundations of secure interactive computing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 377–391. Springer, Heidelberg (1992)

[Blo70] Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. Commun. ACM 13(7), 422–426 (1970)

[BM77] Boyer, R.S., Strother Moore, J.: A fast string searching algorithm. Commun. ACM 20(10), 762–772 (1977)

[Bra05] Brandt, F.: Efficient cryptographic protocol design based on distributed el gamal encryption. In: Won, D.H., Kim, S. (eds.) ICISC 2005. LNCS, vol. 3935, pp. 32–47. Springer, Heidelberg (2006)

[Can00] Canetti, R.: Security and composition of multi-party cryptographic protocols. Journal of Cryptology 13, 143–202 (2000)

[CGS97] Cramer, R., Gennaro, R., Schoenmakers, B.: A secure and optimally efficient multi-authority election scheme. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 103–118. Springer, Heidelberg (1997)

[CP93] 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)

[DH76] Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Transactions on Information Theory 22(6), 644–654 (1976)

[ElG85] ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory 31(4), 469–472 (1985)

[FNP04] Freedman, M.J., Nissim, K., Pinkas, B.: Efficient private matching and set-intersection. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 1–19. Springer, Heidelberg (2004)

[GHS10] Gennaro, R., Hazay, C., Sorensen, J.S.: Automata evaluation and text search protocols with simulation based security. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 145–160. Springer, Heidelberg (2010)

[GL91] Goldwasser, S., Levin, L.A.: Fair computation of general functions in presence of immoral majority. In: Menezes, A., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 77–93. Springer, Heidelberg (1991)

[GMW87] Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game. In: Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing, STOC 1987, pp. 218–229. ACM, New York (1987)

[Gol04] Goldreich, O.: Foundations of Cryptography: Basic Applications, vol. 2. Cambridge University Press, New York (2004)

[Gro03] Groth, J.: A verifiable secret shuffle of homomorphic encryptions. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 145–160. Springer, Heidelberg (2002)

[HL08] Hazay, C., Lindell, Y.: Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 155–175. Springer, Heidelberg (2008)

[IR07] Iliopoulos, C.S., Sohel Rahman, M.: Pattern matching algorithms with don't cares. In: van Leeuwen, J., Italiano, G.F., van der Hoek, W., Meinel, C., Sack, H., Plášil, F. (eds.) SOFSEM 2007. LNCS, vol. 4362, pp. 116–126. Springer, Heidelberg (2007)

[JP09] Jarrous, A., Pinkas, B.: Secure hamming distance based computation and its applications. In: Abdalla, M., Pointcheval, D., Fouque, P.-A., Vergnaud, D. (eds.) ACNS 2009. LNCS, vol. 5536, pp. 107–124. Springer, Heidelberg (2009)

[KM10] Katz, J., Malka, L.: Secure text processing with applications to private dna matching. In: To appear CCS (2010)

[KMP77] Knuth, D.E., Morris Jr., J.H., Pratt, V.R.: Fast pattern matching in strings. SIAM J. Comput. 6(2), 323–350 (1977)

[KS05] Kissner, L., Song, D.X.: Privacy-preserving set operations. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 241–257. Springer, Heidelberg (2005)

[LP02] Lindell, Y., Pinkas, B.: Privacy preserving data mining. Journal of Cryptology 15(3), 177–206 (2002)

[LP07] 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)

[MR91] Micali, S., Rogaway, P.: Secure computation (abstract) (This is preliminary version of unpublished 1992 manuscript). In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 392–404. Springer, Heidelberg (1992); This is preliminary version of unpublished (1992) (manuscript)

[NM07] Navarro, G., Mäkinen, V.: Compressed full-text indexes. ACM Comput. Surv. 39(1), 2 (2007)

[Ped91] 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)

[Sch89] Schnorr, C.P.: Efficient identification and signatures for smart cards. In: Quisquater, J.-J., Vandewalle, J. (eds.) CRYPTO 1989. LNCS, vol. 434, pp. 239–252. Springer, Heidelberg (1990)

[TPKC07] Troncoso-Pastoriza, J.R., Katzenbeisser, S., Celik, M.: Privacy preserving error resilient dna searching through oblivious automata. In: Proceedings of the 14th ACM conference on Computer and communications security, CCS 2007, pp. 519–528. ACM, New York (2007)

[Yao86] Yao, A.C.-C.: How to generate and exchange secrets. In: Proceedings of the 27th Annual Symposium on Foundations of Computer Science, SFCS 1986, Washington, DC, USA, pp. 162–167. IEEE Computer Society, Los Alamitos (1986)

Linear-Complexity Private Set Intersection Protocols Secure in Malicious Model

Emiliano De Cristofaro[1], Jihye Kim[2], and Gene Tsudik[1]

[1] Computer Science Department, University of California, Irvine
[2] Department of Mathematical Sciences, Seoul National University

Abstract. Private Set Intersection (PSI) protocols allow one party ("client") to compute an intersection of its input set with that of another party ("server"), such that the client learns nothing other than the set intersection and the server learns nothing beyond client input size. Prior work yielded a range of PSI protocols secure under different cryptographic assumptions. Protocols operating in the semi-honest model offer better (linear) complexity while those in the malicious model are often significantly more costly. In this paper, we construct PSI and Authorized PSI (APSI) protocols secure in the malicious model under standard cryptographic assumptions, with both *linear* communication and computational complexities. To the best of our knowledge, our APSI is the first solution to do so. Finally, we show that our linear PSI is appreciably more efficient than the state-of-the-art.

1 Introduction

Private set intersection (PSI) protocols allow two parties – a server and a client – to interact on their respective input sets, such that the client only learns the intersection of the two sets, while the server learns nothing (beyond the client input set size). PSI addresses several realistic privacy issues. Typical application examples include:

1. **Aviation Security:** The U.S. Department of Homeland Security (DHS) needs to check whether any passenger on each flight from/to the United States must be denied boarding or disembarkation, based on so-called *Terror Watch List*. Today, airlines surrender their entire passenger manifests to DHS, together with other sensitive information, such as credit card numbers. Besides privacy implications, this modus operandi poses liability issues with regard to (for the most part) innocent passengers' data and concerns about potential data losses. Ideally, DHS would obtain information *only* pertaining to passengers on the list, while not disclosing any information to the airlines.
2. **Healthcare:** Insurance companies often need to obtain information about their insured patients from other parties, such as other insurance carriers or hospitals. The former cannot disclose the identity of inquired patients, whereas, the latter cannot provide any information on other patients.
3. **Law Enforcement:** Investigative agencies (e.g., the FBI) need to obtain information on suspects from other agencies, e.g., local police departments, the military, DMV, IRS, or employers. In many cases, it is dangerous (or simply forbidden) for the FBI to disclose subjects of investigation. For their part, other parties cannot

M. Abe (Ed.): ASIACRYPT 2010, LNCS 6477, pp. 213–231, 2010.

disclose their entire data-sets and need the FBI to access only desired information. Also, the FBI requests might need to be pre-*authorized* by some appropriate trusted authority (e.g., a federal judge, via a warrant). This way, the FBI can only obtain information related to legitimate requests.

1.1 Adversaries in PSI

Over the last years, PSI-related research has yielded several PSI constructs, with a wide range of adversarial models, security assumptions, and efficiency characteristics. One major distinguishing factor is the adversarial model which is typically either semi-honest or malicious. (Note that, in the rest of this paper, the term *adversary* refers to insiders, i.e., protocol participants. Outside adversaries are not considered, since their actions can be mitigated via standard network security techniques.)

Following Goldreich's definition [Gol04], protocols secure in the presence of **semi-honest adversaries** (or honest-but-curious) assume that parties faithfully follow all protocol specifications and do not misrepresent any information related to their inputs, e.g., set size and content. However, during or after protocol execution, any party might (passively) attempt to infer additional information about the other party's input. This model is formalized by requiring that each party does not learn more information that it would in an ideal implementation relying on a trusted third party (TTP).

In contrast, security in the presence of **malicious parties** allows *arbitrary* deviations from the protocol. In general, however, it does not prevent parties from refusing to participate in the protocol, modifying their private input sets, or prematurely aborting the protocol. Security in the malicious model is achieved if the adversary (interacting in the real protocol, without the TTP) can learn no more information than it could in the ideal scenario. In other words, a secure PSI emulates (in its real execution) the ideal execution that includes a trusted third party. This notion is formulated by requiring the existence of adversaries in the ideal execution model that can *simulate* adversarial behavior in the real execution model.

1.2 Authorized (Client) Input

Malicious parties cannot be prevented from modifying their input sets, even if a protocol is proven secure in the malicious model. Considering that the client learns the intersection while the server learns nothing, this appears a severe threat to server's privacy. For instance, suppose that a malicious client faithfully follows the protocol, but populates its input set with its best guesses of the server set (especially, if the set is easy to exhaustively enumerate). This would maximize the amount of information it learns. In the extreme case, the client could even claim that its set contain all possible elements. Although the server could impose a limit on this size, the client could still vary its set over multiple protocol runs.

We claim that this issue cannot be effectively addressed without some mechanism to *authorize* client inputs. Consequently, a trusted certification authority (CA) is needed to certify input sets, as proposed in [DJKT09, CZ09]. This variant is called "Authorized Private Set Intersection" (APSI) in [DT10]. Note that the CA is an off-line entity; it is neither trusted, nor involved in, computing the intersection.

As discussed above, input authorization ensures that malicious clients cannot manipulate their inputs to harm server privacy. However, this does not help at all as far as manipulation of server inputs. One way towards security against malicious servers would be to introduce authorization for server input, along the same lines as client input authorization. Although this would likely yield protocols secure in the malicious model, we choose not to pursue this direction. The main reason is that, it is more natural for the client (who learns the intersection) to be authorized on its input, than for the server (who learns nothing). However, though it is outside the scope of this paper, we believe that enforcing both server and client input authorization is a subject worth investigating. Finally, we leave as an open question whether we can *reduce* security of PSI in the malicious model to authorization of both client and server inputs.

1.3 Technical Roadmap and Contributions

Over the last few years, several elegant (if not always efficient) PSI and APSI protocols have been proposed, that are secure in the malicious model, under standard assumptions [KS05, HL08, DSMRY09, CZ09, CKRS09, HN10]. Only [JL09] presents a linear-complexity PSI protocol secure in the malicious setting. Its proof requires that the domain of inputs to be restricted to polynomial in the security parameter and requires a Common Reference String model (CRS), where the reference string, including a safe RSA modulus, must be generated by a mutually trusted third party. Other results (such as [DT10]) construct linear-complexity PSI and APSI protocols secure in the semi-honest model, under assumptions of the *one-more-XXX* type [BNPS03], with much lower computational and communication complexity. (Note that we overview prior work in Section 2). As shown in [DT10], via both analysis and experiments, there is an appreciable efficiency gap between the two "families" of PSI/APSI protocols: those secure in the malicious and in the semi-honest models. In this paper, our main goal is to construct efficient PSI and APSI protocols secure under standard assumptions, with malicious participants (both server and client).

Our starting point are the linear-complexity protocols from [DT10] (specifically, Figure 2 and 3), which are secure only in the semi-honest model. First, we modify the APSI construct of [DT10] and obtain APSI protocol secure in the malicious model, under the standard RSA assumption (in ROM). Then, we modify its PSI counterpart: while the linear-complexity PSI protocol in [DT10] is secure under the One-More-Gap-DH assumption [BNPS03] against semi-honest parties, our modified variant is secure in the malicious model under the standard DDH assumption (again, in ROM). We present formal proofs for all proposed protocols.

Contributions of our work are:

1. To the best of our knowledge, our APSI protocol is the *first* result with *linear* communication and computational complexity, in the malicious model. (Previous work achieved quadratic computational complexity.)
2. Our PSI protocol also offers linear complexity. Although some prior work (i.e., [JL09]) also achieves the same asymptotic bound, we do not require the CRS model and our proof does not restrict input domain size. We also show that our protocol incurs *significantly reduced* constant factors.

3. We prove security of proposed protocols, in presence of malicious adversaries, under standard cryptographic (RSA and DDH) assumptions, in ROM.

Organization. Section 2 overviews previous work. Then, after some preliminaries in Section 3, we present our constructions in Sections 4 and 5. Next, Section 6 discusses the efficiency of our constructs and Section 7 concludes the paper.

2 Related Work

This section overviews prior work on PSI and APSI.

2.1 Prior Work on PSI

It is well known that PSI could be realized via general secure two-party computation [Yao82]. However, it is usually far more efficient to have dedicated protocols (see [FNP04, KS05]); which is the direction we pursue in this paper. From here on, we consider PSI as an interaction between a server S and a client C. The server set contains w items, while the client set – v.

Freedman, et al. [FNP04] introduce the concept of PSI and presented protocols based on *Oblivious Polynomial Evaluation* (OPE) [NP06]. The basic idea is to represent a set as a polynomial, with individual elements as its roots. The construction for the semi-honest setting incurs linear communication, and quadratic computational, complexity. Using Horner's rule and balanced bucket allocation, the number of modular exponentiations can be reduced to $O(w \log\log v)$ exponentiations for the server and $O(w + v)$ exponentiations for the client. [FNP04] also gives constructions for a malicious client and semi-honest server. This protocol uses a cut-and-choose strategy, thus, the overhead is increased by a statistical security parameter. Also presented is a protocol secure in the presence of a malicious server and a semi-honest client in ROM.

Kissner and Song [KS05] propose OPE-based protocols for *mutual* PSI (as well as for additional set operations), and may involve more than two players. Protocols are secure in the standard model against semi-honest and also malicious adversaries. The former incurs quadratic $(O(wv))$ computation (but linear communication) overhead. The latter uses (expensive) generic zero-knowledge proofs to prevent parties from deviating to the protocol. Later, Dachman-Soled, et al. [DSMRY09] present an improved PSI construction, based on [KS05]. Their construction incorporates a secret sharing of polynomial inputs. Since Shamir's secret sharing [Sha84] implies Reed Solomon codes, they do not need generic zero-knowledge proofs. Complexity of their protocol amounts to $O(wk^2 \log^2(v))$ in communication and $O(wvk \log(v) + wk^2 \log^2(v))$ in computation, being k the security parameter.

Another family of protocols rely on so-called *Oblivious Pseudo-Random Functions* (OPRF-s). An OPRF is a two-party protocol (between a sender and a receiver) that securely computes a pseudorandom function $f_k(\cdot)$ on key k contributed by the sender and input x contributed by the receiver, such that the former learns nothing from the interaction, and the latter learns only the value $f_k(x)$. OPRF-based PSI-s work as follows: Server S holds a secret random key k. Then, for each $s_j \in \mathcal{S}$, S computes $u_j = f_k(s_j)$, and publishes (or sends the client) the set $\mathcal{U} = \{u_1, \cdots, u_w\}$. Then, C and S engage in

an OPRF computation of $f_k(c_i)$ for each $c_i \in \mathcal{C}$ (of size v), such that S learns nothing about $\mathcal{C}$ (except the size) and C learns $f_k(c_i)$. Finally, C obtains $c_i \in \mathcal{C} \cap \mathcal{S}$ if and only if $f_k(c_i) \in \mathcal{U}$. The idea of using OPRFs for PSI protocols is due to Hazay and Lindell [HL08], who propose one solution with security against malicious adversaries with one-sided simulatability, and one – against covert adversaries [AL07].

This protocol has been later improved by Jarecki and Liu [JL09], who proposed a protocol secure in the standard model in the presence of both malicious parties, based on the Decisional q-Diffie-Hellman Inversion assumption, in the Common Reference String (CRS) model, where a safe RSA modulus is generated by a trusted third party. Encryption operations are performed using an additively homomorphic encryption scheme, such as Camenisch and Shoup [CS03]. As pointed out in [JL09], this approach can be further optimized, based on the concurrent work in [BCC+09]. Assuming such improved construction, [JL09] incurs the following computational complexity: Let m be the number of bits needed to represent each set item; the server performs at least $O(w)$ PRF evaluations, i.e., both m-bit and group exponentiations, plus $O(v)$ group exponentiations, whereas, the client at least $O(v)$ m-bit exponentiations plus $O(v)$ group exponentiations. We discuss in details the complexity of this solution later in the paper. Finally, note that the proof in [JL09] requires the ability to exhaustively search over the input domain, i.e., the input domain size of the PRF should be polynomial in the security parameter.

A recent result by Hazay and Nissim [HN10] presents an improved construction of OPE-based PSI based on [FNP04], but without ROM. Specifically, it introduces zero-knowledge proofs that allow client to demonstrate that encrypted polynomials are correctly produced. Also, it uses a technique based on a perfectly hiding commitment scheme with an OPRF evaluation protocol to prevent the server from deviating from the protocol. The PSI protocol in [HN10] incurs $O(v + w(\log\log v + m))$ computational and $O(v + w \cdot m)$ communication complexity, where m is the number of bits needed to represent a set item. Note that execution of the underlying OPRF in [HN10] requires m oblivious transfer invocations, and hence $O(m)$ modular exponentiations, for each set item. However, such overhead can be avoided by instantiating the protocol in ROM. This protocol can be also optimized if the size of the intersection is allowed to be leaked to the server, in contrast to our strict privacy definitions (see Section 3.3). Nonetheless, the resulting protocol is of sending $O(v + |\mathcal{S} \cap \mathcal{C}| \cdot m)$ and computing $O(v + w \cdot \log\log v + |\mathcal{S} \cap \mathcal{C}| \cdot m)$, which is still not linear. (Also recall that it is not clear how to enable convert the PSI construct of [HN10] into APSI.)

In another recent result, [DT10] (Fig.4) presents an adaptive PSI protocol based on blind-RSA signatures [Cha83], secure in the semi-honest model, under the One-More-RSA assumption [BNPS03], in ROM. Specifically, during an initialization phase, the server generates RSA keys (N, e, d) and commits to its set, by publishing the hash of the RSA signature of each item. During the interaction, the client obtains blind-RSA signatures of its items from the server. Thus, the server needs to compute $O(w)$ RSA signatures during the initialization phase, and $O(v)$ online. Whereas, the client (assuming $e = 3$) only computes $O(v)$ multiplications, thus making this construct particularly appealing for clients running on limited-resource devices.

[DT10] (Fig.3) includes another PSI secure in the presence of semi-honest adversaries, under the One-More-Gap-DH assumption, in ROM. Common inputs are primes p, q (with $q|p-1$, the order of a subgroup of $\mathbb{Z}_p^*$) and a generator of the subgroup, g. First, the client computes the accumulator $PCH = \prod_{i=1}^{v}(H(c_i))$ and sends $X = PCH \cdot g^{R_c}$ for R_c random in $\mathbb{Z}_q^*$. Also, for $i = 1, \ldots, v$, it computes $PCH_i = PCH/H(c_i)$ and sends $x_i = PCH_i \cdot g^{R_{c:i}}$ for $R_{c:i}$ in $\mathbb{Z}_q^*$. The server picks a random R_s in $\mathbb{Z}_q^*$, sends $Z = g^{R_s}$, and, for each x_i, sends back $x_i' = {x_i}^{R_s}$. Then, for $j = 1, \ldots, w$, it computes $T_{s:j} = H'((X/H(s_j))^{R_s})$. Finally, the client computes $T_{c:i} = x_i' \cdot Z^{R_c} \cdot Z^{-R_{c:i}}$, and learns that $c_i \in \mathcal{C} \cap \mathcal{S}$ if $T_{c:i} = T_{s:j}$. Computational complexity of this protocol is $O(w+v)$ and $O(v)$ exponentiations (with short exponents) for the server and client, respectively.

2.2 Prior Work on APSI

Authorized Private Set Intersection (APSI) is defined in [DT10] to extend PSI to support *authorization* of client inputs. Each client input must be authorized (via signing) by some trusted authority, e.g., a CA. Recall the third example in Section 1: to obtain information on a suspect from her employer, the FBI needs to be duly authorized. APSI represents an authorization using a digital signature. Note that authorizations obtained from the CA are private to the client and cannot be disclosed to the server.

[DT10] shows that the PSI protocol in its Fig.3 (reviewed at the end of Section 2.1) can be instantiated in a RSA setting, where client input is a set of RSA signatures and the server obliviously verifies them by modifying the protocol as follows. The client C needs to obtain from the CA signatures $\sigma_i = H(c_i)^d$ (for input set $\mathcal{C} = \{c_1, \ldots, c_v\}$). C computes the accumulator $PCH^* = \prod_{i=1}^{v} \sigma_i$ and sends $X = PCH^* \cdot g^{R_c}$ for random R_c. Also, it computes $PCH_i^* = PCH^*/\sigma_i$ and sends $x_i = PCH_i^* \cdot g^{R_{c:i}}$ for random $R_{c:i}$. The server picks a random R_s, sends $Z = g^{eR_s}$, and, for each x_i, sends back $x_i' = {x_i}^{eR_s}$. Then, for $j = 1, \ldots, w$, it computes $T_{s:j} = H'((X^e/H(s_j))^{R_s})$. Finally, the client computes $T_{c:i} = x_i' \cdot Z^{R_c} \cdot Z^{-R_{c:i}}$, and learns that $c_i \in \mathcal{C} \cap \mathcal{S}$ if $T_{c:i} = T_{s:j}$. Asymptotic complexity of this solution is the same as that of the standard PSI presented above, i.e., $O(w+v)$ and $O(v)$ exponentiations for the server and client, respectively. (Although short exponents are replaced with "RSA" exponents.) The resulting protocol is secure in the semi-honest model, under the standard RSA assumption, in ROM. Note that the use of "authorized" client inputs seems to increase server privacy: under the RSA assumption, the client does not learn any information about server inputs, unless it holds a valid RSA signature. In other words, there appears to be a strong correlation between server privacy and client's difficulty of forging signatures.

A similar concept (adaptable to APSI) is Public-Key Encryption with Oblivious Keyword Search in [CKRS09]. It proposes an Identity-based cryptosystem (inspired by PEKS in [BDOP04]), where the client obtains authorized search trapdoors from a CA, and uses them to search over data encrypted by the server. The client learns only the information matching the authorized trapdoors, whereas, the server learns nothing. The protocol is secure in the presence of malicious adversaries in the standard model, under the Decision Bilinear Diffie-Hellman assumption [BF03]. It uses a modification of the Boyen-Waters IBE [BW06]. Even without taking into account zero-knowledge proofs,

the server would compute $O(w)$ encryptions of [BW06] (each requiring 6 exponentiations and a representation of 6 group elements). The client would need to test each of the $O(w)$ PEKS against its $O(v)$ trapdoors, hence performing $O(w \cdot v)$ decryptions.

Finally, [CZ09] introduces another similar notion – Private Intersection of Certified Sets. This construct allows a trusted third party to ensure that all protocol inputs are valid and bound to each protocol participant. The proposed protocol is *mutual* (i.e., both parties receive the intersection) and builds upon oblivious polynomial evaluation and achieves *quadratic* computation and communication overhead.

3 Preliminaries

In this section, we present our cryptographic assumptions and tools, as well as security model. We introduce our notation in Table 1.

Table 1. Notation

$a \leftarrow_r \mathcal{A}$	variable a is chosen uniformly at random from set $\mathcal{A}$
κ	security parameter
$N = pq$	safe RSA modulus with at least κ-bit security
e, d	public and private exponents of RSA
$Z_{N/2}$	1/2 of bit-size of N
$H()$	random oracle $H : \{0,1\}^* \rightarrow \{0,1\}^\kappa$
$H_1()$	random oracle $H_1 : \{0,1\}^* \rightarrow \mathbb{G}$ ($\mathbb{G}$ depends on the context)
$H_2()$	random oracle $H_2 : \mathbb{G} \times \mathbb{G} \times \{0,1\}^* \rightarrow \{0,1\}^\kappa$ ($\mathbb{G}$ depends on the context)
$\mathcal{C}, \mathcal{S}$	client and server sets, respectively
v, w	sizes of $\mathcal{C}$ and $\mathcal{S}$, respectively
$i \in [1, v]$	indices of elements of $\mathcal{C}$
$j \in [1, w]$	indices of elements of $\mathcal{S}$
c_i, s_j	i-th and j-th elements of $\mathcal{C}$ and $\mathcal{S}$, respectively
hc_i, hs_j	$H_1(c_i)$ and $H_1(s_j)$, respectively
σ_i	$H_1(c_i)^d$, RSA-signature on client item

3.1 Cryptographic Assumptions

Definition 1. *Let $\mathbb{G}$ be a cyclic group and let g be its generator. Assume that the bit-length of the group size is l. The DDH problem is hard in $\mathbb{G}$ if for every efficient algorithm A the probability:*

$$\left|\Pr[x, y \leftarrow_r \{0,1\}^l : A(g, g^x, g^y, g^{xy}) = 1] - \Pr[x, y, z \leftarrow_r \{0,1\}^l : A(g, g^x, g^y, g^z) = 1]\right|$$

is a negligible function of κ.

Definition 2. *Let $\mathsf{RSA\text{-}Gen}(1^\kappa)$ be an algorithm that outputs so-called "safe RSA instances", i.e. pairs (n, e) where $n = pq$, e is a small prime such that $gcd(e, \phi(n)) = 1$, and p, q are random κ-bit primes subject to the constraint that $p = 2p' + 1$, $q = 2q' + 1$ for prime p', q', $p' \neq q'$. The RSA problem is hard if, for every efficient algorithm A, the probability:*

$$Pr[(n, e) \leftarrow \mathsf{RSA\text{-}Gen}(1^\kappa), z \leftarrow \mathbb{Z}_n^* : A(n, e, z) = y \text{ s.t. } y^e = z \pmod n]$$

is a negligible function of κ.

3.2 Tools

In this section, we consider signature of knowledge of a discrete logarithm and equality of two discrete logarithms in a cyclic group $\mathbb{G} = \langle g \rangle$. In particular, we consider $\mathbb{G}$ where either the order of $\mathbb{G}$ is known or the order of $\mathbb{G}$ is unknown but its bit-length l is publicly known. Fujisaki and Okamoto [FO97] show that (under the strong RSA assumption) standard proofs of knowledge that work in a group of known order are also proofs of knowledge in this setting. We define discrete logarithm of $y \in \mathbb{G}$ with respect to base g as any integer $x \in \mathbb{Z}$ such that $y = g^x$ in $\mathbb{G}$. We assume a security parameter $\epsilon > 1$.

Definition 3. *(ZK of DL over a known order group) Let $y, g \in \mathbb{G}$ of order q. A pair $(c, s) \in \{0,1\}^\kappa \times \mathbb{Z}_q$ verifying $c = H(y||g||g^s y^c||m)$ is a signature of knowledge of the discrete logarithm of $y = g^x$ w.r.t. base g, on message $m \in \{0,1\}^*$.*

Definition 4. *(ZK of DL over an unknown order group) Let $y, g \in \mathbb{G}$ where the group order is unknown, but its bit-length is known as l bits. A pair $(c, s) \in \{0,1\}^\kappa \times \pm\{0,1\}^{\epsilon(l+\kappa)+1}$ verifying $c = H(y||g||g^s y^c||m)$ is a signature of knowledge of the discrete logarithm of $y = g^x$ w.r.t. base g, on message $m \in \{0,1\}^*$.*

The player in possession of the secret $x = \log_g y$ can compute the signature by choosing a random $t \in \mathbb{Z}_q$ (or $\pm\{0,1\}^{\epsilon(l+\kappa)}$) and then computing c and s as: $c = H(y||g||g^t||m)$ and $s = t - cx$ in $\mathbb{Z}_q$ (or in $\mathbb{Z}$).

Definition 5. *(ZK of EDL over a known order group) Let $y_1, y_2, g, h \in \mathbb{G}$ of order q. A pair $(c, s) \in \{0,1\}^\kappa \times \mathbb{Z}_q$ verifying $c = H(y_1||y_2||g||h||g^s y_1^c||h^s y_2^c||m)$ is a signature of knowledge of the discrete logarithm of both $y_1 = g^x$ w.r.t. base g and $y_2 = h^x$ w.r.t. base h, on message $m \in \{0,1\}^*$.*

Definition 6. *(ZK of EDL over an unknown order group) Let $y_1, y_2, g, h \in \mathbb{G}$ where the group order is unknown, but its bit-length is known as l bits. A pair $(c, s) \in \{0,1\}^\kappa \times \pm\{0,1\}^{\epsilon(l+\kappa)+1}$ verifying $c = H(y_1||y_2||g||h||g^s y_1^c||h^s y_2^c||m)$ is a signature of knowledge of the discrete logarithm of both $y_1 = g^x$ w.r.t. base g and $y_2 = h^x$ w.r.t. base h, on message $m \in \{0,1\}^*$.*

The player in possession of the secret $x = \log_g y_1 = \log_h y_2$ can compute the signature by choosing a random $t \in \mathbb{Z}_q$ (or $\pm\{0,1\}^{\epsilon(l+\kappa)}$) and then computing c and s as: $c = H(y_1||y_2||g||h||g^t||h^t||m)$ and $s = t - cx$ in $\mathbb{Z}_q$ (or in $\mathbb{Z}$).

3.3 Security Model

We assume a malicious adversary that behaves arbitrarily. Informally, a protocol is secure in this model if no adversary interacting in the real protocol (where no TTP exists) can learn any more from a real execution than from an execution that takes place in the ideal world. In other words, for any adversary that successfully attacks a real protocol, there exists a simulator that successfully attacks the same protocol in the ideal world.

We now define ideal functionalities of PSI and APSI. In particular, in contrast to PSI, APSI employs an (off-line) CA with algorithms (KGen, Sign, Ver). The CA generates a key-pair $(sk, pk) \leftarrow \mathsf{KGen}$, publishes its public key pk, and, on client input c_i, it issues a signature $\sigma_i = \mathsf{Sign}(sk, c_i)$ s.t. $\mathsf{Ver}(pk, \sigma_i, c_i) = 1$.

Definition 7. *The ideal functionality $\mathcal{F}_{APSI}$ of an APSI protocol betw. server S on input $\mathcal{S} = \{s_1, \cdots, s_w\}$ and client C on input $\mathcal{C} = \{(c_1, \sigma_1), \cdots, (c_v, \sigma_v)\}$ is defined as:*

$$\mathcal{F}_{APSI} : (\mathcal{S}, \mathcal{C}) \rightarrow (\perp, \mathcal{S} \cap \{c_i \mid c_i \in \mathcal{C} \ \wedge \ \mathsf{Ver}(pk, \sigma_i, c_i) = 1\})$$

where w, v are the public input to $\mathcal{F}_{APSI}$.

Definition 8. *The ideal functionality $\mathcal{F}_{PSI}$ of a PSI between server S on input $\mathcal{S} = \{s_1, \cdots, s_w\}$ and client C on input $\mathcal{C} = \{c_1, \cdots, c_v\}$ is defined as follows:*

$$\mathcal{F}_{PSI} : (\mathcal{S}, \mathcal{C}) \rightarrow (\perp, \mathcal{S} \cap \mathcal{C})$$

where w, v are the public input to $\mathcal{F}_{PSI}$.

4 APSI Protocol

We now present our protocol for secure computation of authorized set intersection. We start from the APSI protocol of [DT10] (reviewed in Section 2.2), secure in the semi-honest model. We describe a modified version that securely implements the $\mathcal{F}_{\mathsf{APSI}}$ functionality in the *malicious* model, in ROM, under the RSA and DDH assumptions.

The CA (trusted third party that authorizes client input) is realized with the following algorithms:

- KGen: On input of security parameter κ, this algorithm generates safe RSA modulus $N = pq$ where $p = 2p' + 1$, $q = 2q' + 1$ and picks a random element g, g' s.t. $\langle -1 \rangle \times \langle g \rangle \equiv \langle -1 \rangle \times \langle g' \rangle \equiv \mathbb{Z}_N^*$. RSA exponents (e, d) are chosen in the standard way: e is a small prime and $d = e^{-1} \bmod \phi(N)$. The algorithm also fixes hash function $H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_N^*$ and $H_2 : \mathbb{Z}_N^* \times \mathbb{Z}_N^* \times \{0,1\}^* \rightarrow \{0,1\}^\kappa$. The secret key is (p, q, d) and the public parameters are: $N, e, g, g', H_1(), H_2()$.
- Sign: On input of c_i, this algorithm issues an authorization $\sigma_i = H_1(c_i)^d \bmod N$.
- Ver: On input of (σ_i, c_i), this algorithm verifies whether ${\sigma_i}^e = H_1(c_i) \bmod N$.

The resulting protocol is presented in **Figure 1**.

Theorem 1. *If RSA and DDH problems are hard, and π, π' are zero-knowledge proofs, then the protocol in Figure 1 is a secure computation of $\mathcal{F}_{APSI}$, in ROM.*

Proof. [*Construction of an ideal world SIM_s from a malicious real-world server S^**]

The simulator SIM_s is built as follows:

- **Setup:** SIM_s executes KGen and publishes public parameters N, e, g, g'.
- **Hash queries to H_1 and H_2:** SIM_s constructs two tables $T_1 = (q, h_q)$ and $T_2 = ((k, h'_q, q'), t)$ to answer, respectively, the H_1 and H_2 queries. Specifically:
  - On query q to H_1, SIM_s checks if $\exists (q, h_q) \in T_1$: If so, it returns h_q, otherwise it responds $h_q \leftarrow_r \mathbb{Z}_N^*$, and stores (q, h_q) to T_1.
  - On query (k, h'_q, q') to H_2, SIM_s checks if $\exists ((k, h'_q, q'), t) \in T_2$: If so, it returns t, otherwise it responds $t \leftarrow_r \{0,1\}^\kappa$ to H_2, and stores $((k, h'_q, q'), t)$ to T_2.

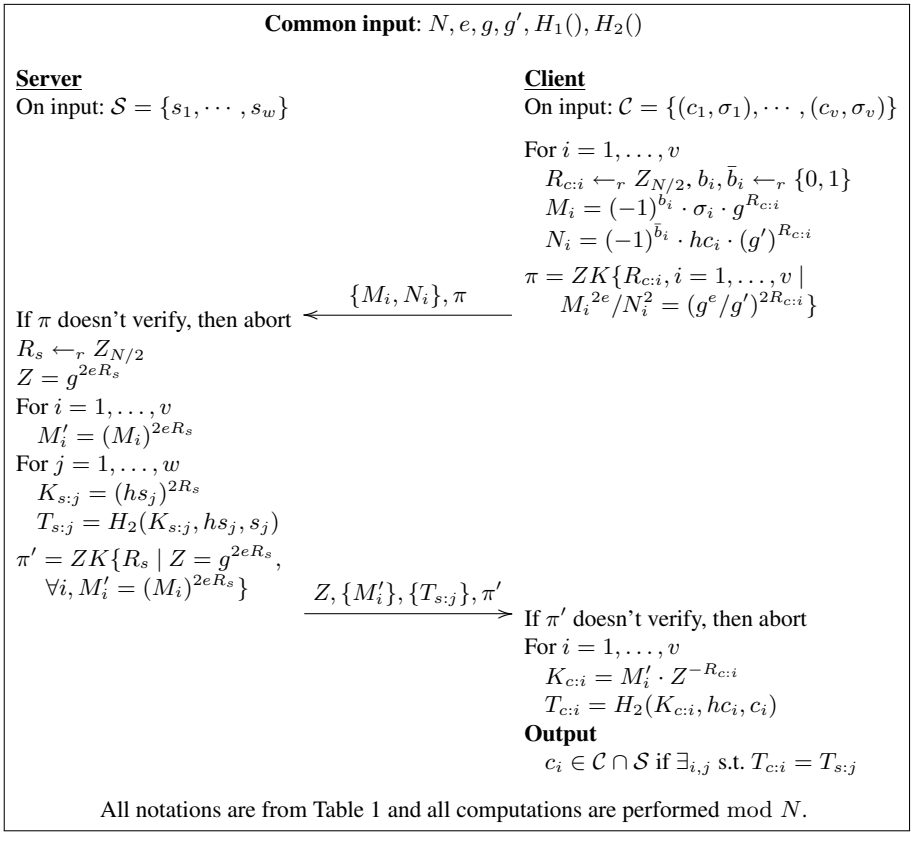


Fig. 1. Our APSI Protocol with linear complexity secure against malicious adversaries

– **Simulation of the real-world client C and the ideal-world server $\overline{S}$:**
  1. SIM_s picks $M'_i \leftarrow_r \mathbb{Z}_N^*$, $N'_i \leftarrow_r \mathbb{Z}_N^*$ and computes $M_i = (M'_i)^2$, $N_i = (N'_i)^2$ for each $i = 1, \cdots, v$.
  2. SIM_s sends $\{M_i, N_i\}_{i=1,\ldots,v}$ and simulates the proof π.
  3. After getting $(Z, \{M'_i\}_{i=1,\ldots,v}, \{T_{s:j}\}_{j=1,\ldots,w})$, and interacting with S^* as verifier in the proof π', if the proof π' verifies, SIM_s runs the extractor algorithm for R_s. Otherwise, it aborts.
     (a) For each $T_{s:j}$, SIM_s checks if $\exists(q, h_q) \in T_1$ and $\exists((k, h'_q, q'), t) \in T_2$, s.t. $q = q'$, $h_q = h'_q, k = (h_q)^{2R_s}$ and $t = T_{s:j}$. If so, add q to $\mathcal{S}$; otherwise, add a dummy item into $\mathcal{S}$.
     (b) Then SIM_s plays the role of the ideal-world server, which uses $\mathcal{S}$ to respond to ideal client $\overline{C}$'s queries.

Since the distribution of $\{M_i, N_i\}_{i=1,\ldots,v}$ sent by SIM_s is identical to the distribution produced by the real client C and the π proof system is zero-knowledge, S^*'s views when interacting with the real client C and with the simulator SIM_s are indistinguishable.

[*Output of (honest) real client C interacting with S^**]

Now we consider the output of the honest real client C interacting with S^*. By soundness of proof π', message Z and M'_i sent by S^* is $Z = g^{eR_s}$ and $M'_i = (M_i)^{eRs}$ for $i = 1, \cdots, v$. Then, C's final output is a set containing all c_i's, such that $H_2(M'_i \cdot Z^{-R_{c:i}}, hc_i, c_i) \in \{T_{s:j}\}$. In other words, for each c_i, C outputs c_i if $\exists\ j$ s.t. $H_2(M'_i \cdot Z^{-R_{c:i}}, hc_i, c_i) = T_{s:j}$. Since H_2 is a random oracle, there are two possibilities:

1. S^* computes $T_{s:j}$ from $H_2((hs_j)^{2Rs}, hs_j, s_j)$ for $s_j = c_i$. Since SIM_s described above extracts $s_j = c_i$ and adds s_j in $\mathcal{S}$, the ideal world $\overline{C}$ also output c_i on its input c_i.
2. S^* did not query H_2 on $(M'_i \cdot Z^{-R_{c:i}}, hc_i, c_i)$ but $H_2(M'_i \cdot Z^{-R_{c:i}}, hc_i, c_i)$ happens to be equal to $T_{s:j}$. This event occurs with negligible probability bounded by $v \cdot\ w \cdot\ 2^{-\kappa}$.

Therefore, with probability $1 - v \cdot\ w \cdot\ 2^{-\kappa}$, the real-world client C interacting with S^* and the ideal-world client $\overline{C}$ interacting with SIM_s yield identical outputs.

[*Construction of an ideal world SIM_c from a malicious real-world client C^**]

The simulator SIM_c is formed as follows:

– **Setup and hash queries to H_1 and H_2:** Same as Setup and H_1 and H_2 responses described above in construction of SIM_s.
– **Authorization queries:** On input m, SIM_c responds with (m, σ) where $\sigma = (H_1(m))^d$ and stores (m, σ) to table T_3.
– **Simulation of real-world server S and ideal-world client $\overline{C}$:**

1. After getting $\{M_i, N_i\}_{i=1,\ldots,v}$, and interacting with C^* as verifier in the proof π, SIM_c checks if proof π verifies. If not, it aborts. Otherwise, it runs the extractor algorithm for $\{R_{c:i}\}$ and computes $\pm(hc_i, \sigma_i)$ s.t. $hc_i = \sigma^e$.
2. For each $\pm(hc_i, \sigma_i)$:
   - If $\nexists (q, h_q) \in T_1$ s.t. $h_q = \pm hc_i$ then add a dummy item (δ, σ_δ) to $\mathcal{C}$ where δ and σ_δ are randomly selected from the respective domain.
   - If $\exists (q, h_q) \in T_1$ s.t. $h_q = \pm hc_i$, but $\nexists (m, \sigma) \in T_3$ s.t. $\sigma = \pm\sigma_i$ then output fail_1 and abort.
   - If $\exists (q, h_q) \in T_1$ s.t. $h_q = \pm hc_i$ and $\exists (m, \sigma) \in T_3$ s.t. $\sigma = \pm\sigma_i$, then add $(q, \pm\sigma)$ to the set $\mathcal{C}$.
3. SIM_c plays the role of the client in the ideal-world. On input $\mathcal{C} = \{(c_1, \sigma_1), \cdots, (c_v, \sigma_v)\}$, SIM_c interacts with the ideal-world server $\bar{S}$ through the TTP.
4. On getting intersection $L = \{c'_1, \cdots, c'_{|L|}\}$, with $|L| \leq v$ from the ideal-world interaction, SIM_c forms $\mathcal{S} = \Pi\left(c'_1, \cdots, c'_{|L|}, \delta'_1, \cdots, \delta'_{w-|L|+1}\right)$, where δ''s are dummy items and Π is a permutation function.
5. SIM_c picks $R_s \leftarrow_r \mathbb{Z}_{N/2}$, and computes $Z = g^{2eR_s}$ and $M'_i = (M_i)^{2eR_s}$ for $i = 1, ..., v$.
6. For each $s_j \in \mathcal{S}$:
   - If $s_j \in L$, compute $T_{s:j} = H_2((hs_j)^{2R_s}, hs_j, s_j)$.
   - If $s_j \notin L$, compute $T_{s:j} \leftarrow_r \{0,1\}^\kappa$.
7. SIM_c returns $Z, \{M'_i\}_{i=1,\ldots,v}, \{T_{s:j}\}_{j=1,\ldots,w}$ to C^* and simulates the proof π'.

Claim 1. If event fail_1 occurs with non-negligible probability, then C^* can be used to break the RSA assumption.

We describe the reduction algorithm using a modified simulator algorithm called $\mathcal{C}h_1$ that takes an RSA challenge (N', e', z) as an input and tries to output $z^{(e')^{-1}}$. $\mathcal{C}h_1$ follows the SIM_c as described above, except:

- **Setup:** On input (N', e', z), $\mathcal{C}h_1$ sets $N = N'$, $e = e'$ and picks generator g, $g' \leftarrow_r \mathbb{Z}_N^*$. (Note that random g in $\mathbb{Z}_N^*$ matches that chosen by a real key generation with probability about 1/2.)
- **Authorization queries:** On input m, $\mathcal{C}h_1$ responds with (m, σ) with $\sigma \leftarrow_r \mathbb{Z}_N^*$, assign $H_1(m) = \sigma^e$, and records (m, σ) to T_3.
- **Hash queries to H_1:** On query H_1 on q, if $\nexists(q, h_q) \in T_1$ then $\mathcal{C}h_1$ responds $h_q = z(r_q)^e$ where $r_q \leftarrow_r \mathbb{Z}_N$, and stores (q, r_q, h_q) to T_1. (Since r_q is uniformly distributed in $\mathbb{Z}_N$, the distribution of h_q is also uniformly distributed in $\mathbb{Z}_N$.)

Assume that fail_1 occurs on (hc_i, σ_i). Then, $\mathcal{C}h_1$ extracts entry $(q, r_q, h_q) \in T_1$ s.t. $h_q = hc_i$ and outputs σ_i / r_q, which breaks the RSA assumption.

Now unless the fail_1 event occurs, the views interacting with the SIM_c and with the real protocol are different only in the computation of $T_{s:j}$ for $s_j \in \mathcal{S}$ but $s_j \notin L$. Let fail_2 be the event that C^* queries H_2 on $((hs_j)^{2R_s}, hs_j, s_j)$ for $s_j \in \mathcal{S}$ and $s_j \notin L$.

Claim 2. If event fail_2 occurs with non-negligible probability, then C^* can be used to break the DDH assumption.

We describe reduction algorithm $\mathcal{C}h_2$ that takes a DDH challenge $(N', f, \alpha = f^a \pmod{N'}, \beta = f^b \pmod{N'}, \gamma)$ as input and outputs the DDH answer using C^*. $\mathcal{C}h_2$ follows the SIM_c algorithm as we describe above, except that:

- **Setup:** On input $(N', f, \alpha, \beta, \gamma)$, $\mathcal{C}h_2$ sets $N = N'$, $g = f$ and picks generator $g' \leftarrow_r \mathbb{Z}_N^*$ and odd $e \leftarrow_r \mathbb{Z}_N$.
- **Authorization queries:** Same as in $\mathcal{C}h_1$ simulation.
- **Hash queries to H_1:** On query q to H_1, if $\nexists(q, h_q) \in T_1$ then $\mathcal{C}h_2$ responds with $h_q = \beta g^{r_q}$ where $r_q \leftarrow_r \mathbb{Z}_{N/2}$, and records (q, r_q, h_q) to T_1. (Since r_q is random $\mathbb{Z}_N/2$, the distribution of h_q is computationally indistinguishable from the uniform distribution of $\mathbb{Z}_N^*$.)
- **In computation for $Z, \{M_i\}, \{T_{s:j}\}$:**
  - $\mathcal{C}h_2$ sets $Z = A^{2e}$ and computes $M_i' = \gamma^2(\alpha)^{2r_q + 2eR_{c:i}}$ for $i = 1, \ldots, v$ (instead of picking R_s and computing $Z = g^{2eR_s}$ and $M_i' = (M_i)^{2eR_s}$).
  - For each $s_j \in \mathcal{S}$, if $s_j \in L$, $\mathcal{C}h_2$ computes $T_{s:j} = H_2(\gamma^2(\alpha)^{2r_q}, hs_j, s_j)$.

Given $\alpha = g^a (= g^{R_s})$ and $\beta = g^b$, we replace g^{ab} by γ in the above simulation of M_i and $T_{s:j}$. Thus, C^*'s views when interacting with the real server S and with the simulator $\mathcal{C}h_2$ are indistinguishable under that DDH assumption. Assume that fail_2 occurs, i.e., C^* makes a query to H_2 on $((hs_j)^{2R_s}, hs_j, s_j)$ for $s_j \in \mathcal{S}$ but $s_j \notin L$. $\mathcal{C}h_2$ checks if $\exists(q, r_q, h_q) \in T_1$ and $\exists((k, h_q', q'), t) \in T_2$ s.t. $q = q'$, $h_q = h_q'$, $k = \gamma^2(\alpha)^{2r_q}$ for each $q \in \mathcal{S}$ but $q \notin L$. If so, $\mathcal{C}h_2$ outputs True. Otherwise, $\mathcal{C}h_2$ outputs False. Thus, the DDH assumption is broken.

Therefore, since fail_1 and fail_2 events occur with negligible probability, C^*'s view in the protocol with the real-world server S and in the interaction with SIM_c is negligible.

[*The output of honest real server S interacting with C^**]

Finally, the real-world S interacting with C^* in the real protocol outputs $\perp$ and the ideal-world $\bar{S}$ interacting with SIM_c gets $\perp$. This ends proof of Theorem 1.

Our APSI protocol differs from the one in [DT10] in the following:

- We modify inputs to the protocol and add efficient zero-knowledge proofs to prevent client and server from deviating from the protocol and to enable extraction of inputs.
- We multiply client inputs by -1 to a random bit to: (1) ensure that they are uniformly distributed in QR_N, and (2) simplify reduction to the RSA problem.
- We do not use "accumulated" values, such as PCH_i^*, as they are not needed either for protocol security or for input extraction during simulation.

5 PSI Protocol

This section presents our protocol for secure computation of set intersection. It is a modified version of the PSI protocol of [DT10] (reviewed in Section 2.1), secure in the semi-honest model under the One-More-Gap-DH assumption (in ROM). We amend it to obtain a protocol that securely implements $\mathcal{F}_{\mathsf{PSI}}$ in the *malicious* model under the DDH assumptions (in ROM). We assume that KGen generates p, q, g, g', g'' where p and q are primes, such that $q|p-1$ and g, g', g'' are generators of $\mathbb{Z}_q^*$.

The resulting protocol is presented in **Figure 2**.

Theorem 2. *If the DDH problem is hard and π, π' are zero-knowledge proofs, the protocol in Figure 2 is a secure computation of $\mathcal{F}_{\mathsf{PSI}}$, in ROM.*

Proof. [*Construction of an ideal world SIM_s from malicious real-world server S^**]

Simulator SIM_s is built as follows:

- **Setup:** SIM_s executes KGen and publishes public parameters p, q, g, g', g''.
- **Queries H_1 and H_2:** SIM_s creates two tables $T_1 = (q, h_q)$ and $T_2 = ((k, h'_q, q'), t)$ to answer, respectively, H_1 and H_2 queries. Specifically,
  - On query q to H_1, SIM_s checks if $\exists (q, h_q) \in T_1$: If so, it returns h_q, otherwise it responds $h_q \leftarrow_r \mathbb{Z}_p^*$, and stores (q, h_q) to T_1.
  - On query (k, h'_q, q') to H_2, SIM_s checks if $\exists ((k, h'_q, q'), t) \in T_2$: If so, it returns t, otherwise it responds $t \leftarrow_r \{0,1\}^\kappa$ to H_2, and stores $((k, h'_q, q'), t)$ to T_2.
- **Simulation of real-world client C and ideal-world server $\overline{S}$:**
  1. SIM_s picks $X \leftarrow_r \mathbb{Z}_p^*$ and $\{M_i, N_i \mid M_i \leftarrow_r \mathbb{Z}_p^*, N_i \leftarrow_r \mathbb{Z}_p^*\}$ (for $i = 1, \ldots, v$).
  2. SIM_s sends $X, \{M_i, N_i\}_{i=1,\ldots,v}$ and simulates proof π.
  3. After getting $(Z, \{M'_i\}_{i=1,\ldots,v}, \{T_{s:j}\}_{j=1,\ldots,w})$, and interacting with S^* as verifier in proof π', if π' verifies, SIM_s runs the extractor algorithm for R_s. Otherwise, it aborts.
     (a) For each $T_{s:j}$, SIM_s checks if $\exists (q, h_q) \in T_1$ and $\exists ((k, h'_q, q'), t) \in T_2$, s.t. $q = q'$, $h_q = h'_q, k = (h_q)^{R_s}$ and $t = T_{s:j}$. If so, add q to $\mathcal{S}$; otherwise, add a dummy item into $\mathcal{S}$.
     (b) Then SIM_s plays the role of the ideal-world server, which uses $\mathcal{S}$ to respond to ideal client $\overline{C}$'s queries.

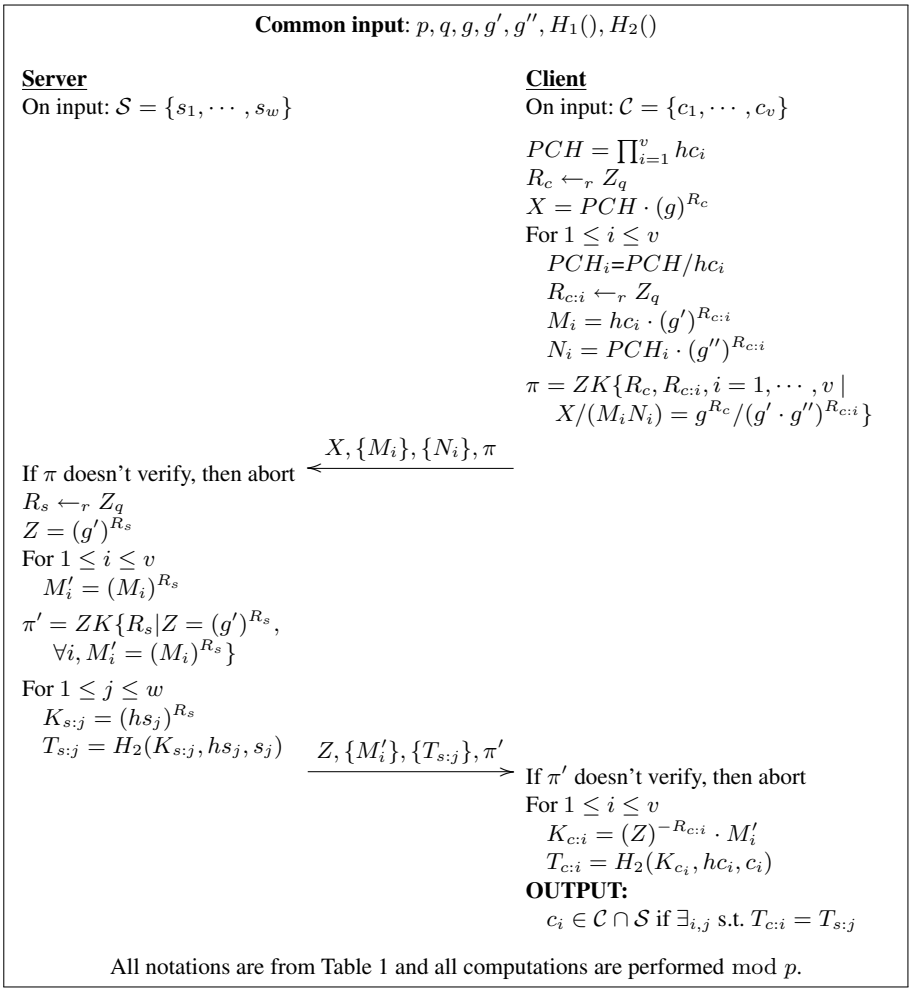


Fig. 2. Our PSI Protocol with linear complexity secure against malicious adversaries

Since the distribution of $X, \{M_i, N_i\}_{i=1,\ldots,v}$ sent by SIM_s is identical to the distribution produced by the real client C and the π proof system is zero-knowledge, S^*'s views when interacting with real client C and with simulator SIM_s are indistinguishable.

[*Output of the honest real client C interacting with S^**]

Now we consider output of honest real client C interacting with S^*. By soundness of π', message Z and M'_i sent by S^* is $Z = (g')^{R_s}$ and $M'_i = (M_i)^{Rs}$ for $i = 1, \cdots, v$. Then C's final output is a set containing all c_i's such that $H_2(M'_i Z^{-R_{c:i}}, hc_i, c_i) \in \{T_{s:j}\}$. In other words, for each c_i, C outputs c_i if $\exists\ j$ s.t. $H_2(M'_i Z^{-R_{c:i}}, hc_i, c_i) = T_{s:j}$. Since H_2 is a random oracle, there are two possibilities:

1. S^* computes $T_{s:j}$ from $H_2((hs_j)^{2Rs}, hs_j, s_j)$ for $s_j = c_i$. Since SIM_s described above extracts $s_j = c_i$ and adds s_j in $\mathcal{S}$, ideal world $\overline{C}$ also output c_i on its input c_i.
2. S^* did not query H_2 on $(M'_i Z^{-R_{c:i}}, hc_i, c_i)$ but $H_2(M'_i Z^{-R_{c:i}}, hc_i, c_i)$ happens to be equal to $T_{s:j}$. This event occurs with negligible probability bounded by $v \cdot\ w \cdot\ 2^{-\kappa}$.

Therefore, with probability $1 - v \cdot\ w \cdot\ 2^{-\kappa}$, real-world client C interacting with S^* and ideal-world client $\overline{C}$ interacting with SIM_s produce identical output.

[*Construction of ideal world SIM_c from malicious real-world client C^**]

Simulator SIM_c is formed as follows:

– **Setup and hash queries to H_1 and H_2:** Same as Setup and H_1 and H_2 responses described above in construction of SIM_s.
– **Simulation of real-world server S and ideal-world client $\overline{C}$:**

1. After getting $(X, \{M_i\}, \{N_i\})$, and interacting with C^* as verifier in proof π, SIM_c checks if π verifies. If not, it aborts. Otherwise, it runs the extractor algorithm for $R_c, \{R_{c:i}\}$ and computes $hc_1, \cdots, hc_v$.
2. For each hc_i, if $\exists (q, h_q) \in T_1$ s.t. $h_q = hc_i$ then add q to the set $\mathcal{C}$. Otherwise, add a dummy item to $\mathcal{C}$.
3. SIM_c plays the role of the client in the ideal-world. On input $\mathcal{C} = \{c_1, \cdots, c_v\}$, SIM_c interacts with the ideal-world server $\bar{S}$ through the TTP.
4. On getting intersection $L = \{c'_1, \cdots, c'_{|L|}\}$, with $|L| \leq v$ from the ideal-world interaction, SIM_c forms $\mathcal{S}$= $\Pi\left(c'_1, \cdots, c'_{|L|}, \delta'_1, \cdots, \delta'_{w-|L|+1}\right)$, where δ''s are dummy items and Π is a permutation function.
5. SIM_c picks $R_s \leftarrow_r \mathbb{Z}_q$, and computes $Z = g^{R_s}$ and $M'_i = (M_i)^{R_s}\}$ for $_{i=1,\ldots,v}$.
6. For each $s_j \in \mathcal{S}$:
   - If $s_j \in L$, compute $T_{s:j} = H_2((hs_j)^{R_s}, hs_j, s_j)$.
   - If $s_j \notin L$, compute $T_{s:j} \leftarrow_r \{0,1\}^{\kappa}$.
7. SIM_c returns $Z, \{M'_i\}, \{T_{s:j}\}$ to C^* and simulates proof π'.

Let fail be the event that C^* queries H_2 on $((hs_j)^{R_s}, hs_j, s_j)$ for $s_j \in \mathcal{S}$ and $s_j \notin L$. Similar to the argument in the proof of Theorem 1, if fail event does not occur, since the π' is zero-knowledge, we argue that C^*'s views in the real game with real-world server S and in the interaction with simulator SIM_c constructed above are indistinguishable .

Claim. If event fail occurs with non-negligible probability, then C^* can be used to break the DDH assumption.

We describe the reduction algorithm called $\mathcal{C}h$ that takes a DDH problem $(p', q', f, \alpha = f^a \pmod{p'}, \beta = f^b \pmod{p'}, \gamma)$ as an input and tries to output the answer using C^*. $\mathcal{C}h$ follows the SIM_c algorithm as we describe above, except that:

– **Setup:** On input $(p', q', f, \alpha, \beta, \gamma)$, $\mathcal{C}h_2$ sets $p = p'$, $q = q'$, $g' = f$ and picks generator g, $g'' \leftarrow_r \mathbb{Z}_q^*$.
– **Hash queries to H_1:** On query q to H_1, if $\nexists (q, h_q) \in T_1$ then $\mathcal{C}h_2$ responds with $h_q = \beta(g')^{r_q}$ where $r_q \leftarrow_r \mathbb{Z}_q$, and records (q, r_q, h_q) to T_1.

– **In computation for** $Z, \{M'_i\}, \{T_{s:j}\}$**:**
  - $\mathcal{C}h_2$ sets $Z = A$ and computes $M'_i = C(A)^{r_q + R_{c:i}}$.
  - For each $s_j \in \mathcal{S}$, if $s_j \in L$, $\mathcal{C}h_2$ computes $T_{s:j} = H_2(C(A)^{r_q}, hs_j, s_j)$.

Using an argument similar to that in the proof of Theorem 1, C^*'s views, when interacting with real server S and with simulator $\mathcal{C}h_2$, are indistinguishable under the DDH assumption. Assume that fail occurs, i.e., C^* makes a query to H_2 on $((hs_j)^{R_s}, hs_j, s_j)$ for $s_j \in \mathcal{S}$ but $s_j \notin L$. $\mathcal{C}h$ checks if $\exists (q, r_q, h_q) \in T_1$ and $\exists ((k, h'_q, q'), t) \in T_2$ s.t. $q = q'$, $h_q = h'_q$, $k = C(A)^{r_q}$ for each $q \in \mathcal{S}$ and $q \notin L$. If so, $\mathcal{C}h$ outputs True. Otherwise, $\mathcal{C}h_2$ outputs False. Thus, $\mathcal{C}h$ solves the DDH problem.

Since fail occurs with negligible probability, C^*'s view in the protocol with the real-world server S and in interaction with SIM_c is negligible.

[*Output of honest real server S interacting with C^**]

Finally, real-world S interacting with C^* in the real protocol outputs $\perp$ and ideal-world $\bar{S}$ interacting with SIM_c gets $\perp$.

6 Protocols Efficiency

In this section, we analyze the efficiency of our protocols and compare them to prior results. We summarize different features and estimated asymptotic complexities of prior work on Authorized Private Set Intersection and Private Set Intersection, respectively, as well as those of our protocols, in Table 2 and 3. Recall that we use w and v to denote the number of elements in the server and client input sets, respectively. Also, we specify whether they can support the extension for *data transfer* – a PSI variant introduced in [DT10] and discussed in details in the extended version of the paper [DKT10].

Note that our APSI protocol (in Figure 1) is, to the best of our knowledge, the only such construct, secure in the malicious model, with *linear* communication and computational complexity.

Comparing our PSI [Fig. 2] to [JL09]. Our PSI protocol achieves the same (linear) asymptotic overhead as in prior work [JL09], although, in ROM. However, the underlying cryptographic operations of [JL09], hidden in the big O() notation, are much more expensive than those in Figure 2, as we discuss below.

First, recall that, on average, each q-bit multi-exponentiation mod p involves $(1.5 \cdot |q|)$ multiplications of p-bit numbers. Whereas, each q-bit fixed-based exponentiation mod p incurs only $(0.5 \cdot |q|)$ multiplications. From now on, we denote with m a modular multiplication of p-bit numbers, and we assume $|p| = 1024$.

Observe that the PSI protocol in Figure 2, in the malicious model, incurs the total cost of $(240w + 960v)\mathsf{m}$. Due to space limitation, we refer to our extended version [DKT10] for all the details of our estimation.

In order to count the number of operations of [JL09], we use the optimized OPRF construction due to [BCC+09] and we use standard non-interactive ZK in ROM. We select set items to be drawn from a 40-bit domain.[1] The total cost of [JL09], in the malicious model, amounts to $(80w + 81320v)\mathsf{m}$. (Again, refer to [DKT10] for the details).

[1] Recall that, in the proof of [JL09], there seems to be a trade-off between the input domain size and security loss, as the simulator needs to exhaustively search over the input domain.

Table 2. Comparison of Authorized Private Set Intersection protocols

Protocol	Standard Model	Malicious Model	Assumption	Communication Complexity	Server	Client	Data Transfer
[CKRS09]	✓	✓	BDH	$O(w)$	$O(w)$ enc's of [BW06]	$O(w \cdot v)$ dec's of [BW06]	✓
[CZ09]	✓	✓	Strong RSA	$O(w+v)$	$O(w \cdot v)$ exps	$O(w \cdot v)$ exps	✗
Fig.2 of [DT10]	✗	✗	RSA	$O(w+v)$	$O(w+v)$ exps	$O(v)$ exps	✓
Our APSI	✗	✓	RSA	$O(w+v)$	$O(w+v)$ exps	$O(v)$ exps	✓

Table 3. Comparison of Private Set Intersection protocols

Protocol	Standard Model	Malicious Model	Assumption	Communication Complexity	Server	Client	Data Transfer
[FNP04]	✓	✗	Homom. Encr.	$O(w+v)$	$O(w \log\log v)$ exps	$O(w+v)$ exps	✓
[KS05]	✓	✓	Homom. Encr.	$O(w+v)$	$O(w \cdot v)$ exps	$O(w \cdot v)$ exps	✗
[JL09]	✓	✓	Decisional q-DH, CRS	$O(w+v)$	$O(w+v)$ exps	$O(v)$ exps	✓
[HN10]	✓	✓	DDH	$O(w+v)$	$O(w \log\log v)$ exps	$O(w+v)$ exps	✗
Fig.3 of [DT10]	✗	✗	One-More Gap-DH	$O(w+v)$	$O(w+v)$ exps	$O(v)$ exps	✓
Fig.4 of [DT10]	✗	✗	One-More RSA	$O(w+v)$	$O(w+v)$ exps	$O(v)$ mults	✓
Our PSI	✗	✓	DDH	$O(w+v)$	$O(w+v)$ exps	$O(v)$ exps	✓

Selecting, for instance, $w = v$, protocol in Figure 2 would require **as low as 1.5**% of the total modular multiplications incurred by [JL09] (even with the optimized OPRF construction and using non-interactive ZK in ROM). Only when $w/v >> 500$, [JL09] incurs lower cost. Furthermore, recall that, although secure in the standard model, the PSI construct in [JL09], when compared to ours, has three major drawbacks: (1) The size of set items should be polynomial in the security parameter, whereas, in our protocol, items can be taken from $\{0,1\}^*$, (2) It requires Decisional q-DH assumption and Common Reference String (CRS) model, where a safe RSA modulus must be generated by a mutually trusted party, and (3) It is not clear how to convert it into APSI.

We conclude that, though in ROM (as opposed to [JL09]), our PSI protocol significantly improves performance of prior PSI results, secure in the malicious model, while avoiding several restrictions.

7 Conclusion

In this paper, we presented PSI and APSI protocols secure in the malicious model under standard cryptographic assumptions, with linear communication and computational complexities. Proposed protocols offer better efficiency that prior work. In particular, our APSI protocol is the first technique to achieve linear computational complexity. Our efficiency claims are supported by detailed performance comparison.

Acknowledgements. This research was supported by the US Intelligence Advanced Research Projects Activity (IARPA) under grant number FA8750-09-2-0071 as well as by the Basic Science Research Program through the National Research Foundation of Korea, funded by the Ministry of Education, Science and Technology (2009-0070095) and by 'Developing Future Internet Network Model - Mathematical Approach' of the National Institute for Mathematical Sciences. Jihye Kim is the corresponding author for this paper. We also would like to thank Xiaomin Liu and Jae Hong Seo for their helpful comments.

References

[AL07] Aumann, Y., Lindell, Y.: Security Against Covert Adversaries: Efficient Protocols for Realistic Adversaries. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 137–156. Springer, Heidelberg (2007)

[BCC+09] Belenkiy, M., Camenisch, J., Chase, M., Kohlweiss, M., Lysyanskaya, A., Shacham, H.: Randomizable Proofs and Delegatable Anonymous Credentials. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 108–125. Springer, Heidelberg (2009)

[BDOP04] 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)

[BF03] Boneh, D., Franklin, M.K.: Identity-based encryption from the weil pairing. SIAM Journal of Computing 32(3), 586–615 (2003)

[BNPS03] 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)

[BW06] Boyen, X., Waters, B.: Anonymous Hierarchical Identity-Based Encryption (Without Random Oracles). In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 290–307. Springer, Heidelberg (2006)

[Cha83] Chaum, D.: Blind signatures for untraceable payments. In: CRYPTO (1983)

[CKRS09] Camenisch, J., Kohlweiss, M., Rial, A., Sheedy, C.: Blind and Anonymous Identity-Based Encryption and Authorised Private Searches on Public Key Encrypted Data. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 196–214. Springer, Heidelberg (2009)

[CS03] 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)

[CZ09] Camenisch, J., Zaverucha, G.: Private intersection of certified sets. In: Dingledine, R., Golle, P. (eds.) FC 2009. LNCS, vol. 5628, pp. 108–127. Springer, Heidelberg (2009)

[DJKT09] De Cristofaro, E., Jarecki, S., Kim, J., Tsudik, G.: Privacy-Preserving Policy-Based Information Transfer. In: Goldberg, I., Atallah, M.J. (eds.) Privacy Enhancing Technologies. LNCS, vol. 5672, pp. 164–183. Springer, Heidelberg (2009)

[DKT10] De Cristofaro, E., Kim, J., Tsudik, G.: Linear-Complexity Private Set Intersection Protocols Secure in Malicious Model, Cryptology ePrint Archive (2010), http://eprint.iacr.org/2010/469

[DSMRY09] Dachman-Soled, D., Malkin, T., Raykova, M., Yung, M.: Efficient Robust Private Set Intersection. In: Abdalla, M., Pointcheval, D., Fouque, P.-A., Vergnaud, D. (eds.) ACNS 2009. LNCS, vol. 5536, pp. 125–142. Springer, Heidelberg (2009)

[DT10] De Cristofaro, E., Tsudik, G.: Practical Private Set Intersection Protocols with Linear Complexity. In: Financial Cryptography and Data Security (2010)

[FNP04] Freedman, M.J., Nissim, K., Pinkas, B.: Efficient private matching and set intersection. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 1–19. Springer, Heidelberg (2004)

[FO97] 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)

[Gol04] Goldreich, O.: Foundations of cryptography: Basic applications. Cambridge Univ. Press, Cambridge (2004)

[HL08] Hazay, C., Lindell, Y.: Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 155–175. Springer, Heidelberg (2008)

[HN10] Hazay, C., Nissim, K.: Efficient Set Operations in the Presence of Malicious Adversaries. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 312–331. Springer, Heidelberg (2010)

[JL09] Jarecki, S., Liu, X.: Efficient Oblivious Pseudorandom Function with Applications to Adaptive OT and Secure Computation of Set Intersection. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 577–594. Springer, Heidelberg (2009)

[KS05] Kissner, L., Song, D.: Privacy-preserving set operations. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 241–257. Springer, Heidelberg (2005)

[NP06] Naor, M., Pinkas, B.: Oblivious polynomial evaluation. SIAM Journal on Computing 35(5), 1254–1284 (2006)

[Sha84] Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)

[Yao82] Yao, A.C.: Protocols for secure computations. In: FOCS, pp. 160–164 (1982)

Generic Compilers for Authenticated Key Exchange*

Tibor Jager, Florian Kohlar, Sven Schäge, and Jörg Schwenk

Ruhr-University Bochum

Abstract. So far, all solutions proposed for *authenticated key agreement* combine key agreement and authentication into a single cryptographic protocol. However, in many important application scenarios, key agreement and entity authentication are clearly separated protocols. This fact enables efficient attacks on the naïve combination of these protocols. In this paper, we propose new compilers for two-party key agreement and authentication, which are provably secure in the standard Bellare-Rogaway model. The constructions are generic: key agreement is executed first and results (without intervention of the adversary) in a secret session key on both sides. This key (or a derived key) is handed over, together with a transcript of all key exchange messages, to the authentication protocol, where it is combined with the random challenge(s) exchanged during authentication.

Keywords: authenticated key agreement, protocol compiler, TLS.

1 Introduction

Authenticated key agreement (AKE) is a basic building block in modern cryptography. Many secure protocols for two-party and group key agreement have been proposed, including generic compilers that transform simple key agreement protocols into authenticated key agreement protocols, with many additional security properties.

However, all known constructions (including e.g. the modular approach of [1], and the Katz-Yung compiler [22]) result in a single cryptographic protocol, whereas many security-critical *real-world* applications combine two or more clearly separated protocols:

– **(Client) Authentication and SSL/TLS.** The most prominent example is SSL/TLS. Although server and browser can be authenticated in a provably secure way [20,25] within a single cryptographic protocol (the TLS handshake protocol), nearly all known web applications authenticate the client through a different protocol on top of the TLS channel. The security of these protocols is based on the sole assumption that the (human) user is able to

* The research leading to these results has received funding from the European Community (FP7/2007-2013) under grant agreement number ICT-2007-216646 - European Network of Excellence in Cryptology II (ECRYPT II).

M. Abe (Ed.): ASIACRYPT 2010, LNCS 6477, pp. 232–249, 2010.

authenticate the server on the basis of security indicators of the browser, which was shown to be false in [17]. We do not rely on this assumption. Instead, we regard SSL/TLS simply as a key agreement protocol, which cannot be changed due to the large number of implementations that are running worldwide. We may however change the authentication protocol, since the authentication protocol is often implemented in HTML/Javascript[1].

– **Browser based Single Sign-On (SSO).** This scenario is perhaps the most complex one and a formalization is out of scope of this paper. However, it may serve as an illustration of how cryptographic protocols are combined today to implement key exchange (KE) and authentication functionalities. In SSO protocols, two key agreement protocols, and two different authentication protocols are combined to achieve the desired goal. Cryptographically secure SSO protocols have e.g. been described in [19].

In this work, we present a new compiler that handles these scenarios. Moreover, we can use our compiler to combine existing authentication protocols in a novel way with key exchange protocols. This includes:

– **Zero-Knowledge Authentication.** Zero-knowledge protocols have been developed with the goal to authenticate entities. However, in all known compilers, they cannot be combined with key agreement, except if they are transformed into digital signature schemes using the Fiat-Shamir heuristic. With our second compiler, ZK protocols can be used directly, which enables many interesting new protocols.
– **Privacy-preserving authentication.** With our compiler, we can easily combine privacy-preserving authentication protocols like Direct Anonymous Attestation with different key agreement protocols.

Man-in-the-middle Attack. Our real world attack scenario is as follows (cf. Figure 1): the adversary E ("Eve") acts as an active (wo) man-in-the-middle (MITM) between A and B during key exchange, and then acts as a passive "wire" during authentication. As a result, E has successfully authenticated as "A" towards B, and as "B" towards A, and shares (different) keys with A and B.

To counter this attack, one could of course apply standard cryptographic primitives to turn the key exchange protocol into an authenticated key exchange protocol (AKE) [1], but this is not possible in the cases cited above, *because the implementation of the KE protocol cannot be changed*, or the desired security goals (e.g. privacy) cannot be reached with standard compilers. Our compiler turns the combination of the two protocols into a provably secure AKE protocol. During compilation, only the authentication protocol is changed slightly.

[1] At first glance, it seems that the security of TLS as a key agreement protocol could easily be proven in the Bellare-Rogaway model, since we only have to consider passive adversaries, and the TLS ciphersuites includes e.g. ephemeral Diffie-Hellman key exchange. However, there are some subtle problems with the `Reveal` query and the fact that the final `Finished` message of the TLS handshake is already encrypted. Therefore it is still unclear if TLS fits in our theoretical framework.

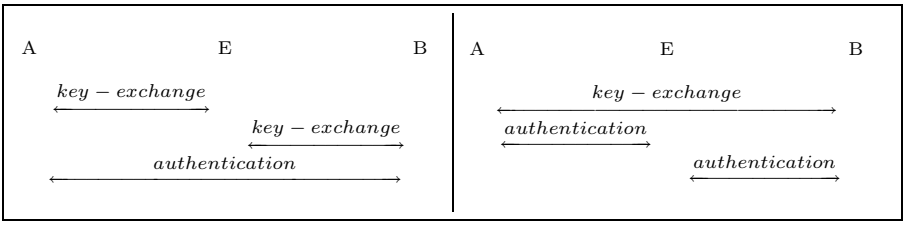


Fig. 1. Attack Scenario: Real world man-in-the-middle attack (left), and unknown key share attack (right)

Unknown Key Share (UKS) Attacks. To be able to prove the security in the standard Bellare-Rogaway (BR) model, the resulting AKE protocol must also be secure against unknown key share (UKS) attacks [14,13] that do not directly lead to an attack in the real world, but invalidate security proofs in the model. Interestingly, in our scenario this is a kind of orthogonal attack to MITM attacks (cf. Figure 1): The adversary acts as a man-in-the-middle on the authentication protocol. To achieve security against both (MITM and UKS) attacks, one usually needs two compilers: One compiler who adds authenticators to each message [1], and one compiler who includes the complete state of the session into the computation of the session key [15]. Our compilers achieve this in one step, because we force the adversary to prove knowledge of the session key k through the derived key dk during authentication. Thus the adversary cannot authenticate to A or B without knowing k, and neither A nor B will accept.

Practical AKE protocols. If the two parties accept, they share a common state. This state consists of the secret key k, and the transcript of all messages sent and received. This transcript plays an important role in the BR model, since it defines the attack possibilities of the adversary. In practically relevant AKE protocols, a hash of this transcript is included in a final message secured with a MAC, to protect against MITM attacks.

The A&KE Compilers. To protect against MITM attacks in our generic scenario, it is sufficient to simply include the transcript of the KE protocol into the authentication protocol. (Many authentication protocols offer the possibility to authenticate arbitrary strings chosen by A od B, e.g. authentication protocols based on digital signatures, or the MAP2 protocol from [2].) Such a compiler protects against MITM attacks because (a) any modification of messages in the KE protocol automatically results in a modification of messages in the authentication protocol (since the transcript is included), which results in an abort of the authentication protocol if this protocol is secure in the BR model. Thus (b) the adversary is restricted to a passive role when attacking the KE protocol, but this protocol is by definition secure against passive adversaries.

Unfortunately, this simple compiler cannot be proven secure in the BR model, because the adversary also has access to the transcript of the protocol, and can use this in both instances of the authentication protocol (cf. right side of Fig. 1.) To avoid this attack, a secret value only known to A and B (i.e. the

session key k) must be used in the authentication protocol in a generic way. There are at least two different methods (besides [15]) how to achieve this:

- An additional pair of messages can be sent after the KE and the authentication protocol. These messages contain a cryptographic checksum over the transcripts of both protocols. This checksum is basically a MAC, computed over the transcript of both the KE and the authentication protocol, using a key $K_{\mathsf{mac}} = \mathsf{PRF}(k, \text{“MAC”})$ derived from the key k returned by the KE protocol and some *pseudo-random function* PRF. The actual session key K returned by the compiled protocol (i.e., the value returned by a Reveal or Test query in the BR model) is also derived from k as $K = \mathsf{PRF}(k, \text{“KE”})$. In Section 3, we describe the compiler for this in detail, and prove its security in the standard model.
- Alternatively, we can modify a value that is present in *all* secure authentication protocols, in such a way that it does not change the security properties of the protocol:

  In a generic authentication protocol, a random challenge r_A guaranteeing the freshness of the message(s) must be sent from the challenger A to the prover B, which is answered with a response s_B from B. Ideally, this challenge is chosen from a large message space with uniform distribution.

  We assume that r_A is chosen uniformly from $\{0,1\}^t$, for some security parameter t. The answer $s_B := f(sk_B, r_A)$ is computed using the secret long-lived key sk_B of B, and the challenge r_A.

  Our compiler changes the computation of s_B slightly. Instead of using the challenge r_A directly, we use a derived value r'_A from the same distribution:

$$r'_A := \mathcal{H}(K_{\mathsf{mac}}, r_A, transcript_{KE}),\ \ s'_B := f(sk_B, r'_A),$$

  where $\mathcal{H}$ is some *hash function* modeled as a random oracle. Please note that r'_A is never sent (cf. Figure 3), but has to be computed by A and B. Thus the adversary E does not learn r'_A. This construction does not alter the security properties of the authentication protocol.

  In Section 4, we give a security proof for this compiler in the random oracle model.

1.1 Related Work

In their seminal papers [2,1] on two-party authenticated key agreement, Bellare et al. started a line of research that has expanded in two directions: group key agreement [9], [8,22,10], and refined models to cover different types of attacks [11,23,24]. All these models cover concurrent execution of the protocol, and at least corruption of non-related session keys.

All models can roughly be classified in two groups: models that require a unique session ID before the start of the protocol, and models that construct this session ID. [11] is the prototype of the former case: proofs and definitions are easier, but it is unclear how a session ID can be defined for practical applications. (E.g. in case of an SSL man-in-the-middle, browser and server do not share any

common state.) Newer models like [24] or [23] thus avoid this assumption, and construct the session identifiers from the messages sent and received by the intended communication partners.

Unknown key share [5] attacks do not threaten the real world security of cryptographic protocols, but invalidate security proofs in the formal models that follow [2]: If the adversary is able to force two protocol participants into accepting the same session key, but without a matching conversation, a `Reveal` query to one of the participants will help to win the `Test` game against the other participant. Choo, Boyd and Hitchcock have shown how to invalidate security proofs of various protocols in the different models [14,13], and how to fix the problem by including the whole session information in the computation of the session key [15]. They were able to compare the relative strengths of the different models assuming that session identifiers are constructed as a concatenation of the exchanged messages.

Canetti and Krawczyk in [12] consider a practically important protocol (IPSec IKE), which has a structure that places authentication after key exchange. Still, this is a single AKE protocol, and thus not comparable to our construction. In 2008 Morissey et al. studied the security of the TLS key agreement protocol [25] and provided a modular and generic proof of security for the established application keys.

Katz and Yung presented in [22] a first scalable compiler that transforms any passively secure group key-exchange protocol to an actively secure AKE. Their compiler adds one round and constant size (per user) to the original scheme, by appending an additional signature to each message of the protocol.

1.2 Contribution

In this paper, we describe two new compilers that allow us to combine key agreement protocols (which, in the BR model, need only be secure against passive adversaries) with arbitrary authentication protocols to form an authenticated key agreement (AKE) protocol in the sense of [2].

These compilers enable us to formally prove the security of real world protocols in the BR model, which was not possible before. The most important case here is TLS with an authentication protocol on top of the TLS channel, which can be proven secure if the authentication protocol is secure in the BR model. This is possible since we consider TLS only as a key agreement protocol, and not as an AKE protocol, and it seems likely that the security of (some ciphersuites of) TLS against passive adversaries can be proven.

Additionally, the compilers allow for a modular design of new AKE protocols, using existing protocols (e.g. TLS, IPSec IKE) or new ones (e.g. zero-knowledge authentication, group signatures). The formal security proof is simplified considerably, since the security of key agreement and authentication protocols can be proven separately, and our theorems yield the security of the combined protocol.

2 Preliminaries and Definitions

In this section, we recall the syntax and security definitions of the building blocks for our protocol compilers.

2.1 Digital Signature Schemes

A digital signature scheme is a triple $\Sigma = (\mathsf{SIG.Gen}, \mathsf{SIG.Sign}, \mathsf{SIG.Vfy})$, consisting of a key generation algorithm $(sk, pk) \xleftarrow{\$} \mathsf{SIG.Gen}(1^\kappa)$ generating a (public) verification key pk and a secret signing key sk on input of security parameter κ, signing algorithm $\sigma \xleftarrow{\$} \mathsf{SIG.Sign}(sk, m)$ generating a signature for message m, and verification algorithm $\mathsf{SIG.Vfy}(pk, m, \sigma)$ returning 1, if σ is a valid signature for m under key pk, and 0 otherwise.

Consider the following security experiment played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

1. The challenger generates a public/secret key pair $(sk, pk) \xleftarrow{\$} \mathsf{SIG.Gen}(1^\kappa)$, the adversary receives pk as input.
2. The adversary may query arbitrary messages m_i to the challenger. The challenger replies each query with a signature $\sigma_i = \mathsf{SIG.Sign}(sk, m_i)$. Here i is an index, ranging between $1 \leq i \leq q$ for some polynomial $q = q(\kappa)$. Queries can be made adaptively.
3. Eventually, the adversary outputs a message/signature pair (m, σ).

Definition 1. *We say that Σ is secure against* existential forgeries under adaptive chosen-message attacks *(EUF-CMA), if*

$$\Pr\left[(m, \sigma) \xleftarrow{\$} \mathcal{A}^{\mathcal{C}}(1^\kappa, pk) : \mathsf{SIG.Vfy}(pk, m, \sigma) = 1 \wedge m \notin \{m_1, \ldots, m_q\}\right] \leq \epsilon.$$

for all probabilistic polynomial-time (in κ) adversaries $\mathcal{A}$, where $\epsilon = \epsilon(\kappa)$ is some negligible function in the security parameter.

2.2 Message Authentication Codes

A *message authentication code* is an algorithm MAC. This algorithm implements a deterministic function $w = \mathsf{MAC}(K_{\mathsf{mac}}, m)$, taking as input a (symmetric) key $K_{\mathsf{mac}} \in \{0,1\}^\kappa$ and a message m, and returning a string w.

Consider the following security experiment played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

1. The challenger samples $K_{\mathsf{mac}} \xleftarrow{\$} \{0,1\}^\kappa$ uniformly random.
2. The adversary may query arbitrary messages m_i to the challenger. The challenger replies each query with $w_i = \mathsf{MAC}(K_{\mathsf{mac}}, m_i)$. Here i is an index, ranging between $1 \leq i \leq q$ for some polynomial $q = q(\kappa)$. Queries can be made adaptively.
3. Eventually, the adversary outputs a pair (m, w).

Definition 2. *We say that* MAC *is a* secure *message authentication code, if*

$$\Pr\left[(m, w) \xleftarrow{\$} \mathcal{A}^{\mathcal{C}}(1^{\kappa}) : w = \mathsf{MAC}(K_{\mathsf{mac}}, m) \wedge m \notin \{m_1, \ldots, m_q\}\right] \leq \epsilon$$

for all probabilistic polynomial-time (in κ) adversaries $\mathcal{A}$, where $\epsilon = \epsilon(\kappa)$ is some negligible function in the security parameter.

2.3 Pseudo-random Functions

A *pseudo-random function* is an algorithm PRF. This algorithm implements a deterministic function $z = \mathsf{PRF}(k, x)$, taking as input a key $k \in \{0,1\}^{\kappa}$ and some bit string x, and returning a string $z \in \{0,1\}^{\kappa}$.

Consider the following security experiment played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

1. The challenger samples $k \xleftarrow{\$} \{0,1\}^{\kappa}$ uniformly random.
2. The adversary may query arbitrary values x_i to the challenger. The challenger replies each query with $z_i = \mathsf{PRF}(k, x_i)$. Here i is an index, ranging between $1 \leq i \leq q$ for some polynomial $q = q(\kappa)$. Queries can be made adaptively.
3. Eventually, the adversary outputs value x and a special symbol $\top$. The challenger sets $z_0 = \mathsf{PRF}(k, x)$ and samples $z_1 \xleftarrow{\$} \{0,1\}^{\kappa}$ uniformly random. Then it tosses a coin $b \xleftarrow{\$} \{0,1\}$, and returns z_b to the adversary.
4. Finally, the adversary outputs a guess $b' \in \{0,1\}$.

Definition 3. *We say that* PRF *is a* secure *pseudo-random function, if*

$$|\Pr[b = b'] - 1/2| \leq \epsilon$$

for all probabilistic polynomial-time (in κ) adversaries $\mathcal{A}$, where $\epsilon = \epsilon(\kappa)$ is some negligible function in the security parameter.

2.4 Key Exchange Protocols

A (two-party) key-exchange protocol is a protocol executed among two parties A and B. At the end of the protocol, both A and B obtain the same key K_0 as the output of the protocol.

Definition 4. *We say that a key-exchange protocol is* passively-secure *if for all polynomial-time adversary holds that $|\Pr[b = b'] - 1/2| \leq \epsilon$ for some negligible function ϵ in the following experiment.*

1. *A challenger generates the public parameters Λ of the protocol (e.g. a generator describing a group etc.).*
2. *The adversary receives Λ as input, and may query the challenger. To this end, it submits a symbol $\perp$. Then, the challenger runs a protocol instance, and obtains the transcript T of all messages exchanged during the protocol and a key K_0. The challenger returns (T, K_0).*

3. *Eventually, the adversary outputs a special symbol* $\top$*. Given* $\top$*, the challenger runs a protocol instance, obtaining the transcript* T *and key* K_0*, samples* K_1 *uniformly at random from the key space of the protocol, and tosses a fair coin* $b \in \{0,1\}$*. Then it returns* (T, K_b) *to the adversary.*
4. *The adversary may continue making* $\perp$*-queries to the challenger.*
5. *Finally, adversary* E *outputs a bit* b'*.*

We say that the adversary wins the game, if $b = b'$*.*

Simple protocols satisfying the above definition are the Diffie-Hellman protocol (under the DDH assumption), or key-transport using an IND-CPA secure encryption scheme (i.e., party A samples a random key k, encrypts k under B's public key, and sends the ciphertext to B).

2.5 Secure Authenticated Key Exchange

While the security model for passively-secure key-exchange protocols is very simple, a more complex model is required to model the capabilities of active adversaries to define secure authenticated key-exchange. We must describe the subtleties of executions that we expect from the implementations of the protocol, the attacks against which our protocol should be secure, and which outcome we expect if we run the protocol with the defined adversary. In accordance with the line of research [5,11,24,16] initiated by Bellare and Rogaway [2], we model our adversary by providing an "execution environment", which emulates the real-world capabilities of an active adversary. That is, the adversary has full control over the communication network, thus may forward, alter, or drop any message sent by the participants, or insert new messages.

Execution Model. Let $I = I(\kappa)$ and $S = S(\kappa)$ be polynomials in the security parameter κ. Our model is characterized by a collection of oracles

$$\{\pi_{i,j}^s : i, j \in [I], s \in [S]\}$$

An oracle $\pi_{i,j}^s$ represents an entity i running the protocol with entity j for the s-th time. Each oracle maintains its own internal state (e.g. nonces), all oracles representing some entity i share the same long-term secrets of entity i. Moreover, each oracle $\pi_{i,j}^s$ maintains a variable T storing an ordered list of all messages sent and received by $\pi_{i,j}^s$ so far.

An oracle aborts, if it receives a message which is not valid according to the protocol specification, or terminates after it has sent or received the last protocol message according to the protocol specification. When a process terminates, it outputs "accept" or "reject" and (possibly) a key k.

An adversary may interact with these oracles by issuing different types of queries. Before the first query is asked, long-term secret/public key pairs (pk_i, sk_i) for each entity i are generated. An adversary $\mathcal{A}$ receives as input the long-term public keys $(pk_1, \ldots, pk_l)$ of all parties, and may then ask the following query:

– $\mathsf{Send}(\pi_{i,j}^s, m)$: The adversary can use this query to send any message m of his own choice to oracle $\pi_{i,j}^s$. The oracle will respond according to the protocol

specification. If $m = \emptyset$, where $\emptyset$ denotes the empty string, then $\pi^s_{i,j}$ will respond with the first protocol message.

Secure Authentication Protocols. An authentication protocol is a protocol run between two processes $\pi^s_{i,j}$ and $\pi^t_{j,i}$ of two parties P_i and P_j, where both processes output either "accept" or "reject" at the end of the protocol. We define correctness and security of an authentication protocol following the idea of *matching conversations*, as introduced by Bellare and Rogaway [2].

In the following let $T_{i,s}$ denote the transcript of all messages sent and received by process $\pi^s_{i,j}$. Intuitively, we would like to say that a protocol is *correct*, if a process $\pi^s_{i,j}$ outputs "accept" *if* there exists a process $\pi^t_{j,i}$ with $T_{i,s} = T_{j,t}$. Likewise, we would like to say that a protocol is *secure*, if a process accepts *only if* there exists a process $\pi^t_{j,i}$ with $T_{i,s} = T_{j,t}$.

As in [2], we face a minor technical obstacle here, which is inherent to authentication protocols. Suppose that P_j sends the last message of the protocol (thus, P_i has initiated the protocol run if the number of protocol rounds is even, or P_j has initiated the protocol if the number of rounds is odd). Party P_j does not get any response to its last message, thus has to accept without knowing whether P_i received the last message.[2] To overcome this obstacle, we let $T'_{i,s}$ be the transcript $T_{i,s}$ truncated by the last message, and we have to define correctness and security in a slightly more complicated way.

Definition 5. *We say that two processes $\pi^s_{i,j}$ and $\pi^t_{j,i}$ have* matching conversations*, if either*

- *P_i sends the last message of the protocol according to the protocol specification and it holds that $T'_{j,t} = T'_{i,s}$, or*
- *P_j sends the last message of the protocol according to the protocol specification and it holds that $T_{j,t} = T_{i,s}$.*

Definition 6. *We say that an authentication protocol is* correct*, if for all processes $\pi^s_{i,j}$ holds that $\pi^s_{i,j}$ "accepts" if there exists a process $\pi^t_{j,i}$ such that $\pi^s_{i,j}$ and $\pi^t_{j,i}$ have matching conversations.*

Definition 7. *We say that an authentication protocol is* secure *in the Bellare-Rogaway model, if for all probabilistic polynomial-time (PPT) adversaries $\mathcal{A}$, interacting with the black-box $\mathcal{O}(\Pi)$ as described above in the execution model, holds that:*

Each process $\pi^s_{i,j}$ of $\mathcal{O}(\Pi)$ "accepts" only if *there exists a process $\pi^t_{j,i}$ such that $\pi^s_{i,j}$ and $\pi^t_{j,i}$ have matching conversations, except for some negligible probability $\epsilon = \epsilon(\kappa)$ in the security parameter.*

Secure Authenticated Key-Exchange Protocols. An authenticated key-exchange protocol is an authentication protocol, where additionally both parties

[2] In contrast, a protocol can be designed such that the party receiving the last message accepts only if it has received this message correctly according to the protocol specification.

obtain a key k after accepting. Intuitively, we would like to say that a authenticated key-exchange protocol is secure, if

- the protocol is a secure authentication protocol, and
- an adversary can not distinguish a key k computed in a protocol run from a uniformly random value from the key space. This should hold even if the adversary is able to learn the key computed in other protocol instances.

We formalize this by extending the execution model by two more type of queries, which may be asked by the adversary.

- $\mathsf{Test}(\pi^s_{i,j})$: This query may only be asked once throughout the game. If process $\pi^s_{i,j}$ has not (yet) "accepted", the black-box returns some failure symbol $\perp$. Otherwise the black-box flips a fair coin b. If $b = 0$, a random element from the keyspace is returned. If $b = 1$ then the session key k computed in process $\pi^s_{i,j}$ is returned.
- $\mathsf{Reveal}(\pi^s_{i,j})$: The adversary may learn the encryption key K computed in process $\pi^s_{i,j}$ by asking this type of query. The adversary submits $\pi^s_{i,j}$ to the black-box. If process $\pi^s_{i,j}$ has "accepted", the black-box responds with the key k in process $\pi^s_{i,j}$. Otherwise some failure symbol $\perp$ is returned.

Definition 8. *Let $\mathcal{A}$ be a PPT adversary, interacting with the black-box $\mathcal{O}(\Pi)$ described in the above execution model (denoted with $\mathcal{A}^{\mathcal{O}(\Pi)}$).*

We say that an authenticated key-exchange protocol Π is secure *in the Bellare-Rogaway model, if 1.) Π is a secure authentication protocol according to Definition 7, and 2.)*

$$\left| \Pr[\mathcal{A}^{\mathcal{O}(\Pi)}(1^\kappa) = b] - \frac{1}{2} \right| \leq \epsilon$$

for all $\mathcal{A}$.

As Shoup pointed out in [27, §15], we do not have to explicitly model a Corrupt-query, as one can efficiently reduce the standard BR-Model to a model without Corrupt-queries (see also [6, p. 70 ff.]).

3 Authenticated Key Exchange Compiler in the Standard Model

Let us now describe our generic AKE compiler. The compiler takes as input the following building blocks (which have been defined in Section 2).

- A key-exchange protocol KE,
- a digital signature scheme $\Sigma = (\mathsf{SIG.Gen}, \mathsf{SIG.Sign}, \mathsf{SIG.Vfy})$,
- a message authentication code MAC,
- and a pseudorandom function PRF.

The compiled protocol between two parties A and B proceeds as follows (see also Figure 2).

1. A and B run the key exchange protocol. For instance, both parties may run the well-known Diffie-Hellman protocol [18]. Throughout this protocol run, both parties compute key k and record a transcript T^A_{KE} and T^B_{KE}, where T^C_{KE} consists of the list of all messages sent and received by party $C \in \{A, B\}$.
2. The key k computed by KE is used to derive two keys $K = \mathsf{PRF}(k, \text{“KE”})$ and $K_{\mathsf{mac}} = \mathsf{PRF}(k, \text{“MAC”})$, where “KE” and “MAC” are some arbitrary fixed constants such that “KE” $\neq$ “MAC”.[3]
3. Then A samples a random nonce $r_A \xleftarrow{\$} \{0,1\}^\lambda$ and sends it to B, B samples $r_B \xleftarrow{\$} \{0,1\}^\lambda$ and sends it to A.
4. Party A computes a signature $\sigma_A \xleftarrow{\$} \mathsf{SIG.Sign}(sk_A, T^A_1)$ under A's secret key sk_A, where $T^A_1 = (T^A_{\mathsf{KE}}||r_A||r^A_B)$ is the transcript of all messages sent and received by A so far. Then B computes a signature over the transcript $T^B_1 = (T^B_{\mathsf{KE}}||r^B_A||r_B)$ of all messages sent and received by B. Let $T^A_2 = (\sigma_A||\sigma^A_B)$ denote the signatures sent and received by A, and $T^B_2 = (\sigma^B_A||\sigma_B)$ be the signatures sent and received by B.
5. A sends a MAC $t_A = \mathsf{MAC}(K_{\mathsf{mac}}, T^A_2||0)$ over transcript T^A_2 using the key K_{mac} computed in 2. B replies with $t_B = \mathsf{MAC}(K_{\mathsf{mac}}, T^B_2||1)$.
6. Party A accepts, if $\mathsf{SIG.Vfy}(pk_B, T^A_1, \sigma^A_B) = 1$ and $t_B = \mathsf{MAC}(K_{\mathsf{mac}}, T^A_2||1)$, that is, if σ^A_B is a valid signature for T^A_2 under B's verification key pk_B and if w_B is a valid MAC under key K_{mac} for $T^A_2||1$. B accepts if it holds that $\mathsf{SIG.Vfy}(pk_A, T^B_1, \sigma^B_A) = 1$ and $w_A = \mathsf{MAC}(K_{\mathsf{mac}}, T^B_2||0)$. Finally, if both parties accept then the key K is returned.

Observe that the signatures and MACs are verified using the *internal* transcripts of party A and B. The intention behind the idea of embedding the transcripts in the protocol is to detect any changes that an active adversary makes to the messages sent by A and B. Informally, in the two-layer authentication consisting of the signature scheme and MAC, the signature is used to authenticate users and thwart *man-in-the-middle* attacks on the key-exchange protocol, while the MAC is used as an implicit “key confirmation” step to avoid *unknown key-share* attacks [14,13].

This allows us to prove security requiring only pretty weak security properties from the utilized building blocks, namely we require that KE is secure against *passive* adversaries only, that the digital signatures are existential unforgeable under (non-adaptive) chosen-message attacks, and that the MAC and PRF meet their standard security notions.

Remark 1. The digital signatures sent in the first round after running KE are merely a concrete instantiation of a *tag-based authentication scheme* as introduced in [21]. It is possible to generalize the above protocol by replacing the digital signatures with a tag-based authentication scheme, without making substantial changes to the protocol or the security proof given below.

[3] Note that we assume here implicitly, that the output key space of KE matches the input key space of PRF. This fact is not only important for correctness, but also for the security proof.

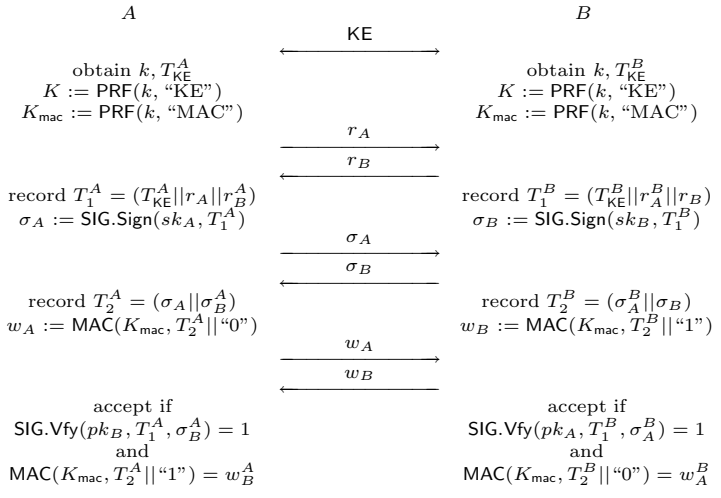


Fig. 2. AKE Protocol

3.1 Security Analysis

Theorem 1. *If the* KE *protocol, the signature scheme, the message authentication code and the pseudo-random function are secure with respect to the definitions in Section 2, then the above protocol is a secure* AKE *protocol in the sense of Definition 8.*

We prove the above theorem by two lemmas. Lemma 1 states that the AKE protocol meets property 1) of Definition 8, Lemma 2 states that it meets property 2) of Definition 8.

Lemma 1. *If the key exchange protocol (*KE*), the signature scheme (*SIG*), the message authentication code (*MAC*) and the pseudo-random function (*PRF*) are secure with respect to the definitions in Section 2, then the above protocol holds property 1) of Definition 8.*

Proof. (Sketch) The proof proceeds in a *sequence of games*, following [3,28]. The first game is the real security experiment. By assumption there exists an adversary $\mathcal{A}$ that breaks the security of the above protocol. We then describe several intermediate games that step-wisely modify the original game. Next we show that in the final security game the adversary has only negligible advantage in breaking the security of the protocol. Finally we prove that (under the stated security assumptions) no adversary can distinguish any of these games X_{i+1} from its predecessor X_i. Let X_i be the event that $\mathcal{A}$ wins in Game i. In the following let $\mathsf{negl}(\kappa)$ be some (unspecified) negligible function in the security parameter κ.

Game 0. This is the original security game with $b = 1$, that is, the adversary receives always the "real" key. By assumption $\mathcal{A}$ can distinguish K from a random key (i.e. correctly answer the $\mathsf{Test}(\pi^s_{i,j})$ query) when given access to the $\mathsf{Send}(\pi^s_{i,j}, m)$ and $\mathsf{Reveal}(\pi^s_{i,j})$ oracles while A and B accept.

Game 1. This game proceeds exactly like the previous game, except that the simulator aborts if A or B accept and $T_1^A \neq T_1^B$.

Claim 1. We claim that

$$|\Pr[X_1] - \Pr[X_0]| \leq \mathsf{negl}(\kappa)$$

by the EUF-CMA security of the digital signature scheme. The proof of this claim exploits that from A's perspective the transcript T_1^A is unique with overwhelming probability (due to the honest random choice of r_A) and from B's perspective T_1^B is unique for (honestly-chosen) r_B with overwhelming probability.

Game 2. This game proceeds exactly like the previous game, except that the simulator now chooses a uniformly random key $\hat{k}$ to derive K_{mac} and K as $K_{\mathsf{mac}} = \mathsf{PRF}(\hat{k},$ "MAC") and $K = \mathsf{PRF}(\hat{k},$ "KE").

Claim 2. We claim that

$$|\Pr[X_2] - \Pr[X_1]| \leq \mathsf{negl}(\kappa)$$

by the security of KE against passive adversaries. In the proof we exploit that we must have $T_1^A = T_1^B$ if A or B accept, as otherwise we abort due to Game 1.

Game 3. This game proceeds exactly like the previous game, except that the simulator now chooses a uniformly random key $\tilde{k}$ (instead of K_{mac}) to compute w_A and w_B as $w_A = \mathsf{MAC}(\tilde{k}, T_2||0)$ and $w_B = \mathsf{MAC}(\tilde{k}, T_2||1)$.

Claim 3. We claim that

$$|\Pr[X_3] - \Pr[X_2]| \leq \mathsf{negl}(\kappa)$$

by the security of the pseudorandom function PRF. In the proof we exploit that we have exchanged the "real" key k computed in KE with a "random" key $\tilde{k}$ in Game 2.

Observe here that, since the output key space of KE must match the input key space of PRF, and PRF is assumed to be secure, it follows implicitly here that the output key space of KE needs to be super-polynomially large.

Game 4. This game proceeds exactly like the previous game, except that the simulator aborts if A or B accepts and $T_2^A \neq T_2^B$.

Claim 4. We claim that

$$|\Pr[X_4] - \Pr[X_3]| \leq \mathsf{negl}(\kappa)$$

Recall that in Game 4 we must have $T_1^A = T_1^B$ due to our abort condition from Game 1, and that we have replaced the key k computed in KE with a uniformly random key $\tilde{k}$ in Game 3 to compute the MACs in the considered protocol instance. Thus, if we have $T_2^A \neq T_2^B$, then the adversary must have forged a MAC to make A or B accept. We can therefore use the adversary to break the security of MAC.

Game 5. This game proceeds exactly like the previous game except that the simulator aborts if A or B accepts and $T_3^A \neq T_3^B$, where $T_3^A = (w_A, w_B^A)$ consists of the MACs sent and received by A and $T_3^B = (w_A^B, w_B)$ consists of the MACs sent and received by B.

Claim 5. We have

$$\Pr[X_5] = \Pr[X_4].$$

This follows from the fact that we have defined MAC as a deterministic function, and we have $T_1^A = T_1^B$ due to Game 1 and $T_2^A = T_2^B$ due to Game 4.

Collecting probabilities from Game 0 to 5, we obtain that both A and B accept *only if* they have *matching conversations*, except for some negligible error probability.

Lemma 2. *If* KE, SIG, MAC *and* PRF *are secure with respect to the definitions in Section 2, then the above protocol holds property 2) of Definition 8.*

Proof. (Sketch). Again we proceed in a sequence of games. The first 5 games of the proof are identical to the sequence in the proof of Lemma 1. We merely add one further game.

Game 6. This game proceeds exactly like the previous game except that the simulator now chooses K uniformly at random from the keyspace.

Claim 6. We claim that

$$|\Pr[X_6] - \Pr[X_5]| \leq \mathsf{negl}(\kappa).$$

This again follows from the security of the PRF, where we use that the seed $\hat{k}$ is chosen uniformly random and independent (cf. Game 2).

In Game 6, the adversary receives a uniformly random key K. However, by collecting probabilities from Game 0 to 6 we obtain that Game 6 is (computationally) indistinguishable from Game 0, which proves indistinguishability of "real" from "random" keys. Thus, the protocol is secure in the sense of Definition 8.

4 An Alternative AKE Compiler for Practical Protocols

Our second compiler is designed for practical applications, where we cannot change the session key K resulting from the KE protocol [15], or where we want to avoid an additional round of protocol messages after the authentication protocol. In this compiler, we directly integrate the transcript of the KE protocol, and the secret value K_{mac}, into the authentication protocol. To do so, we first have to define a "generic" scheme for an authentication protocol.

We only have minimal requirements on the authentication protocols. The party ("challenger") who wants to authenticate the other party ("prover") has to include a random value of high entropy into one of its protocol messages. (Otherwise an adversary may just query different instances of the prover for responses for the most probable challenges to increase her advantage.) The prover

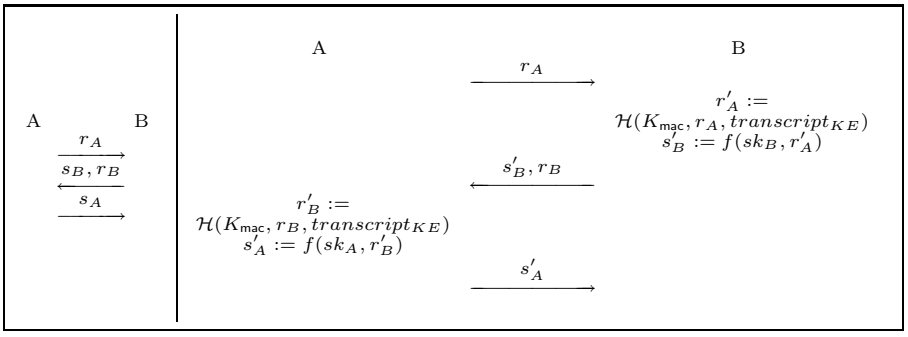


Fig. 3. Scheme of a standard mutual authentication protocol Γ (left), and the version Γ' modified by our compiler (right)

must answer with a value that was computed using his long-lived key sk, and the challenge itself.

The following protocols fulfill our requirements:

- AKEP1 and AKEP2 as defined in [2]
- Sigma- and Schnorr protocols (see [26])
- Zero-Knowledge Authentication protocols as introduced in [7]
- Zero-Knowledge Password-Proof protocols as introduced in [4].
- Signature based authentication protocols.

In this respect, our compiler may even enhance the security of the authentication protocol. This applies to the authentication of both parties, or of one party only.

Let Γ be an authentication protocol as depicted in Fig. 3. Then we denote by r_A a value (the challenge) that is sent from A to B, and by $s_B = f(sk_B, r_A)$ the value (response) returned to A that allows A to check the authenticity of B. The values r_B and s_A are defined analogously.

The main idea in the construction of a modified authentication protocol Γ' is to transmit r_A and r_B according to the protocol specification of Γ, but to compute the response based on both the received challenge, the transcript $transcript_\Pi$ of the key agreement protocol Π, and secret value K_{mac}. This is done using a random oracle $\mathcal{H}$. Our compiler $Comp$, which takes as input a key agreement protocol Π secure against passive adversaries, and a secure authentication protocol Γ, outputs an authenticated key agreement protocol $Comp(\Pi, \Gamma)$ which works as follows:

A&KE-2 Compiler: Let $(\pi^s_{A,B}, \gamma^s_{A,B})$ and $(\pi^t_{B,A}, \gamma^t_{B,A})$ be two pairs of oracles for Π and Γ.

1. Π is executed by $\pi^s_{A,B}$ and $\pi^t_{B,A}$ without any change. The resulting secret value is $k = (K, K_{\mathsf{mac}})$ for $\pi^s_{A,B}$, and $k' = (K', K'_{\mathsf{mac}})$ for $\pi^t_{B,A}$. (Ideally $k = k'$, but we have to take into account actions by the adversary.) The session key K (K', resp.) is used for encryption and integrity protection, and the secret value K_{mac} (K'_{mac}, resp.) is sent locally to the processes $\gamma^s_{A,B}$

and $\gamma^t_{B,A}$, together with the local transcript of the messages of Π. (The values K and K_{mac} are computed as described in Section 3.)

2. Now Γ is executed by $\gamma^s_{A,B}$ and $\gamma^t_{B,A}$ with the following change: In the computation of s_A and s_B, the values r_A and r_B are replaced with $r'_A := \mathcal{H}(K_{\mathsf{mac}}, r_B, transcript_\Pi)$ and $r'_B := \mathcal{H}(K'_{\mathsf{mac}}, r_A, transcript'_\Pi)$, and thus we get $s'_A = f(sk_A, r'_A)$ and $s'_B = f(sk_B, r'_B)$, where $\mathcal{H}(\cdot,\cdot,\cdot)$ is a random oracle. If $\gamma^s_{A,B}$ accepts, the local output is K, and K' for $\gamma^t_{B,A}$.

Lemma 3. *If Π is a key agreement protocol secure against passive adversaries, then it is impossible that three different oracles accept with the same (secret) state $(k, transcript_\Pi)$, where $k = (K, K_{\mathsf{mac}})$ is the secret value computed by Π, and $transcript_\Pi$ is the transcript of all protocol messages.*

Proof. If this was the case, then A, B and the (active) adversary E all would be able to compute k, but the adversary would not have modified any message exchanged between A and B (since the transcripts are identical). Thus E, acting as a passive adversary, would be able to compute k, a contradiction.

Lemma 4. *In $Comp(\Pi, \Gamma)$, any two oracles $\gamma^s_{A,B}$ and $\gamma^t_{B,A}$ with matching conversations have access to a unique random oracle that is defined as $\mathcal{H}_{A^tB^s}(\cdot) := \mathcal{H}(K_{\mathsf{mac}}, \cdot, transcript_\Pi)$. Neither E, nor any other oracle has access to this random oracle.*

Proof. Since the pair $(K_{\mathsf{mac}}, transcript_\Pi)$ is unique for any pair of oracles, $\mathcal{H}_{A^tB^s}(\cdot)$ is unique, too.

Theorem 2. *If Γ is a secure authentication protocol, then Γ' as defined in Fig. 3 also is a secure authentication protocol.*

Proof. Let $\gamma'^s_{A,B}$ and $\gamma'^t_{B,A}$ be two process (oracle) instances of A and B in Γ'. It should be clear that if $\gamma'^s_{A,B}$ and $\gamma'^t_{B,A}$ have matching conversations, then both oracles will accept.

We have to show that the probability that $\gamma'^s_{A,B}$ or $\gamma'^t_{B,A}$ accepts without a matching conversation is negligible. Now assume on the contrary that there is an adversary E' that is able to make $\gamma'^s_{A,B}$ or $\gamma'^t_{B,A}$ accept without a matching conversation, with non-negligible probability ϵ. Then we can define an adversary E that achieves the same goal with the protocol Γ: Since E' has no access to the random oracle $\mathcal{H}_{AB}$, she can only try to guess the challenge r'_A (r'_B, resp.). Now E is simply ignoring the challenge r_X she sees, and simply guesses a random challenge r''_X, and tries to compute s'_Y from this challenge. This strategy succeeds with non-negligible probability ϵ, and we have thus contradicted our assumption that Γ is a secure authentication protocol.

Theorem 3. *If Π is a key agreement protocol secure against passive adversaries, and if Γ is a secure authentication protocol, then $Comp(\Pi, \Gamma)$ is a secure authenticated key agreement protocol.*

Proof (Sketch). $\gamma^s_{A,B}$ and $\gamma^t_{B,A}$ will accept in Γ' if and only if they have access to the same random oracle $\mathcal{H}_{A^tB^s}(\cdot)$. (Otherwise they have to guess the challenge r'_X, which succeeds only with negligible probability.) If they have access to the same random oracle, then $\pi^s_{A,B}$ and $\pi^t_{B,A}$ completed Π with the same state $(k, transcript_\Pi)$. If $\gamma^s_{A,B}$ and $\gamma^t_{B,A}$ accept, Π and Γ were both completed by the same endpoints A and B. This excludes active attacks on Π (since the transcript is unchanged), and UKS attacks on Γ. Thus E may only mount a passive attack on Π, which succeeds only with negligible probability.

References

1. Bellare, M., Canetti, R., Krawczyk, H.: A modular approach to the design and analysis of authentication and key exchange protocols (extended abstract). In: STOC, pp. 419–428 (1998)
2. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 232–249. Springer, Heidelberg (1994)
3. 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)
4. Bellovin, S.M., Merritt, M.: Encrypted key exchange: Password-based protocols secure against dictionary attacks. In: IEEE Symposium On Research In Security And Privacy, pp. 72–84 (1992)
5. Blake-Wilson, S., Menezes, A.: Unknown key-share attacks on the station-to-station (STS) protocol. In: Imai, H., Zheng, Y. (eds.) PKC 1999. LNCS, vol. 1560, pp. 154–170. Springer, Heidelberg (1999)
6. Boyd, C., Mathuria, A.: Protocols for Authentication and Key Establishment, 1st edn. Springer, Heidelberg (2003)
7. Brandt, J., Damgård, I.B., Landrock, P., Pedersen, T.: Zero-knowledge authentication scheme with secret key exchange. In: Goldwasser, S. (ed.) CRYPTO 1988. LNCS, vol. 403, pp. 583–588. Springer, Heidelberg (1990)
8. Bresson, E., Chevassut, O., Pointcheval, D.: Dynamic group Diffie-Hellman key exchange under standard assumptions. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 321–336. Springer, Heidelberg (2002)
9. Bresson, E., Chevassut, O., Pointcheval, D., Quisquater, J.-J.: Provably authenticated group Diffie-Hellman key exchange. In: ACM Conference on Computer and Communications Security, pp. 255–264 (2001)
10. Bresson, E., Manulis, M.: Securing group key exchange against strong corruptions. In: Abe, M., Gligor, V.D. (eds.) ASIACCS, pp. 249–260. ACM, New York (2008)
11. Canetti, R., Krawczyk, H.: Analysis of key-exchange protocols and their use for building secure channels. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 453–474. Springer, Heidelberg (2001)
12. Canetti, R., Krawczyk, H.: Security analysis of IKE's signature-based key-exchange protocol. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 143–161. Springer, Heidelberg (2002)
13. Choo, K.-K.R., Boyd, C., Hitchcock, Y.: Errors in computational complexity proofs for protocols. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 624–643. Springer, Heidelberg (2005)

14. Choo, K.-K.R., Boyd, C., Hitchcock, Y.: Examining indistinguishability-based proof models for key establishment protocols. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 585–604. Springer, Heidelberg (2005)
15. Choo, K.-K.R., Boyd, C., Hitchcock, Y.: On session key construction in provably-secure key establishment protocols. In: Dawson, E., Vaudenay, S. (eds.) Mycrypt 2005. LNCS, vol. 3715, pp. 116–131. Springer, Heidelberg (2005)
16. Cremers, C.J.F.: Session-state reveal is stronger than ephemeral key reveal: Attacking the NAXOS authenticated key exchange protocol. In: Abdalla, M., Pointcheval, D., Fouque, P.-A., Vergnaud, D. (eds.) ACNS 2009. LNCS, vol. 5536, pp. 20–33. Springer, Heidelberg (2009)
17. Dhamija, R., Tygar, J.D., Hearst, M.A.: Why phishing works. In: Grinter, R.E., Rodden, T., Aoki, P.M., Cutrell, E., Jeffries, R., Olson, G.M. (eds.) CHI, pp. 581–590. ACM, New York (2006)
18. Diffie, W., Hellman, M.: New directions in cryptography. IEEE Transactions on Information Theory 22, 644–654 (1976)
19. Gajek, S., Jager, T., Manulis, M., Schwenk, J.: A browser-based kerberos authentication scheme. In: Jajodia, S., Lopez, J. (eds.) ESORICS 2008. LNCS, vol. 5283, pp. 115–129. Springer, Heidelberg (2008)
20. Gajek, S., Manulis, M., Pereira, O., Sadeghi, A.-R., Schwenk, J.: Universally composable security analysis of TLS. In: Baek, J., Bao, F., Chen, K., Lai, X. (eds.) ProvSec 2008. LNCS, vol. 5324, pp. 313–327. Springer, Heidelberg (2008)
21. Jager,T., Kohlar, F., Schäge, S., Schwenk, J.: Generic compilers for authenticated key exchange Full version. To appear on eprint (2010), http://eprint.iacr.org/
22. Katz, J., Yung, M.: Scalable protocols for authenticated group key exchange. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 110–125. Springer, Heidelberg (2003)
23. Krawczyk, H.: HMQV: A high-performance secure Diffie-Hellman protocol. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 546–566. Springer, Heidelberg (2005)
24. LaMacchia, B.A., Lauter, K., Mityagin, A.: Stronger security of authenticated key exchange. In: Susilo, W., Liu, J.K., Mu, Y. (eds.) ProvSec 2007. LNCS, vol. 4784, pp. 1–16. Springer, Heidelberg (2007)
25. Morrissey, P., Smart, N.P., Warinschi, B.: A modular security analysis of the TLS handshake protocol. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 55–73. Springer, Heidelberg (2008)
26. 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)
27. Shoup, V.: On formal models for secure key exchange. IBM Research Report RZ, 3120 (1999)
28. Shoup, V.: Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332 (November 2004)

A Forward-Secure Symmetric-Key Derivation Protocol

How to Improve Classical DUKPT

Eric Brier and Thomas Peyrin

Ingenico, France
{forename.name}@ingenico.com

Abstract. In this article, we study an interesting and very practical key management problem. A server shares a symmetric key with a client, whose memory is limited to R key registers. The client would like to send private messages using each time a new key derived from the original shared secret and identified with a public string sent together with the message. The server can only process N computations in order to retrieve the derived key corresponding to a given message. Finally, the algorithm must be forward-secure on the client side: even if the entire memory of the client has leaked, it should be impossible for an attacker to retrieve previously used communication keys. Given N and R, the total amount T of keys the system can handle should be as big as possible.

In practice such a forward-secure symmetric-key derivation protocol is very relevant, in particular in the payment industry where the clients are memory-constraint paying terminals and where distributing symmetric keys on field is a costly process. At the present time, one standard is widely deployed: the Derive Unique Key Per Transaction (DUKPT) scheme defined in ANSI X9.24. However, this algorithm is complicated to apprehend, not scalable and offers poor performances.

We provide here a new construction, *Optimal-DUKPT* (or O-DUKPT), that is not only simpler and more scalable, but also more efficient both in terms of client memory requirements and server computations when the total number of keys T is fixed. Finally, we also prove that our algorithm is optimal in regards to the client memory R / server computations N / number of keys T the system can handle.

keywords: key management, key derivation, DUKPT, forward-security.

1 Introduction

In information security, one of the most complicated part related to practical cryptography is the key management. Many different scenarios can exist and a different method is often required for each of them. The banking industry is well used to face strong constraints regarding key management. The financial transactions are processed in the paying terminal that reads the user magnetic card or chip card data. In order to validate the PIN input from the user, protocols based

M. Abe (Ed.): ASIACRYPT 2010, LNCS 6477, pp. 250–267, 2010.

on symmetric-key cryptography [5,6] are usually implemented. Also, because of the recent devastating attacks on back-office payment servers, the industry found strong incentives in protecting as well the user card data that is sent to the bank. Leading solutions [10,4], deploying format-preserving encryption, are also based on symmetric-key cryptography.

Of course, the symmetric key shared between the terminal and the bank must be securely distributed beforehand. In most cases, this is done by manually injecting the terminals with the key in a secure room. Clearly, such a process is costly for the stakeholders and can not be done frequently. For that reason, the symmetric keys stored in the terminal are static and not often changed.

Recently, the industry has seen the rising of side-channel attacks [8,9,13], practical and devastating methods that aims at recovering the secret key inside cryptographic modules by analyzing unusual information channels such as computation time, power consumption, electromagnetic emission, etc. As the paying terminals environment can not be considered as secure, side-channels attacks have to be taken in account seriously. For this reason, the banking industry has actively promoted an improved and more secure key management protocol: Derive Unique Key Per Transaction (DUKPT), defined in ANSI X9.24 [7].

The idea of DUKPT is to derive from the originally shared key a unique key per transaction. This feature greatly reduces the applicability of side-channel attacks, for which many measurement traces of encryption processes with the same symmetric key must be obtained. Moreover, this method is done is a forward-secure way on the client side (the servers are considered as located in a secure environment): if the internal state of the client is recovered, the attacker can not retrieve any of the transaction keys previously used. The algorithm can handle up to one million derived keys, which seems a reasonable upper bound for the number of transactions performed during a paying terminal life-cycle. Thus, the costly key injection process only has to be performed once.

DUKPT is standardized and now widely deployed in a majority of payment solutions. However, this protocol consumes a lot of memory in the devices, which are strongly memory-constraints. This is particularly problematic when a terminal has to be able to communicate with several distinct servers, and thus to handle many DUKPT instances at the same time. Moreover, DUKPT can also cause troubles on the server side, since it is costly in terms of computations required to retrieve the transaction key. Of course, this issue is even worsen by the fact that the server receives many financial transactions at the same time.

Our Contribution. In this article, we propose an improvement over the DUKPT technique described in ANSI X9.24 [7]. Our forward-secure symmetric-key derivation protocol offers scalability, simplicity and memory/computations performance gains. Yet the problem we study here is more general than just the sole case of paying terminals in the banking industry: memory-constraint clients that want to share with a computation-limited server unique symmetric keys per message sent in a forward-secure way. After having described our new proposal O-DUKPT,

we show that it is optimal in terms of client's memory requirements R / server computations N / number of keys T handled by the construction. Note that we restrict ourselves to using symmetric-key cryptography only. For forward-secure encryption schemes using public-key, see for example [3].

2 State-of-the-Art

2.1 Goals and Constraints

Derive Unique Key Per Transaction (DUKPT, see [7]) is a symmetric-key derivation scheme that offers several nice security properties for the payment industry scenarios (or any asymmetric situation where one server securely communicates with many memory-constrained clients). First, the symmetric keys used for each payment transaction are distinct. Moreover, a *forward-security* feature is incorporated: if the internal state of the paying terminal is entirely or partially compromised by any means, no useful information on the keys derived in previously processed transactions can be retrieved thanks to this leakage. Usually, the DUKPT scheme is utilized for the derivation of symmetric keys ciphering PIN blocks (for user authentication), or more recently for deriving symmetric keys that encrypts sensitive banking data such as Personal Account Number (PAN), expiration date, etc.

In practice, we have one server $\mathcal{S}$ that communicates with many paying terminals and each of these clients $\mathcal{C}_i$ must first share a symmetric key with $\mathcal{S}$. For obvious security reasons, two clients can not share the same key with the server (except by chance). This constraint could lead to memory problems for $\mathcal{S}$ if it has to deal with many clients. The issue is avoided by starting from a Base Derivation Key (BDK), directly available to $\mathcal{S}$. The k-bit key shared between a client $\mathcal{C}_i$ and the server $\mathcal{S}$ is denoted IK_i (for Initial Key) and is derived from the BDK as follows:

$$IK_i = F(BDK, i)$$

where $F : \{0,1\}^* \longmapsto \{0,1\}^k$ is a pseudo-random function. We give in Appendix A how F is obtained in practice. Thus, the system is initialized by giving BDK to the server, and IK_i to each client $\mathcal{C}_i$. This key distribution is not in the scope of this article, but in general it is done on the client side with manual key injection in secure room or with remote key injection using public-key cryptography. Note that when a transaction is sent from the client to the server, the identity of the client $\mathcal{C}_i$ has to be sent as well, so the server can appropriately derive the key originally shared IK_i. The initialization process is depicted in Figure 1. The problem studied in this paper can be directly reduced to the case of a single client. Therefore, from now on, we will only consider two entities: a server $\mathcal{S}$ and a client $\mathcal{C}$ that initially share a key IK.

We would like to derive the unique transaction keys in a forward-secure way, using only the function F as a black-box. There is a very natural and inefficient way of achieving this: the client $\mathcal{C}$ maintains an internal state of one key, initialized with IK. The internal state after having processed the j-th transaction is

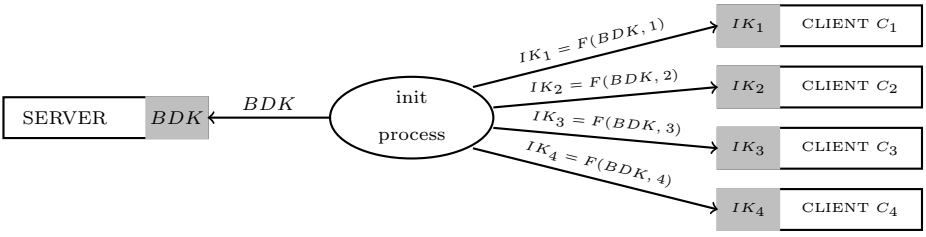


Fig. 1. Server and clients initialization

denoted S_j and we have $S_0 = IK$. Then, the key used for transaction j is simply the key stored in the internal state, S_{j-1}, and we update the internal state with $S_j = F(S_{j-1}, j)$. At each transaction, the client sends the value j so that the server can understand which key is shared for this transaction. It is clear that each key derived will be unique (except by chance) and that we have the forward-security property: F is a non-invertible process, so one can not obtain any information on the previous keys by recovering the internal state. However, it is also clear that this will be very inefficient on the server side. If one would like to handle say 1 million transactions, the server may have to go through one million computations of F to obtain the key in the worst case.

The idea of ANSI X9.24 DUKPT is to allow for the client to store more data than just one key, so as to lower the computation cost on the server side. More precisely, DUKPT allows the client to store $R = 21$ key registers and the server to compute F at maximum $N = 10$ times for one key derivation (except $IK_i = F(BDK, i)$). Overall, a total of $T = 1048575$ transactions can be handled.

In this paper we will show that DUKPT is not optimal when studying the following problem: given at maximum R key storage registers in one client $\mathcal{C}$ and N computations of F on the server $\mathcal{S}$ for one key derivation, what is the maximum number T of distinct keys the system can handle while ensuring the forward-security property if all the secret information contained in $\mathcal{C}$ is compromised? We provide an optimal algorithm, named *Optimal-DUKPT* (or O-DUKPT) that can handle up to $T = \binom{N+R}{R} - 1$ derived keys. For example, with the original parameters of DUKPT, $R = 21$ and $N = 10$, we are able to generate $T = 44352164$ keys. Otherwise, if the goal is to be able to derive about one million keys, one can use our solution with only $R = 13$ key registers. Our solution is not only more attractive in terms of memory and computations, but it is also much simpler to apprehend and to implement. Finally, *Optimal-DUKPT* is completely scalable and depending on the expected/desired memory/computation constraints of the system, it offers very valuable tradeoffs for practical applications.

2.2 ANSI X9.24 DUKPT Description

For completeness and future comparison, we provide in this section a description of the original DUKPT algorithm as defined in the ANSI X9.24-2009 document [7]. We assume that the shared symmetric key IK has been securely given

to the client and the server. We define $hw(x)$ as the hamming weight of the word x. The base 2 representation is denoted $(\cdot)_2$, i.e. 13 in base 2 is written $(1101)_2$, the most significant bit being located on the left. Also, for $x \neq 0$, we define $y = \tilde{x}$ to be the value of x with the least significant "1" bit set to zero. For example, if $x = (10110)_2$ we have $y = \tilde{x} = (10100)_2$ and $\tilde{y} = (10000)_2$.

In order to identify the key derived, for each transaction a 21-bit counter tc is sent from the client to the server. Then, for transaction j, the counter value $tc = j$ is sent and the key identified by j is used. The initial key IK is considered as a key identified by $j = 0$. DUKPT intrinsically defines a hierarchy between the keys: each key used for transaction $j \neq 0$ is the daughter of the key identified by $\tilde{j}$ (by A is the daughter of B, we mean that A is directly derived from B with F). For example, the key identified by $j = (0\ldots010000)_2$ has four daughters identified by $j_1 = (0\ldots010001)_2$, $j_2 = (0\ldots010010)_2$, $j_3 = (0\ldots010100)_2$ and $j_4 = (0\ldots011000)_2$, since $j = \tilde{j_1} = \tilde{j_2} = \tilde{j_3} = \tilde{j_4}$. More precisely, we have

$$K_j = F(K_{\tilde{j}}, j).$$

Before describing the process on the client and server side, one may ask why a 21-bit counter is needed (20 bits would suffice). The reason is that not all values of the counter and the corresponding keys will be used. Indeed, only the counter values with a non-zero hamming weight lower or equal to 10 will be considered and one can aim for a total key amount of

$$T = \sum_{j=1}^{10} \binom{21}{j} = 2^{20} - 1 = 1048575.$$

On the Server Side. $\mathcal{S}$ receives the 21-bit transaction counter tc. The server will derive the transaction key with only $hw(tc)$ computations of F (since we forced $hw(tc) \leq 10$, we do have the property that at maximum $N = 10$ computations of F are required). First, $\mathcal{S}$ deduces the bit position p_1 of the most significant "1" bit of tc and computes $c_{temp} = 2^{p_1}$ and $K_{temp} = F(IK, c_{temp})$. Then, the server deduces the bit position p_2 of the second most significant "1" bit of tc and computes $c_{temp} = c_{temp} + 2^{p_2}$ and $K_{temp} = F(K_{temp}, c_{temp})$. One continues until all the $hw(tc)$ "1" bits of tc have been processed. Then, the final key stored in K_{temp} is the shared key for this transaction. One can see that the server derivation simply consists in following the key hierarchy starting from IK and ending to K_{tc}. For example, if $tc = (0\ldots011010)_2$, the server first computes $K_{temp} = F(IK, (0\ldots010000)_2)$, then $K_{temp} = F(K_{temp}, (0\ldots011000)_2)$ and finally $K_{temp} = F(K_{temp}, (0\ldots011010)_2)$.

On the Client Side. The derivation on the client side is a little bit more complicated. First, the client is initialized as follows: each register r is filled with the value $F(IK, 2^{r-1})$ with $r \in \{1, \ldots, R\}$, i.e. each register r is filled with $K_{2^{r-1}}$. Then, IK is erased from the client's memory. One can note that those $R = 21$ keys are in fact the mothers of all future keys. For the first transaction, the key corresponding to $tc = 1$ is located in the first register (since the key stored in

Table 1. Chronological accesses to the key registers on the client side for DUKPT. A value i in a cell means that the key K_i is derived and stored in the corresponding column register at the corresponding row iteration. An "X" means that for this transaction the client used the key located in the corresponding register and then erased its contents.

counter tc		R_{21}	R_{20}	R_{19}		R_{12}	R_{11}		R_5	R_4	R_3	R_2	R_1
dec	hex												
init		2^{20}	2^{19}	2^{18}	...	2^{11}	2^{10}	...	16	8	4	2	1
1	1												X
2	2											X	3
3	3												X
4	4										X	6	5
5	5												X
6	6											X	7
7	7												X
8	8									X	12	10	9
9	9												X
10	A											X	11
11	B												X
12	C											14	13
13	D												X
14	E											X	15
15	F												X
16	10				...			...	X	24	20	18	17
17	11												X
						⋮							
2045	7FD												X
2046	7FE											X	
2048	800					X	3072		2064	2056	2052	2050	2049
2049	801												X
						⋮							
1047552	FFC00						X						
1048576	100000	X	$2^{20}+2^{19}$	$2^{20}+2^{18}$		$2^{20}+2^{11}$	$2^{20}+2^{10}$		$2^{20}+16$	$2^{20}+8$	$2^{20}+4$	$2^{20}+2$	$2^{20}+1$
1048577	100001												X
						⋮							
2095104	1FF800					X							

this register is $K_1 = F(IK, 1)$). Once the transaction completed, the content of this register is erased in order to preserve the forward-secrecy: only IK is the mother of K_1 and it has already been erased. Note that one can freely erase K_1 because it has no daughter, so one does not lose any important information for later derivation. Then, when $tc = 2$, one uses the content from register 2 as transaction key. However, since K_2 is the mother of K_3, before erasing it, one derives $K_3 = F(K_2, 3)$ and stores this key in register 1. One continues this process until all registers contain no more information.

To summarize, for a transaction j, the client picks and uses the key K_j located in register r, where r is the bit position of the least significant "1" bit of j. Then, before erasing K_j from its memory, the client derives and stores all the $r-1$ direct daughters of K_j in the $r-1$ least significant registers. The forward-secrecy is always maintained since after a transaction key have been used, it is always ensured that this key and its mother (or (grand)*-mothers) will no more be

present in the client's memory. Also, remember that all counter values with hamming weight strictly greater than 10 are skipped. We give in Table 1 an illustration of the chronological accesses to the key registers.

3 A New Proposal: O-DUKPT

3.1 General Idea

Our underlying idea for improving DUKPT can be understood with the following simple observation: for the very first transaction $tc = 1$ of DUKPT, the key K_1 located in the first register is used and directly erased. Note that this key has no daughter in the key hierarchy and that its mother is IK (it is at distance 1 from the IK). Said in other words, the server can retrieve K_1 from IK with only one computation of F. Instead of erasing K_1 directly and since we are yet far from reaching 10 computations of F on the server side, we could have derived another key from K_1 and placed it in this first register. Continuing this idea, we could have generated 9 more keys only with the first register.

Now, this can be generalized to the other registers as well. Once the first register contains a key located at a distance of 10 from the IK, we can not derive it anymore. Then, we have to utilize the key located in the second register, but before erasing it from the client's memory, we can derive from it two new keys that we will place in the first and second registers. Those two new keys are at distance 2 from the IK. Again, we can derive many keys only using the first register, but one less than before since we started from a key at distance 2 (and not 1) from the IK. This idea is finally iterated to all the registers.

3.2 Description

In order to preserve the scalability of the algorithm, Optimal-DUKPT will be defined as a family of key management schemes. Each member of the family is identified by the amount R of key registers available on the client side and the number N of maximum computations required to derive one key on the server side. Moreover, we will show later that each member can handle a maximum number of keys $T = \binom{R+N}{N} - 1$.

As for the original DUKPT, we assume that the shared symmetric key IK has been securely given to the client and the server. In order to identify the key derived, for each transaction a public string st is sent from the client to the server. This string is composed of R integers st_i, with $1 \leq st_i \leq N$ for $1 \leq i \leq R$. An integer st_i represents the distance from IK of the key stored in register i of the client's memory before processing the transaction. For example, the string sent for the very first transaction is $1 \ldots 1\,1$, for the second one $1 \ldots 1\,2$, etc.

On the Client Side. The client maintains two tables. First, the classical R key registers, denoted R_i for $1 \leq i \leq R$. They are simply initialized with $R_i = F(IK, 0^R, i)$ and once this process is over, the IK is erased from the client's memory. Secondly, the client maintains a table D of R integers that

we denote D_i, where D_i represents the distance from IK of the key stored in register R_i. The content of D is exactly what is sent to the server in the string st. Naturally, it is initialized with $D_i = 1$ for $1 \leq i \leq R$.

When requested to process a new transaction, the client builds $st = D$ and looks for the least significant register having a corresponding distance D_i strictly smaller than $N + 1$. This register, that we denote R_p, contains the transaction key TK that will be used. Then, once the transaction over,

- If $D_p < N$ (i.e. more keys can be derived from TK), the client updates the p registers R_p, R_{p-1}, ..., R_1 with $R_i = F(TK, D, i)$ and updates the distance table with $D_i = D_p + 1$ with $1 \leq i \leq p$.
- Otherwise, if $D_p = N$ (i.e. TK does not have any daughter), the client simply erases the content of register R_p and updates $D_p = D_p + 1 = N + 1$.

Note that in the key derivation process, the data used as input of F is always unique. Indeed, D will be different for each transaction. This guarantees the security of the system. The forward-secrecy is always maintained since after a transaction key has been used, it is always ensured that this key (and its predecessors) will no more be present in the client's memory. We give an example of the clients internal state evolution in Table 2 or an alternate tree view in Figure 2.

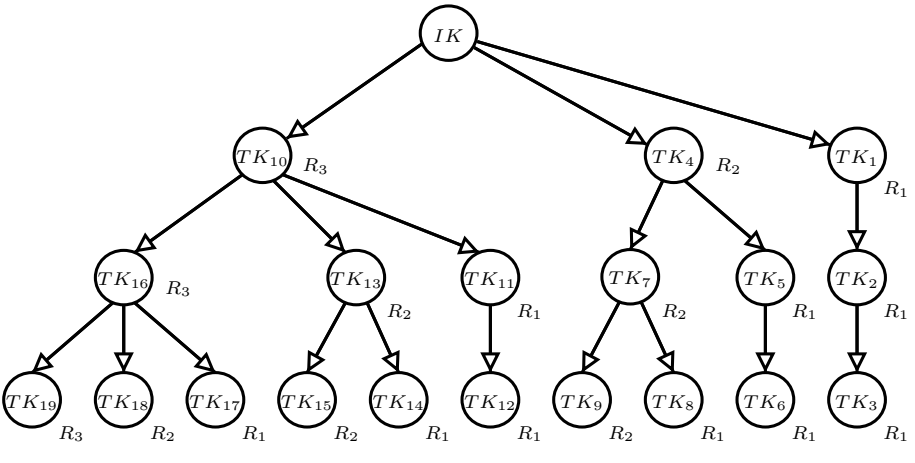


Fig. 2. Tree view of the client side key derivation example from Table 2, with system parameters $N = 3$ and $R = 3$. We denote TK_j the key used for the j-th iteration, and the R_i aside the circles represent the register in which each key TK_i is stored during the process.

Table 2. Example of key registers and distance tables evolution on the client side, with system parameters $N = 3$ and $R = 3$. We denote TK_i the key used for the i-th iteration. An "X" in the key registers evolution columns means that the client erases the contents from this register.

iter.	*st* sent	transaction key used and key registers update	key registers evolution				distance table		
			R_3	R_2	R_1		D_3	D_2	D_1
init		$R_3 = F(IK, 000, 3)$	TK_{10}	TK_4	TK_1	in	0	0	0
		$R_2 = F(IK, 000, 2)$, $R_1 = F(IK, 000, 1)$				out	1	1	1
1	111	$TK_1 = R_1$			TK_2	in	1	1	1
		$R_1 = F(TK_1, 111, 1)$				out	1	1	2
2	112	$TK_2 = R_1$			TK_3	in	1	1	2
		$R_1 = F(TK_2, 112, 1)$				out	1	1	3
3	113	$TK_3 = R_1$			X	in	1	1	3
		erase R_1				out	1	1	4
4	114	$TK_4 = R_2$		TK_7	TK_5	in	1	1	4
		$R_2 = F(TK_4, 114, 2)$, $R_1 = F(TK_4, 114, 1)$				out	1	2	2
5	122	$TK_5 = R_1$			TK_6	in	1	2	2
		$R_1 = F(TK_5, 122, 1)$				out	1	2	3
6	123	$TK_6 = R_1$			X	in	1	2	3
		erase R_1				out	1	2	4
7	124	$TK_7 = R_2$		TK_9	TK_8	in	1	2	4
		$R_2 = F(TK_7, 124, 2)$, $R_1 = F(TK_7, 124, 1)$				out	1	3	3
8	133	$TK_8 = R_1$			X	in	1	3	3
		erase R_1				out	1	3	4
9	134	$TK_9 = R_2$		X		in	1	3	4
		erase R_2				out	1	4	4
10	144	$TK_{10} = R_3$, $R_3 = F(TK_{10}, 144, 3)$	TK_{16}	TK_{13}	TK_{11}	in	1	4	4
		$R_2 = F(TK_{10}, 144, 2)$, $R_1 = F(TK_{10}, 144, 1)$				out	2	2	2
11	222	$TK_{11} = R_1$			TK_{12}	in	2	2	2
		$R_1 = F(TK_{11}, 222, 1)$				out	2	2	3
12	223	$TK_{12} = R_1$			X	in	2	2	3
		erase R_1				out	2	2	4
13	224	$TK_{13} = R_2$		TK_{15}	TK_{14}	in	2	2	4
		$R_2 = F(TK_{13}, 224, 2)$, $R_1 = F(TK_{13}, 224, 1)$				out	2	3	3
14	233	$TK_{14} = R_1$			X	in	2	3	3
		erase R_1				out	2	3	4
15	234	$TK_{15} = R_2$		X		in	2	3	4
		erase R_2				out	2	4	4
16	244	$TK_{16} = R_3$, $R_3 = F(TK_{16}, 244, 3)$	TK_{19}	TK_{18}	TK_{17}	in	2	4	4
		$R_2 = F(TK_{16}, 244, 2)$, $R_1 = F(TK_{16}, 244, 1)$				out	3	3	3
17	333	$TK_{17} = R_1$			X	in	3	3	3
		erase R_1				out	3	3	4
18	334	$TK_{18} = R_2$		X		in	3	3	4
		erase R_2				out	3	4	4
19	344	$TK_{19} = R_3$	X			in	3	4	4
		erase R_3				out	4	4	4

On the Server Side. The server receives a string st that corresponds to the table D of the client before processing the transaction. Note that the distance values memorized in this table are always increasing from most significant to least significant registers. Moreover, we recall that when the client extracted a transaction key from a register R_p, it means that the distance table was such that $D_i = N + 1$ for $1 \leq i \leq p - 1$. We denote by $g_{p,v}(D)$ the transformation that maps the distance table D to another distance table D' with

$$\begin{cases} D'_i = N + 1, & \text{for } 1 \leq i \leq p - 1 \\ D'_i = \quad v, & \text{for } i = p \\ D'_i = \quad D_i, & \text{for } p + 1 \leq i \leq R \end{cases}$$

The server first initializes a local distance value $d = 1$ and a register position value $p = p'$, with p' being the most significant position with $D_{p'} > 1$. Then, he computes $K = F(IK, 0^R, p')$ and keeps repeating the following process:

- While $d < D_p - 1$, compute $K = F(K, g_{p,d}(D), p)$ and $d = d + 1$.
- If $p = 1$, then $K = F(K, g_{p,d}(D), p)$ is the key shared with the client so the program can stop.
- The server finds the most significant position p' such that $D_{p'} > D_p$. If $D_{p'} \neq N+1$, then he computes $K = F(K, g_{p,d}(D), p')$ and updates the local variables $p = p'$ and $d = d + 1$. Otherwise, $K = F(K, g_{p,d}(D), p' + 1)$ is the key shared with the client so the program can stop.

This algorithm exactly follows the implicit process performed by the client to derive the transaction key TK from the initial key IK. For example, reusing the

Table 3. Example of key derivation on the server side, with system parameters $N = 8$ and $R = 8$ and $st = \texttt{12466689}$. The key is derived with 8 server computations.

iter.	key update		local values d	p	p'
init	$K = F(IK, \texttt{00000000}, 7)$	in			7
		out	1	7	
1	$K = F(K, \texttt{11999999}, 6)$	in	1	7	6
		out	2	6	
2	$K = F(K, \texttt{12299999}, 6)$	in	2	6	5
	$K = F(K, \texttt{12399999}, 5)$	out	4	5	
3	$K = F(K, \texttt{12449999}, 5)$	in	4	5	2
	$K = F(K, \texttt{12459999}, 2)$	out	6	2	
4	$K = F(K, \texttt{12466669}, 2)$	in	6	2	1
	$K = F(K, \texttt{12466679}, 2)$	out			

scenario from Table 2, assume that the server receives $st = 224$. He will first set $d = 1$, $p = 3$ and compute $K = F(IK, 000, 3)$. Then, he does not enter the while loop nor the first if. He computes $p' = 1$ and since $D_{p'} = 4 = N + 1$, the key $K = F(K, 144, 2)$ is the key shared with the client. We give in Table 3 a more complicated example.

3.3 Performance Analysis

Now that we defined our construction, we would like to precisely compute its theoretical performance. It is clear that the client only needs to maintain a table of R registers and that the maximum number of computations required on the server side to derive the key is at most N calls to the function F. What is the number T of keys the system can support (note that IK is not counted as transaction key)? Since T depends on N and R, we denote $T(N, R)$ the maximum number of keys that can be generated with R client's registers and N server computations.

One can be easily convinced that $T(N, 1) = N$. Indeed, with only a single register R_1, the derivation process will simply use and self-update the key stored in R_1 until it exceeds the maximal distance N. Also, we have $T(1, R) = R$ since with a maximal allowable distance of 1, our construction would simply fill each of the R registers with a key directly derived from the IK and that has no daughter.

Let $t(n, r)$ denote the number of distinct keys stored by register r with a distance n from IK during the entire course of the algorithm, with $1 \leq n \leq N$ and $1 \leq r \leq R$. Since the registers are ordered (a register r can only be updated by a register r' with $r' \geq r$), we deduce that $t(n, R) = 1$, because the most significant register can only be updated by itself. It is also clear that $t(1, r) = 1$, since the only keys at distance 1 are the very first keys stored in each register.

The only way for a register r to hold a key at distance $n > 1$ is to be updated from a key at distance $n - 1$ stored in a register $r' \geq r$. Thus, for $2 \leq n \leq N$ and $1 \leq r \leq R$, we have

$$t(n, r) = \sum_{i=r}^{R} t(n-1, i)$$

which simplifies to

$$t(n, r) = t(n-1, r) + \sum_{i=r+1}^{R} t(n-1, i) = t(n-1, r) + t(n, r+1).$$

We define the function $g(n, r) = \binom{n+R-r-1}{R-r}$ and it is well known that

$$\binom{a}{b} = \binom{a-1}{b} + \binom{a-1}{b-1} \tag{1}$$

$$\sum_{i=b}^{a} \binom{i}{b} = \binom{a+1}{b+1} \tag{2}$$

where (2) is derived by induction from (1). Thus, using (1) we obtain

$$g(n,r) = \binom{n+R-r-2}{R-r} + \binom{n+R-r-2}{R-r-1} = g(n-1,r) + g(n,r+1).$$

Since we have $g(n,R) = 1$ and $g(1,r) = 1$, we conclude that $t(n,r) = g(n,r)$ for $1 \leq n \leq N$ and $1 \leq r \leq R$.

The total amount of key handled by the system is computed by

$$T(N,R) = \sum_{i=1}^{N}\sum_{j=1}^{R} t(i,j) = \sum_{i=1}^{N}\sum_{j=1}^{R}\binom{i+R-j-1}{R-j} = \sum_{i=0}^{N-1}\sum_{j=0}^{R-1}\binom{i+j}{j}$$

Finally, using identities (1) and (2) we obtain

$$\begin{aligned} T(N,R) &= \sum_{i=0}^{N-1}\sum_{j=0}^{R-1}\binom{i+j}{j} = \sum_{i=0}^{N-1}\sum_{j=0}^{R-1}\binom{i+j}{i} = \sum_{i=0}^{N-1}\binom{i+R}{i+1} = \sum_{i=0}^{N-1}\binom{i+R}{R-1} \\ &= -1 + \sum_{i=-1}^{N-1}\binom{i+R}{R-1} = \binom{R+N}{N} - 1 = \binom{R+N}{R} - 1 \end{aligned}$$

3.4 Optimality Proof

In this section we show that when N and R are fixed, the amount of keys T handled by our algorithm is the maximal reachable value. First, note that we do not count the initial key in this amount. Indeed, as potentially sensitive in practice, we consider that the initial key must not be stored on the client side after the initialization process has been performed. Of course, if needed, our algorithm O-DUKPT can be trivially modified in order to also utilize IK as one of the transaction keys: the initialization process only stores the IK in one of the register, then the first transaction key is this initial key and the first registers update simply performs the initialization process from O-DUKPT.

We assume that the server and the client can only call the non-invertible function F to derive the keys, in a black-box manner. This pseudo-random function manipulates an input of arbitrary length and outputs a key.

After the initialization phase and during the protocol run, the client can only store R keys in its memory. We do not count here the temporary memory required for the key derivations. Those R registers represent the number of keys that can be memorized in the client's memory once the transaction is over. Once the transaction key has been deduced by the client, he processes the transaction with this key and sends the public data st to the server. Once st received, the server can only use a maximum of N calls of F in order to derive the transaction key from the initial key IK.

The key generation algorithm must be forward-secure on the client side. That is, when a transaction has been performed, it must be impossible for an attacker that just recovered the R internal registers to retrieve any transaction

key previously utilized. We call such an algorithm a *forward-secure DUKPT algorithm* and we denote by T the maximum number of distinct keys the system can handle.

At each transaction i, the client can first do some computations from the R registers contents to deduce the transaction key TK_i, then he stores TK_i in its local memory in order to use it, and he finally updates its R internal registers. Because F is as pseudo-random function, there is no need to use several keys on its input. One can generate as many distinct outputs as needed from F with a single key input by using distinct additional data (such as a counter for example). Thus, when deriving keys or transaction keys with F, we can assume that only one key is used on its input.

Now, since we would like to preserve the forward-security on its side (and since TK_i only depends on one key from the R registers), the client model can be simplified: at each transaction, he picks the transaction key TK from one of its internal registers R_i, he stores it in its local memory and finally updates the R registers (i.e. the computation phase can be merged with the update). Moreover, since the forward-security forces only the TK to be erased, the client only uses this key for derivation (there is no advantage in doing a derivation from a key that we do not have to erase yet). Therefore, for each transaction, the client picks the transaction key TK from one of its internal registers R_i, stores it in its local memory and finally updates the R registers exclusively from it.

When studying theoretically the optimality of a DUKPT algorithm, there is no need to consider the server behavior. Indeed, the only requirement for the server is that it must be able to compute the transactions with at most N calls to F. Since F is as pseudo-random function, this only depends on how the client generated the transaction keys. This constraint is modeled on the client side with a distance value assigned to each key, representing the number of calls to F required to reach this key from IK. Obviously, no transaction key can have a distance strictly greater than N (and it is useless to memorize any key with a distance strictly greater than N).

Theorem 1. *A forward-secure DUKPT algorithm with R client registers and N maximal server computations can derive at most T distinct keys, with*

$$T = \binom{R+N}{N} - 1 = \binom{R+N}{R} - 1.$$

Let $\mathcal{A}$ be an optimal algorithm, i.e. reaching the maximum value T of keys handled. We prove this theorem with several very simple lemmas concerning $\mathcal{A}$.

Lemma 1. *After the initialization process of $\mathcal{A}$, the R registers of the client are filled with R new distinct keys.*

Proof. It is clear that not filling all R registers during the initialization phase is not an optimal method. Let $\mathcal{B}$ be such a forward-secure DUKPT algorithm. We can trivially build another forward-secure DUKPT algorithm $\mathcal{B}'$ that generates strictly more keys than $\mathcal{B}$: during the initialization phase, $\mathcal{B}'$ memorizes one more

key derived from IK in one of the registers left blank by $\mathcal{B}$. It uses this key for the very first transaction and erases the contents of the corresponding register. Once this key used and erased from the client's memory, $\mathcal{B}'$ behaves identically as $\mathcal{B}$. Overall, one more key has been generated in the process. □

Lemma 2. *When $\mathcal{A}$ derives keys on the client side during the registers update, it only memorizes newly derived keys in empty registers.*

Proof. Indeed, let $\mathcal{B}$ be a forward-secure DUKPT algorithm that memorizes a newly derived key in a non empty register during one transaction. We can trivially build another forward-secure DUKPT algorithm $\mathcal{B}'$ that generates strictly more keys than $\mathcal{B}$: $\mathcal{B}'$ behaves identically as $\mathcal{B}$, but instead of directly erasing this particular register, it first uses the key stored in it and erases the register contents once the transaction is over. Overall, one more key has been generated in the process. □

Lemma 3. *When $\mathcal{A}$ derives keys on the client side during the registers update, all previously empty registers are filled at the end of the process.*

Proof. Let $\mathcal{B}$ be a forward-secure DUKPT algorithm that does not fill all empty registers when deriving new keys during one transaction. We can trivially build another forward-secure DUKPT algorithm $\mathcal{B}'$ that generates strictly more keys than $\mathcal{B}$: $\mathcal{B}'$ behaves identically as $\mathcal{B}$, but instead fills one of the empty register that $\mathcal{B}$ left blank with a new distinct key K (this is possible since we already assumed that $\mathcal{B}$ possess some key content to derive from at this moment). Then, during the next transaction, $\mathcal{B}'$ will use K, erase it and finally continue as $\mathcal{B}'$ in the previous transaction. Overall, one more key has been generated in the process. □

The direct corollary of the two last lemmas is that the update derives keys in every empty register only.

Lemma 4. *The transaction key chosen by $\mathcal{A}$ is always one of the keys at the maximal available distance from IK (different from $N+1$).*

Proof. Let $\mathcal{B}$ be a forward-secure DUKPT algorithm that extracts a transaction key TK from a register R_i containing a key at distance $d < d_{max}$ from the IK, where d_{max} denotes the maximal distance available among the R registers. From previous lemmas we know that, after erasure of R_i, all empty registers must be filled with keys derived from TK. We can trivially build another forward-secure DUKPT algorithm $\mathcal{B}'$ that generates strictly more keys than $\mathcal{B}$: $\mathcal{B}'$ behaves identically as $\mathcal{B}$, but instead does one more transaction. First $\mathcal{B}'$ extracts a transaction key TK_+ among the set of registers containing keys located at distance d_{max} from IK. We denote by R_+ this register. Then, the update simply consists in erasing R_+. For the next iteration, $\mathcal{B}'$ extracts the transaction key TK from R_i and performs the update exactly as $\mathcal{B}$. The only difference for $\mathcal{B}'$ is that R_+ will be updated as well, because it is now empty. The update is done with TK, located at distance $d < d_{max}$: we make $(d_{max} - d)$ calls to F to perform the

derivation, so that R_+ finally contains a key at distance d_{max}. Thus, at this point, $\mathcal{B}'$ generated one more key (i.e. TK^+) than $\mathcal{B}$, while reaching the same distance table situation (since R_+ has distance d_{max} for both $\mathcal{B}$ and $\mathcal{B}'$). □

We showed than an optimal algorithm $\mathcal{A}$ must fulfill several properties stated in the previous lemmas. Namely, the initialization phase must fill all the R client's registers. Then, for each transaction, the client must use (and erase) one of the keys stored with the maximal distance from IK and derive exclusively from it distinct keys that will be stored in each and every empty register only. This already almost completely specifies what is an optimal algorithm. The only freedom remaining concerns which key is picked when several have the same maximal distance from IK and this has absolutely no incidence on the maximum number T of keys one can generate. Thus, all algorithms verifying the lemmas are equivalent and optimal. Since our proposal O-DUKPT does fulfill those conditions, we can conclude that we reach the optimal value of T:

$$T = \binom{R+N}{N} - 1 = \binom{R+N}{R} - 1.$$

4 Discussions

Knowing the maximum number of computations N on the server is a good guarantee of minimal performance (note that the maximal number of computations on the client side is equivalent for both algorithms: $R-1$ for DUKPT and R for O-DUKPT.). However, one could also estimate the average number of computations and for this we need to know the relative amount of keys at distance i from the IK. Let $A(i)$ represent the number of keys at distance i. Of course, we have $T = \sum_{i=1}^{N} A(i)$. The more keys we have at a large distance, the bigger will be the average number of computations per key on the server side. The average number of computations on the server side is

$$C_S = \left(\sum_{i=1}^{N} i.A(i) \right) / T$$

and on the client side it is

$$C_C = \left(\sum_{i=1}^{N} A(i) \right) / T = 1.$$

In the case of O-DUKPT, we have

$$A(i) = \sum_{j=1}^{R} t(i,j) = \sum_{j=1}^{R} \binom{i+R+-j-1}{R-j} = \binom{R+i-1}{i}$$

and for classical DUKPT, we have $A(i) = \binom{21}{i}$, with $i \leq 10$.

Table 4. Performance comparison between DUKPT (parameters $R = 21/N = 10$) and O-DUKPT (for parameters $R = 21/N = 7$, $R = 13/N = 10$ and $R = 17/N = 8$)

	DUKPT ($R = 21$, $N = 10$)	O-DUKPT ($R = 21$, $N = 7$)	O-DUKPT ($R = 13$, $N = 10$)	O-DUKPT ($R = 17$, $N = 8$)
T	1048575	1184039	1144065	1081574
$A(1)/T$	$2^{-15.6}$	$2^{-15.8}$	$2^{-16.4}$	$2^{-16.0}$
$A(2)/T$	$2^{-12.3}$	$2^{-12.3}$	$2^{-13.6}$	$2^{-12.8}$
$A(3)/T$	$2^{-9.6}$	$2^{-9.4}$	$2^{-11.3}$	$2^{-10.1}$
$A(4)/T$	$2^{-7.4}$	$2^{-6.8}$	$2^{-9.3}$	$2^{-7.8}$
$A(5)/T$	$2^{-5.7}$	$2^{-4.5}$	$2^{-7.5}$	$2^{-5.7}$
$A(6)/T$	$2^{-4.3}$	$2^{-2.4}$	$2^{-5.9}$	$2^{-3.9}$
$A(7)/T$	$2^{-3.2}$	$2^{-0.4}$	$2^{-4.5}$	$2^{-2.1}$
$A(8)/T$	$2^{-2.4}$		$2^{-3.2}$	$2^{-0.6}$
$A(9)/T$	$2^{-1.8}$		$2^{-2.0}$	
$A(10)/T$	$2^{-1.6}$		$2^{-0.8}$	
C_S	8.65	6.68	9.28	7.56

As a comparison with classical DUKPT, if we use the same amount of registers $R = 21$ for O-DUKPT, we only need to do at maximum $N = 7$ computations to handle an equivalent number of keys: $\binom{21+7}{21} - 1 = 1184039$. If we allow the same amount of maximum computations $N = 10$ for O-DUKPT, then we only need to maintain $R = 13$ key registers to handle an equivalent number of keys: $\binom{13+10}{13} - 1 = 1144065$. The Table 4 gives the numerical application for A_i, C_S and C_C. Thus, the performance improvement is twofold: for T and R fixed, not only O-DUKPT has a lower maximum number of computations on server side, but the average number of computations is also lower. Finally, we give an example for which O-DUKPT provides better results in regards to every performance aspects ($R = 17$ and $N = 8$ gives $T = 1081574$ and $C_S = 7.56$).

Variants. We proved in a previous section the optimality of our algorithm. However, one may derive variants concerning its implementation and most specifically concerning how the client communicates the identity of the key to the server and how the server processes its corresponding key derivation.

Our O-DUKPT implementation proposed requires the client to send a string st of R integers in $[1, \ldots, N]$. This could be coded on $log_2((N-1)^R)$ bits. The algorithm is simple to understand and implement, but it is not optimal in terms of message size since $T < (N-1)^R$. For example, classical DUKPT requires to send a 21-bit counter, while O-DUKPT (with parameters $R = 17$, $N = 8$) requires 48 bits.

One can think of several variants if message size is an issue in practice. For example, a very easy way to lower the message size is to leverage the memory available at the server side. For example, instead of sending the D table before

processing the transaction, the client could simply send the transaction counter (thus coded on $log_2(T)$ bits, the smallest possible message size). The server would have to recover the corresponding table D from the transaction counter received. This could be done very simply by a table lookup. This table, filled during initialization of the system, would require $T.log_2((N-1)^R)$ bits (roughly 5MB for O-DUKPT with parameters $R = 17$ and $N = 8$).

Acknowledgments

The authors would like to thank the ASIACRYPT 2010 committee for their helpful comments and for pointing us the reference [3].

References

1. Bellare, M.: New Proofs for NMAC and HMAC: Security Without Collision-Resistance. Cryptology ePrint Archive, Report 2006/043 (2006), http://eprint.iacr.org/
2. Bellare, M., Kilian, J., Rogaway, P.: The Security of Cipher Block Chaining. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 341–355. Springer, Heidelberg (1994)
3. Canetti, R., Halevi, S., Katz, J.: A Forward-Secure Public-Key Encryption Scheme. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 255–271. Springer, Heidelberg (2003)
4. Brier, E., Peyrin, T., Stern, J.: BPS: a Format Preserving Encryption Proposal. NIST submission (April 2010), http://csrc.nist.gov/groups/ST/toolkit/BCM/modes_development.html
5. American National Standards Institute. ISO 9564-1:2002 Banking – Personal Identification Number (PIN) management and security – Part 1: Basic principles and requirements for online PIN handling in ATM and POS systems (2002)
6. American National Standards Institute. ANSI X9.8-1:2003 Banking - Personal Identification Number Management and Security - Part 1: PIN protection principles and techniques for online PIN verification in ATM and POS systems (2003)
7. American National Standards Institute. ANSI X9.24-1:2009 Retail Financial Services Symmetric Key Management Part 1: Using Symmetric Techniques (2009)
8. 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)
9. Kocher, P.C., Jaffe, J., Jun, B.: Differential Power Analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
10. Bellare, M., Rogaway, P., Spies, T.: Format-preserving Feistel-based Encryption Mode. NIST submission (April 2010), http://csrc.nist.gov/groups/ST/toolkit/BCM/modes_development.html
11. NIST. FIPS 198 – The Keyed-Hash Message Authentication Code, HMAC (2002)
12. National Institute of Standards and Technology. SP800-67: Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher (May 2004), http://csrc.nist.gov
13. Quisquater, J.-J., Samyde, D.: ElectroMagnetic Analysis (EMA): Measures and Counter-Measures for Smart Cards. In: E-smart, pp. 200–210 (2001)

Appendix A: How to Instantiate F in Practice?

In ANSI X9.24, the DUKPT implementation described is intended to derive 112-bit TDES keys [12] (128-bit keys with 8 bits of parity checks). The F function is therefore itself based on TDES. A 128-bit incoming key K (the first input) is divided into two 64-bit parts K_L and K_R, and the new key $K' = K'_L||K'_R$ is derived with

$$K'_L = \texttt{TDES}_{K_L}(C \oplus K_R) \oplus K_R$$
$$K'_R = \texttt{TDES}_{K_L}(C' \oplus C \oplus K_R) \oplus K_R$$

where C is a known value depending on the counter (the second input), and C' is a fixed constant. The parity bits of K'_L and K'_R are then appropriately adjusted.

As F function, we advice to use commonly deployed MAC algorithms such as CBC-MAC [2] or HMAC [11,1] with the incoming key as MAC key and transaction related input as MAC message input.

Efficient String-Commitment from Weak Bit-Commitment

Kai-Min Chung[1,⋆], Feng-Hao Liu[2,⋆⋆], Chi-Jen Lu[3], and Bo-Yin Yang[3]

[1] School of Engineering & Applied Sciences, Harvard University, Cambridge MA, USA
kmchung@fas.harvard.edu
[2] Department of Computer Science, Brown University, Providence RI, USA
fenghao@cs.brown.edu
[3] Institute of Information Science, Academia Sinica, Taipei, Taiwan
cjlu@iis.sinica.edu.tw, by@crypto.tw

Abstract. We study security amplification for commitment schemes and improve the efficiency of black-box security amplification in the computational setting, where the security holds against PPT active adversaries. We show that $\omega(\log s)$ black-box calls to a weak bit-commitment scheme with constant security is sufficient to construct a commitment scheme with standard negligible security, where s denotes the security parameter and $\omega(\log s)$ denotes any super-logarithmic function of s. Furthermore, the resulting scheme is a string commitment scheme that can commit to $O(\log s)$-bit strings. This improves on previous work of Damgård et al. [DKS99] and Halevi and Rabin [HR08], whose transformations require $\omega(\log^2 s)$ black-box calls to commit a single bit.

As a byproduct of our analysis, we also improve the efficiency of security amplification for message authentication codes, digital signatures, and pseudorandom functions studied in [DIJK09]. This is from an improvement of the "Chernoff-type Theorems" of dynamic weakly-verifiable puzzles of [DIJK09].

1 Introduction

1.1 Security Amplification for Commitment Schemes

Security amplification for weak cryptographic primitives is a basic question that has been studied since the seminal work of Yao [Yao82]. This question has been extensively studied in recent years for a variety of primitives in various settings. To name a few, amplification has been studied for encryption schemes [DNR04, HR05], commitment schemes [DKS99, Wul07, HR08], oblivious transfer [DKS99, Wul07], message authentication codes (MACs), digital signatures, and pseudorandom functions (PRFs) [DIJK09]. Some of these works consider information-theoretic security (e.g., [DKS99]), and others consider computational security. The

⋆ Supported by US-Israel BSF grant 2006060 and NSF grant CNS-0831289.
⋆⋆ Supported by NSF grant CNS-0347661 and CNS-0831293.

M. Abe (Ed.): ASIACRYPT 2010, LNCS 6477, pp. 268–282, 2010.

various security properties of primitives present different interactive settings, for example, commitment schemes are more interactive than encryption schemes, and the chosen-message-attack for MACs introduces a different type of interaction. Proving amplification results tend to be more challenging in an interactive and computational setting.

In this paper, we continue the study of security amplification for commitment schemes, which was previously studied in [DKS99, Wul07, HR08]. We focus on black-box security amplification in the computational setting, where the security holds against probabilistic polynomial time (PPT) active adversaries. Namely, the starting point is a (weak) bit-commitment scheme Com_0 that is *p-hiding* in the sense that no PPT adversarial receiver, who may deviate from the prescribed protocol arbitrarily, can guess the committed bit correctly with probability better than $(1 + p)/2$, and *q-binding* in the sense that no PPT adversarial sender can open in two ways with probability better than q, and the goal is to transform Com_0 to a *secure* commitment scheme Com that makes black-box calls to Com_0 and achieves negligible security for both properties.

Previous works focus on feasibility results. Namely, for what values of p and q is the security amplification achievable. In the information-theoretic setting (i.e., the security holds for unbounded adversaries), Damgård, Kilian and Salvail [DKS99] showed that a black-box transformation is possible if and only if $p + q \leq 1 - 1/\mathrm{poly}(s)$, where s is the security parameter. Halevi and Rabin [HR08] analyzed the transformation of [DKS99] in the computational setting and proved that a black-box transformation is possible whenever $p+q \leq 1-1/\mathrm{polylog}(s)$. Recently and independent of our work, Holenstein and Schoenebeck [HS10] improved the result to optimal. They showed that in the computational setting, black-box security amplification is achievable if and only if $p + q \leq 1 - 1/\mathrm{poly}(s)$.

However, the existing transformations are not very efficient. To measure the efficiency, let us consider the number of black-box calls to Com_0 that Com makes when p and q are constants with $p + q < 1$. Note that the number of black-box calls affects not only the communication complexity, but also the round complexity of the resulting protocol, because in the computational setting, each black-box call needs to be done sequentially.[1] All existing solutions requires $\omega(\log^2 s)$ black-box calls to securely commit a single bit. At a high level, the reason is that they amplify the hiding and binding property *separately*. Amplifying each property from constant to negligible seems to require $\omega(\log s)$ black-box calls, which is the case of the existing constructions and results in $\omega(\log^2 s)$ black-box calls in total. On the other hand, the existing constructions give bit commitment schemes, but there are applications that require *string* commitment schemes. Since it requires $\omega(\log s)$ black-box calls to amplify the security anyway, perhaps we can obtain a string commitment scheme instead of just committing to a single bit, which

[1] In general, the commit stage of can consist of multiple rounds. If the black-box calls are done in parallel, one can show by modifying the counter example of Bellare, Impagliazzo, and Naor [BIN97] for interactive arguments that the security may not be amplified at all.

also improves efficiency in terms of the *rate*, i.e., the number of black-box calls per committed bit. These motivate us to ask the following question.

> **Main question:** How many black-box calls does it require to amplify a (weak) bit commitment scheme with constant security to one with negligible security? What is the length of the string that the resulting Com can commit to, and what is the achievable rate?

Our Results. We give a transformation that amplify a (weak) bit commitment scheme with constant security to a $O(\log s)$-bit string commitment scheme with negligible security using only $\omega(\log s)$ black-box calls, where $O(\log s)$ (resp., $\omega(\log s)$) denotes any $O(\log s)$ (resp., $\omega(\log s)$) function of the security parameter s. In terms of rate, we achieve $\omega(1)$ black-box calls per committed bit. A summary of our result and existing results can be found in Figure 1.

	Efficiency (constants p, q)			Feasibility
Work	Number of black-box calls	Length of committed string	Rate	Applicable range of parameters
[HR08]	$\omega(\log^2 s)$	1	$\omega(\log^2 s)$	$p + q < 1 - 1/\text{poly}\log(s)$
[HS10]	$\omega(\log^2 s)$	1	$\omega(\log^2 s)$	$p + q < 1 - 1/\text{poly}(s)$
Ours	$\omega(\log s)$	$O(\log s)$	$\omega(1)$	$p + q < 1 - 1/\text{poly}\log(s)$
Ours + [HS10]	$\omega(\log s)$	$O(\log s)$	$\omega(1)$	$p + q < 1 - 1/\text{poly}(s)$

Fig. 1. Summary of results on security amplification for commitment schemes in the computational setting. Efficiency measures the cost of amplifying commitment schemes from constant security to negligible security. Feasibility refers to the parameter range that security amplification is possible.

To bypass the $\omega(\log^2 s)$ barrier of the previous transformations, we use error-correcting codes and randomness extractors to amplify both hiding and binding properties *simultaneously*. To analyze our construction, we model the security of commitment schemes as (the hardness of) solving "two-phase" (interactive) puzzle systems, and study the hardness of solving at least r out of n puzzles. Our result on puzzle systems also applies to the dynamic weakly-verifiable puzzle systems of [DIJK09], and hence improves the efficiency of security amplification for MACs, digital signatures, and PRFs.

Due to the space limit, we focus on presenting our results on security amplification of commitment schemes. We discuss our results of puzzle systems in the following section, and defer the detailed definitions and proofs to the full version of this paper.

1.2 Puzzle Systems and Security Amplification for Other Primitives

Informally, in a puzzle system, there is a puzzle generator generates a puzzle and there is a solver trying to solve the puzzle. At a high level, puzzle systems provide

a nice way to abstract the security property of cryptographic protocols – the hardness of solving a puzzle models the hardness for an adversary to break the security. Previously, Canetti, Halevi, and Steiner [CHS05] define *weakly-verifierable puzzle systems* to capture the CAPTCHA scenario, and Dodis, Impagliazzo, Jaiswal, and Kabanets [DIJK09] generalized the model to *dynamic weakly-verifiable puzzle systems* to capture the security of MACs, digital signatures, and PRFs. In this paper, we introduce *two-phase puzzle systems*, which also generalize the model of [CHS05], to capture both hiding and binding properties of commitment schemes.

One natural way to achieve hardness/security amplification is via repetition. Suppose solving a puzzle is δ-hard in the sense that no efficient solver S can successfully solve a puzzle with probability higher than δ. If successfully solving different puzzles were independent events, then successfully solving n puzzles should be δ^n-hard. However, since a solver can correlate his answers to different puzzles, the events are not independent and the hardness bound may not hold. In the literature, there are various *(parallel) repetition theorems* for aforementioned puzzle systems saying that the hardness bounds match that of independent events and/or that the hardness is amplified in an exponential rate, which are useful to deduce security amplification results [CHS05, IJK07, DIJK09, Jut10]. In general, hardness amplification results for one puzzle systems do not imply the same results for another puzzle systems. Furthermore, for general interactive protocols, which can be viewed as "interactive puzzle systems," there are counter examples (under reasonable assumptions) showing that the hardness may not be amplified at all under *parallel* repetition [BIN97, PW07].

Previous Results. For weakly-verifiable puzzle systems, Canetti, Halevi, and Steiner [CHS05] prove a tight *Direct Product Theorem*, saying that solving n puzzles is δ^n-hard[2] if solving a single puzzle is δ-hard, and Impagliazzo, Jaiswal, and Kabanets [IJK07] prove a more general *Chernoff-type Theorem*, saying that solving at least $(1.1) \cdot \delta \cdot n$ out of n puzzles is $2^{-\Omega(\delta \cdot n)}$-hard if solving a single puzzle is δ-hard. The bound of [IJK07] was recently improved by Jutla [Jut10] to nearly optimal. Dodis, Impagliazzo, Jaiswal, and Kabanets [DIJK09] extend the Chernoff-type Theorem to dynamic weakly-verifiable puzzle systems, and use it to achieve security amplification for MACs, digital signatures, and PRFs. However, the proof techniques of [IJK07, DIJK09, Jut10] seem not applicable to the two-phase puzzle systems.

To analyze their transformations for security amplification for commitment schemes, Halevi and Rabin [HR08] prove a Hardness Degradation Theorem for two-phase puzzle systems (without formally defining the model), saying that solving at least one out of n puzzles is $(1 - (1 - \delta)^n)$-hard if solving a single puzzle is δ-hard (matching the bound for independent events).

Our Results. We show that the three types of hardness results (Direct Product, Chernoff-type, Hardness Degradation) actually hold for the three aforementioned puzzle systems (weakly-verifiable puzzles, dynamic weakly-verifiable puzzles, and

[2] We omit the negligible slackness in this informal discussion.

two-phase puzzles.) We establish a *Full-Spectrum Amplification Theorem*, which essentially says that the hardness of solving at least r puzzles out of n puzzles matches the bound of independent events if solving a single puzzle is δ-hard for some constant δ. Note that such a bound is optimal, since a solver can always solve each puzzle independently. A summary of our results and previous results can be found in Figure 2.

	Weakly Verifiable	Dynamic Weakly Verifiable	Two-Phase
Direct Product	[CHS05]	[DLJK09], Ours	[HR08]
Chernoff-type	[IJK07, Jut10], Ours	[DLJK09], Ours	Ours, [HS10]
Hardness Degradation	[HR08]	Ours*	[HR08]
Full-Spectrum	Ours, [HS10]	Ours*	Ours, [HS10]

Fig. 2. Summary of results on different types of puzzle systems. "Ours" means that either we obtain new results or we improve bounds over previous ones. The work of [HS10] and our work are independent works. (*): Our hardness degradation and full-spectrum results only hold for a variant of the dynamic weakly verifiable puzzle systems (see the full version of this paper for details).

We prove the Full-Spectrum Amplification Theorem by a single reduction algorithm that is applicable to all three puzzle systems. The reduction algorithm can be viewed as a generalization of the reduction algorithm of Canetti, Halevi, and Steiner [CHS05].

As a consequence, our improvement on the Chernoff-type Theorem for dynamic weakly verifiable puzzle systems of Dodis et al. [DLJK09] implies improvement on the efficiency of security amplification for MACs, digital signatures, and PRFs.

Historical Notes. The work of Holenstein and Schoenebeck [HS10] and our work were done independently, but have significant overlap. We briefly compare the results and make some historical notes as follows. For security amplification for commitment schemes, both works improve the result of Halevi and Rabin [HR08], but in complementary ways. Holenstein and Schoenebeck shows a feasibility result saying that security amplification is possible if and only if $p+q \leq 1-1/\mathrm{poly}(s)$. We improve the efficiency of the transformation, saying that only $\omega(\log s)$ black-box calls is sufficient to amplify security from constant to negligible and the resulting commitment scheme can commit to a $O(\log s)$-bit string. The constructions in both work are very different. As shown in the figure 1, the two results can be combined to obtain both improvements simultaneously.

For puzzle systems, Holenstein and Schoenebeck [HS10] present essentially the same idea and reduction algorithm as in our work. However, they have a cleaner way to deal with the parameters, and hence their result holds for every δ as opposed to constant δ in our result. Also, they consider more general "monotone combining functions" in addition to the threshold functions considered in our work. On the other hand, the application to efficiency improvement of security amplification for MACs, digital signatures, and PRFs was pointed out by us.

2 Preliminaries

All log's are base 2. s is the security parameter, and $\mathsf{ngl} = \mathsf{ngl}(s)$ denotes a negligible function of the security parameter, i.e. $s^{-\log(s)}$. We use U_n to denote uniform distribution over n-bit strings. We identify $\{0,1\}$ with $\mathbb{F}_2$, the finite field of size 2. If $x, y \in \{0,1\}^n$ are vectors in $\mathbb{F}_2^n$, then $x \oplus y \in \{0,1\}^n$ denotes their sum, (i.e. bit-wise xor) and $x \cdot y \stackrel{\text{def}}{=} \sum_i x_i y_i \in \{0,1\}$ denotes their inner product.

We review the facts we need about error-correcting codes. The lemma below says that a short random linear code has good minimum distance with overwhelming probability. It can be proved by standard probabilistic methods, and we omit the proof. The constants in the lemma are actually small.

Definition 1. *The* Hamming distance *of two strings x and y is the number of coordinates i such that $x_i \neq y_i$. Let $C : \{0,1\}^n \to \{0,1\}^{n'}$ be a code. The* minimum distance *of C is the minimum Hamming distance over all parts of codewords $C(x)$ and $C(y)$ such that $x \neq y$.*

Lemma 1. *There exist universal constants d_0, d_1 such that the following holds. Let k be a positive integer, and $\gamma, \delta \in [0,1]$ be numbers such that $\gamma > d_0 \cdot \delta \log(1/\delta)$. Let n be an integer such that $n > d_1 \cdot k/\delta$. Let $C : \{0,1\}^n \to \{0,1\}^{(1+\gamma)n}$ be a random linear code defined by $C(m) = (m, Am)$, where $A \in \{0,1\}^{\gamma n \times n}$ is a random* 0-1 *matrix. Then C has minimum distance at least $\delta \cdot n$ with probability at least $1 - 2^{-k}/2$.*

3 Definitions and Main Theorems

3.1 Commitment Schemes

In this section, we formally define commitment schemes and present our main theorem. We consider a standard model where the communication is over the classical noiseless channel and the decommitment is non-interactive [Gol01, HR08].

Definition 2 (Commitment Scheme). *A* **commitment scheme** *is an interactive protocol* $\mathrm{Com} = (S, R)$ *with the following properties:*

1. *Scheme* Com *consists of two stages: a* **commit stage** *and a* **reveal stage**. *In both stages, the* **sender** *S and the* **receiver** *R receive a security parameter 1^s as common input.*
2. *At the beginning of the commit stage, sender S receives a private input $v \in \{0,1\}^t$, which denotes the string to which S is supposed to commit. The commitment stage results in a joint output, which we call the* **commitment** *$x = \mathrm{output}((S(v), R)(1^s))$, and a private output for S, which we call the* **decommitment** *string $d = \mathrm{output}_S((S(v), R)(1^s))$. Without loss of generality, x can be taken to be the full transcript of the interaction between S and R, and d to be the private coin tosses of S.*
3. *In the reveal stage, sender S sends the pair (v, d), where d is the decommitment string for string v. Receiver R accepts or rejects based on v, d, x.*

4. *Both sender S and receiver R are efficient, i.e., both run in probabilistic polynomial time in the security parameter s.*
5. *R will always accept with probability $1-\mathsf{ngl}$ if both the sender S and the receiver R follow their prescribed strategy. If R accepts with probability 1, we say* Com *has* **perfect correctness**.
6. *When the commit string v is just a bit in $\{0,1\}$, we call* Com *a* **bit-commitment scheme**. *Otherwise, we call* Com *a t-***bit string-commitment scheme**.

Remark 1. The assumption of non-interactive reveal stage is essential in both our work and the previous work [HR08]. This assumption can be made without loss of generality as long as no additional property (e.g., if the sender wants to decommit in a zero-knowledge manner) is required, because in the reveal stage, the sender S can send his coin tosses to the receiver R, who can check the consistency and simulate the protocol. On the other hand, the assumption of perfect correctness can be relaxed to $(1-\mathsf{ngl})$-correctness in both works.

We proceed to define the hiding and binding properties of commitment schemes. To facilitate the presentation of our results and analysis, we are precise about the adversary's running time in the definition and define the binding property in terms of binding games.

Definition 3 (p-hiding against time T). *A commitment scheme* $\mathrm{Com} = (S, R)$ *is* p-hiding against uniform time T *if for every probabilistic time T cheating receiver R^*, the distributions* $(\mathrm{view}_{R^*}(S(U_t), R^*), U_t)$ *and* $(\mathrm{view}_{R^*}(S(U_t), R^*), U'_t)$ *are p-indistinguishable for time T, where U'_t is an i.i.d. copy of U_t. That is, for every probabilistic time T distinguisher D,*

$$|\Pr[D(\mathrm{view}_{R^*}(S(U_t), R^*), U_t) = 1] - \Pr[D(\mathrm{view}_{R^*}(S(U_t), R^*), U'_t) = 1]| \leq p/2$$

We say Com *is* p**-hiding** *if* $\mathrm{Com}(1^s)$ *is p-hiding against time s^c for every constant c, and sufficiently large security parameter s.*

We remark that the hiding property above is defined as the indistinguishability for *random values*, which does not generally imply the standard semantic security for the hiding property. Nevertheless, it is easy to transform a commitment scheme Com with the above hiding property to one with standard semantic security – one can use Com to commit to a random string $r \in_R \{0,1\}^t$, and use r as a one-time pad to hide the actual string v that the sender wants to commit to.

Remark 2. For bit-commitment schemes, p-hiding is equivalent to saying that the receiver can guess the committed bit with probability at most $1/2+p/2$. Formally, for every time T predictor P,

$$\Pr[P(\mathrm{view}_{R^*}(S(U_1), R^*)) = U_1)] \leq 1/2 + p/2.$$

Definition 4 (Binding Game). *The* **binding game** *for a commitment scheme* $\mathrm{Com} = (S, R)$ *is played between a honest receiver R, and (S^*, F), a cheating sender S^* with a decommitment finder F. The game consists of two stages:*

1. *In the commit stage, S^* interacts with R to produce a view* $\text{view}_{S^*}(S^*, R)$.
2. *In the decommitment finding stage, F gets the view* $\text{view}_{S^*}(S^*, R)$*, and produces two decommitment strings (s, d) and (s', d').*

(S^*, F) **succeeds** *if in the reveal stage, R accepts both decommitment strings (s, d) and (s', d').*

Definition 5 (q-binding against time T). *A commitment scheme* $\text{Com} = (S, R)$ *is* **q-binding against time** T*, if in the binding game, for every time T pair (S^*, F), the success probability of (S^*, F) is at most q. We say* Com *is* **q-binding** *if* $\text{Com}(1^s)$ *is q-binding against time s^c for every constant c, and sufficiently large security parameter s.*

Definition 6 (security of commitment schemes). *A commitment scheme* Com *is* **(p, q)-secure (against time T)** *if* Com *is p-hiding and q-binding (against time T).* Com *is* **secure** *if* $\text{Com}(1^s)$ *is (s^{-c}, s^{-c})-secure for every constant c, and sufficiently large security parameter s.*

We proceed to state our main result on efficient security amplification for commitment schemes. The following theorem says that we can securely commit a $O(\log s)$-bit string using only $\omega(\log s)$ black-box call to a weak commitment scheme Com_0 with constant hiding and binding properties.

Theorem 1. *Let $p, q \in (0, 1)$ be constants with $p + q < 1$. Suppose there exists a (p, q)-secure bit commitment scheme* Com_0*. Then for every $t(s) = O(\log s)$, $n(s) = \omega(t + \log s)$ where s is the security parameter, there exists a secure t-bit string-commitment scheme* Com *that makes only n black-box call to* Com_0*.*

4 Efficient Security Amplification for Commitment Schemes

In this section, we present our result on efficient black-box security amplification for commitment schemes in the computational setting, where the security holds against PPT active adversaries. We start by reviewing the previous construction of Halevi and Rabin [HR08], and then discuss its limitation and our improvement. The construction in [HR08] uses the following two transformations, each of which improves one property significantly without hurting the other property too much.

– **Secret-sharing transformation.** Let Com_0 be a bit commitment scheme, and $n \in \mathbb{N}$ be a parameter. The transformation gives a bit commitment scheme $\text{Com} = (S, R)$ as follows. To commit a bit $b \in \{0, 1\}$ to R, S generates random $b_1, b_2, \ldots b_n \in \{0, 1\}$ such that $\bigoplus_{i \in [n]} b_i = b$, i.e. a secret sharing of b, and then uses Com_0 to sequentially commit to each b_i to R.

  Intuitively, this transformation improves the hiding property, since an adversarial R^* needs to learn all bits $b_1, \ldots, b_n$ to recover b, but hurts the binding property, since an adversarial S^* only needs to cheat on any single bit b_i to decommit in two ways. Indeed, Halevi and Rabin proved that if Com_0 is (p, q)-secure, then Com is $(p^n, 1 - (1 - q)^n)$-secure.[3]

[3] We omit the negligible slackness in the informal discussion.

– **Repetition transformation.** Let Com_0 be a bit commitment scheme, and $n \in \mathbb{N}$ be a parameter. The transformation gives a bit commitment scheme $\mathrm{Com} = (S, R)$ as follows. To commit a bit $b \in \{0,1\}$ to R, S sequentially uses Com_0 n times to commit to the same bit b to R.

  Intuitively, this transformation improves the binding property, since an adversarial S^* needs to cheat on all copies of Com_0 to decommit in two ways, but hurts the hiding property, since an adversarial R^* can learn the bit b from any copy of the commitments. Indeed, Halevi and Rabin proved that if Com_0 is (p,q)-secure, then Com is $(1-(1-p)^n, q^n)$-secure.

Halevi and Rabin showed that, as long as p and q satisfy $p+q \leq 1-1/\mathrm{polylog}(s)$, then given a (p,q)-secure (weak) bit commitment scheme Com_0, one can apply the above two transformations alternately to obtain a secure bit commitment scheme Com. To measure the efficiency, consider the case where both p and q are constants with $p+q<1$. Since improving either hiding or binding property from constant to negligible requires $\omega(\log s)$ invocations to Com_0, and the above transformations improve two properties *separately*, the construction of Halevi and Rabin requires at least $\omega(\log^2 s)$ black-box calls to Com_0.

Remark 3. Independent of our work, Holenstein and Schoenebeck [HS10] present a different construction that improves the result of Halevi and Rabin in the following sense.For any (p,q)-secure bit commitment scheme Com_0 with $p+q \leq 1-1/\mathrm{poly}(s)$, their construction gives a secure bit commitment scheme Com using $\mathrm{poly}(s)$ black-box calls to Com_0. Their construction uses Valiant's monotone formula for majority [Val84] to improve both properties. However, a closer inspection shows that their construction is equivalent to applying the secret sharing transformation and a variant of repetition transformation (with the same effect on the parameters) alternately. Hence, in terms of the efficiency, their construction also requires at least $\omega(\log^2 s)$ black-box calls to amplify a (p,q)-secure weak commitment scheme with constant p and q to a secure one.

To bypass the $\omega(\log^2 s)$ barrier of the existing constructions, our main idea is to use error-correcting codes and randomness extractors to amplify both hiding and binding properties *simultaneously*. For intuition, we give an informal description of our transformation first. Let us informally use $\mathrm{Com}_0(b)$ to denote a commitment of a bit b, and let $C : \{0,1\}^n \to \{0,1\}^{n'}$ be an error-correcting code with minimum distance at least $\delta \cdot n'$, and $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^t$ a strong randomness extractor. Our transformation uses Com_0, C and Ext to commit to a string $v \in \{0,1\}^t$ as follows (recall that we obtain string commitment schemes as opposed to bit commitment schemes of other existing constructions).

– **Commit Stage:** the sender S samples a message $m \in_R \{0,1\}^n$ uniformly at random, and sequentially commits to each bit of the codeword $C(m)$ using Com_0, which generates commitments $\mathrm{Com}_0(C(m)) \stackrel{\mathrm{def}}{=} (\mathrm{Com}_0(C(m)_1), \ldots, \mathrm{Com}_0(C(m)_{n'}))$. Then S samples a uniform seed $z \in_R \{0,1\}^d$, and sends the seed z with $v \oplus \mathrm{Ext}(m,z)$ to the receiver R. In sum, the commitment is $\mathrm{Com}(v) = (\mathrm{Com}_0(C(m)), z, v \oplus \mathrm{Ext}(m,z))$.

– **Reveal Stage:** the sender S sends the value v, the message m and reveals each committed bit of $C(m)$ to R, who checks consistency and accepts or rejects accordingly.

Intuitively, the binding property is improved because for an adversarial sender S^* to cheat, S^* needs to decommit $C(m)$ into two valid codewords. Since the code C has good minimum distance, S^* needs to successfully cheat on at least $\delta \cdot n'$ committed bits out of n' commit bits. The q-binding property of Com_0 says that, for each committed bit, S^* can cheat with probability at most q. Thus, in expectation, S^* can cheat on only $q \cdot n'$ commit bits. If $q < (0.9)\delta$, the Chernoff bound suggests that S^* should be able to cheat on at least $\delta \cdot n'$ commit bits with only exponentially small probability in n'. On the other hand, the hiding property is improved because after seeing the commitments of $C(m)$, an adversarial receiver R^* has only partial information about m by the p-hiding property of Com_0. Thus, Ext extracts the remaining (computational) entropy from m, which is used to hide the value v. Ideally, when both p and q are constants, we can set both $n, n' = \omega(\log s)$ and commit to $\Omega(n)$-bit string.

In sum, our efficient security amplification for commitment schemes consists of three steps: given a (p, q)-secure bit commitment scheme Com_0 with constants $p + q < 1$, (1) we first apply the transformations of Halevi and Rabin to obtain a (p', q')-secure bit commitment scheme Com_1 with sufficiently small constants p', q', which costs a constant number of black box calls, (2) we apply the above construction to obtain a (s^{-c}, s^{-c})-secure $O(\log s)$-bit string commitment scheme Com_2, which costs $O(\log s)$ black box calls, and (3) we apply a string version of the transformations of Halevi and Rabin [HR08] to obtain a secure $O(\log s)$-bit string commitment scheme Com_3, which costs $\omega(1)$ black box calls. The number of black-box calls multiply over steps, and hence the resulting Com_3 uses $\omega(\log s)$ black-box calls to Com_0.

We proceed to give a formal description of the above construction and its analysis in Section 4.1, and present a string version of the transformations of Halevi and Rabin used in the third step in Section 4.2.

4.1 Efficient Security Amplification in the Known-Security Setting

In this section, we present a transformation that converts a (p, q)-secure bit commitment scheme Com_0 to a (s^{-c}, s^{-c})-secure $O(\log s)$-bit string commitment scheme Com using $O(\log s)$ black-box calls to Com_0, where c is an arbitrary constant. Our transformation uses error-correcting codes and randomness extractors to amplify both hiding and binding properties *simultaneously*. The transformation requires to use a *systematic* code with good distance and the "Goldreich-Levin" extractor. We will discuss the reason when we prove the security below. A formal description of our transformation can be found in Figure 3.

We will show that if Com_0 is a (p, q)-secure bit commitment scheme for small constants p, q, then by setting $n, \ell, t = O(\log s)$, the resulting string commitment scheme is (s^{-c}, s^{-c})-secure for some constant c. Note that both parties in Com run in time polynomial in n, ℓ, t, and the running time of Com_0, which is efficient. Formally, we prove the following theorem.

Transformation $\mathcal{T}(\mathrm{Com}_0, n, \ell, t)$:

- **Inputs.** A bit commitment scheme Com_0, and parameters $n, \ell, t \in \mathbb{N}$.
- **Outputs.** A t-bit string-commitment scheme $\mathrm{Com} = (S, R)$ defined as follows.
- **Commit Stage.** Let $v \in \{0,1\}^t$ be the string to which S is committing to.
  1. R samples a uniformly random matrix $A \leftarrow \{0,1\}^{\ell \times n}$, and sends A to S.
     /* i.e., R selects a random systematic linear code $C(m) \stackrel{\text{def}}{=} (m, Am)$. */
  2. S samples the following uniformly at random: a message $m \leftarrow \{0,1\}^n$ and a matrix $Z \leftarrow \{0,1\}^{t \times n}$.
     /* Z is a random seed for a (strong) randomness extractor $\mathrm{Ext}(m, Z) \stackrel{\text{def}}{=} Zm$.*/
  3. S uses Com_0 to commit to each bit of m and each bit of Am to R sequentially. Let $\boldsymbol{x} = (x_1, \ldots, x_n)$ and $\boldsymbol{y} = (y_1, \ldots, y_\ell)$ denote the commitment of each bit respectively.
     /* i.e., S commits to each bit of the codeword $C(m)$. */
  4. S sends $(Z, v \oplus Zm)$ to R, where $v \oplus Zm$ is the bit-wise xor of v and Zm.
     /* i.e., S uses $\mathrm{Ext}(m, Z)$ as a one-time pad to hide the commit string v. */

  In sum, the commitment of v is $(A, \boldsymbol{x}, \boldsymbol{y}, Z, v \oplus Zm)$.
- **Reveal Stage.** S sends v and its coin tosses r to R, and R checks that v and r are consistent with the honest sender's algorithm.

Fig. 3. Our black-box transformation $\mathcal{T}(\mathrm{Com}_0, n, \ell, t)$

Theorem 2. *The following holds for all sufficiently small constants* $p, q \in (0,1)$, *and* $k = O(\log s)$*: Suppose there exists a* (p,q)*-secure (weak) bit commitment scheme* Com_0*, then there exists a* $(2^{-k}, 2^{-k})$*-secure* $t = \Omega(k)$*-bit string-commitment scheme* Com *that makes* $O(k)$ *black-box calls to* Com_0*. Specifically,* $\mathrm{Com} = \mathcal{T}(\mathrm{Com}_0, n, \ell, t)$ *for appropriate* $n, \ell = O(k)$*, and* $t = \Omega(k)$*.*

We formalize the aforementioned intuition to analyze the hiding and binding properties in the below subsections.

Analysis of the Binding Property. In this section, we analyze the binding property of our transformation $\mathcal{T}(\mathrm{Com}_0, n, \ell, t)$. We first recall the intuition of why the binding property is improved. Recall that in the construction, the sender S is supposed to commit to each bit of a valid codeword $C(m) = (m, Am)$ using Com_0, where C is a random linear code chosen by the receiver R. By Lemma 1, C has good min-distance $\delta \cdot n$ with overwhelming probability. For an adversarial sender S^* to cheat, S^* needs to decommit the $n + \ell$ commitments into two valid codewords $C(m_1), C(m_2)$, which means that S^* needs to successfully cheat on at least $\delta \cdot n$ commitments out of $n + \ell$ commitments. Intuitively, suppose breaking the binding property of each commitment were independent events with success probability at most q, and if $\delta \cdot n \geq (1.1) \cdot q \cdot (n + \ell)$, then by Chernoff bounds, the success probability of S^* should be exponentially small in n.

Of course, the events are not independent since S^* has chance to correlate his strategy for different instances. However, breaking the binding property of sequentially committed bits can be modeled as repetition of two-phase puzzle systems,

and hence the above intuition can be formalized using the Full-Spectrum Amplification Theorem (appeared in the full version of this paper), which says the success probability of S^* behaves the same as the case of independent events.

Formally, we prove the following lemma, which essentially says that when q is sufficiently smaller than the min-distance of the code, the binding property is amplified in an exponential rate. We formulate the lemma in concrete parameters for preciseness. For intuition, think of $n, \ell = \Theta(k)$, $k = O(\log s)$, $T_0 = \mathrm{poly}(s)$, and $T = s^{\omega(1)}$.

Lemma 2 (Binding). *Let d_0 be the universal constant in Lemma 1. There exist universal constants c_1 such that the following holds. For any $q \in (0,1), n, k, \ell, t, T_0, T \in \mathbb{N}$ satisfying (i) $d_0 \cdot (3q) \cdot \log(1/3q) < 1$, (ii) $2c_1 \cdot k/q \geq n \geq c_1 \cdot k/q$, (iii) $n > \ell \geq d_0 \cdot (3q) \cdot \log(1/3q) \cdot n$, if a bit-commitment scheme $\mathrm{Com}_0 = (S_0, R_0)$ with runtime T_0 is q-binding against time T, then $\mathrm{Com} = \mathcal{T}(\mathrm{Com}_0, n, \ell, t)$ is 2^{-k}-binding against time $T' = T/\mathrm{poly}(2^k, T_0, t)$.*

Analysis of the Hiding Property. In this section, we analyze the hiding property of our transformation $\mathcal{T}(\mathrm{Com}_0, n, \ell, t)$. We first recall the intuitive entropy argument of why the hiding property is improved. Recall that in the construction, the sender S samples a random n-bit message m, which contains n bits of entropy. Then S commits to each bit of the codeword $C(m) = (m, Am)$, each of which leaks information about m. Intuitively, if we set the parameters so that there are entropy left in m, S can use randomness extractor to extract a string $\mathrm{Ext}(x, Z)$ that is (pseudo-)random from an adversarial receiver R^*'s point of view, and use it as one-time-pad to hide the commit value v.

We argue that it is very hard for R^* to predict the whole message m after he sees the $n+\ell$ commitments, and hence one can apply the Goldreich-Levin theorem to extract pseudo-random bits. This is why our transformation requires to use the Goldreich-Levin extractor. To argue that m is hard to predict from the commitments $(\boldsymbol{x}, \boldsymbol{y})$, we first argue that m is hard to predict from $\boldsymbol{x}$. We can view predicting n sequentially committed message bits of m from the commitments $\boldsymbol{x}$ as n-fold direct product of a two-phase puzzle system. By Direct Product Theorem of Halevi and Rabin [HR08], the success probability of R^* is at most $((1+p)/2)^n$ (up to a negligible factor). Observing that $\boldsymbol{y}$ contains at most ℓ bits of information about m, the success probability of R^* to predict m from $(\boldsymbol{x}, \boldsymbol{y})$ is at most $2^\ell \cdot ((1+p)/2)^n$. Hence, by the Goldreich-Levin theorem, we can extract $\Omega(\log(2^\ell \cdot ((1+p)/2)^n))$ pseudorandom bits.

Formally, we prove the following lemma, which essentially says that we can extract $\Omega(\log(2^\ell \cdot ((1+p)/2)^n))$ pseudorandom bits. Again, we formulate the lemma in concrete parameters for preciseness, and we use parameter $\alpha = 1-p$ for clarity. For intuition, think of $n, \ell = \Theta(k)$, $k = O(\log s)$, $T_0 = \mathrm{poly}(s)$, and $T = s^{\omega(1)}$.

Lemma 3 (Hiding). *There exist universal constants c_2 such that the following holds. For every $\alpha \in (0,1), n, k, \ell, t, T_0, T \in \mathbb{N}$ satisfying (i) $2c_2 \cdot k/\alpha \geq n \geq c_2 \cdot k/\alpha$, (ii) $\ell, t \leq \alpha n/12$, if $\mathrm{Com}_0 = (S_0, R_0)$ with runtime T_0 is a $(1-\alpha)$-hiding against time T, then $\mathrm{Com} = \mathcal{T}(\mathrm{Com}_0, n, \ell, t)$ is 2^{-k}-hiding against time $T' = T/\mathrm{poly}(2^k, T_0)$.*

We leave the proofs of Lemma 2 and 3 in the full version of this paper.

Proof of Theorem 2. Theorem 2 follows by applying Lemma 2 and 3 with properly chosen parameters.

Proof. (**of Theorem 2**) We set the parameters n, k, ℓ as follows: $n = \max\{\frac{c_1 k}{q}, \frac{c_2 k}{1-p}\} = \Theta(k)$, $\ell = d_0(3q)\log(3q) \cdot n$, and $t = \frac{(1-p)n}{12} = \Omega(k)$, where c_1, c_2, d_0 are the constants in the Lemma 1, 2, and 3. The theorem follows directly from Lemma 2 and 3.

4.2 Security Amplification for String Commitment Schemes

In this section, we generalize the transformations of Halevi and Rabin [HR08] to the case of string commitment schemes, with the goal of amplifying the (s^{-c}, s^{-c})-secure string commitment scheme obtained from our transformation to achieve negligible security. This is a simpler task, and can be done by applying a secret-sharing transformation first and then a repetition transformation. A formal description of the transformations can be found in Figure 4.

Secret-sharing $\mathcal{SS}(\mathrm{Com}_0, u)$. Let Com_0 be a t-bit string commitment scheme, and $u \in \mathbb{N}$ be a parameter. The transformation gives a t-bit string commitment scheme $\mathrm{Com} = (S, R)$ as follows. To commit a value $v \in \{0,1\}^t$ to R, S generates random $v_1, v_2, \ldots v_n \in \{0,1\}^t$ such that $v_1 \oplus v_2 \oplus \cdots \oplus v_u = v$, where $\oplus$ denotes the bit-wise xor of v_i's (i.e. a secret sharing of v), and then uses Com_0 to sequentially commit to each v_i to R.

Repetition $\mathcal{R}(\mathrm{Com}_0, u)$. Let Com_0 be a t-bit string commitment scheme, and $u \in \mathbb{N}$ be a parameter. The transformation gives a t-bit string commitment scheme $\mathrm{Com} = (S, R)$ as follows. To commit a value $v \in \{0,1\}^t$ to R, S sequentially uses Com_0 u times to commit to the same value v to R.

Fig. 4. Secret-sharing and repetition transformation for string commitment schemes

We proceed to analyze the binding and hiding properties of the resulting commitment schemes of the two transformations. For the binding property, the analysis is essentially the same as in [HR08]: for repetition, it requires to break all u commitments of Com_0, and for secret-sharing, it requires to break only 1 out of u commitments of Com_0, which can be modeled as solving corresponding repetition of two-phase puzzles. The Direct Product Theorem and Hardness Degradation Theorem of Halevi and Rabin [HR08] (or our Full-Spectrum Amplification Theorem) imply the following lemma.

Lemma 4 ([HR08]). *Let* Com_0 *be a* t*-bit string-commitment scheme,* $u = u(s) \leq \mathrm{poly}(s)$ *a efficiently computable function, and* $q \in (0,1)$ *a constant. Suppose* Com_0 *is* q*-binding, then* $\mathcal{R}(\mathrm{Com}_0, u)$ *is* $(q^u + \mathsf{ngl})$*-binding, and* $\mathcal{SS}(\mathrm{Com}_0, u)$ *is* $(1-(1-q)^u + \mathsf{ngl})$*-binding.*

On the other hand, analyzing the hiding property is tricker. For the secret-sharing transformation, we need a string version of XOR Lemma to show that the hiding property is amplified. Maurer and Tessaro [MT09] proved a more general result (Theorem 2 of [MT09]) in the context of system composition, which implies the following lemma.

Lemma 5 ([MT09]). *Let* Com_0 *be a* t*-bit string-commitment scheme, and* $\mathrm{Com} = \mathcal{SS}(\mathrm{Com}_0, u)$ *with efficiently computable* $u = u(s) \leq \mathrm{poly}(s)$*. If* Com_0 *is* p*-hiding, then* Com *is* $(p^u + \mathsf{ngl})$*-binding.*

We next show that repetition transformation preserves the (negligible) hiding property. This is sufficient for our purpose since we will apply the secret-sharing transformation to amplify the hiding property to negligible before applying the repetition transformation.

Lemma 6. *Let* $\mathrm{Com}_0 = (S_0, R_0)$ *be a* t*-bit string-commitment scheme, and* $\mathrm{Com} = \mathcal{R}(\mathrm{Com}_0, u)$ *with efficiently computable* $u = u(s) \leq \mathrm{poly}(s)$*. If* Com_0 *is* ngl*-hiding, so does* Com*.*

We leave the proof in the full version of this paper.

4.3 Put Things Together

We are ready to present a formal description of our efficient security amplification for commitment schemes (in Figure 5) and prove Theorem 1.

Final Construction.

- **Inputs.** A (p, q)-secure bit commitment scheme Com_0 with $p + q < 1$.
- **Outputs.** A secure t-bit string-commitment scheme Com with $t = O(\log s)$.

1. Apply the transformations of Halevi and Rabin alternately to obtain a (p', q')-secure bit commitment scheme Com_1 with sufficiently small constants p', q'.
2. Apply our transformations $\mathcal{T}(\mathrm{Com}_1, n, \ell, t)$ to obtain a (s^{-c}, s^{-c})-secure t-bit string commitment scheme Com_2, where $n, \ell = O(\log s)$, and c is some constant.
3. Let a be an arbitrary $\omega(1)$ function. Apply $\mathcal{SS}(\mathrm{Com}_2, a)$ to obtain a $(\mathsf{ngl}, a \cdot s^{-c} + \mathsf{ngl})$-secure t-bit string commitment scheme Com_3.
4. Apply $\mathcal{R}(\mathrm{Com}_3, a)$ to obtain a secure t-bit string commitment scheme Com.

Fig. 5. Efficient security amplification of commitment schemes

Proof (of Theorem 1). The fact that Com is a secure string commitment scheme follows straightforwardly from Theorem 2 and Lemma 4, 5, 6. Observing the Com_1 makes $O(1)$ black-box calls to Com_0, Com_2 makes $O(\log s)$ black-box calls to Com_1, Com_3 makes $\omega(1)$ black-box calls to Com_2, and finally Com makes $\omega(1)$ black-box calls to Com_3, the total number of black-box calls that Com makes to Com_0 is $\omega(\log s)$, as desired.

Acknowledgments

We thank Salil Vadhan for his discussions throughout the whole work of this paper. We also thank Anna Lysyanskaya and Yevgeniy Dodis for sharing their insights.

References

[BIN97] Bellare, M., Impagliazzo, R., Naor, M.: Does parallel repetition lower the error in computationally sound protocols? In: FOCS, pp. 374–383 (1997)

[CHS05] Canetti, R., Halevi, S., Steiner, M.: Hardness amplification of weakly verifiable puzzles. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 17–33. Springer, Heidelberg (2005)

[DIJK09] Dodis, Y., Impagliazzo, R., Jaiswal, R., Kabanets, V.: Security amplification for interactivecryptographic primitives. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 128–145. Springer, Heidelberg (2009)

[DKS99] Damgård, I., Kilian, J., Salvail, L.: On the (im)possibility of basing oblivious transfer and bit commitment on weakened security assumptions. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 56–73. Springer, Heidelberg (1999)

[DNR04] 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)

[Gol01] Goldreich, O.: Foundations of Cryptography. Basic tools. Cambridge University Press, Cambridge (2001)

[HR05] Holenstein, T., Renner, R.: 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)

[HR08] Halevi, S., Rabin, T.: Degradation and amplification of computational hardness. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 626–643. Springer, Heidelberg (2008)

[HS10] Holenstein, T., Schoenebeck, G.: General hardness amplification of predicates and puzzles. CoRR, abs/1002.3534 (2010)

[IJK07] Impagliazzo, R., Jaiswal, R., Kabanets, V.: Chernoff-type direct product theorems. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 500–516. Springer, Heidelberg (2007)

[Jut10] Jutla, C.S.: Almost optimal bounds for direct product threshold theorem. In: Micciancio, D. (ed.) Theory of Cryptography. LNCS, vol. 5978, pp. 37–51. Springer, Heidelberg (2010)

[MT09] Maurer, U., Tessaro, S.: Computational indistinguishability amplification: Tight product theorems for system composition. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 350–368. Springer, Heidelberg (2009)

[PW07] Pietrzak, K., Wikström, D.: Parallel repetition of computationally sound protocols revisited. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 86–102. Springer, Heidelberg (2007)

[Val84] Valiant, L.G.: Short monotone formulae for the majority function. J. Algorithms 5(3), 363–366 (1984)

[Wul07] Wullschleger, J.: Oblivious-transfer amplification. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 555–572. Springer, Heidelberg (2007)

[Yao82] Yao, A.C.-C.: Theory and applications of trapdoor functions (extended abstract). In: FOCS, pp. 80–91 (1982)

On the Static Diffie-Hellman Problem on Elliptic Curves over Extension Fields

Robert Granger*

Claude Shannon Institute
School of Computing, Dublin City University
Glasnevin, Dublin 9, Ireland
rgranger@computing.dcu.ie

Abstract. We show that for any elliptic curve $E(\mathbb{F}_{q^n})$, if an adversary has access to a Static Diffie-Hellman Problem (Static DHP) oracle, then by making $O(q^{1-\frac{1}{n+1}})$ Static DHP oracle queries during an initial learning phase, for fixed $n > 1$ and $q \to \infty$ the adversary can solve *any* further instance of the Static DHP in *heuristic* time $\tilde{O}(q^{1-\frac{1}{n+1}})$. Our proposal also solves the *Delayed Target DHP* as defined by Freeman, and naturally extends to provide algorithms for solving the *Delayed Target DLP*, the *One-More DHP* and *One-More DLP*, as studied by Koblitz and Menezes in the context of Jacobians of hyperelliptic curves of small genus. We also argue that for *any* group in which index calculus can be effectively applied, the above problems have a natural relationship, and will *always* be easier than the DLP. While practical only for very small n, our algorithm reduces the security provided by the elliptic curves defined over $\mathbb{F}_{p^2}$ and $\mathbb{F}_{p^4}$ proposed by Galbraith, Lin and Scott at EUROCRYPT 2009, should they be used in any protocol where a user can be made to act as a proxy Static DHP oracle, or if used in protocols whose security is related to any of the above problems.

1 Introduction

In recent years, there has been a steadily growing appreciation of the existence of an apparent separation, in some cases, between the hardness of breaking discrete logarithms in a particular group, and the hardness of solving in that group certain problems to which the security of a cryptosystem is provably related. This situation can arise when auxiliary information is provided to an attacker in the form of limited access to a particular oracle, either within the game played by the attacker in the security proof, or in practice when a user can be made to act as a proxy oracle by virtue of the nature of the protocol itself.

For example, in 2004 Brown and Gallant studied the *Static Diffie-Hellman Problem* (Static DHP), in which a party reuses the same Diffie-Hellman secret

* Research supported by the Claude Shannon Institute, Science Foundation Ireland Grant No. 06/MI/006.

M. Abe (Ed.): ASIACRYPT 2010, LNCS 6477, pp. 283–302, 2010.

for multiple Diffie-Hellman key agreements [6]. The authors proved that if the associated DLP for the static secret is hard, then so is the Static DHP. However, their reduction naturally becomes an algorithm for solving the DLP if an attacker has access to a Static DHP oracle. In the protocols [15,17] and [7] for instance, a user can indeed be made to act as a proxy Static DHP oracle, thus rendering such systems vulnerable to this attack. In the best case (from an attacker's perspective), one can compute a static Diffie-Hellman secret in a group of order r with only $O(r^{1/3})$ Static DHP oracle queries and $O(r^{1/3})$ group operations [6]. For cryptographically interesting elliptic curves, i.e., those for which generic attacks are the best known, this result is in stark contrast to the time required to compute discrete logarithms, namely $O(r^{1/2})$. So while solving the Static DHP in this case may still be hard, it has lower complexity than the best DLP algorithms.

Koblitz and Menezes have shown that several other problems exhibit a similar apparant hardness separation[1] from the DLP, in the context of Jacobians of hyperelliptic curves of small genus [38]; namely the *Delayed Target DHP* [18], the *Delayed Target DLP* [38], the *One-More DHP* [3] and the *One-More DLP* [1,2]. For each of these problems, it is the use of oracle queries that creates these separations. For instance, for the *Delayed Target DHP*, the Brown-Gallant algorithm can be applied immediately, since the game played by the attacker gives him initial access to a Static DHP oracle.

In 2006 Cheon rediscovered the Brown-Gallant algorithm when the requisite information is provided in the guise of the Strong Diffie-Hellman Problem (Strong DHP) [8]. Cheon also extended the attack to utilise divisors of $r+1$ as well as of $r-1$, as with the Brown-Gallant algorithm; indeed both algorithms can be regarded as instances of the well-known reduction from the DLP to the DHP due to den Boer, Maurer, Wolf *et al.,* (see [42] for a survey), but with restricted access to a DHP oracle. Incidently, Cheon's break of the Strong DHP does not in itself reveal any weakness in the protocols that depend upon it, since reductions given in security proofs until that time were in the wrong direction, i.e., they showed that breaking the system enables one to solve the Strong DHP, but not the other way around. Hence, if an algorithm that efficiently solves the Strong DHP is found, then while all security proofs that assume its hardness would no longer provide any security assurance, no actual break of the associated systems would result. In the case of Boneh-Boyen signatures [4], which relies on the Strong DHP in the above manner, Jao and Yoshida have given a reduction in the reverse direction, thus strengthening the proof of security for these signatures, and at the same time providing an attack on the scheme with complexity $O(r^{2/5+\epsilon})$, if $O(r^{1/5+\epsilon})$ signature queries are permitted to be performed [32]. This result suggests that if one can establish an equivalence between a given protocol and a problem that exhibits an apparent hardness separation from the DLP, then in some attack models the security assurances provided by these arguments will likely be lower than that provided by the DLP.

[1] We use the prefix *apparent*, since these separations exist only relative to the current understanding of the respective problems, which could of course change.

Similarly, for the RSA problem, in 2007 Joux, Naccache and Thomé showed that with initial subexponential access to an e-th root oracle an attacker can later compute the e-th root of any element with complexity lower than that required to factor the modulus [36]. This algorithm was then adapted to solve the oracle-assisted Static DHP in finite fields [34], with similar efficiency improvements, demonstrating an apparent hardness separation in this case also.

It is therefore natural to ask whether initial access to a Static DHP oracle can aid in solving later Static DHP instances faster than solving the DLP, in the context of elliptic curves? As previously mentioned, in the best case the Brown-Gallant-Cheon algorithm requires $O(r^{1/3})$ oracle queries and group operations. However, for elliptic curves defined over extension fields $\mathbb{F}_{q^n}$, we present an algorithm which for fixed $n > 1$ and $q \to \infty$ requires $O(q^{1-\frac{1}{n+1}})$ oracle queries and has *heuristic* time complexity $\tilde{O}(q^{1-\frac{1}{n+1}})$. This should be compared with the best known DLP algorithm for these curves which has complexity $\tilde{O}(q^{2-2/n})$ [24], hence our proposal approaches being a square-root faster than the DLP with increasing n. Note that for $n = 2$ our complexity is the same as the best-case Brown-Gallant-Cheon complexity, but applies to all elliptic curves over $\mathbb{F}_{q^2}$ and not just those with appropriate divisors of $r \pm 1$, while for $n > 2$ our result is superior. We also present an *heuristic* subexponential oracle-assisted Static DHP algorithm for elliptic curves over a special family of extension fields.

Our proposal also naturally extends to provide algorithms for solving the four problems studied by Koblitz and Menezes in [38]. In this work it was found that the relationships between the hardness of these problems do not appear to behave as one might expect (cf. [39,40]). We correct a minor oversight in the analysis and argue that in the context of *any* group in which index calculus is effective, i.e., one in which index calculus provides the best known algorithm to solve the given problems — which includes Jacobians of hyperelliptic curves and elliptic curves over extension fields — the aforementioned problems do indeed have natural relationships, and are *always* easier than the DLP (this statement naturally only applies with respect to the state of the art in index calculus algorithms). However a central conclusion of [38], namely that it is difficult to assess what security assurances are provided by security proofs when the games played are interactive or have complicated inputs, still holds.

Due to the fact that the implicit constant in the complexity of our algorithm grows very quickly with n, it is practical only for small values of n, namely $n = 2, 3$ or 4 (and whenever n is divisible by $2, 3$ or 4). However, based on the results of timing estimates arising from an implementation of the components of the attack, the security provided by the elliptic curves defined over $\mathbb{F}_{p^2}$ and $\mathbb{F}_{p^4}$ proposed by Galbraith, Lin and Scott at EUROCRYPT 2009, would be significantly reduced, should these curves be used in any protocol where a user can be made to act as a proxy Static DHP oracle, or if used in protocols whose security is related to any of the problems studied in [38].

Independently of this work, Joux and Vitse [37] have proposed the same basic algorithm as the one presented here for the oracle-assisted Static DHP, namely

Heuristic Result 1. Although Joux and Vitse did not consider our factor base reduction method that leads to Heuristic Result 2, their main idea improves upon Gaudry's relation finding technique — which is used in the present work — enabling one to find relations for elliptic curves over degree five extension fields, which is currently impractical with Gaudry's method.

One goal of the present work was to assess the impact the algorithms presented herein had upon the oracle-assisted Static DHP on the Oakley Well-known Groups 3 and 4, which form part of the IPSEC set of protocols [31] (see §4 and §5), and which are defined over degree five extension fields in characteristic two. Noting from our results how much easier it is to find relations in characteristic two than for large prime characteristic, the author contacted the authors of [37] in order to apply their idea to these curves. Using the results of this paper and [37], we recently announced the experimental verification of the feasibility of solving the oracle-assisted Static DHP on the 152-bit Group 3 curve [28] over $\mathbb{F}_{2^{155}}$, which has long been suspected of weakness, but had until now resisted many attacks [49,25,22,44].

The sequel is organised as follows. In §2 we recall the Static DHP. In §3 we motivate our main idea, present our basic algorithm, and analyse asymptotic variants of it. Then in §4 we detail curves in the literature that are vulnerable to our attack. In §5 we give a full account of our experimental implementation at the 128-bit security level for extension degrees $n = 2, 3, 4$ and 5, over large prime and characteristic two fields, assess their impact on the above curves and report on the Oakley Group 3 curve. In §6 we present algorithms for three other problems which arise in cryptographic protocols and analyse their impact, and make some concluding remarks.

2 The Static Diffie-Hellman Problem

Let $\mathbb{G}$ be a cyclic group of prime order r, and let g be a fixed generator of $\mathbb{G}$. The classical Diffie-Hellman problem in $\mathbb{G}$ can be stated as follows [12]:

Problem 1. (DHP): Given g and random g^x and g^y, find g^{xy}.

In Diffie-Hellman (DH) key agreement between two parties, Alice chooses a random secret $x \in \mathbb{Z}/r\mathbb{Z}$ and computes g^x, while Bob chooses a random secret $y \in \mathbb{Z}/r\mathbb{Z}$ and computes g^y, which are then exchanged. Upon receipt each party computes the shared secret g^{xy} by exponentiating the other party's group element by their own secret. A fundamental security requirement of DH key agreement is that the DHP should be hard.

Should Alice for any reason repeatedly reuse the same secret, $x = d$ say, then the resulting set of DHP problem instances forms a tiny subset of all problem instances featured in the DHP, and it is thus not *a priori* clear that these instances should be hard, even if the DHP is hard. This problem is referred to as the Static DHP$_d$, which we state as follows:

Problem 2. (Static DHP$_d$): Given fixed g and g^d, and random g^y, find g^{dy}.

Observe that this situation need not just arise as an efficiency measure during multiple DH key agreements — Alice need only compute g^d once and reuse this value for multiple key agreements — but also arises in text-book El-Gamal encryption [15], Ford-Kaliski key retrieval [17] and Chaum-Van Antwerpen's undeniable signatures [7]. As mentioned in §1, Brown and Gallant have shown that if the associated DLP is hard, then so is the Static DHP$_d$ [6]. However, in the above three protocols, one of the system entities acts as a Static DHP$_d$ oracle, thus turning the Brown-Gallant reduction into an attack.

As in [6] we define an oracle for solving the Static DHP$_d$ as follows:

Definition 1. *(Static DHP$_d$ Oracle). Let $\mathbb{G}$ be a cyclic group of prime order r, written additively. For a fixed base element $P \in \mathbb{G}$ and a fixed element $Q \in \mathbb{G}$ let $d \in \mathbb{Z}/r\mathbb{Z}$ be such that $Q = dP$. Then a Static DHP$_d$ oracle (with respect to $\mathbb{G}$) computes the function $\delta : \mathbb{G} \to \mathbb{G}$ defined by:*

$$\delta(X) = dX.$$

Likewise a Static DHP$_d$ algorithm is said to be *oracle-assisted* if during an initial learning phase, it can make a number of Static DHP$_d$ queries, after which, given a previously unseen challenge element X, it outputs dX. We now consider how to solve the oracle-assisted Static DHP when $\mathbb{G} = E(\mathbb{F}_{q^n})$.

3 An Oracle-Assisted Static DHP Algorithm for $E(\mathbb{F}_{q^n})$

In this section we motivate and present our algorithm for solving the oracle-assisted Static DHP$_d$ in the present context.

The key observation in [36] is that if one is able to define a suitable 'factor base' in the group under consideration, i.e., a relatively small subset of group elements over which a non-negligible proportion of all group elements can be 'factored' via the group operation, then it is possible to solve the Static DHP$_d$ with input an arbitrary group element, given knowledge of the action of the Static DHP$_d$ oracle on the factor base elements alone. This follows from the simple fact that if in an additively written group $\mathbb{G}$ we have $R = P_1 + \cdots + P_n$, with P_i in some factor base $\mathcal{F}$, then

$$\delta(R) = dR = dP_1 + \cdots + dP_n = \delta(P_1) + \cdots + \delta(P_n).$$

Note that if an arbitrary group element R is not expressible over the factor base, then by adding a random element $Q \in \mathcal{F}$ (or any linear combination thereof) to R and testing expressibility, one can produce an element $R + Q$ which factors over $\mathcal{F}$, thus permitting the Static DHP$_d$ to be solved as before. Therefore a good factor base over which a non-negligible proportion of elements may be expressed, combined with randomisation, enables one to solve the Static DHP$_d$ for arbitrary group elements.

Observe that for the oracle-assisted Static DHP$_d$, one does not ever need to know d in order to compute the action of multiplication by d on an arbitrary element of $\mathbb{G}$, i.e., one can solve the Static DHP$_d$ without solving the DLP. This is because implicit information is 'leaked' via the Static DHP$_d$ oracle queries which enables one to solve the Static DHP$_d$ using the above observations, more readily than one is able to solve the DLP, for which there is no such information. This idea is central to both [34] and [38].

When $\mathbb{G}$ is the multiplicative group of a finite field, the problem of how best to construct a factor base, and how to express arbitrary elements over such a factor base is well studied [35,33,34]. For finite fields there exists a natural notion of size for elements, or equivalently a norm function, given by either the absolute value of an element for prime fields, or the degree of an element for extension fields, or a combination of both depending on the algorithm being used to generate multiplicative relations. A norm function imbues a notion of smoothness for a group and those elements of small norm generate more group elements than those elements of larger norm, hence the best choice for a factor base is those elements of norm up to some bound.

In the context of elliptic curves over prime fields, there does not appear to be a utilisable notion of norm that enables the selection of a factor base that generates a higher proportion of group elements than any other, nor a means by which to factor elements over one should one be chosen. It is precisely this issue that has so far precluded the discovery of a successful native index calculus algorithm for computing discrete logarithms on such curves[2], which is why they are so attractive from a security perspective.

For elliptic curves over extension fields, the story is very different. While the 'Weil descent' methodology [19,25,30] has proven successful for solving or weakening the DLP in some cases, this involves mapping to a generally larger group, which although possessing a natural factor base, does not allow the requisite Static DHP oracle queries to be made on the preimages of the factor base elements, since in general such preimages will not exist. There does however exist a notion of smoothness for such elliptic curves, as remarkably discovered by Gaudry [24].

3.1 Gaudry's Insight

Developing upon an intriguing idea due to Semaev [47], in 2004[3] Gaudry showed how to define a useful factor base for $E(\mathbb{F}_{q^n})$, over which elements can be 'factored', or more properly, decomposed, which leads to an index calculus algorithm for computing logarithms over these curves [24]. For fixed $n > 1$ and $q \to \infty$,

[2] There are of course attacks that apply to a very small minority of elliptic curves [43,20,48,46,45], though these are well understood and are easily avoided, or in the case of pairing-based cryptography, which relies on curves which are susceptible to [43,20], are employed.

[3] Gaudry's algorithm was initially posted to the IACR preprint server in 2004 (paper number 2004/073), but was not published until 2009, in [24].

the algorithm has heuristic complexity $\tilde{O}(q^{2-\frac{2}{n}})$, which is much faster than the Pollard rho complexity $\tilde{O}(q^{n/2})$.

We begin by recalling Semaev's *Summation Polynomials* [47].

Definition 2. *For* $\mathrm{char}(\mathbb{F}_q) > 3$ *let* E *be an elliptic curve defined over* $\mathbb{F}_{q^n}$ *by the equation* $y^2 = x^3 + ax + b$. *The summation polynomials* f_n *of* E *are defined by the following recurrence, with initial values for* $n = 2$ *and* 3 *given by* $f_2(X_1, X_2) = X_1 - X_2$, *and*

$$f_3(X_1, X_2, X_3) = (X_1 - X_2)^2 X_3^2 - 2((X_1 + X_2)(X_1X_2 + a) + 2b)X_3 + ((X_1X_2 - a)^2 - 4b(X_1 + X_2)),$$

and for $n \geq 4$ *and* $1 \leq k \leq n - 3$,

$$f_n(X_1, \ldots, X_n) = Res_X(f_{n-k}(X_1, \ldots, X_{n-k-1}, X), f_{k+2}(X_{n-k}, \ldots, X_n, X)).$$

While this definition may appear rather mysterious, Semaev derived the above formulae by insisting that f_n satisfies the following property, which relates f_n to the addition law on E.

Theorem 1. *(Semaev [47]). Let* E *be an elliptic curve over a field* k, $n \geq 2$ *and* f_n *its* n*-th summation polynomial. Let* $x_1, \ldots, x_n$ *be* n *elements of an algebraic closure* $\overline{k}$ *of* k. *Then* $f_n(x_1, \ldots, x_n) = 0$ *iff there exists an* n*-tuple* $(y_1, \ldots, y_n)$ *of elements in* $\overline{k}$ *such that for all* i, $P_i = (x_i, y_i)$ *is a point on* E *and*

$$P_1 + \cdots + P_n = \mathcal{O}.$$

One can therefore see immediately that f_n provides an encoding for all sets of n points on a given curve whose sum is the identity element. For an elliptic curve E over a prime field $\mathbb{F}_p$, Semaev proposed setting the factor base to be the set be all points on E whose abscissa have magnitude less than $p^{1/n}$. Then one computes random multiples of some base point P, say $R_i = r_iP$, and attempts to write each such R_i as a sum of n points in the factor base. To do this one need only solve

$$f_{n+1}(x_1, \ldots, x_n, x_{R_i}) = 0. \tag{1}$$

By symmetry, one heuristically expects this to be possible for a proportion $1/n!$ of points R_i, and when $O(p^{1/n})$ points that decompose have been found (the approximate size of the factor base) one can obtain their logarithms with respect to P via a sparse linear algebra elimination, which has complexity $\tilde{O}(p^{2/n})$. Finding the logarithm of an arbitrary group element is then easy. Therefore, if finding small roots of (1) were possible, for fixed $n \geq 5$ and $p \to \infty$ this algorithm would be faster than Pollard rho.

Unfortunately, finding such small roots, at least for more than two variables [9], appears hard. Gaudry's insight was to observe that for elliptic curves over $\mathbb{F}_{q^n}$, if one uses a factor base consisting of points with abscissae in the base field $\mathbb{F}_q$, then assuming the field of definition of the curve is $\mathbb{F}_{q^n}$, the Weil restriction of scalars of equation (1) from $\mathbb{F}_{q^n}$ to $\mathbb{F}_q$ forms an algebraic system of n

equations in n indeterminates over $\mathbb{F}_q$, which is nearly always zero-dimensional and which can be solved via elimination theory [24]. Note that the above assumption crucially also ensures that the factor base elements do not form a subgroup. Using a 'double large prime variation' [26] this leads to a DLP algorithm with complexity $\tilde{O}(q^{2-\frac{2}{n}})$. We are now ready to present the basic version of our algorithm, in which we detail how this Weil restriction approach works.

3.2 Basic Oracle-Assisted Static DHP Algorithm

Let E be an elliptic curve whose field of definition is $\mathbb{F}_{q^n}$. We define a factor base $\mathcal{F}$ à la Gaudry [24] as follows:

$$\mathcal{F} = \{P = (x, y) \in E(\mathbb{F}_{q^n}) \mid x \in \mathbb{F}_q\}.$$

On heuristic grounds, one expects $|\mathcal{F}| \approx q$, see [24]. For each $P \in \mathcal{F}$ we make an oracle call to the Static DHP oracle, to give $\delta(P) = dP$.

For an arbitrary point $R \in E(\mathbb{F}_{q^n})$, the goal is to find dR. We attempt write R as a sum of n elements of $\mathcal{F}$, i.e.,

$$R = P_1 + \cdots + P_n.$$

By symmetry, one heuristically expects the proportion of elements expressible in such a way to be approximately $1/n!$. To perform this decomposition one uses Semaev's summation polynomial f_{n+1}, and attempts to solve

$$f_{n+1}(x_1, \ldots, x_n, x_R) = 0 \in \mathbb{F}_{q^n}. \tag{2}$$

Note that the expression on the left of equation (2) involves the defining coefficients of the curve E, and the abscissa x_R, all of which are in $\mathbb{F}_{q^n}$. Fix a polynomial basis $\{1, t, \ldots, t^{n-1}\}$ for the extension $\mathbb{F}_{q^n}/\mathbb{F}_q$. Then each one of the n coefficients of powers of t must be zero. Since each of the n abscissae x_i are in $\mathbb{F}_q$, equation (2) defines a variety with n equations in n indeterminates over $\mathbb{F}_q$, which one solves via a Grobner basis computation, see §3.3.

If there is a solution $(x_1, \ldots, x_n)$ to the system (2), then one needs to compute all 2^n possible combinations $\pm P_1 \pm \cdots \pm P_n$ for the corresponding ordinates in order to find the correct combination which sums to R. Then the solution to the Static DHP for R is immediate:

$$\delta(R) = dR = dP_1 + \cdots + dP_n,$$

where all the terms on the right hand side are already known, due to the oracle queries on $\mathcal{F}$.

As already discussed, if a solution does not exist, then one adds to R a random element $Q \in \mathcal{F}$ (or any linear combination thereof) and attempts to decompose this point once again. One expects this to succeed after approximately $n!$ attempts. When it does we have the following equation:

$$R + Q = P_1 + \cdots + P_n,$$

which implies that

$$\delta(R) = dR = dP_1 + \cdots + dP_n - dQ,$$

where again all the terms on the right hand side are already known. Hence our Static DHP$_d$ instance is solved.

3.3 Discussion

Our first observation is that the above algorithm and this discussion of it are entirely heuristic; however we believe that the algorithm and its complexity can be made completely rigorous using the results of Diem [10,11], should one choose to do so, see §3.4.

Our second observation — which is fundamental to the complexity of the algorithm — is that in contrast to the DLP, there is no linear algebra elimination, since only a single relation is sought. So once the initial oracle querying phase is complete, the complexity of the algorithm depends only on the problem of computing one relation. We therefore analyse this cost now.

For $n \geq 3$, Semaev's summation polynomials $\{f_n\}$ are symmetric and are of degree 2^{n-2} in each variable. Hence equation (2) is of degree 2^{n-1} each variable. In order to simplify the system greatly, it pays to express f_{n+1} in terms of the elementary symmetric functions $e_1, \ldots, e_n$ in the variables $x_1, \ldots, x_n$. We then have a system of n equations in the n indeterminates $e_1, \ldots, e_n$ each of which again has degree bounded by 2^{n-1} in each variable. In order to solve this system, we perform a Gröbner basis computation.

In practice our experiments (see §5) showed that the Gröbner basis w.r.t the lexicographic ordering always satisfies the so-called shape lemma [27,41], i.e., it is of the following form:

$$e_1 - g_1(e_n), e_2 - g_2(e_n), \ldots, e_{n-1} - g_{n-1}(e_n), g_n(e_n), \tag{3}$$

where $g_i(e_n)$ is a univariate polynomial in e_n for each i. In general the degree of the univariate polynomial in e_n that we obtain will be $2^{n(n-1)}$ and indeed in our experiments this is borne out. The complexity of Faugère's algorithm F4 [16] to compute this basis is therefore at least

$$\tilde{O}(Poly(2^{n(n-1)})).$$

Since this is doubly exponential in n, this makes the algorithm practical only for very small values of n. However for fixed n and $q \to \infty$, this is polynomial in $\log q$.

To find whether or not the system has roots $e_1, \ldots, e_n \in \mathbb{F}_q$, one extracts the linear factors of the univariate polynomial $g_n(e_n)$ using a gcd computation with $e_n^q - e_n$ followed by Cantor-Zassenhaus and then substitutes each $\mathbb{F}_q$ root e_n into $g_i(e_n)$ to find $e_{n-1}, \ldots, e_1$. For each such vector of $\mathbb{F}_q$ roots $(e_1, \ldots, e_n)$ one tests whether the polynomial

$$p(x) = x^n - e_1 x^{n-1} + e_2 x^{n-2} - \cdots - (-1)^n e_n \tag{4}$$

splits over $\mathbb{F}_q$. If it does then these roots are the abscissae of points in $E(\mathbb{F}_q)$, and there exists a linear combination

$$\epsilon_1 P_1 + \cdots + \epsilon_n P_n \tag{5}$$

with $\epsilon_i \in \{-1, 1\}$ which sums to R. This step is also polynomial in $\log q$.

On average one expects to have to perform $n!$ such decompositions in order to find a relation. Therefore the complexity of the our basic Static DHP algorithm for fixed $n > 1$ and $q \to \infty$ is polynomial in $\log q$. This gives the following heuristic result.

Heuristic Result 1. *For any elliptic curve $E(\mathbb{F}_{q^n})$, by making $O(q)$ queries to a Static DHP$_d$ oracle during an initial learning phase, for fixed $n > 1$ and $q \to \infty$, an adversary can solve any further instance of the Static DHP$_d$ in time Poly$(\log q)$.*

Note that prior to the learning phase, the adversary needs to construct the factor base by testing whether a given abscissa $x \in \mathbb{F}_q$ gives a point lying on E or not. We incorporate this computation into the learning phase, since it has the same complexity of $\tilde{O}(q)$. It is of course possible to balance the cost of the learning and relation-finding phases, which we now consider.

3.4 Balancing the Setup and Relation-Finding Costs

To balance the cost of the oracle querying phase and the relation finding phase, one needs to reduce the size of the factor base by some proportion. To this end, Let $|\mathcal{F}| = q^\alpha$, with $0 < \alpha \leq 1$. Then given the decomposition of a random point $R \in E$ as a sum of points whose abscissa are in $\mathbb{F}_q$, the probability that a single abscissa is in $\mathcal{F}$ is $q^{\alpha-1}$. Assuming these events are independent, the probability that all n abscissae are in $\mathcal{F}$ is $q^{n(\alpha-1)}$. Hence in order to obtain one relation, one expects to have to perform $1/q^{n(\alpha-1)} = q^{n(1-\alpha)}$ successful decompositions.

Asymptotically for fixed $n > 1$ and $q \to \infty$ one can regard the cost of a decomposition as unital (modulo some log factors) and hence to balance the two stages α must satisfy:

$$q^\alpha = q^{n(1-\alpha)},$$

and so $\alpha = n/(n+1) = 1 - \frac{1}{n+1}$. This gives the following heuristic result as stated in the abstract.

Heuristic Result 2. *For any elliptic curve $E(\mathbb{F}_{q^n})$, by making $O(q^{1-\frac{1}{n+1}})$ queries to a Static DHP$_d$ oracle during an initial learning phase, for fixed $n > 1$ and $q \to \infty$, an adversary can solve any further instance of the Static DHP$_d$ in time $\tilde{O}(q^{1-\frac{1}{n+1}})$.*

Observe that there is no possibility (nor necessity) for considering so-called large primes, i.e., those with absicissa in $\mathbb{F}_q$ but not lying in $\mathcal{F}$, since there is no linear algebra elimination step on the single relation. If we compare the above complexity to that obtained by Gaudry for the DLP, namely $\tilde{O}(q^{2-\frac{2}{n}})$, which uses a

double large-prime variant, we see that our algorithm for solving the Static DHP approaches being a square root faster for increasing n. Intuitively this difference in complexity arises from there not being a linear algebra step in the solution of the Static DHP$_d$.

We note that Diem has given a rigorous algorithm that is essentially equivalent to Gaudry's DLP algorithm above [10], which for fixed $n \geq 2$ solves the DLP on any elliptic curve over $\mathbb{F}_{q^n}$ in proven expected time $q^{2-2/n}(\log q)^{O(1)}$. We believe his treatment can be adapted *mutatis mutandis* to transform the above two heuristic results into theorems, though since it is not the primary focus of this paper, we have not verified this here.

Observe that in practice the limiting factor is not the decompositions, but the oracle queries, since these would typically be performed on a single server, whereas the former can be easily distributed. One can therefore reduce the number of such queries below the above threshold, at the expense of needing to perform more decompositions. Such a trade-off is easily optimised, based on the amount of computing power available, but will nevertheless require an exponential number of oracle queries, for fixed n and $q \to \infty$. We now consider how and when the number of oracle queries may be made subexponential in the size of the group.

3.5 Subexponential Oracle-Assisted Static DHP Algorithm

Diem has also proven the following remarkable result [11]. For $n \to \infty$ and assuming $n = O(\sqrt{\log q})$, the DLP over any elliptic curve $E(\mathbb{F}_{q^n})$ can be solved in expected time $q^{O(1)} = e^{O(\log(q^n)^{2/3})}$. Thus for a family of finite fields, any elliptic curve DLP can be solved using a *native* subexponential index calculus algorithm.

While Diem is not precise in his analysis of the exponents in the complexity of the constituent parts of the algorithm, it is clear that since for the oracle-assisted Static DHP$_d$ there is no linear algebra step, one expects a similar improvement over the DLP algorithm in this context to the fixed n case, i.e., nearly square root, and that this also can be rigorously proven. This therefore provides an oracle-assisted Static DHP$_d$ algorithm that requires a subexponential number of oracle queries. We leave it as an open problem to find the precise complexity of Diem's algorithm, and the resulting complexity of our algorithm in this context.

4 Potentially Vulnerable Curves

At EUROCRYPT 2009, Galbraith, Lin and Scott proposed the use of special elliptic curves over $\mathbb{F}_{p^2}$ and $\mathbb{F}_{p^4}$ [21], which possess efficiently computable homomorphisms that permit a particularly efficient version of Gallant-Lambert-Vanstone point multiplciation method [23]. As well as the single bit speed-up of Pollard rho available on these curves, both the GHS attack [25] and Gaudry's attack [24] are considered, and appropriate recommendations are made in light of these. In particular, for curves over $\mathbb{F}_{p^2}$, neither of these attacks is faster than

Pollard rho, and so these curves were believed to attain the desired security level. For curves over $\mathbb{F}_{p^4}$, in light of the latter attack the authors recommend that primes of length 80 bits should be used to achieve 128-bit security, rather than of length 64 bits, although it is stated that this is a very conservative choice, since Gaudry's algorithm requires expensive computations, and so potentially smaller primes could be used. Similarly Hankerson, Karabina and Menezes have considered the GLS point multiplication method over binary fields of the form $\mathbb{F}_{q^2}$ [29].

Prior to our attack, the only potential weakness of cryptographically interesting curves over $\mathbb{F}_{p^2}$ would be due to the Brown-Gallant-Cheon attack. In the best case (from an adversary's perspective), should the group order ± 1 be divisible by an integer of size $O(p^{2/3})$, then the Static DHP$_d$ secret d can be computed in time $\tilde{O}(p^{2/3})$. Such a condition can be easily avoided should this attack be a concern. For the curves considered in [29], the Weil descent method is analysed and it is shown that the proportion of susceptible curves is negligible and can be provably avoided with a feasible computation. However, regardless of the divisibility properties of the group order ± 1, the balanced oracle-assisted Static DHP$_d$ algorithm from §3.4 achieves a complexity of $\tilde{O}(p^{2/3})$ (and similarly for the binary curves). Assuming that point decompositions over the factor base can be computed efficiently, this attack therefore poses a real threat.

For curves over $\mathbb{F}_{p^4}$, our attack has complexity $\tilde{O}(p^{4/5})$, which is much faster than Gaudry's attack on the DLP, which has complexity $\tilde{O}(p^{3/2})$. Again assuming that point decompositions can be performed efficiently, curves over degree 4 extensions are also vulnerable.

Also of interest are the legacy curves which until recently formed part of the Oakley Key Determination Protocol, a part of IPSEC. These are the 'Well Known Groups' 3 and 4 [31] which are elliptic curves defined over the fields $\mathbb{F}_{2^{155}}$ and $\mathbb{F}_{2^{185}}$, and which have been the target of numerous attempted attacks via the Weil descent method [49,25,22,44], since their inception.

5 Experimental Results

We implementated our oracle-assisted Static DHP$_d$ algorithm using the computational algebra system MAGMA [5] (V2.16-5), which was run on an Intel Xeon running at 3.16GHz with 32G of memory. We considered two sets of curves. The first set consisted of four randomly selected curves of prime order, each of which were 256 bits in length, for fields of the form $\mathbb{F}_{p^2}$, $\mathbb{F}_{p^3}$, $\mathbb{F}_{p^4}$ and $\mathbb{F}_{p^5}$, see §5.1 and §5.2. These curves were chosen in order to measure how vulnerable the curves proposed in [21] are to our algorithm. We also provide estimates for solving the DLP on these curves via Pollard rho and the state of the art index calculus algorithms. The second set consisted of four randomly selected curves of order $4 \cdot p$ with p of bitlength 256 over the binary fields $\mathbb{F}_{2^{ln}}$, for $n = 2, 3, 4$ and 5, so that ln was as close to 256 as possible. The reason for implementing the attack on these curves was twofold: firstly to assess the security of the curves proposed in [29]; and secondly to compare the efficiency of the attack with the prime

field case, with a view to assessing the difficulty of breaking the oracle-assisted Static DHP$_d$ on the Oakley curves, which we report in §5.3.

While our implementations in MAGMA are clearly sub-optimal, our goal was to provide a proof-of-concept implementation, and to give a reasonable indication of what can be achieved in practice. Indeed our results provide an upper bound for the time required to solve the oracle-assisted Static DHP$_d$ in each case. With a tailored and optimised low-level implementation our attack times can be improved significantly, as exemplified by the result reported in [28].

5.1 Large Prime Characteristic

For each of $n = 2, 3, 4$ and 5 we used curves of the form

$$E(\mathbb{F}_{p^n}) : y^2 = x^3 + ax + b,$$

for a and b randomly chosen elements of $\mathbb{F}_{p^n}$, such that $\#E(\mathbb{F}_{p^n})$ was a prime of bitlength 256.

For $n = 2, 3$ and 4 we computed the symmetrised summation polynomials f_3, f_4 and f_5 respectively, and all experiments were completed within two hours. For the computation of f_6, we surprisingly ran out of memory, and so instead independently symmetrised the two f_4 polynomials used in the resultant computation to reduce the number of terms, and substituted x_R into this partially symmmetrised version of f_6. One can extract the elementary symmetric polynomials from these two independent sets by appropriately recombining them. However the resulting Gröbner basis computation eventually exhausted the available memory and so the $n = 5$ experiments were unable to be completed. Without an accurate idea of how long the Gröbner basis computation might take were we to have sufficient available memory, we consider finding relations for curves over these fields to be impractical given our resources at the present time. Note however that for prime base fields, we know of no proposals in the literature for the use of degree five extension fields for elliptic curve cryptography. We therefore include results only for $n = 2, 3$ and 4, in Table 1.

The column titles in the table denote respectively: the degree of the extension field; the size of the prime base field in bits; the number of monomials in f_{n+1}; the number of monomials in f_{n+1} once symmetrised; the average time required to perform a Gröbner basis computation; and the average time required to find the points that sum to the point being decomposed respectively.

Table 1. Data for testing and decomposing points for elliptic curves over extension fields. Times are in seconds.

n	$\log p$	$\#f_{n+1}$	$\#$ symf_{n+1}	T(GB)	T(roots)
2	128	13	5	0.001	0.009
3	85.3	439	43	0.029	0.027
4	64	54777	1100	5363	3.68

As per §3.3 the last of these consists of the extraction of the degree one factors of the polynomial $g_n(e_n)$ and then substitutes the roots into the remaining polynomials $g_i(e_n)$ in equation (3). This is followed by the desymmetrisation factorisation (equation (4)) and then computation of the correct linear combination of factor base elements that sum to P (equation (5)).

As one can see, symmetrisation reduces the size of the system greatly. Note that the only setup cost comes from computing f_{n+1} and its symmetrisation; the final two columns give the average decomposition cost per input point, which for $n = 2$ and 3 is over 1000 inputs includes both those that do decompose over $\mathcal{F}$, as well as those that do not.

For $n = 4$, since the computation is significantly more costly, we report the time for one input point only; note that the input system for the Gröbner basis computation always has the same form but with different coefficients, and hence one expects this part of the computation to be very consistent. With regards to the root finding time, the three stages described above took $3.68s, 0.00s$ and $0.04s$ respectively, and so the dominant cost is the initial factorisation, which is necessary whether an input point decomposes or not. Hence we estimate the average time over uniformly chosen input points to be $\approx 3.68 + 0.04/4! \approx 3.68s$, since a point decomposes with probability $1/4!$.

5.2 Upper Bounds on Attack Times

From the data in Table 1 and the time required to compute a scalar multiplication, one can compute an upper bound on the time required to carry out the attack in §3.4. Setting $|\mathcal{F}| = p^\alpha$, a minimising α balances the two stages of the attack, namely the oracle calls, and the relation finding stage. We ignore the cost of constructing the factor base since this only involves a handful of field operations and a Legendre symbol computation. A more careful version of the argument of §3.4 leads to the following equation:

$$p^{n(1-\alpha)} \cdot n! \cdot (T(\text{GB}) + T(\text{roots})) = p^\alpha \cdot T(\text{scalar}),$$

where $T(\text{scalar})$ denotes the average cost of a scalar multiplication. With our implementation the latter costs approximately $0.008s$, $0.011s$ and $0.012s$ on the curves defined over $\mathbb{F}_{p^2}$, $\mathbb{F}_{p^3}$ and $\mathbb{F}_{p^4}$ respectively.

Table 2 details the resulting values of alpha for $n = 2, 3, 4$ and the corresponding estimated attack times. As stated in §3.4, these estimates assume that each set of q^α factor base elements has the same probability of expressing the decomposition a random decomposable point as a linear sum of elements from that set.

The Pollard rho attack times have been estimated as $\sqrt{\pi \cdot 2^{256}/2}$ group operations, where the cost of a group operation has been estimated using the $T(\text{scalar})$ times above, assuming use of the double and add algorithm. We have incorporated the speed-up afforded by performing random walks on equivalence classes of points [14,51] when the set of points $\{\pm\psi^i(P) : 0 \leq i < m\}$ for a given point P are deemed to be equivalent, where ψ is the homomorphism from [21]. This results in the three curves have virtually identical security.

Table 2. Attack time estimates for our implementation. Times are in seconds.

n	α	Attack time	Pollard rho
2	0.6701 (2/3)	$2^{79.8}$	$2^{111.3}$
3	0.7645 (3/4)	$2^{59.7}$	$2^{111.4}$
4	0.8730 (4/5)	$2^{50.5}$	$2^{111.4}$

Pollard rho however is not the fastest asymptotic DLP algorithm in this context. In the basic index calculus one finds $O(p)$ relations with a linear algebra cost of $O(p^2)$. Assuming the decomposition cost is sufficiently small, one can reduce the size of the factor base to balance the cost of the two stages, to $O(p^{2-\frac{2}{n+1}})$, which is originally due to Harley. In addition, one can also use single and double large prime variations [50,26], resulting in complexities of $O(p^{2-\frac{2}{n+1/2}})$ and $O(p^{2-\frac{2}{n}})$ respectively.

Our implementation allows one to give upper bounds for the attack times for each of these approaches, and consequently provides information regarding what size of p should be chosen to provide 128 bit security, for each n, subject to our attack implementation. This security level is the length of time required to compute 2^{128} basic group operations. Note that in the double large prime variation, for the most interesting case $n = 4$ the number of relations required is $O(p^{3/2})$. With our decomposition implementation, the time for the relation generation stage is $p^{3/2} \cdot 4! \cdot 5366.68s \approx 2^{113.0}s$, which is comparable to Pollard rho. Hence for this security level, p of length 64 bits would appear to be secure. However, in an optimised implementation the decomposition time could clearly be improved, necessitating increasing p accordingly to compensate. Furthermore, since the relation generation stage is more costly than the linear algebra, to balance the two stages of the algorithm one would need to increase the factor base size marginally. These intricacies mean that although our implementation provides an upper bound for the attack time, how to select an appropriate size p to ensure security for elliptic curves over these extension fields remains an open issue.

5.3 Characteristic Two

For each of $n = 2, 3, 4$ and 5 we used curves of the form

$$E(\mathbb{F}_{2^{ln}}) : y^2 + xy = x^3 + b, \tag{6}$$

for b a randomly chosen element of $\mathbb{F}_{2^{ln}}$, such that $\#E(\mathbb{F}_{2^{ln}})$ was a four times a prime of bitlength 256. Note that (6) is the form of the Oakley curves [31]. Also note that the base fields $\mathbb{F}_{2^l}$ in each case are not necessarily of prime extension degree over $\mathbb{F}_2$. Since our focus was to compare the effect of characteristic for fields of a given size with particular small extension degrees, we disregard any possible DLP weaknesses due to Weil descent for these example curves.

For these curves the summation polynomials are surprisingly simple, and very sparse, making their computation easy, in contrast to the prime base field case.

Observe that as a result the size of the f_i and their symmetrisation is much smaller than before, faciliting a much faster Gröbner basis computation for $n = 4$.

As was the case for prime base fields, for $n = 5$ we also had insufficient memory to complete a decomposition using Gaudry's method. However, as stated in §1 and announced in [28], by attempting to write a random point on the curve as a sum of four factor base elements as in [37] one is able to find such a decomposition, at the expense of reducing the probability from $1/5!$ to $1/(2^l \cdot 4!)$. As with Gaudry's decomposition method, this method is much faster in characteristic two than in large prime characteristic. Thus for the Oakley Group 3 curve, the oracle-assisted Static DHP$_d$ problem is practical. Whether this can be extended to the Oakley Group 4 curve is worthy of further investigation.

For $n = 2, 3$ and 4 the time for a scalar multiplication is $0.014s$. Table 3 details the results using Gaudry's decomposition technique, with the final row detailing the results from the announcement [28] on the Oakley Group 3 curve.

Table 3. Data for testing and decomposing points for elliptic curves over binary extension fields and attack time estimates. Times are in seconds.

n	l	$\#f_{n+1}$	$\#$ symf_{n+1}	Time GB	Time roots	α	Attack time
2	129	5	3	0.000	0.008	0.6672 (2/3)	$2^{80.9}$
3	86	24	6	0.005	0.008	0.7572 (3/4)	$2^{60.0}$
4	65	729	39	247	0.88	0.8575 (4/5)	$2^{50.6}$
5	52	148300	638	N/A	N/A	N/A	N/A
5	31	729	39	0.021	(total time)	30/31	$2^{30.0}$

Note that despite the α values being smaller for binary fields — due to faster decompositions — the attack times are slightly higher, because the fields are 258 and 260 bits in size, as opposed to 256 bits. Due to the scalar multiplication time being very similar to the prime field case (with our implementation), the Pollard rho times are similar and hence the curves in [29] should also be considered vulnerable to our attack.

6 Other Cryptographically Relevant Assumptions

Our proposed oracle-assisted Static DHP$_d$ algorithm also solves the *Delayed Target DHP*, as defined by Freeman [18], which may be phrased as follows: A solver is given initial access to a Static DHP$_d$ oracle for the element $Q = dP \in \mathbb{G}$; when the Static DHP$_d$ oracle is removed, the solver is given a random element $X \in \mathbb{G}$ and must solve the DHP for input (Q, X), namely, output dX.

Koblitz and Menezes studied this problem in the context of Jacobians of hyperelliptic curves of small genus [38], along with several other problems, including the *Delayed Target DLP*, the *One-More DHP* and the *One-More DLP*. In the Delayed Target DLP, rather than given access to a Static DHP$_d$ oracle, the solver is given access to a discrete logarithm oracle but the problem is otherwise

identical to the Delayed Target DHP. In the One-More DHP and One-More DLP the solver is supplied with a challenge oracle that outputs random elements of the group, as well as a Static DHP_d oracle or a DLP oracle respectively. This time however the solver chooses an integer t and must solve t instances of the Static DHP_d or the DLP, but is only allowed to use the Static DHP_d or the DLP oracle at most $t-1$ times.

The One-More DHP was first formulated in [3] while the One-More DLP was first formulated in [1] and [2]. Using Jacobians of hyperelliptic curves of small genus as example groups, Koblitz and Menezes argue that the constituents of each of the two pairs of similar problems — the Delayed Target DHP and Delayed Target DLP, and the One-More DHP and One-More DLP — should each be incomparable to one another. In particular there very probably does not exist a reduction between the Delayed Target DHP and the Delayed Target DLP, since in some groups the former appears to be easier than the latter, while in others the converse is true, and similarly for the One-More problems. However, their analysis of the Delayed Target DHP and One-More DHP contains a minor oversight, since it only considers the impact of the Brown-Gallant-Cheon algorithm and not the index calculus methods they used for studying the corresponding DLP versions. Doing so for Jacobians of hyperelliptic curves of genus ≥ 3, one sees that the complexities for the Delayed Target problems are identical, and similarly for the One-More problem variants.

Indeed, taking the basic Static DHP_d algorithm presented in §3.2, one sees that by changing the Static DHP_d oracle calls to DLP oracle calls, one obtains an otherwise unaltered algorithm and hence the complexities of the two delayed target problems are the same. Similarly any variation in factor base size will give rise to algorithms of the same complexity; the oracle calls themselves are not relevant to the structure of the algorithm, so it should be clear that for any group in which one can identify and use a factor base to generate relations, the Delayed Target DHP and Delayed Target DLP will have identical complexities, *whenever this method provides the most effective means to solve both problems.* Exceptions to this condition arise, for instance, when a faster algorithm applies to just one problem, as with the Brown-Gallant-Cheon algorithm for the Delayed Target DHP, for an elliptic curve over $\mathbb{F}_p$ whose group order ± 1 is divisible by an integer of size $\approx p^{1/3}$.

For the One-More problem variants, in our context we have the following simple algorithm. We choose the same factor base as in §3.2, and perform $|\mathcal{F}|$ Static DHP_d oracle calls on its elements. Then for each of the $|\mathcal{F}|+1$ challenge elements, we solve the appropriate problem exactly as before. The only difference between the one-more and the delayed target problems is that for the one-more variants we must solve $|\mathcal{F}|+1$ such challenges, and not just one. If we perform the analysis of §3.4 once more we find that the optimal size of $\mathcal{F}$ is given by $\alpha = 1$, exactly as in §4.5 of [38]. As before either oracle can be applied to a given relation and so the One-More DHP and One-More DLP have the same complexity, and again will do so in any group *for which this method provides the most effective means to solve both problems.*

Interestingly this means that even when one can not find a natural reduction between two problems, the presence of an effective index calculus ensures that in some circumstances the problems have the same complexity. Furthermore the two pairs of problems considered above (as well as oracle-assisted Static DHP_d) are easier to solve than the DLP, for elliptic curves over extension fields, Jacobians of hyperelliptic curves of genus ≥ 3, and indeed for any group for which index calculus provides the best means to solve each of these problems.

Acknowledgements

The author would like to thank both Steven Galbraith and Alfred Menezes for their comments on an earlier draft of this article, and the reviewers for their suggestions.

References

1. 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, 185–215 (2003)
2. Bellare, M., Palacio, A.: GQ and Schnorr identification schemes: proofs of security against impersonation under active and concurrent attacks. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 149–162. Springer, Heidelberg (2002)
3. Boldyreva, A.: Efficient threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 31–46. Springer, Heidelberg (2003)
4. Boneh, D., Boyen, X.: Short signatures without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004)
5. Bosma, W., Cannon, J., Playoust, C.: The Magma algebra system I: The user language. J. Symbolic Comput., 24(3-4), 235–265 (1997)
6. Brown, D.R.L., Gallant, R.P.: The Static Diffie-Hellman Problem, Cryptology ePrint Archive, Report 2004/306 (2004)
7. Chaum, D., van Antwerpen, H.: Undeniable signatures. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 212–217. Springer, Heidelberg (1990)
8. Cheon, J.: Security analysis of the Strong Diffie-Hellman problem. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 1–11. Springer, Heidelberg (2006)
9. 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)
10. Diem, C.: On the discrete logarithm problem in class groups of curves. Mathematics of Computation (to appear)
11. Diem, C.: On the discrete logarithm problem in elliptic curves (2009) (preprint)
12. Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Trans. Inform. Theory 22(6), 644–654 (1976)
13. Digital Signature Standard (DSS). FIPS PUB 186-2 (2000)
14. Duursma, I., 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)

15. El Gamal, T.: A public-key cryptosystem and a signature scheme based on discrete logarithms. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 10–18. Springer, Heidelberg (1985)
16. Faugère, J.C.: A new efficient algorithm for computing Gröbner bases (F4). Journal of Pure and Applied Algebra 139(1-3), 61–88 (1999)
17. Ford, W., Kaliski, B.: Server-assisted generation of a strong secret from a password. In: 9th International Workshop on Enabling Technologies, WET ICE 2000, IEEE Press, Los Alamitos (2000)
18. Freeman, D.: Pairing-based identification schemes, technical report HPL-2005-154, Hewlett-Packard Laboratories (2005)
19. Frey, G.: How to disguise an elliptic curve, Talk at Waterloo Workshop on the ECDLP (1998), http://cacr.math.uwaterloo.ca/conferences/1998/ecc98/slides.html
20. Frey, G., Rück, H.G.: A remark concerning m-divisibility and the discrete logarithm problem in the divisor class group of curves. Math. Comp., 62, 865–874 (1994)
21. Galbraith, S.D., Lin, X., Scott, M.: Endomorphisms for Faster Elliptic Curve Cryptography on a Large Class of Curves. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 518–535. Springer, Heidelberg (2010)
22. Galbraith, S.D., Hess, F., Smart, N.P.: Extending the GHS Weil Descent Attack. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 29–44. Springer, Heidelberg (2002)
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.: Index calculus for abelian varieties of small dimension and the elliptic curve discrete logarithm problem. Journal of Symbolic Computation 44, 1690–1702 (2009)
25. Gaudry, P., Hess, F., Smart, N.P.: Constructive and destructive facets of Weil descent on elliptic curves. Journal of Cryptology 15, 19–46 (2002)
26. Gaudry, P., Thomé, E., Thériault, N., Diem, C.: A Double Large Prime Variation for Small Genus Hyperelliptic Index Calculus. Math. Comp. 76(257), 475–492 (2007)
27. Gianni, P., Mora, T.: Algebraic solution of systems of polynomial equation using Gröbner bases. In: Huguet, L., Poli, A. (eds.) AAECC 1987. LNCS, vol. 356, pp. 247–257. Springer, Heidelberg (1989)
28. Granger, R., Joux, A., Vitse, V.: New timings for oracle-assisted SDHP on the IPSEC Oakley 'Well Known Group' 3 curve. Web announcement on Number Theory List (July 8th, 2010), http://listserv.nodak.edu/archives/nmbrthry.html
29. Hankerson, D., Karabina, K., Menezes, A.J.: Analyzing the Galbraith-Lin-Scott point multiplication method for elliptic curves over binary fields. IEEE Transactions on Computers 58, 1411–1420 (2009)
30. Hess, F.: The GHS Attack Revisited. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 374–387. Springer, Heidelberg (2003)
31. IETF, The Oakley Key Determination Protocol, IETF RFC 2412 (November 1998)
32. Jao, D., Yoshida, K.: Boneh-Boyen Signatures and the Strong Diffie-Hellman Problem. In: Shacham, H., Waters, B. (eds.) Pairing-Based Cryptography – Pairing 2009. LNCS, vol. 5671, pp. 1–16. Springer, Heidelberg (2009)
33. 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)

34. Joux, A., Lercier, R., Naccache, D., Thomé, E.: Oracle-Assisted Static Diffie-Hellman Is Easier Than Discrete Logarithms. In: Parker, M.G. (ed.) Cryptography and Coding. LNCS, vol. 5921, pp. 351–367. Springer, Heidelberg (2009)
35. Joux, A., Lercier, R., Smart, N.P., Vercauteren, F.: The Number Field Sieve in the Medium Prime Case. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 326–344. Springer, Heidelberg (2006)
36. Joux, A., Naccache, D., Thomé, E.: When e-th roots become easier than factoring. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 13–28. Springer, Heidelberg (2007)
37. Joux, A., Vitse, V.: Elliptic Curve Discrete Logarithm Problem over Small Degree Extension Fields. Application to the static Diffie-Hellman problem on $E(Fq^5)$, Cryptology ePrint Archive, Report 2010/157 (2010)
38. Koblitz, N., Menezes, A.J.: Another look at non-standard discrete log and Diffie-Hellman problems. Journal of Mathematical Cryptology 2(4), 311–326 (2008)
39. Koblitz, N., Menezes, A.J.: Intractable problems in cryptography. In: Proceedings of the 9th International Conference on Finite Fields and Their Applications. AMS, Providence
40. Koblitz, N., Menezes, A.J.: The brave new world of bodacious assumptions in cryptography. Notices of the AMS 57, 357–365 (2010)
41. Lakshman, Y.N.: On the complexity of computing Gröbner bases for zero-dimensional ideals. Ph. D.Thesis, RPI, Troy (1990)
42. Maurer, U.M., Wolf, S.: The Diffie-Hellman Protocol. Designs, Codes, and Cryptography 19, 147–171 (2000)
43. Menezes, A.J., Okamoto, T., Vanstone, S.A.: Reducing elliptic curve logarithms to a finite field. IEEE Trans. Info. Theory 39, 1639–1646 (1993)
44. Menezes, A., Teske, E., Weng, A.: Weak Fields for ECC. In: Okamoto, T. (ed.) CT-RSA 2004. LNCS, vol. 2964, pp. 366–386. Springer, Heidelberg (2004)
45. Satoh, T., Araki, K.: Fermat quotients and the polynomial time discrete log algorithm for anomalous elliptic curves. Comm. Math. Univ. Sancti Pauli 47, 81–92 (1998)
46. Semaev, I.A.: Evaluation of discrete logarithms on some elliptic curves. Math. Comp., 67, 353–356 (1998)
47. Semaev, I.: Summation Polynomials and the discrete logarithm problem on elliptic curves, Cryptology ePrint Archive, Report 2004/031 (2004)
48. Smart, N.P.: The discrete logarithm problem on elliptic curves of trace one. Journal of Cryptology 12, 141–151 (1999)
49. Smart, N.P.: How Secure are Elliptic Curves over Composite Extension Fields? In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 30–39. Springer, Heidelberg (2001)
50. Thériault, N.: Index calculus attack for hyperelliptic curves of small genus. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 75–92. Springer, Heidelberg (2003)
51. 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)

Random Oracles with(out) Programmability*

Marc Fischlin[1], Anja Lehmann[2], Thomas Ristenpart[3], Thomas Shrimpton[4], Martijn Stam[5], and Stefano Tessaro[3]

[1] Darmstadt University of Technology, Germany
[2] IBM Research Zurich, Switzerland
[3] University of California, San Diego, USA
[4] Portland State University, Portland, Oregon, USA
[5] Laboratory for Cryptologic Algorithms (LACAL), EPFL, Lausanne, Switzerland

Abstract. This paper investigates the Random Oracle Model (ROM) feature known as *programmability*, which allows security reductions in the ROM to dynamically choose the range points of an ideal hash function. This property is interesting for at least two reasons: first, because of its seeming artificiality (no standard model hash function is known to support such adaptive programming); second, the only known security reductions for many important cryptographic schemes rely fundamentally on programming. We provide formal tools to study the role of programmability in provable security. This includes a framework describing three levels of programming in reductions (none, limited, and full). We then prove that *no* black-box reductions can be given for FDH signatures when only limited programming is allowed, giving formal support for the intuition that full programming is fundamental to the provable security of FDH. We also show that Shoup's trapdoor-permutation-based key-encapsulation is provably CCA-secure with limited programmability, but no black-box reduction succeeds when no programming at all is permitted. Our negative results use a new concrete-security variant of Hsiao and Reyzin's two-oracle separation technique.

Keywords: hash functions, random oracle model, programmability, indifferentiability framework.

1 Introduction

In the random oracle model (ROM) [1] parties are provided oracle access to a publicly available random function, a random oracle (RO). A random oracle is

* 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. Marc Fischlin and Anja Lehmann were supported by the Emmy Noether Grant Fi 940/2-1 of the German Research Foundation (DFG). Marc is also supported by CASED (`www.cased.de`). Anja's work was done at TU Darmstadt. Thomas Ristenpart is supported by NSF grants CCF-0915675, CNS-0627779. Thomas Shrimpton is supported by NSF grants CNS-0627752, CNS-0845610. Stefano Tessaro's work was done at ETH Zurich, partially supported by the Swiss National Science Foundation (SNF), project no. 200020-113700/1.

M. Abe (Ed.): ASIACRYPT 2010, LNCS 6477, pp. 303–320, 2010.

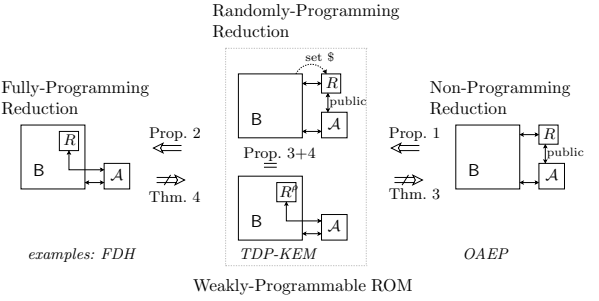


Fig. 1. Relations between the proposed models and example results. A "public" interface indicates that reduction B can see all queries of adversary $\mathcal{A}$, whereas "set $" denotes that B can re-assign random values to R, and R^ρ denotes a weakly-programmable RO. The "example: xxx" labels indicate a scheme xxx that enjoys a proof of security in the model above it, but for which we show black-box separation results implying the difficulty of proving its security in models to the right.

often viewed as the idealization of a cryptographic hash function, and security proofs in the ROM provide heuristic support for actual security when real hash functions are used instead. The ROM enables proofs of security for a multitude of important schemes because reductions may exploit various properties of a RO that can be realized only to a limited extent (if at all) in the standard model.

One such property is *programmability*. Loosely speaking, a random oracle can be "implemented" by dynamically selecting return values, and so long as the distribution of outputs is correct (uniform on the specified range), any method for selecting these values is permitted. The technique of programming RO output values in a security reduction seems crucial in countless positive results, e.g. [1, 3–5,7]. However, no standard model function is known to provide the completely arbitrary and adaptive programmability provided by a RO, making it natural to wonder: which (if any) of these results could have been established *without* exploiting the full programmability of the ROM?

In this paper we formally explore models in which programmability of the random oracle in reductions is restricted. For this, we propose a form of limited programmability that is between full and no programmability. We provide two different but, surprisingly, equivalent characterizations of this limited form of programmability. We use them to show that: (1) one can prove (using a new variant of the Hsiao-Reyzin two-oracle separation technique [11]) the inability to give a programming-limited black-box reduction for the FDH signature scheme [1] and (2) that Shoup's trapdoor-permutation-based key encapsulation scheme (TDP-KEM) [15] is provably CCA secure given only limited programmability, while no black-box reduction works when programming is forbidden. For a diagrammatic summary of our main results, see Figure 1.

Modeling (Non-)Programmability in Reductions. Nielsen [13] was the first to formally investigate the role of programmability in security results. He showed that there is no way to realize a natural cryptographic functionality

(non-committing non-interactive encryption) in a ROM-like model that strictly prohibits programming of RO outputs. His result, and a more recent one by Wee [17] in the context of zero-knowledge, apply to simulation-based notions of security, and in particular restrict the ability of the simulator in these to program the RO. Unfortunately, these approaches are not sufficient to reason about the majority of ROM security proofs, which exploit programmability in *security reductions*, for example to embed an instance of a hard problem into RO outputs.

Our work considers this complementary direction, by investigating security reductions in models equipped with random oracles, but in which the ability of the reduction to program the random oracle is constrained. Along the lines of Nielsen's approach [13] in the simulation-based setting, our first contribution is to formalize *non-programming reductions* in the black-box (BB) setting (i.e. reductions only have oracle access to adversaries) by making the reduction work relative to an external RO (to which the adversary also has access).[1] We then propose a natural relaxation called *randomly-programming reductions.* Intuitively, the external random oracle is realized by a message-indexed table of randomly-chosen points, and while the reduction does not get to pick the range points, it can pick the order they appear in the table. As we shall see, this limitation on programming realizes an interesting middle point between full and no programming, and one that captures the provability of important schemes. Finally, a *fully-programming reduction* allows the reduction to arbitrarily choose output range values, as in traditional ROM proofs.

The Weakly-Programmable Random Oracle Model. A limitation of the above reduction-centric approach is the restriction to BB reductions. Indeed, as observed by Nielsen [13], providing models in which one can argue about limitations on programmability in non-BB reductions is challenging. This is because, in a non-BB setting, the reduction directly simulates all oracle queries made by an adversary and so there is no way to force the reduction to work relative to an external RO. We resolve this difficulty for the case of randomly-programming reductions by proposing a new variant of the ROM.

A Weakly-Programmable Random Oracle (WPRO) works as follows to form an idealized model of a hash function. Let ρ be an arbitrary function (whose range matches that of the hash function). For each distinct input x, the WPRO chooses its output to be $\rho(r)$ for a random coin-string r. Additionally, the WPRO allows *only* adversaries to obtain the coins r used to generate any output. Then in the WPRO model (WPROM) all parties have access to a WPRO that uses a regular, one-way function ρ. The requirements on ρ ensure that the WPROM limits programmability. For example, attempting to program an output of the oracle to a given value y requires computing $\rho^{-1}(y)$ and refuting one-wayness; regularity implies that the output of ρ is uniform, as usually required for random oracles.

[1] The reduction does see the queries made by the adversary and the oracle's replies.

The WPRO model appears to have little to do with randomly-programming reductions. Nevertheless, we prove that the two characterizations of limited programming are strongly related. Namely, Proposition 3 states that any BB reduction in the WPROM implies a randomly-programming reduction, while Proposition 4 states that any randomly-programming reduction implies a reduction in the WPROM. Besides being convenient in proving results (the WPROM typically being easier to work with), the equivalence provides some evidence that this formulation of limited programmability is well-founded. The results discussed next point to this as well.

Implications for Practical Schemes. We put these new tools to use by reconsidering security proofs of various important schemes. These schemes can be viewed as initial and interesting case studies; we expect that one can use our techniques readily to analyze the need for programmability in many further schemes.

A first example is FDH signatures. The only known security proofs [1, 7] use reductions that embed a challenge range point for the underlying trapdoor permutation in one (or more) of the hash query responses. It may be, though, that a clever reduction exists that does not rely on programming. We give formal evidence that this is unlikely: Theorem 4 states that no BB reduction exists that shows FDH is secure in the WPROM, even for a very weak definition of unforgeability. Even if the intuition is clear that programming plays a significant role in existing reductions, we emphasize that *proving* the inability to give a reduction here is technically involved. Previous negative results use inherently *asymptotic* methods to achieve black-box separations in the uniform setting. Instead, our proof of Theorem 4 makes use of a novel approach that is non-asymptotic in nature. This result is complementary to existing negative results about FDH, e.g. [8]. (See the full version for additional discussion.)

A more involved example is Shoup's TDP-KEM [15]. Shoup's (IND-CCA) security proof does not involve embedding a challenge in the output of the RO, but rather programming is used to ensure consistency between simulation of a decapsulation oracle and simulation of the RO. We show the following surprising result: Shoup's TDP-KEM is CCA-secure in the WPROM (Theorem 2), but no non-programming BB reduction exists for showing CCA-security (Theorem 3). The negative result is even more complex than in the FDH case, involving several interesting technical hurdles (e.g. dealing with the fact that reductions can rewind adversaries, explicitly allowed in our non-programming reduction framework).

We also observe that OAEP [2] is an example of a scheme whose proof requires no programming whatsoever. This is actually evident by inspection of the proof given in [9]. We give the details in the full version, where all proofs omitted due to space constraints can also be found.

Discussion. Note that proving security with limited or no programming (still) only provides heuristic evidence for security. That said, it could be the case that proofs in the WPROM or that use a non-programming reduction provide a better heuristic than the ROM. While this would hold for any weakening of the ROM,

we feel that programmability is a particularly interesting case due to its apparent artificiality. Note, for example, that one can actually *run* a non-programming RO reduction when a concrete hash function (e.g. SHA-256) is used to realize the RO. This is not true for fully or randomly-programming reductions.

Further Related Work. Hofheinz and Kiltz [10] offer some insights on programmability from a completely different angle. Generalizing a technique due to Waters [16], they built *standard-model* hash functions that provide a limited form of programmability. Unfortunately, their hash functions are not sufficiently programmable to admit the techniques used in security arguments for ROM schemes like FDH and Fiat-Shamir. Nonetheless, their work indicates that a better understanding of programmability could lead to more broadly applicable standard-model solutions.

2 Reduction-Centric Models

In this section, we first formalize at an abstract level the general concept of a black-box reduction in the random oracle model. Furthermore, we present two variations of the black-box reduction notion where the reduction's capabilities in programming the random oracle are restricted.

2.1 Preliminaries

We begin by establishing notation and execution semantics for algorithms and oracles.

Oracle Access to Adversaries. We model all algorithms (e.g. adversaries and reductions) and ideal primitives (e.g. random oracles) as interactive Turing machines (ITM). In particular these machines can be probabilistic and can keep state. Each machine may have several communication tapes, which we usually call *interfaces*, that connect different machines to each other. We write $\mathcal{A}^{(\cdot)}$ to denote an ITM $\mathcal{A}$ with an interface that expects oracle (i.e. black-box) access to some other ITM. A reduction B with oracle access to adversary $\mathcal{A}^{(\cdot)}$ (denoted as $\mathsf{B}^{\mathcal{A}^{(\cdot)}}$) is allowed to do the following:

- At any time, B can start a copy of the algorithm $\mathcal{A}^{(\cdot)}$ on (chosen) randomness and input, where the random coins are those used to compute the first output (or oracle query).
- Once such a copy is started, B obtains each value output by $\mathcal{A}$ and must provide both the corresponding answer *and* the random coins needed for the execution of $\mathcal{A}$ to continue to its next output. (This includes queries to the given oracles.)
- At any point in time, B may halt the execution of the current copy of $\mathcal{A}$.

Note that the model is general enough so that B can, for example, "rewind" the adversary $\mathcal{A}$ to an arbitrary output by running a new copy of $\mathcal{A}$ with previously given coins.

We stress that if we write that B is given oracle access to $\mathcal{A}^{\mathcal{O}}$ for a *particular* oracle $\mathcal{O}$ (as opposed to $\mathcal{A}^{(\cdot)}$), then B does *not* get to answer $\mathcal{A}$'s queries to $\mathcal{O}$. Queries are sent directly to, and answered directly by $\mathcal{O}$ itself. We write (for example) $\mathcal{A}^{(\cdot,\mathcal{O})}$ when we wish to be explicit that queries to the first oracle are controlled by B, and the second are not. Sometimes we will simply omit some of the oracles which are controlled by B: the understanding is that any oracle which is not explicitly given to $\mathcal{A}$ in our notation can be controlled by the reduction.

Finally, we write $\mathcal{A}^{\mathcal{O}_{\text{pub}}}$ to mean the following: when $\mathcal{A}$ queries x to $\mathcal{O}$, x is forwarded to B, which can then perform some computations, call other oracles, and only after this triggers delivery of $\mathcal{O}(x)$ to $\mathcal{A}$. The answer is however given by $\mathcal{O}$ directly (but visible to B) and there is no way for B to influence it directly.[2] This construct will be useful in a number of contexts.

Security Properties. It is convenient to consider generic security properties Π for cryptographic primitives defined in terms of games involving a candidate f (called a Π-candidate) and an adversary $\mathcal{A}$ (called a Π-adversary). In particular, with each triple f, $\mathcal{A}$ and Π we associate an advantage $\mathbf{Adv}_f^{\Pi}(\mathcal{A})$, and f is said to be Π-secure if $\mathbf{Adv}_f^{\Pi}(\mathcal{A})$ is small for all efficient adversaries $\mathcal{A}$. It is convenient to assume that the advantage satisfies the following *linearity* condition: if an oracle $\mathcal{O}$ behaves as $\mathcal{O}_1$ with probability p and as $\mathcal{O}_2$ with probability $1-p$, then $\mathbf{Adv}_{f^{\mathcal{O}}}^{\Pi}(\mathcal{A}^{\mathcal{O}}) = p \cdot \mathbf{Adv}_{f^{\mathcal{O}_1}}^{\Pi}(\mathcal{A}^{\mathcal{O}_1}) + (1-p) \cdot \mathbf{Adv}_{f^{\mathcal{O}_2}}^{\Pi}(\mathcal{A}^{\mathcal{O}_2})$ for every (oracle) primitive f and all adversaries $\mathcal{A}$. Despite there being a few advantage notions that do not satisfy this property (e.g. distinguishing advantage with absolute values), an equivalent notion satisfying this property can typically be given (e.g. dispense with the absolute values).

2.2 Black-Box Reductions in the ROM

When we talk about black-box reductions, we mean *fully* black-box security reductions as defined by Reingold et al. [14]. Those reductions are paramount in cryptography, especially for random-oracle based schemes with practical efficiency as a design goal.

We present in Definition 1 below our formalization of fully-black-box reductions in the ROM, as well as our two variants with limited and no programmability of the random oracle, which we first introduce in detail.

Fully-Programming Reductions (FPRed). The first notion formalizes the standard concept of black-box reductions in the ROM. As they support the common strategy of programming the ROM without any restriction, we refer to such reductions as *fully-programming reductions*.

Non-programming Reductions (NPRed). The first (stronger) new notion that we introduce captures the fact that the reduction has no control at all on the answers of random oracle queries. Namely, the queries are answered by a random oracle which is chosen once, independently from the reduction

[2] But the answer may be influenced through queries to related oracles.

B and its input(s), and remains the same for every execution of an adversary $\mathcal{A}$ initiated by B. While the reduction B can learn all of the RO-queries issued by $\mathcal{A}$, there is no way for B to influence their distribution. Intuitively, this models the fact that the reduction can be run with an *external* random oracle.

Randomly-Programming Reductions (RPRed). Our second variant only allows the reduction B to program the RO with random instead of arbitrary values, and is hence somewhat between fully- and non-programming reductions, To this end, we first introduce a *randomly-programmable random oracle (RPRO)* which is an idealized object that exposes three interfaces: $R_{eval}, R_{rand}, R_{prog}$ (a conventional RO can be seen as having a single interface to callers). If called via the *evaluation* interface R_{eval}, it behaves as a conventional random oracle mapping $Dom \to Rng$. A second *random* interface R_{rand} implements a random mapping $\{0,1\}^* \to Rng$. Finally, the *programming* interface R_{prog} takes $X \in Dom$ and $Y \in \{0,1\}^*$ as input, and sets $R_{eval}(X)$ to be the same as $R_{rand}(Y)$.

As $\mathcal{A}$'s queries to the evaluation interface of R_{eval} are public, the reduction B is allowed, on query X by $\mathcal{A}$, to perform a number of R_{rand} calls followed by a suitable $R_{prog}(X,Y)$ invocation in order to let the output of $\mathcal{A}$'s query satisfy a certain property before the query is actually answered to $\mathcal{A}$. This allows a minimal amount of programmability, for instance a constant number of output bits can be forced to take some input-dependent value.
We note that these interfaces allow to "reprogram" the random oracle. This supports, among other things, the ability to rewind the adversary and run it on "another", partly consistent random oracle, but where the reduction does not need to choose the actual values. Note that non-programming reductions prevent such forking techniques.

In the following, let $\mathsf{S} = \mathsf{S}^R[f]$ be a cryptographic scheme relying on a primitive f and a random oracle $R : Dom \to Rng$. Let Π and Π' be security properties which can possibly be satisfied by S and f, respectively.

Definition 1 (FPRed, NPRed, RPRed). *Let* $\mathsf{X} \in \{$*fully-programming, non-programming, randomly-programming*$\}$. *A* $(\Pi \to \Pi', \delta, t, q_{\mathcal{O}}, q_A)$*-fully-BB* X *ROM security reduction for* S *is an oracle machine* $\mathsf{B}^{(\cdot,\cdot)}$ *with the property that for* all[3] Π*-adversaries* $\mathcal{A}^{(\cdot)}$ *and all* Π'*-candidates* f*, if*

$$\mathbf{Adv}^{\Pi}_{\mathsf{S}^R[f]}(\mathcal{A}^R) > \epsilon$$

for a random oracle $R : Dom \to Rng$ *and* $\epsilon > 0$*, then*

$$\mathbf{Adv}^{\Pi'}_{f}(\mathsf{B}^{\mathcal{O}_1,\mathcal{A}^{\mathcal{O}_2}}) > \delta(\epsilon, q, \ell),$$

where q *is the total number of queries* $\mathcal{A}$ *makes and* ℓ *is the overall length of these queries. Furthermore,* B *runs in time* t*, makes* $q_{\mathcal{O}}$ *queries to the given oracle(s)*

[3] In particular, including those which are not efficiently implementable.

$\mathcal{O}_1$ and runs q_A instantiations of $\mathcal{A}^{\mathcal{O}_2}$, where all three quantities are functions of ϵ, q, and ℓ. Moreover, when:

$$\mathsf{X} = \text{fully-programming, then } \mathcal{O}_1 = f,\ \mathcal{O}_2 = (\cdot),\ \text{and } q_{\mathcal{O}} = q_F$$
$$\mathsf{X} = \text{non-programming, then } \mathcal{O}_1 = (f, R),\ \mathcal{O}_2 = R_{\text{pub}},\ \text{and } q_{\mathcal{O}} = (q_F, q_R)$$
$$\mathsf{X} = \text{randomly-programming, then } \mathcal{O}_1 = (f, R'_{eval}, R'_{prog}, R'_{rand}),$$
$$\mathcal{O}_2 = R'_{eval,\text{pub}},\ \text{and } q_{\mathcal{O}} = (q_F, q_{ev}, q_{pr}, q_{ra}),$$

where $R' = (R'_{eval}, R'_{prog}, R'_{rand})$ is a RPRO.

2.3 Black-Box Separations

This paper uses a novel approach in order to obtain black-box separations in the concrete setting. The approach applies to all notions of reductions defined in this paper, but we illustrate it in the context of FPRed reductions. In order to disprove the existence of a fully-BB reduction within a certain class of reductions, for every reduction B of interest, we have to prove the existence of a Π'-candidate f and an adversary $\mathcal{A}$ such that $\mathbf{Adv}^{\Pi}_{\mathsf{S}^R[f]}(\mathcal{A}^R)$ is large, but the advantage $\mathbf{Adv}^{\Pi'}_{f}(\mathsf{B}^{f,\mathcal{A}^{(\cdot)}})$ is small. We will achieve this by first showing the existence of a *randomized* Π'-candidate F and an adversary $\mathcal{A}_P$ with *private* random coins P (i.e. not controllable by B) such that $\mathbf{Adv}^{\Pi}_{\mathsf{S}^R[F]}(\mathcal{A}_p^{f,R})$ is large for *all* values p of the random coins P and for all (fixed) primitives f obtained by fixing the coins of F, but $\mathbf{Adv}^{\Pi'}_{F}(\mathsf{B}^{F,\mathcal{A}_P^{(F,\cdot)}})$ is small for all reductions B of interest. Because of the linearity of the advantage measures, we have

$$\mathbf{Adv}^{\Pi'}_{F}(\mathsf{B}^{F,\mathcal{A}_P^{(F,\cdot)}}) = \mathbf{E}_{p,f}\left[\mathbf{Adv}^{\Pi'}_{f}(\mathsf{B}^{f,\mathcal{A}_p^{(f,\cdot)}})\right],$$

where the expected value is taken over the choice of the private coins p and the primitive f realized by F (with the corresponding probability distributions, which may in the general case even be correlated). Therefore, for all reductions B of interest, there must exist some particular f and some adversary $\mathcal{A}'^{(\cdot)} := \mathcal{A}_p^{(f,\cdot)}$ *without* private coins such that $\mathbf{Adv}^{\Pi'}_{f}(\mathsf{B}^{f,\mathcal{A}'^{(\cdot)}}) \leq \mathbf{Adv}^{\Pi'}_{F}(\mathsf{B}^{F,\mathcal{A}_P^{(F,\cdot)}})$ is small, too. Hence, such a statement (for randomized primitives) also implies the inexistence of a reduction working universally for all primitives: in particular, there is no need to apply well-known classical asymptotic (and uniform) de-randomization techniques based on the Borel-Cantelli lemma. To the best of our knowledge, this approach is novel to this paper.

3 The Weakly Programmable ROM

In the previous section programmability (or the lack thereof) is captured by considering a restricted set of reductions; from the point of view of the adversary being employed by the reduction, nothing has changed. In this section we take an

subroutine $R^{\rho}_{hon}(X)$:	**subroutine** $R^{\rho}_{adv}(X)$:
if $\mathtt{T}[X] \neq \perp$ then ret $\mathtt{T}[X]$	if $\mathtt{T}[X] \neq \perp$ then ret $\mathtt{T}[X], \mathtt{R}[X]$
$r \leftarrow\!\!\$\ \mathit{Coins}$; $z \leftarrow \rho(r)$	$r \leftarrow\!\!\$\ \mathit{Coins}$; $z \leftarrow \rho(r)$
$\mathtt{T}[X] \leftarrow z$; $\mathtt{R}[X] \leftarrow r$	$\mathtt{T}[X] \leftarrow z$; $\mathtt{R}[X] \leftarrow r$
ret $\mathtt{T}[X]$	ret $\mathtt{T}[X], \mathtt{R}[X]$

Fig. 2. The weakly-programmable random oracle ideal primitive R^{ρ} for $\rho\colon \mathit{Coins} \to$ Rng. Initially $\mathtt{T}[X] = \perp$ for all X.

alternative approach, modifying the random oracle itself rather than restricting the reduction.

Consider a random oracle as a mapping from *Dom* to *Rng*, where *Rng* is finite and non-empty. Since we model ideal primitives by stateful and probabilistic interactive Turing machines we can imagine the random oracle as being implemented via so-called lazy sampling: whenever a new query $X \in \mathit{Dom}$ appears, the random oracle returns a random value $z \leftarrow\!\!\$\ \mathit{Rng}$ and stores the pair (X, z) for further use. We now restrict the way the random oracle's answers z are determined. Namely, we parameterize the random oracle by a function $\rho\colon \mathit{Coins} \to \mathit{Rng}$ for a finite, non-empty set *Coins*. Each time the random oracle receives a new query X it picks $r \leftarrow\!\!\$\ \mathit{Coins}$ at random and returns $z = \rho(r)$ and stores X together with r.

Now, recall that an ideal primitive can have multiple interfaces. In what follows, we consider two: an *honest* interface for use by honest parties and protocols; and an *adversarial* interface. Loosely, the latter will give the adversary an ability to "validate" that the random oracle is behaving properly. Formally, we give the following definition of a (ρ-restricted) weakly programmable random oracle.

Definition 2 (WPRO). *For a function $\rho : \mathit{Coins} \to \mathit{Rng}$ the ideal primitive $R^{\rho} = (R^{\rho}_{hon}, R^{\rho}_{adv})$ described in Figure 2 is called a ρ-WPRO (or simply WPRO if ρ is implicitly clear).*

Notice that the honest interface of this object returns the range point z associated with the queried input. The adversarial interface returns both that range point *and* the random value r used to generate z.

At this point we have not imposed any restriction on ρ. For example, if ρ is the identity function (and $\mathit{Rng} = \mathit{Coins}$) then the resulting ideal primitive is equivalent to a normal random oracle. On the other end of the spectrum, if ρ is a constant function then it is clear that R^{ρ} would not model an ideal cryptographic hash function. Thus we establish what it means for a function ρ to be *good*.

Definition 3 (Good ρ). *A function $\rho : \mathit{Coins} \to \mathit{Rng}$ is called* good *for Rng if and only if: (1) Coins is finite, (2) $|\mathit{Rng}|$ divides $|\mathit{Coins}|$ and (3) ρ is regular, i.e. for all $y \in \mathit{Rng}$ we have*

$$|\{r \in \mathit{Coins} : \rho(r) = y\}| = \frac{|\mathit{Coins}|}{|\mathit{Rng}|}.$$

Clearly any good ρ is such that, when evaluated on a uniformly chosen domain point, one gets a uniform range point. (And conversely, if a uniform distribution on the domain of ρ induces a uniform distribution on the range, ρ is good.) Said another way, a random oracle $R : \mathit{Dom} \to \mathit{Rng}$ and WPRO R^{ρ}_{hon} (with matching domain and range) are information-theoretically indistinguishable if and only if ρ is good for Rng.

It is easy to see that various kinds of functions ρ will limit a reduction's ability to program. For the scenarios we consider, the crucial property of ρ that make the reductions—the proof of security—fail, is one-wayness of ρ (but stated in the non-asymptotic setting via an upper bound on an algorithm's inversion probability). For any function ρ and owf-adversary $\mathcal{A}$ we define the owf advantage as

$$\mathbf{Adv}^{\mathrm{owf}}_{\rho}(\mathcal{A}) = \Pr\left[\, \rho(r) = \rho(r') \;:\; r \leftarrow\!\!\$\; \mathit{Coins}\,;\; r' \leftarrow\!\!\$\; \mathcal{A}(\rho(r)) \,\right]$$

where a owf-adversary $\mathcal{A}$ is a probabilistic algorithm that takes as input a point $y \in \mathit{Rng}$ and outputs a domain point $x \in \mathit{Coins}$.

One-wayness of ρ ensures non-programmability in the following sense: Consider for example a security reduction like the traditional one for FDH. This reduction receives a random image y under a trapdoor permutation and, at some point, injects this value as the hash value in a black-box simulation for an allegedly successful adversary. But since the adversary can access the R^{ρ}_{adv} interface, the reduction would also need to provide a preimage of y under ρ, violating the one-wayness of ρ.

Reductions in the WPRO Model. One can straightforwardly define a WPRO model (WPROM) by analogy to the ROM (all honest parties have access to R_{hon}, adversarial parties have access to R_{adv}), and the notion of a black-box reduction naturally extends to this model. In particular, we consider a strong notion of reduction that allows any *good* function ρ, regardless of whether ρ is efficiently computable or not.

Definition 4 (WPROM Reduction). *A $(\Pi \to \Pi', \delta, t, q_{\rho}, q_F, q_A)$-fully-BB WPROM security reduction for S is an oracle machine $\mathsf{B}^{(\cdot,\cdot,\cdot)}$ with the property that for* all *Π-adversaries $\mathcal{A}^{(\cdot)}$, all good functions ρ for Rng, and all Π'-candidates f, if*

$$\mathbf{Adv}^{\Pi}_{\mathsf{S}^{R^{\rho}_{hon}}[f]}(\mathcal{A}^{R^{\rho}_{adv}}) > \epsilon$$

for a ρ-WPRO $R^{\rho} = (R^{\rho}_{hon}, R^{\rho}_{\mathrm{adv}})$ with range Rng and $\epsilon > 0$, then

$$\mathbf{Adv}^{\Pi'}_{f}(\mathsf{B}^{\rho,f,\mathcal{A}^{(\cdot)}}) > \delta(\epsilon, q, \ell),$$

where q is the total number of queries $\mathcal{A}$ makes and ℓ is the overall length of these queries. Furthermore, B runs in time t, makes q_{ρ} and q_F queries to the given ρ and f, respectively, and runs q_A instantiations of $\mathcal{A}^{(\cdot)}$, where all three quantities are functions of ϵ, q, and ℓ.

Since the reduction notion quantifies over all good ρ, a reduction must work for one-way ρ. Indeed, it must also work for a ρ chosen randomly from the set of all functions $\mathit{Coins} \to \mathit{Rng}$. In this way reductions must avoid making use of

FDH-style programming: a reduction cannot inject a specific range point into one of WPRO's responses. As we will see in the next section, however, one can take advantage of more limited programming techniques in the WPRO model.

Although all the WPROM reductions given in this paper are fully black-box as per the definition above, we emphasize that the WPRO model is distinct from the formulations in Section 2 in that one can give non-black-box reductions in it.

4 Relationships among Types of Reductions

Having specified our reduction settings, we now establish the relationships among them. We begin by stating the intuitive implications: a non-programming BB reduction implies a randomly programming one, which in turn implies a fully programmable reduction. The straightforward proofs are omitted. Let $\mathsf{S} = \mathsf{S}^{f,R}$ be a scheme relying on a cryptographic primitive f and a random oracle $R : Dom \rightarrow Rng$. Let Π and Π' be security properties which can possibly be satisfied by S and f, respectively.

Proposition 1 (NPRed $\Rightarrow$ RPRed). *If there exists a* non-programming $(\Pi \rightarrow \Pi', \delta, t, q_F, q_R, q_A)$*-fully-BB ROM security reduction for* S*, then there exists a* randomly-programming $(\Pi \rightarrow \Pi', \delta, t, q_F, q_R, 0, 0, q_A)$*-fully-BB ROM security reduction for* S*.*

Proposition 2 (RPRed $\Rightarrow$ FPRed). *If there exists* randomly-programming $(\Pi \rightarrow \Pi', \delta, t, q_F, q_{ev}, q_{pr}, q_{ra}, q_A)$*-fully-BB ROM security reduction for* S*, then there exists a* fully-programming $(\Pi \rightarrow \Pi', \delta, t', q_F, q_A)$*-fully-BB ROM security reduction for* S*, where*[4] $t' = t + \mathcal{O}(\overline{q} \log \overline{q})$ *for* $\overline{q} = q \cdot q_A + q_{ev} + q_{pr} + q_{ra}$*.*

Next we show that schemes are secure in the WPRO model via a black-box reduction if and only if there is a randomly-programming reduction. Hence, restricting the *random oracle* in the WPRO sense, and restricting the *reduction*'s abilities to program a full-fledged random oracle, are equivalent in a black-box sense. The first result, in particular, exploits the fact that a fully-BB reduction in the WPROM must also work for a randomly chosen (regular) function ρ.

Proposition 3 (WPROM Red $\Rightarrow$ RPRed). *If a* $(\Pi \rightarrow \Pi', \delta, t, q_\rho, q_F, q_A)$*-fully-BB WPROM security reduction for* S *exists, then a randomly-programming* $(\Pi \rightarrow \Pi', \delta', t', q_F, q \cdot q_A, q \cdot q_A, q_\rho + q \cdot q_A, q_A)$*-fully-BB ROM security reduction for* S *exists, where* $\delta' = \delta - \frac{(q_A \cdot q + q_\rho)^2}{2|Dom|}$ *and* $t' = t + \mathcal{O}(q \cdot \ell)$*.*

Proposition 4 (RPRed $\Rightarrow$ WPROM Red). *If there exists a randomly-programming* $(\Pi \rightarrow \Pi', \delta, t, q_F, q_{ev}, q_{pr}, q_{ra}, q_A)$*-fully-BB ROM security reduction for* S*, then a* $(\Pi \rightarrow \Pi', \delta, t', q_F, q'_\rho, q_A)$*-fully-BB WPROM security reduction for* S *exists with* $t' = t + \mathcal{O}((q \cdot q_A \cdot \log(q \cdot q_A) \cdot \ell)$ *and* $q'_\rho = q \cdot q_A + q_{ev} + q_{pr} + q_A$*.*

[4] The extra overhead $\mathcal{O}(\overline{q} \log \overline{q})$ is due to the simulation of the RPRO in the reduction.

WPROs are not ROs, but WPROM and ROM are equivalent. Below we will confirm the expected implication that being a WPRO is actually a weaker requirement than being a full-fledged RO. Yet *existentially* WPROs and ROs are equivalent, i.e. we can efficiently construct a RO out of a WPRO.

For these comparisons we adopt the indifferentiability framework of Maurer, Renner and Holenstein [12] to reason about primitives being close to random oracles. We denote by $\mathbf{Adv}^{\text{ind-R}}_{C,H,\mathcal{S}}(\mathcal{D})$ the advantage any distinguisher $\mathcal{D}$ has in distinguishing between a construction C with component H, and an ideal primitive "R" (with an intermediary simulator $\mathcal{S}$). We denote by the superscripts "RO" and "WPRO" in the advantage the fact that the ideal primitive "R" is a random oracle or a WPRO.

First, we derive a WPRO that is not a RO, i.e. it is easily differentiable from a random oracle. This serves to show that in general WPROs and ROs are separated. Consider the composition $C^H_\rho(x) = \rho(H(x))$ of a random oracle $H : \mathit{Dom} \to \mathit{Coins}$ and a regular one-way function $\rho : \mathit{Coins} \to \mathit{Rng}$. In this case, any simulator $\mathcal{S}$ would have to invert ρ for a sound simulation.

Proposition 5 (WPRO $\not\Rightarrow$ RO). *For any function ρ that is good for Rng, for the construction $C^H_\rho(x) = \rho(H(x))$ there exists a simulator $\mathcal{S}^{WPRO}_\rho$ such that*

$$\mathbf{Adv}^{\text{ind-WPRO},\rho}_{C_\rho,H,\mathcal{S}^{WPRO}}(\mathcal{D}) = 0$$

for any distinguisher $\mathcal{D}$, but where there exists a distinguisher $\mathcal{D}^{RO}$ such that for any simulator $\mathcal{S}$,

$$\mathbf{Adv}^{\text{ind-RO}}_{C_\rho,H,\mathcal{S}}(\mathcal{D}^{RO}) \geq 1 - \mathbf{Adv}^{\text{owf}}_\rho(\mathcal{S}).$$

Despite this generic separation, it *is* possible to build a (fully programmable) random oracle out of a WPRO, essentially building RO outputs one bit at a time. Specifically, for $x \in \{0,1\}^*$ let $x|_i$ denote the ith bit of x. Given a function $H : \{0,1\}^* \to \{0,1\}^\ell$ for some $\ell > 1$ we consider the construction $\mathsf{C}^H : \{0,1\}^* \to \{0,1\}^m$ such that

$$\mathsf{C}^H(x) = H(x\|\langle 1\rangle)|_1 \parallel H(x\|\langle 2\rangle)|_1 \parallel \cdots \parallel H(x\|\langle m\rangle)|_1,$$

where $\|$ denotes concatenation of strings and $\langle i\rangle$ is the (suffix-free) binary encoding of an integer i. Note that the construction calls H altogether m times to achieve output length m; one can improve the efficiency by outputting more bits in each iteration at the cost of tightness in the reduction. Furthermore, due to the suffix-freeness of $\langle\cdot\rangle$ one can always decide if a given string is of the form $x\|\langle i\rangle$ for some $i \in \mathbb{N}$.

Theorem 1 (WPROM $\Leftrightarrow$ ROM). *For all good functions ρ, all integers $\tau > 0$, and a WPRO $R^\rho = (R^\rho_{\text{adv}}, R^\rho_{\text{hon}}) : \{0,1\}^* \to \{0,1\}^\ell$ there exists a simulator $\mathcal{S}_{\rho,\tau}$ such that for all distinguishers $\mathcal{D}$ issuing at most q queries to each oracle we have*

$$\mathbf{Adv}^{\text{ind-RO}}_{\mathsf{C},R^\rho,\mathcal{S}_{\rho,\tau}}(\mathcal{D}) \leq q \cdot 2^{-\tau}$$

where the simulator $\mathcal{S}_{\rho,\tau}$ invokes R at most once on each query and has running time $\mathcal{O}(\tau \cdot (\mathsf{Time}_\rho + \mathsf{Time}_{Coins}))$, where Time_ρ and Time_{Coins} are the times needed to compute ρ and to sample a random element from Coins, respectively.

5 Trapdoor-Permutation-Based Key-Encapsulation

5.1 TDP-KEM Security in the WPROM

A key-encapsulation mechanism (KEM) is a triple of algorithms denoted $\mathsf{KEM} = (\mathsf{Key}, \mathsf{Encap}, \mathsf{Decap})$ that operate as follows. The probabilistic *key-generation algorithm* returns a key-pair (pk, sk); we write $(pk, sk) \leftarrow\!\!\$\ \mathsf{Key}$. The *(key) encapsulation algorithm* Encap is a probabilistic algorithm that takes pk as input and returns a key-ciphertext pair (K, C) where $K \in \mathcal{K}$ for some non-empty set $\mathcal{K}$. The *(key) decapsulation algorithm* takes as input a pair (sk, C) and deterministically outputs a key $K \in \mathcal{K}$ or the distinguished symbol $\perp$ to denote invalidity of (sk, C). For proper operation, we require that for all pairs (pk, sk) generated by Key, if $(K, C) \leftarrow\!\!\$\ \mathsf{Encap}(pk)$ then $K \leftarrow \mathsf{Decap}(sk, C)$.

Let $\mathsf{KEM} = (\mathsf{Key}, \mathsf{Encap}, \mathsf{Decap})$ be a KEM, $\mathcal{K}$ be a non-empty set, and $\mathcal{A}$ be a KEM adversary. The security of KEM, in the WPRO model of hash function R with underlying function ρ, against an adversary $\mathcal{A}$ is defined by the following experiment:

$$\begin{boxed}\begin{array}{l}\underline{\mathbf{Exp}^{\text{kem-cca}}_{\mathsf{KEM},R,\rho}(\mathcal{A})}\\ (pk, sk) \leftarrow\!\!\$\ \mathsf{Key};\ b \leftarrow\!\!\$\ \{0,1\};\\ b' \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{Decap}(sk,\cdot),R^\rho_{adv}(\cdot),\mathsf{Encap}(pk,b,\cdot)}(pk)\\ \text{if } b' = b \text{ then return 1 else 0}\end{array}\end{boxed}$$

The $\mathsf{Decap}(sk, \cdot)$ oracle performs the decapsulation algorithm upon its input and returns the result. The $\mathsf{Encap}(pk, b, \cdot)$ oracle takes as input a distinguished symbol Run, picks $K_0 \leftarrow\!\!\$\ \mathcal{K}$, runs the encapsulation algorithm to produce $(K_1, C) \leftarrow\!\!\$\ \mathsf{Encap}(pk)$, and returns the challenge (K_b, C). The encapsulation oracle can be queried only once by the adversary. We then define the KEM-CCA advantage of adversary $\mathcal{A}$ in breaking the KEM scheme via a chosen-ciphertext attack as $\mathbf{Adv}^{\text{kem-cca}}_{\mathsf{KEM},R,\rho}(\mathcal{A}) = \Pr[\mathbf{Exp}^{\text{kem-cca}}_{\mathsf{KEM},R,\rho}(\mathcal{A}) = 1] - 1/2$, where $\mathcal{A}$ is forbidden to ask the challenge ciphertext C to its decapsulation oracle.

TDP-based KEMs in the WPRO Model. We recall that a trapdoor permutation with domain *Dom* is a triple $\mathcal{TP} = (G, F, \overline{F})$ of efficient algorithms such that G returns a pair (pk, td), consisting of the public key and the trapdoor, with the property that $F(pk, \cdot)$ implements a permutation $f_{pk} : Dom \to Dom$, whereas $\overline{F}(td, \cdot)$ implements its inverse $f^{-1}_{pk}(\cdot)$. Consider key encapsulation mechanism $\mathsf{TDP\text{-}KEM}^R[\mathcal{TP}] = (\mathsf{Key}, \mathsf{Encap}, \mathsf{Decap})$ based on a TDP $\mathcal{TP} = (G, F, \overline{F})$ with domain *Dom* and a WPRO $R^\rho : Dom \to \mathcal{K}$ for some underlying good function ρ mapping *Coins* to $\mathcal{K}$, where $\mathcal{K}$ is some non-empty set. The key generation algorithm is defined by $\mathsf{Key} = G$, so it returns a pair (pk, td). The encapsulation algorithm on input pk samples $x \leftarrow\!\!\$\ Dom$, sets $K \leftarrow R^\rho_{hon}(x)$ and $C \leftarrow F(pk, x)$,

and returns (K, C). The decapsulation algorithm on input (td, C) computes $x \leftarrow \overline{F}(td, C)$, sets $K \leftarrow\!\!\$\ R^{\rho}_{hon}(x)$ and returns K.

The KEM-CCA security of this scheme is tightly bound to the OWF security of the underlying TDP. Our proof largely mirrors the one given by Shoup [15] for RSA-KEM in the ROM.

Theorem 2 (WPROM Reduction for TDP-KEM). *Let $\mathcal{TP} = (G, F, \overline{F})$ be a trapdoor permutation with domain Dom. Let $\mathsf{TDP\text{-}KEM}^R[\mathcal{TP}] = (\mathsf{Key}, \mathsf{Encap}, \mathsf{Decap})$ be the TDP-based KEM described above. Let ρ be good for the non-empty set $\mathcal{K}$ and let $\mathcal{A}$ be an adversary suitable for $\mathbf{Exp}^{\text{kem-cca}}_{\mathsf{TDP\text{-}KEM},R,\rho}(\mathcal{A})$ asking q_D queries to its decapsulation oracle, q_R queries to the WPRO and running in time t. Then there exists an adversary $\mathcal{B} = \mathsf{B}^{\rho,F,\mathcal{A}(\cdot)}$ such that*

$$\mathbf{Adv}^{\text{kem-cca}}_{\mathsf{TDP\text{-}KEM},R,\rho}(\mathcal{A}) \leq 2 \cdot \mathbf{Adv}^{\text{owf}}_{\mathcal{TP}}(\mathcal{B}) + \frac{q_D}{|Dom|}$$

where $\mathcal{B}$ uses a single instantiation of $\mathcal{A}^{(\cdot)}$, runs in time at most $t + \overline{q} \log \overline{q} \cdot (\mathsf{Time}_F + \mathsf{Time}_\rho + \mathsf{Time}_{Coins} + \mathsf{Time}_{\mathcal{K}})$ for $\overline{q} = q_D + q_R + 1$ and Time_X denotes the time to execute algorithm X or to sample from set X.

Here we give a very brief overview of the proof. The reduction is required to invert the TDP on some challenge range point C^*, and it will embed this challenge along with a random key K as the response (K, C^*) to the KEM-adversary's encapsulation query. Despite not having access to the trapdoor information, the reduction *can* answer decapsulation queries by (randomly) programming the WPRO to $\rho(r)$, where r is uniform. This simulation is correct, and can easily be made consistent with WPRO queries, except in the case that Decap is queried on the TDP challenge point C^*. This case accounts for the $q_D/|Dom|$ term in the bound. Barring that case, the KEM-adversary wins its game only by querying the WPRO on the preimage of C^*, in which case the reduction succeeds to invert its challenge.

5.2 TDP-KEM Is Not Provable under Non-programming Reductions

In the proof of Theorem 2, a weak form of programmability is needed to allow for consistent simulation of the decapsulation oracle. Namely, the reduction may need to return a *random* key $K \in \mathcal{K}$ for a decapsulation query C, because it does not know the associated preimage r of C under $F(pk, \cdot)$. Consequently, if the adversary queries the random oracle with input r at a later point in time, its output is programmed to K. This fact makes $\mathsf{TDP\text{-}KEM}$ amenable to an attempt to entirely avoid programmability, e.g., by means of rewinding techniques. Yet, any such approach is doomed to fail: we prove that $\mathsf{TDP\text{-}KEM}$ *cannot* be proven secure with respect to non-programming fully-BB reductions, hence showing that $\mathsf{TDP\text{-}KEM}$ is a scheme which *necessarily* requires a mild type of programmability.

This is summarized by the following theorem. Note that the result requires $q_{\mathcal{A}} \leq 2^{q-1}$: For a small number of adversarial queries q a reduction may indeed be feasible (e.g. using rewinding). Yet, for acceptable values of q the value 2^{q-1} is too large for an efficient reduction to be allowed to issue more than 2^{q-1} queries.

Theorem 3 (Non-programming Irreducibility for TDP-KEM). *Let* $\mathsf{TDP\text{-}KEM}^R[\mathcal{TP}] = (\mathsf{Key}, \mathsf{Encap}, \mathsf{Decap})$ *be the* TDP-KEM *scheme with key space* $\mathcal{K}$*, relying on a trapdoor permutation* $\mathcal{TP} = (G, F, \overline{F})$ *with domain Dom and public-key/trapdoor space* $\{0,1\}^k$*, as well as on a random oracle* $R : Dom \to \mathcal{K}$*. Then, for all* $t, q > 0$*, all* $\epsilon \leq \frac{1}{2} - \frac{1}{|\mathcal{K}|}$*, and all* $(\text{kem-cca} \to \text{owp}, \delta, t, (q_G, q_F, q_{\overline{F}}), q_R, q_{\mathcal{A}})$*-fully-BB non-programming reductions* B *for* TDP-KEM*, we have*

$$\delta(\epsilon, q, q \cdot \log|Dom|) \leq \frac{(q_{\mathcal{A}} q + 1) \cdot (2 q_{\mathcal{A}} q + q_F + q_R + 1)}{|Dom|} + \frac{q_{\mathcal{A}} q}{|\mathcal{K}|} + \frac{q_G + q_{\overline{F}}}{2^k}$$

where $q_G, q_F, q_{\overline{F}}$*, and* q_R *are the number of queries of* B *to the respective oracles, and* $q_{\mathcal{A}} \leq 2^{q-1}$ *is the number of adversarial instances run by* B*.*

We provide a high-level description of the proof. We rely on an ideal trapdoor permutation $\mathcal{TP} = \mathcal{TP}^{\mathcal{F}} = (G, F, \overline{F})$ defined using the oracles $\mathcal{F} = (\mathcal{F}_\tau, \mathcal{F}_E, \mathcal{F}_{E^{-1}})$: The oracle $\mathcal{F}$ initially chooses a keyed family of *random* permutations $E : \{0,1\}^k \times Dom \to Dom$ (in other words, $E(pk, \cdot)$ is a random permutation for all k-bit pk), as well as a random permutation τ that associates to each k-bit trapdoor td a corresponding public key $pk = \tau(td)$. The oracles $\mathcal{F}_\tau$ and $\mathcal{F}_E$ allow direct evaluation of τ and E, whereas the oracle $\mathcal{F}_{E^{-1}}$, on input (td, y) computes $E^{-1}(\tau(td), y)$. The associated trapdoor permutation $\mathcal{TP}^{\mathcal{F}} = (G, F, \overline{F})$ is such that the generation algorithm $G^{\mathcal{F}}$ chooses a random uniform trapdoor $td \leftarrow\!\!\$\ \{0,1\}^k$, and sets the public key $pk \leftarrow \tau(td)$. Furthermore, the algorithms $F^{\mathcal{F}}$ and $\overline{F}^{\mathcal{F}}$ simply call $\mathcal{F}_E$ and $\mathcal{F}_{E^{-1}}$, respectively, with their inputs, and return the corresponding output. Note that, even given the public key pk, in order to be able to use $\mathcal{F}_{E^{-1}}$ for inversion of $F(pk, \cdot)$ we are required to guess $\tau^{-1}(pk)$ given only access to τ, which is of course infeasible (at least without an extra oracle).

We show that there exists a *deterministic* adversary $\mathcal{A}$ making q queries from Dom (and hence of length $\log|Dom|$ bits each) and accessing an oracle $\mathcal{O} : \{0,1\}^* \to Dom$ such that $\mathbf{Adv}^{\text{kem-cca}}_{\mathsf{TDP\text{-}KEM}^R[\mathcal{TP}],R}(\mathcal{A}^{\mathcal{O},\mathcal{TP},R}) \geq 1 - \frac{1}{|\mathcal{K}|}$ for all $\mathcal{TP}$ and $\mathcal{O}$, but whenever $\mathcal{O}$ is a random oracle and $\mathcal{TP} = \mathcal{TP}^{\mathcal{F}}$, then

$$\mathbf{Adv}^{\text{owf}}_{\mathcal{TP}}(\mathsf{B}^{\mathcal{TP},R,\mathcal{A}^{\mathcal{O},\mathcal{TP},R_{\text{pub}}}}) \leq \frac{(q_{\mathcal{A}} q + 1) \cdot (2 q_{\mathcal{A}} q + q_F + q_R + 1)}{|Dom|} + \frac{q_{\mathcal{A}} q}{|\mathcal{K}|} + \frac{q_G + q_{\overline{F}}}{2^k}$$

for *all* reductions B as in the statement of the theorem, where in particular B can run $q_{\mathcal{A}}$ instances of $\mathcal{A}$ answering both its encapsulation and the decapsulation queries. The statement of the theorem is obtained by derandomizing $\mathcal{F}$ and $\mathcal{O}$ as described in Section 2.3.

Adversary Description. Ideally, we would like the (inefficient) adversary $\mathcal{A}$ to be capable of determining whether it is being run in the actual kem-cca-game, or whether it is being used by the reduction B in order to break the one-wayness of the underlying trapdoor permutation $\mathcal{TP}$. A naive approach consists of letting $\mathcal{A}$, on input the public key pk, choose a random $r \leftarrow\!\!\$\ Dom$ and compute

$C \leftarrow F(pk, r)$; the adversary subsequently asks C to the decapsulation oracle, obtaining a value K. Finally, it issues a query r to the random oracle R, and checks whether $R(r) = K$. In the affirmative case, it assumes that it is being used in the actual kem-cca-game, and proceeds in breaking the scheme, e.g., by inverting $F(pk, \cdot)$ on the challenge ciphertext and guessing the bit b by making an additional random oracle query. Otherwise, $\mathcal{A}$ just outputs a random guess. Intuitively, since B is efficient, it cannot retrieve r given only C, and thus must give some independent answer K' back to $\mathcal{A}$'s decapsulation query, such that $\mathcal{A}$'s check will then fail.

This argument, however, has two major fallacies. First, the randomness of $\mathcal{A}$ is determined by B, and thus r is chosen (and known) by B. Second, even provided a way for $\mathcal{A}$ to issue a decapsulation query for a ciphertext C with preimage r *unknown* to B, the reduction B can still first run $\mathcal{A}$ by giving a random answer to the decapsulation query, up to the point where the random-oracle check fails, and hence finding out r (as it is queried to R_{pub}). It subsequently *rewinds* $\mathcal{A}$ so that the same query C is issued, for which now B knows the right answer $R(r)$. This allows B to invert the underlying $\mathcal{TP}$, by just giving the challenge output y as the challenge ciphertext to $\mathcal{A}$'s encapsulation query.

We overcome both these problems by using a *random* oracle $\mathcal{O} : \{0,1\}^* \to \mathit{Dom}$ and considering the following adversary $\mathcal{A}$: On input the public key pk, it asks a sequence of decapsulation queries $C_1, C_2, \ldots, C_\ell$ (for $\ell = q - 1$), where C_i is computed by applying the random oracle to pk, to $C_1, \ldots, C_{i-1}$, and to the answers of the previous queries. (We assume that such inputs can injectively be mapped into bit strings.) Then, it checks the correctness of the answers $K_\ell, K_{\ell-1}, \ldots$ in *reverse* order (as above, it checks whether $K_i = R(F^{-1}(pk, C_i))$), but stops checking as soon as the first inconsistency is found. (This is crucial for the analysis to go through.) Finally, it behaves as above depending on whether all checks have been successful or not.

The main idea is that rewinding does not help when $\mathcal{O}$ is a random oracle, since (provided some unlikely events do not occur) the best strategy for B to build a run of an instance of $\mathcal{A}$ where the correctness check is always satisfied requires exponentially many (in ℓ) executions of $\mathcal{A}$. This is proven by showing an interesting connection to a tree-based, purely combinatorial, game. This approach is similar to the schedule used by Canetti et al. [6] to prove a lower bound on the round complexity of black-box concurrent zero-knowledge proofs.

6 FDH Is Not Provably Secure in the WPRO Model

In this section we consider the traditional full-domain hash signature scheme and show that one cannot prove it secure under randomly-programming reductions only.[5] Hence, a stronger version of programmability is required. We carry out our proof in the WPRO model and the result follows for randomly-programming reductions by the equivalence.

[5] In fact, we prove the slightly stronger statement that not even a ρ-dependent black-box reduction in the WPRO model exists for any *one-way* good function ρ.

FULL-DOMAIN HASH. We briefly recall the FDH-signature scheme. The scheme $\mathsf{FDH}^H[\mathcal{TP}] = (\mathsf{Kg}, \mathsf{Sign}, \mathsf{Ver})$ is based on a trapdoor permutation $\mathcal{TP} = (G, F, \overline{F})$. To sign a message $M \in Msg$ one computes $\sigma \leftarrow \overline{F}(sk, H(M))$ for hash function $H : Msg \to Sig$, and to verify one checks that $F(pk, \sigma) = H(M)$, where (pk, sk) are the keys generated through Kg. Below we consider a very weak unforgeability notion for FDH (called *wsig*), where the adversary has to forge a signature for a random message in a key-only attack. This strengthens our result as we show that even WPROM reductions from wsig to the one-wayness of the trapdoor permutation (*owp*) perform badly.

FDH CANNOT BE SECURE IN THE WPROM. We have the following result, which states that FDH cannot be proven secure (by a black-box security analysis) in the WPROM.

Theorem 4 (WPROM Irreducibility of FDH). *Let* $\mathsf{FDH}^R[\mathcal{TP}] = (\mathsf{Kg}, \mathsf{Sign}, \mathsf{Ver})$ *be the* FDH *scheme with message space Msg and signature space Sig, relying on a trapdoor permutation* $\mathcal{TP} = (G, F, \overline{F})$ *with domain Sig and public-key/trapdoor space* $\{0,1\}^k$*, as well as on a random oracle* $R : Msg \to Sig$*. Then, for all* $t > 0$*, all* $\epsilon \leq 1$*, and all* $(\text{wsig} \to \text{owp}, \delta, t, (q_G, q_F, q_{\overline{F}}), q_\rho, q_\mathcal{A})$*-fully-BB WPROM security reductions* B *for* FDH *we have*[6]

$$\delta(\epsilon) \leq \frac{q_G + q_{\overline{F}}}{2^k} + \frac{q_F + 2q_\mathcal{A} + q_\rho + 2}{|Sig|},$$

where $q_G, q_F, q_{\overline{F}}$*, and* q_ρ *are the number of queries of* B *to* $G, F, \overline{F}$*, and* ρ*, respectively, whereas* $q_\mathcal{A}$ *is the number of adversarial instances run by* B*.*

The proof adopts a variant of the *two-oracle separation technique* by Hsiao and Reyzin [11]. For $\mathcal{F}$ and the *ideal* (i.e. random) trapdoor permutation $\mathcal{TP}^\mathcal{F} = (G, F, \overline{F})$ defined as in the proof of Theorem 3, we define for all functions ρ, an oracle $\mathcal{B} = \mathcal{B}_\rho^{\mathcal{TP}^\mathcal{F}}$ such that $\mathcal{TP}^\mathcal{F}$ is one way relative to $\mathcal{B}$ as long as ρ is one way, yet there exists an adversary $\mathcal{A}_{\mathsf{FDH}}$ forging an FDH-signature given access to $\mathcal{B}$ on *any* given message, i.e. it breaks FDH in the strongest possible sense.

Roughly speaking, the oracle $\mathcal{B}$ allows inversion of $F(pk, \cdot)$ on each output y' whenever a preimage r' of y' under ρ is exhibited: This allows inversion of $F(pk, \cdot)$ for *any* output of R_{adv}^ρ, and hence arbitrary forgeries in the WPROM. Yet, in the task of inverting $F(pk, \cdot)$ on a *random* y, coming up with a valid preimage of y under ρ is as hard as inverting ρ, and thus infeasible if ρ is one way. Therefore, the oracle $\mathcal{B}$ is only used to invert $F(pk, \cdot)$ for outputs *other than* the random challenge, which does not help it to win the OWF game.

[6] We remark that wsig adversaries are only permitted to output one forgery, and perform no queries: the function δ hence only depends on ϵ without loss of generality.

References

1. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Proceedings of the Annual Conference on Computer and Communications Security (CCS). ACM Press, New York (1993)
2. Bellare, M., Rogaway, P.: Optimal asymmetric encryption — how to encrypt with RSA. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 92–111. Springer, Heidelberg (1995)
3. Bellare, M., Rogaway, P.: The exact security of digital signatures — how to sign with RSA and Rabin. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 399–416. Springer, Heidelberg (1996)
4. Boneh, D., Franklin, M.K.: Identity-based encryption from the weil pairing. SIAM J. Comput. 32(3), 586–615 (2003)
5. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the Weil pairing. Journal of Cryptology 17(4), 297–319 (2004)
6. Canetti, R., Kilian, J., Petrank, E., Rosen, A.: Black-box concurrent zero-knowledge requires $\tilde{\Omega}(\log n)$ rounds. In: Proceedings of the Annual Symposium on the Theory of Computing, STOC 2001. pp. 570–579. ACM Press, New York (2001)
7. 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)
8. Dodis, Y., Oliveira, R., Pietrzak, K.: On the generic insecurity of the full domain hash. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 449–466. Springer, Heidelberg (2005)
9. Fujisaki, E., Okamoto, T., Pointcheval, D., Stern, J.: RSA-OAEP is secure under the RSA assumption. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, p. 260. Springer, Heidelberg (2001)
10. Hofheinz, D., Kiltz, E.: Programmable hash functions and their applications. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 21–38. Springer, Heidelberg (2008)
11. Hsiao, C.Y., Reyzin, L.: Finding collisions on a public road, or do secure hash functions need secret coins? In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 92–105. Springer, Heidelberg (2004)
12. 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)
13. 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)
14. Reingold, O., Trevisan, L., Vadhan, S.: Notions of reducibility between cryptographic primitives. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 1–20. Springer, Heidelberg (2004)
15. Shoup, V.: A proposal for an ISO standard for public key encryption (version 2.1). No. 2001/112 in Cryptology eprint archive (2001), `eprint.iacr.org`
16. Waters, B.: Efficient identity-based encryption without random oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)
17. Wee, H.: Zero knowledge in the random oracle model, revisited. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 417–434. Springer, Heidelberg (2009)

Short Pairing-Based Non-interactive Zero-Knowledge Arguments

Jens Groth*

University College London
j.groth@ucl.ac.uk

Abstract. We construct non-interactive zero-knowledge arguments for circuit satisfiability with perfect completeness, perfect zero-knowledge and computational soundness. The non-interactive zero-knowledge arguments have sub-linear size and very efficient public verification. The size of the non-interactive zero-knowledge arguments can even be reduced to a constant number of group elements if we allow the common reference string to be large. Our constructions rely on groups with pairings and security is based on two new cryptographic assumptions; we do not use the Fiat-Shamir heuristic or random oracles.

Keywords: Sub-linear size non-interactive zero-knowledge arguments, pairing-based cryptography, power knowledge of exponent assumption, computational power Diffie-Hellman assumption.

1 Introduction

Zero-knowledge proofs introduced by Goldwasser, Micali and Rackoff [24] are fundamental building blocks in cryptography that are used in numerous protocols. Zero-knowledge proofs enable a prover to convince a verifier of the truth of a statement without leaking any other information. The central properties are captured in the notions of completeness, soundness and zero-knowledge.

Completeness: The prover can convince the verifier if the prover knows a witness testifying to the truth of the statement.

Soundness: A malicious prover cannot convince the verifier if the statement is false. We distinguish between computational soundness that protects against polynomial time cheating provers and statistical or perfect soundness where even an unbounded prover cannot convince the verifier of a false statement. We will call computationally sound proofs for *arguments*.

Zero-knowledge: A malicious verifier learns nothing except that the statement is true. We distinguish between computational zero-knowledge, where a polynomial time verifier learns nothing from the proof and statistical or perfect zero-knowledge, where even a verifier with unlimited resources learns nothing from the proof.

* Supported by EPSRC grant EP/G013829/1.

M. Abe (Ed.): ASIACRYPT 2010, LNCS 6477, pp. 321–340, 2010.

The first zero-knowledge proofs relied on interaction between the prover and the verifier. Many cryptographic tasks are carried out off-line though; for instance signing or encrypting messages. For these tasks it is desirable to have *non-interactive* zero-knowledge (NIZK) proofs, where there is no interaction and a proof just consists of a single message from the prover to the verifier. Unfortunately, only languages in BPP have NIZK proofs in the plain model without any setup [22,21]. However, Blum, Feldman and Micali [6] introduced NIZK proofs in the *common reference string model*, where both the prover and verifier have access to a common reference string generated in a trusted way. Such NIZK proofs have many applications, ranging from early chosen ciphertext attack secure public-key cryptosystems [17,38] to recent advanced signature schemes [11,7]. For this reason there has been a lot of research into the underlying assumptions [19,2,28], the efficiency [13,15,33,27], and the security guarantees offered by NIZK proofs [16,38,14].

NIZK proofs based on standard cryptographic assumptions used to be inefficient and not useful in practice. To get around this inefficiency, applied cryptographers have relied on the so-called Fiat-Shamir heuristic for transforming public-coin interactive zero-knowledge proofs into NIZK arguments by using a cryptographic hash-function to compute the verifier's challenges. The Fiat-Shamir heuristic can give very efficient NIZK arguments that are secure in the random oracle model [5], where the cryptographic hash-function is modeled as a random function. It is for instance possible to use the Fiat-Shamir heuristic to transform sub-linear size interactive public-coin zero-knowledge arguments [32] into sub-linear size non-interactive zero-knowledge arguments [35]. Unfortunately, there are several examples of protocols that are secure in the random oracle model, but do not have any secure standard model instantiation no matter which hash-function is used [9,10,34,3,37]. Particularly relevant here is Goldwasser and Kalai's [23] demonstration of a signature scheme built from a public-coin identification scheme that is secure in the random oracle model but insecure in real life.

Recent works on NIZK proofs has used bilinear groups to improve efficiency. Groth, Ostrovsky and Sahai [30,29] gave NIZK proofs for circuit satisfiability where the proof consists of $O(|C|)$ group elements, with $|C|$ being the number of gates in the circuit. Their NIZK proofs have the property that they can be set up to give either perfect soundness and computational zero-knowledge, or alternatively computational soundness and perfect zero-knowledge. Works by Boyen, Waters, Groth and Sahai [7,8,25,31] have explored how to build efficient NIZK proofs that are directly applicable in bilinear groups instead of going through circuit satisfiability. In some special cases, for instance in the ring signature of Chandran, Groth and Sahai [11], these techniques lead to sub-linear size NIZK proofs but in general the number of group elements in an NIZK proof grows linearly in the size of the statement. Abe and Fehr [1] gave a construction based on commitments instead of encryptions, but since there is a commitment for each wire they also get a linear growth in the size of the circuit.

Looking at the NP-complete problem of circuit satisfiability, the reason the NIZK proofs grow linearly in the circuit size is that they encrypt the value of each wire in the circuit. Gentry's new fully homomorphic cryptosystem [20] can reduce the NIZK proof to being linear in the size of the witness: The prover encrypts the inputs to the circuit and uses the homomorphic properties of the cryptosystem to compute the output of the circuit. The prover then gives NIZK proofs for the input ciphertexts being valid and the output ciphertext containing 1. Fully homomorphic encryption only helps when the circuit has a small witness though; if the circuit has a linear number of input wires the resulting NIZK proof will also be linear in the circuit size.

1.1 Our Contribution

Micali's CS proofs [35] indicated the possibility of sub-linear size NIZK arguments, but despite more than a decade of research the Fiat-Shamir heuristic is the only known strategy for constructing sub-linear size NIZK arguments. Our goal is to introduce a new type of sub-linear size NIZK arguments where security does not rely on the random oracle model.

We construct NIZK arguments for circuit satisfiability with perfect completeness, computational soundness and perfect zero-knowledge (see Section 2 for definitions). The NIZK arguments are short and very efficient to verify, but the prover uses a super-linear number of group operations. We first give an NIZK argument consisting of a constant number of group elements but having a long common reference string. We then show that it is possible to reduce the size of the common reference string at the cost of increasing the size of the NIZK argument making them simultaneously sub-linear in the circuit size.

The soundness of our NIZK argument relies on the q-computational power Diffie-Hellman and the q-power knowledge of exponent assumptions (see Section 3). The q-CPDH assumption is a normal computational intractability assumption but the q-PKE is a so-called knowledge of exponent assumption. Knowledge of exponent assumptions have been criticized for being unfalsifiable [36] but the use of a non-standard assumption may be unavoidable since Abe and Fehr [1] have demonstrated that no statistical zero-knowledge NIZK argument for an NP-complete language has a "direct black-box" reduction to a standard cryptographic assumption unless NP $\subseteq$ P/poly.[1][2]

[1] Abe and Fehr do not rule out the existence of statistical NIZK arguments with non-adaptive soundness, where the adversary chooses the statement obliviously of the common reference string. Since the common reference string is public it is more natural to define soundness adaptively though; indeed we do not know of any practical applications of NIZK arguments with non-adaptive soundness.

[2] The very assumption that an NIZK argument is sound seems to be unfalsifiable as well since even if an adversary outputs a false statement and a convincing NIZK argument it may be hard to verify that the statement is false. Groth, Ostrovsky and Sahai [30] circumvented this problem by defining co-soundness for languages in NP $\cap$ coNP, which is falsifiable since the adversary can produce a coNP-witness certifying that the statement is false.

Table 1. Comparison of NIZK proofs and arguments

	CRS size	Proof size	Prov. comp.	Ver. comp.	Assumption
Groth [27]	$\tilde{O}(\|C\|)$ G	$\tilde{O}(\|C\|)$ G	$\tilde{O}(\|C\|)$ E	$\tilde{O}(\|C\|)$ M	trapdoor perm.
Groth [27]	$\tilde{O}(\|C\|)$ bits	$\tilde{O}(\|C\|)$ bits	$\tilde{O}(\|C\|)$ M	$\tilde{O}(\|C\|)$ M	Naccache-Stern
Gentry [20]	$O(1)$ G	$\|w\|k^{O(1)}$ G	$\|C\|k^{O(1)}$ M	$\|C\|k^{O(1)}$ M	lattice-based
G-Ostrovsky-Sahai	$O(1)$ G	$O(\|C\|)$ G	$O(\|C\|)$ E	$O(\|C\|)$ P	pairing-
[30,29]	$O(1)$ G	$O(\|C\|)$ G	$O(\|C\|)$ E	$O(\|C\|)$ E	based
Abe-Fehr [1]	$O(1)$ G	$O(\|C\|)$ G	$O(\|C\|)$ E	$O(\|C\|)$ E	knowledge of expo.
Groth [26]	$O(\|C\|^{\frac{1}{2}})$ G	$O(\|C\|^{\frac{1}{2}})$ G	$O(\|C\|)$ M	$O(\|C\|)$ M	random oracle
This paper	$O(\|C\|^{2})$ G	$O(1)$ G	$O(\|C\|^{2})$ M	$O(\|C\|)$ M	PKE and CDHP
This paper	$O(\|C\|^{\frac{2}{3}})$ G	$O(\|C\|^{\frac{2}{3}})$ G	$O(\|C\|^{\frac{4}{3}})$ M	$O(\|C\|)$ M	PKE and CDHP

Table 1 gives a comparison to other NIZK proofs and arguments for circuit satisfiability, where k is a security parameter, G stands for the size of a group element, M and E are the costs of respectively multiplications and exponentiations, and P is the cost of a pairing in a bilinear group (see Section 3).

Compared to other pairing-based NIZK arguments, our arguments are smaller and faster to verify. The prover uses a super-linear number of multiplications and the computational cost may grow beyond a linear number of exponentiations. The public verifiability means that the NIZK arguments are transferable though; they can be copied and distributed to many different entities that can do their own independent verification. The prover only pays a one-time cost for computing the NIZK argument, while all verifiers enjoy the benefits of low transmission bandwidth and efficient verification.

Perfect Zaps. The common reference string model assumes a trusted setup for generating common reference strings and making them available to the prover and verifier. In case no such setup is available[3] we can still get a sub-linear size 2-move publicly verifiable witness-indistinguishable argument where the verifiers first message can be reused many times, a so-called Zap [18], as follows: The verifier generates a common reference string. The prover verifies that the common reference string is well-formed (our common reference string is not a random bit-string, but it does have a certain structure that makes it possible to verify that it is well-formed) and can now make arbitrarily many Zaps using the verifier initial message as the common reference string. Since our NIZK argument is perfectly zero-knowledge, the Zaps will be perfectly witness-indistinguishable.

1.2 Outline of Our NIZK Argument

We will construct NIZK arguments for the existence of an input to a binary circuit C making it output 1. At a loss of a constant factor, we may assume C consists of NAND-gates. Furthermore, if we label the output wire a we may add

[3] We remark that even if the common reference string is adversarially chosen the sub-linearity of our NIZK arguments impose an information theoretic upper bound on how much information can be leaked.

a self-loop to the circuit consisting of a NAND-gate $a = \neg(a \wedge b)$ forcing a to be 1. This reduces the challenge to prove that there is an assignment of truth-values to the wires that respect all the $N = |C|$ NAND-gates in the circuit.

The NIZK argument relies on length-reducing commitments where we commit to n values in a finite field $\mathbb{Z}_p$ using only a constant number of group elements. We will also use non-interactive arguments consisting of a constant number of group elements for proving the following properties about committed values:

Entry-wise product: Commitments c, d, v contain values $a_1, \ldots, a_n, b_1, \ldots, b_n$ and $u_1, \ldots, u_n$ that satisfy $u_i = a_i b_i$ for all i.

Permutation: Commitments c, d contain values $a_1, \ldots, a_n$ and $b_1, \ldots, b_n$ that satisfy $b_i = a_{\rho(i)}$ for all i, where ρ is a publicly known permutation of n elements.

Let us sketch how commitments combined with these two types of non-interactive arguments give us a constant size NIZK argument for circuit satisfiability when $n = 2N$. The prover gets as a witness for the satisfiability of the circuit $a_1, \ldots, a_N$ and $b_1, \ldots, b_N$ such that a_i, b_i are the inputs to gate i and all the values are consistent with the wires and respect the NAND-gates. We use the convention that -1 corresponds to false and $+1$ corresponds to true, so if u_i is the output of gate i we have $u_i = -a_i b_i$.

The prover makes commitments to the $2N$-tuples

$$(a_1, \ldots, a_N, b_1, \ldots, b_N) \qquad (b_1, \ldots, b_N, 0, \ldots, 0) \qquad (-u_1, \ldots, -u_N, 0, \ldots, 0).$$

The prover gives an entry-wise product argument on the commitment to $(a_1, \ldots, a_N, b_1, \ldots, b_N)$ with itself to show $a_i^2 = 1$ and $b_i^2 = 1$ for all i. This shows that $a_1, \ldots, a_N, b_1, \ldots, b_N \in \{-1, 1\}$ are appropriate truth values.

An output of one NAND-gate may be the input of other NAND-gates, which means the corresponding values $a_{i_1}, \ldots, a_{i_\ell}, b_{j_1}, \ldots, b_{j_m}$ have to have the same assignment. The prover picks a permutation ρ that contains cycles $i_1 \to i_2 \to \ldots \to i_\ell \to j_1 + N \to j_2 + N \to \ldots \to j_m + N \to i_1$ for all such sets of values that have to be consistent and gives a permutation argument on the commitment to $(a_1, \ldots, a_N, b_1, \ldots, b_N)$. This shows for each set of values corresponding to the same output wire that $a_{i_2} = a_{i_1}, \ldots, b_{j_1} = a_{i_\ell}, \ldots, b_{j_m} = b_{j_{m-1}}$ so the values $(a_1, \ldots, a_N, b_1, \ldots, b_N)$ are consistent with the wiring of the circuit.

The prover uses additional commitments, entry-wise product and permutation arguments to show that the other committed values $(b_1, \ldots, b_N, 0, \ldots, 0)$ and $(-u_1, \ldots, -u_N, 0, \ldots, 0)$ are consistent with the wiring of the circuit and the values $(a_1, \ldots, a_N, b_1, \ldots, b_N)$, we refer to Section 8 for the details.

Finally, the prover uses the entry-wise product argument to show that the entry-wise product of $(a_1, \ldots, a_N, b_1, \ldots, b_N)$ and $(b_1, \ldots, b_N, 0, \ldots, 0)$ is $(-u_1, \ldots, -u_N, 0, \ldots, 0)$ so all the values respect the NAND gates.

This outline shows how to get a constant size NIZK argument for circuit satisfiability, but to enable the entry-wise product arguments and the permutation arguments the common reference string has size $O(N^2)$ group elements. In Section 9 we reduce the common reference string size by using commitments to n

elements where $n < N$. With n smaller than $2N$ we need to give permutation arguments that span accross multiple commitments though. Using permutation network techniques [12] we manage to build such large permutations from many smaller permutations.

The technical contribution of this paper is the construction of an appropriate commitment scheme with corresponding non-interactive entry-wise product and permutation arguments. The commitment scheme is a variant of the Pedersen commitment scheme, where the commitment key is of the form $(g, g^x, \ldots, g^{x^q})$. A commitment to $a_1, \ldots, a_q$ is a single group element computed as $g^r \prod_{i=1}^{q} (g^{x^i})^{a_i}$.

The nice thing about such a commitment is that the discrete logarithm is a polynomial $r + \sum_{i=1}^{q} a_i x^i$. When we pair two commitments with each other we get a product of two polynomials in the exponent. By taking appropriate linear combinations over products of polynomials, we can express entry-wise products and permutations as equations over the coefficients of these polynomials. The q-CPDH assumption then allows us to conclude that these coefficients are identical and therefore the committed values satisfy an entry-wise multiplication relationship or a permutation relationship to each other.

When pairing commitments (equivalent to multiplying polynomials in the exponent) there will be various cross-terms. The role of the non-interactive arguments will be to cancel out these terms. Usually, a single group element paired with g suffices to cancel out all the cross-terms, so the non-interactive arguments for entry-wise products and permutations are highly efficient themselves.

To prove that our NIZK argument is sound, we need to reason about the coefficient of these polynomials. However, a cheating prover might create a commitment without knowing an opening of it. This is where the q-PKE assumption comes in handy: the prover gives non-interactive arguments demonstrating that it "knows" the openings of the commitments. By this we mean that there is an extractor that given the same input as the prover can reconstruct the commitments together with the openings of the commitments.

2 Definitions

Let R be an efficiently computable binary relation. For pairs $(C, w) \in R$ we call C the statement and w the witness. Let L be the NP-language consisting of statements with witnesses in R. When we restrict ourselves to statements of size N, we write respectively L_N and R_N.

A non-interactive argument for a relation R consists of a common reference string generator algorithm K, a prover algorithm P and a verifier algorithm V that run in probabilistic polynomial time. The common reference string generator takes as input a security parameter k and the statement size N and produces a common reference string σ. The prover on input (σ, C, w) produces an argument π. The verifier on input (σ, C, π) outputs 1 if the argument is acceptable and 0 if rejecting the argument. We call (K, P, V) an argument for R if it has the completeness and soundness property described below.

PERFECT COMPLETENESS. Completeness captures the notion that an honest prover should be able to convince an honest verifier if the statement is true. For $N = k^{O(1)}$ and all adversaries $\mathcal{A}$ outputting $(C, w) \in R_N$:

$$\Pr\left[\sigma \leftarrow K(1^k, N); (C, w) \leftarrow \mathcal{A}(\sigma); \pi \leftarrow P(\sigma, C, w) : V(\sigma, C, \pi) = 1\right] = 1.$$

COMPUTATIONAL SOUNDNESS. Soundness captures the notion that it should be infeasible for an adversary to come up with an accepting argument for a false statement. For $N = k^{O(1)}$ and all non-uniform polynomial time adversaries $\mathcal{A}$:

$$\Pr\left[\sigma \leftarrow K(1^k, N); (C, \pi) \leftarrow \mathcal{A}(\sigma) : C \notin L \text{ and } V(\sigma, C, \pi) = 1\right] \approx 0.$$

PERFECT WITNESS-INDISTINGUISHABILITY. We say a non-interactive argument (K, P, V) is perfectly witness-indistinguishable if it is impossible to tell which witness the prover when there are many possible witnesses. For $N = k^{O(1)}$ and all stateful interactive adversaries $\mathcal{A}$ outputting $(C, w_0), (C, w_1) \in R_N$:

$$\begin{aligned}&\Pr\left[\sigma \leftarrow K(1^k, N); (C, w_0, w_1) \leftarrow \mathcal{A}(\sigma); \pi \leftarrow P(\sigma, C, w_0) : \mathcal{A}(\pi) = 1\right] \\ =\ &\Pr\left[\sigma \leftarrow K(1^k, N); (C, w_0, w_1) \leftarrow \mathcal{A}(\sigma); \pi \leftarrow P(\sigma, C, w_1) : \mathcal{A}(\pi) = 1\right].\end{aligned}$$

PERFECT ZERO-KNOWLEDGE. An argument is zero-knowledge if it does not leak any information besides the truth of the statement. We say a non-interactive argument (K, P, V) is perfect zero-knowledge if there exists a polynomial time simulator $S = (S_1, S_2)$ with the following zero-knowledge property. S_1 outputs a simulated common reference string and a simulation trapdoor. S_2 takes the common reference string, the simulation trapdoor and a statement as input and produces a simulated argument. For $N = k^{O(1)}$ and all stateful interactive adversaries $\mathcal{A}$ outputting $(C, w) \in R_N$:

$$\begin{aligned}&\Pr\left[\sigma \leftarrow K(1^k, N); (C, w) \leftarrow \mathcal{A}(\sigma); \pi \leftarrow P(\sigma, C, w) : \mathcal{A}(\pi) = 1\right] \\ =\ &\Pr\left[(\sigma, \tau) \leftarrow S_1(1^k, N); (C, w) \leftarrow \mathcal{A}(\sigma); \pi \leftarrow S_2(\sigma, \tau, C) : \mathcal{A}(\pi) = 1\right].\end{aligned}$$

3 Bilinear Groups

NOTATION. Given two functions $f, g : \mathbb{N} \to [0, 1]$ we write $f(k) \approx g(k)$ when $|f(k) - g(k)| = O(k^{-c})$ for every constant $c > 0$. We say that f is *negligible* when $f(k) \approx 0$ and that it is *overwhelming* when $f(k) \approx 1$.

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 the set S. We will assume it is possible to sample uniformly at random from sets such as $\mathbb{Z}_p$. We define $[n]$ to be the set $\{1, 2, \ldots, n\}$.

BILINEAR GROUPS. Let $\mathcal{G}$ take a security parameter k written in unary as input and output a description of a bilinear group $(p, G, G_T, e) \leftarrow \mathcal{G}(1^k)$ such that

1. p is a k-bit prime.
2. G, G_T are cyclic groups of order p.
3. $e : G \times G$ is a bilinear map (pairing) such that $\forall a, b : e(g^a, g^b) = e(g, g)^{ab}$.
4. If g generates G then $e(g, g)$ generates G_T.
5. Membership in G, G_T can be efficiently decided, group operations and the pairing e are efficiently computable, generators are efficiently sampleable, and the descriptions of the groups and group elements each have size $O(k)$ bits.

The security of our NIZK arguments will be based on two new assumptions, which we call respectively the q-power knowledge of exponent assumption and the q-computational power Diffie-Hellman assumption.

The q-power knowledge of exponent assumption. The knowledge of exponent (KEA) assumption says that given g, g^α it is infeasible to create $c, \hat{c}$ so $\hat{c} = c^\alpha$ without knowing a so $c = g^a$ and $\hat{c} = (g^\alpha)^a$. Bellare and Palacio [4] extended this to the KEA3 assumption, which says that given $g, g^x, g^\alpha, g^{\alpha x}$ it is infeasible to create $c, \hat{c}$ so $\hat{c} = c^\alpha$ without knowing a_0, a_1 so $c = g^{a_0}(g^x)^{a_1}$ and $\hat{c} = (g^\alpha)^{a_0}(g^{\alpha x})^{a_1}$.

The q-power knowledge of exponent assumption is a generalization of KEA and KEA3. It says that given $(g, g^x, \ldots, g^{x^q}, g^\alpha, g^{\alpha x}, \ldots, g^{\alpha x^q})$ it is infeasible to create $c, \hat{c}$ so $\hat{c} = c^\alpha$ without knowing $a_0, \ldots, a_q$ so $c = \prod_{i=0}^{q}(g^{x^i})^{a_i}$ and $\hat{c} = \prod_{i=0}^{q}(g^{\alpha x^i})^{a_i}$.

We will now give the formal definition of the q-power knowledge of exponent assumption. Following Abe and Fehr [1] we write $(y; z) \leftarrow (\mathcal{A} \parallel \mathcal{X}_\mathcal{A})(x)$ when $\mathcal{A}$ on input x outputs y and $\mathcal{X}_\mathcal{A}$ on the same input (including the random tape of $\mathcal{A}$) outputs z.

Definition 1 (q-PKE). *The $q(k)$-power knowledge of exponent assumption holds for $\mathcal{G}$ if for every non-uniform probabilistic polynomial time adversary $\mathcal{A}$ there exists a non-uniform probabilistic polynomial time extractor $\mathcal{X}_\mathcal{A}$ so*

$$\Pr\Big[(p, G, G_T, e) \leftarrow \mathcal{G}(1^k)\ ;\ g \leftarrow G \setminus \{1\}\ ;\ \alpha, x \leftarrow \mathbb{Z}_p^*\ ;\ \sigma = (p, G, G_T, e, g, g^x, \ldots, g^{x^q}, g^\alpha, g^{\alpha x}, \ldots, g^{\alpha x^q})\ ;\ (c, \hat{c}\ ;\ a_0, \ldots, a_q) \leftarrow (\mathcal{A} \parallel \mathcal{X}_\mathcal{A})(\sigma)\ :\ \hat{c} = c^\alpha \wedge c \neq \prod_{i=0}^{n} g^{a_i x^i}\Big] \approx 0.$$

The q-computational power Diffie-Hellman assumption. The computational Diffie-Hellman (CDH) assumption says that given g, g^β, g^x it is infeasible to compute $g^{\beta x}$. The q-computational power Diffie-Hellman assumption is a generalization of the CDH assumption that says given $(g, g^x, \ldots, g^{x^q}, g^\beta, g^{\beta x}, \ldots, g^{\beta x^q})$ except for one missing elements $g^{\beta x^j}$, it is hard to compute the missing element.

Definition 2 (q-CPDH). *The $q(k)$-computational power Diffie-Hellman assumption holds for $\mathcal{G}$ if for all $j \in \{0, \ldots, q\}$ and all non-uniform probabilistic*

polynomial time adversaries $\mathcal{A}$ we have

$$\Pr\Big[(p,G,G_T,e) \leftarrow \mathcal{G}(1^k)\ ;\ g \leftarrow G \setminus \{1\}\ ;\ \beta, x \leftarrow \mathbb{Z}_p^*\ ; \\ y \leftarrow (\mathcal{A},\mathcal{X}_{\mathcal{A}})(p,G,G_T,e,g,g^x,\ldots,g^{x^q},g^\beta,g^{\beta x},\ldots, \\ g^{\beta x^{j-1}},g^{\beta x^{j+1}},\ldots,g^{\beta x^q})\ :\ \ y = g^{\beta x^j}\Big]\ \approx\ 0.$$

In the full paper we give heuristic arguments for believing in the q-PKE and q-CPDH assumptions by proving that they hold in the generic group model.

4 Knowledge Commitment

We will use a variant of the Pedersen commitment scheme in our NIZK proof where we commit to $a_1,\ldots,a_q$ as $c = g^r \prod_{i\in[q]} g_i^{a_i}$. In the security proof of our NIZK argument for 3SAT we will need to extract the committed values $a_1,\ldots,a_q$; but the commitment scheme itself is perfectly hiding and does not reveal the committed values. For this reason, we will require the prover to create a related commitment $\hat{c} = \hat{g} \prod_{i\in[q]} \hat{g}_i^{a_i}$ and will rely on the q-PKE assumption for extracting the committed values. We call $(c,\hat{c})$ a knowledge commitment, since the prover cannot make a valid commitment without "knowing" the committed values.

Key generation: Pick $gk = (p,G,G_T,e) \leftarrow \mathcal{G}(1^k)\ \ g \leftarrow G \setminus \{1\}\ ;\ x,\alpha \leftarrow \mathbb{Z}_p^*$. The commitment key is $ck = (gk, g, g_1,\ldots,g_q,\hat{g},\hat{g}_1\ldots,\hat{g}_q) = (gk,g,g^x,\ldots,g^{x^q},g^\alpha,g^{\alpha x},\ldots,g^{\alpha x^q})$ and the trapdoor key is $tk = x$.

Commitment: To commit to $a_1,\ldots,a_q$ pick $r \leftarrow \mathbb{Z}_p$ and compute the knowledge commitment $(c,\hat{c})$ as

$$c = g^r \prod_{i\in[q]} g_i^{a_i} \qquad\qquad \hat{c} = \hat{g}^r \prod_{i\in[q]} \hat{g}_i^{a_i}.$$

Given $(c,\hat{c}) \in G^2$ we can verify that it is well-formed by checking $e(g,\hat{c}) = e(c,\hat{g})$.

Trapdoor commitment: To make a trapdoor commitment sample trapdoor randomness $t \leftarrow \mathbb{Z}_p$ and compute the knowledge commitment $(c,\hat{c})$ as $c = g^t\ ;\ \hat{c} = \hat{g}^t$.

Trapdoor opening: The trapdoor opening algorithm on messages $a_1,\ldots,a_q \in \mathbb{Z}_p$ returns the randomizer $r = t - \sum_{i\in[q]} a_i x^i$. The trapdoor opening satisfies $c = g^r \prod_{i\in[q]} g_i^{a_i}$ and $\hat{c} = \hat{g}^r \prod_{i\in[q]} \hat{g}_i^{a_i}$.

The commitment scheme has properties similar to those of standard Pedersen commitments as the following theorem shows. We refer to the full paper for the proof of the following theorem.

Theorem 1. *The commitment scheme is perfectly trapdoor and computationally binding. Assuming the q-PKE assumption holds, there exists for any non-uniform probabilistic polynomial time committer $\mathcal{A}$ a non-uniform probabilistic polynomial time extractor $\mathcal{X}_{\mathcal{A}}$ that computes the contents of the commitment when given the input of $\mathcal{A}$ (including any random coins).*

4.1 Restriction Argument

Consider a subset $S \subset [q]$ and a commitment c. We will need an argument for the opening $r, a_1, \ldots, a_q$ being such that the indices of non-zero values are restricted to S. In other words, we need an argument for the commitment being of the form $c = g^r \prod_{i \in S} g_i^{a_i}$. The argument will take the form $\pi = h^r \prod_{i \in S} h_i^{a_i}$, which intuitively corresponds to an additional argument of knowledge with respect to a small base $(h, \{h_i\}_{i \in S})$.

Setup: $gk \leftarrow \mathcal{G}(1^k)$; $ck \leftarrow K_{\text{commit}}(gk)$.
Common reference string: Given (ck, S) as input pick at random $\beta \leftarrow \mathbb{Z}_p^*$ and compute the common reference string as $\sigma = (h, \{h_i\}_{i \in S}) = (g^\beta, \{g_i^\beta\}_{i \in S})$.
Statement: A valid knowledge commitment $(c, \hat{c})$.
Prover's witness: Opening $r, \{a_i\}_{i \in S}$ so $c = g^r \prod_{i \in S} g_i^{a_i}$ and $\hat{c} = \hat{g}^r \prod_{i \in S} \hat{g}_i^{a_i}$.
Argument: Compute the argument as $\pi = h^r \prod_{i \in S} h_i^{a_i}$.
Verification: Output 1 if and only if $e(c, h) = e(g, \pi)$.

Theorem 2. *The restriction argument is perfectly complete and perfectly witness-indistinguishable. If the q-CPDH assumption holds, all non-uniform probabilistic polynomial time adversaries have negligible probability of outputting $(r, a_1, \ldots, a_q, \pi)$ so $a_i \neq 0$ for some $i \notin S$ and π is an acceptable restriction argument for the commitment corresponding to the opening.*

We refer to the full paper for the proof. Observe that we phrase the soundness of the restriction argument as the inability to find an opening of the commitment that violates the restriction. Since the commitment scheme is perfectly hiding we cannot exclude the existence of openings that violate the restriction. However, if it holds that it is a knowledge commitment (Theorem 1) we see that the opening we extract from the committer must respect the restriction.

5 Common Reference String

We will now describe how to generate the common reference string for our NIZK argument. The common reference string will consist of a knowledge commitment key ck for $q = n^2 + 3n - 2$ values together with three common reference strings for restriction to the sets

$$\tilde{S} = \{1, \ldots, n\} \ , \ \bar{S} = \{(n+1), \ldots, n(n+1)\} \ , \ \dot{S} = \{\ell \in [q] \mid \ell \neq 0 \bmod n+2\}.$$

The zero-knowledge simulation of the argument will use the same type of common reference string, and the simulation trapdoor for our NIZK argument will be the trapdoor for the knowledge commitment.

Common Reference String Generation:
On input 1^k and n do
1. Generate $(p, G, G_T, e) \leftarrow \mathcal{G}(1^k)$ and set $gk = (p, G, G_T, e)$.

2. Pick $g \leftarrow G \setminus \{1\}$; $x, \alpha \leftarrow \mathbb{Z}_p^*$ and compute

$$ck = (gk, g, \ldots, g_q, \hat{g}, \ldots, \hat{g}_q) = (gk, g, \ldots, g^{x^{n^2+3n-2}}, g^{\alpha}, \ldots, g^{\alpha x^{n^2+3n-2}}).$$

3. Generate $\tilde{\sigma} \leftarrow K_{\text{restrict}}(ck, \tilde{S})$ where $\tilde{S} = \{1, 2, \ldots, n\}$.
4. Generate $\bar{\sigma} \leftarrow K_{\text{restrict}}(ck, \bar{S})$ where $\bar{S} = \{(n+1), 2(n+1), \ldots, n(n+1)\}$.
5. Generate $\dot{\sigma} \leftarrow K_{\text{restrict}}(ck, \dot{S})$ where $\dot{S} = \{\ell \in [q] \mid \ell \neq 0 \bmod n+2\}$.

The common reference string is $\sigma = (ck, \tilde{\sigma}, \bar{\sigma}, \dot{\sigma})$ and the simulation trapdoor is $tk = x$.

Given a common reference string, it is hard to find a non-trivial linear combination of $1, x, \ldots, x^q$ because we could run a polynomial factorization algorithm in $\mathbb{Z}_p[X]$ to compute the root x. We will repeatedly use this fact, so we prove the following Lemma in the full paper.

Lemma 1. *If the q-CPDH assumption holds for $\mathcal{G}$ with $q = n^2 + 3n - 2$, a non-uniform probabilistic polynomial time adversary has negligible chance of finding a non-trivial linear combination $(a_0, \ldots, a_q)$ such that $\sum_{i=0}^{q} a_i x^i = 0$ given a random common reference string σ.*

6 Product Argument

Consider three commitments

$$c = g^r \prod_{i \in [n]} g_i^{a_i} \qquad d = g^s \prod_{j \in [n]} g_{j(n+1)}^{b_j} \qquad v = g^t \prod_{i \in [n]} g_i^{u_i} \qquad \forall i \in [n] : u_i = a_i b_i.$$

With the corresponding restriction arguments, $\hat{c}, \tilde{c}, \hat{d}, \bar{d}, \hat{v}, \tilde{v}$ we can assume the committer knows openings to values $a_1, \ldots, a_n, b_1, \ldots, b_n$ and $u_1, \ldots, u_n$. We will give an argument $(\pi, \hat{\pi}, \dot{\pi})$ consisting of three group elements for the committed values satisfying $u_1 = a_1 b_1, \ldots, u_n = a_n b_n$.

In order to explain the intuition in the argument, let us consider the following toy example $c = \prod_{i \in [n]} g_i^{a_i}$ and $d = \prod_{j \in [n]} g_{j(n+1)}^{b_j}$, where we want to show $a_1 b_1 = 0, \ldots, a_n b_n = 0$. The discrete logarithms of the two commitments are $\sum_{i \in [n]} a_i x^i$ and $\sum_{j \in [n]} b_j x^{j(n+1)}$ and the discrete logarithm of $e(c, d)$ is

$$\left(\sum_{i \in [n]} a_i x^i \right) \cdot \left(\sum_{j \in [n]} b_j x^{j(n+1)} \right) = \sum_{i \in [n]} a_i b_i x^{i(n+2)} + \sum_{i \in [n]} \sum_{j \in [n] \setminus \{i\}} a_i b_j x^{j(n+1)+i}.$$

In the final sum, the left term contains the coefficients $a_1 b_1, \ldots, a_n b_n$ that are supposed to be 0, however, the right term complicates matters. The argument π will be constructed such that it can be used to cancel out the latter term.

Notice that the left term isolates the coefficients of $x^{n+2}, \ldots, x^{n(n+2)}$, while the right term does not contain any such coefficients. By giving an appropriate restriction argument, the prover can guarantee that she only cancels out the

right term without interfering with the left term containing $x^{n+2}, \ldots, x^{n(n+2)}$. The prover computes $\pi = \prod_{i\in[n]} \prod_{j\in[n]\setminus\{i\}} g_{j(n+1)+i}^{a_i b_j}$ and gives corresponding $\hat{\pi}, \dot{\pi}$ values demonstrating that it knows an opening $(z, \{z_\ell\}_{\ell\in\dot{S}})$ of π restricted to $\dot{S}$. The verifier will check

$$e(c,d) = e(g,\pi).$$

Let us now argue that we have soundness: given $\pi, \hat{\pi}, \dot{\pi}$ such that $e(c,d) = e(g,\pi)$ the verifier can be assured that $a_1 b_1 = 0, \ldots, a_n b_n = 0$. Taking discrete logarithms, the verification equation tells us that

$$\sum_{i\in[n]} a_i b_i x^{i(n+2)} + \sum_{i\in[n]} \sum_{j\in[n]\setminus\{i\}} a_i b_j x^{j(n+1)+i} = z + \sum_{\ell\in\dot{S}} z_\ell x^\ell.$$

Recall, $\dot{S} = \{\ell \in [n^2+3n-2] \mid \ell \neq 0 \bmod n+2\}$ so the argument π will not contain any coefficients of the form $x^{n+2}, \ldots, x^{n(n+2)}$. This means the coefficients of $x^{n+2}, \ldots, x^{n(n+2)}$ are $a_1 b_1, \ldots, a_n b_n$. If there is an i such that $a_i b_i \neq 0$, then we have a non-trivial polynomial equation in x. By Lemma 1 this would allow us to recover x and breaking the q-PKE assumption.

In the general case we want to give an argument for $a_i b_i = u_i$ instead of just $a_i b_i = 0$. However, if we evaluate $e(v, \prod_{j\in[n]} g_{j(n+1)})$ we can view the latter as a commitment to $(1,1,\ldots,1)$ and we will get their products $u_1 \cdot 1, \ldots, u_n \cdot 1$ as coefficients of $x^{n+2}, \ldots, x^{n(n+2)}$. If $u_1 = a_1 b_1, \ldots, u_n = a_n b_n$ the two pairings $e(c,d)$ and $e(v, \prod_{j\in[n]} g_{j(n+1)})$ therefore have the same coefficients of $x^{n+2}, \ldots, x^{n(n+2)}$ and otherwise the coefficients are different. As in the toy example above, we may choose π such that it cancels out all the other terms. Due to the restriction to $\dot{S}$ the argument will not have any $x^{n+2}, \ldots, x^{n(n+2)}$ terms and we therefore get soundness. In the general case, the commitments also have randomizers and we will choose π such that it also cancels out these terms.

Statement: Commitments $c, d, v \in G$.
Prover's witness: Openings $r, a_1, \ldots, a_n$ and $s, b_1, \ldots, b_n$ and $t, u_1, \ldots, u_n$ so

$$c = g^r \prod_{i\in[n]} g_i^{a_i} \ , \ d = g^s \prod_{j\in[n]} g_{j(n+1)}^{b_j} \ , \ v = g^t \prod_{i\in[n]} g_i^{u_i} \ , \ \forall i \in [n] : u_i = a_i b_i.$$

Argument: Compute the argument $(\pi, \hat{\pi}, \dot{\pi})$ as

$$\pi = g^{rs} \prod_{i\in[n]} g_i^{a_i s} \prod_{j\in[n]} g_{j(n+1)}^{b_j r - t} \prod_{i\in[n]} \prod_{j\in[n]\setminus\{i\}} g_{j(n+1)+i}^{a_i b_j - u_i}$$
$$\hat{\pi} = \hat{g}^{rs} \prod_{i\in[n]} \hat{g}_i^{a_i s} \prod_{j\in[n]} \hat{g}_{j(n+1)}^{b_j r - t} \prod_{i\in[n]} \prod_{j\in[n]\setminus\{i\}} \hat{g}_{j(n+1)+i}^{a_i b_j - u_i}$$
$$\dot{\pi} = \dot{h}^{rs} \prod_{i\in[n]} \dot{h}_i^{a_i s} \prod_{j\in[n]} \dot{h}_{j(n+1)}^{b_j r - t} \prod_{i\in[n]} \prod_{j\in[n]\setminus\{i\}} \dot{h}_{j(n+1)+i}^{a_i b_j - u_i}$$

Verification: Output 1 if and only if

$$e(g,\hat{\pi}) = e(\pi,\hat{g}) \ \wedge \ e(g,\dot{\pi}) = e(\pi,\dot{h}) \ \wedge \ e(c,d) = e(v,\prod_{j\in[n]} g_{j(n+1)})e(g,\pi).$$

Theorem 3. *The product argument has perfect completeness and perfect witness-indistinguishability. If the q-CPDH assumption holds, then a non-uniform probabilistic polynomial time adversary has negligible chance of outputting commitments (c,d,v) and an accepting argument π with corresponding openings of the commitments and the argument such that for some $i \in [n]$ we have $a_i b_i \neq u_i$.*

The proof can be found in the full paper.

The product argument has two commitments with restriction to $\tilde{S}$ and one commitment restricted to $\bar{S}$. It is quite easy to translate commitments back and forth between $\tilde{S}$ and $\bar{S}$ though. If we have two commitments v and d restricted to respectively $\tilde{S}$ and $\bar{S}$, we can give a product argument for the values in v being the product of the values in $c = \prod_{i\in[n]} g_i$ and d. Since c is a commitment to $(1,\ldots,1)$ this proves that v and d contain the same values.

The product argument makes it possible to prove that the committed values in a commitment c are bits encoded as ± 1. If we give a product argument for $\prod_{i\in[n]} g_i$ (a commitment to $(1,\ldots,1)$) being the product of the values in c and in d, where d contains the same values as c, then we have that the values satisfy $a_i^2 = 1$, which implies $a_i = \pm 1$.

7 Permutation Argument

Consider two commitments and a permutation

$$c = g^r \prod_{i\in[n]} g_i^{a_i} \qquad d = g^s \prod_{i\in[n]} g_i^{b_i} \qquad \rho \in S_n \qquad \forall i \in [n]: \ b_i = a_{\rho(i)}.$$

We will now give an argument for the committed values satisfying $b_i = a_{\rho(i)}$, where $\rho \in S_n$ is a publicly known permutation.

The idea behind the permutation argument is to show

$$\sum_{i\in[n]} a_i x^{i(n+2)} = \sum_{i\in[n]} b_i x^{\rho(i)(n+2)}.$$

By Lemma 1 this implies $b_i = a_{\rho(i)}$ for all $i \in [n]$.

To get the desired linear combination we compute $e(c, \prod_{j\in[n]} g_{j(n+1)})$ and $e(d, \prod_{j\in[n]} g_{\rho(j)(n+2)-j})$. They have discrete logarithms

$$r\sum_{j\in[n]} x^{j(n+1)} + \sum_{i\in[n]} a_i x^{i(n+2)} + \sum_{i\in[n]}\sum_{j\in[n]\setminus\{i\}} a_i x^{j(n+1)+i}$$

$$s\sum_{j\in[n]} x^{\rho(j)(n+2)-j} + \sum_{i\in[n]} b_i x^{\rho(i)(n+2)} + \sum_{i\in[n]}\sum_{j\in[n]\setminus\{i\}} b_i x^{\rho(j)(n+2)+i-j}$$

We have the desired sums $\sum_{i\in[n]} a_i x^{i(n+2)}$ and $\sum_{i\in[n]} b_i x^{\rho(i)(n+2)}$ but due to the extra terms it is not the case that $e(c, \prod_{j\in[n]} g_{j(n+1)}) = e(d, \prod_{j\in[n]} g_{\rho(j)(n+2)-j})$.

The prover will construct an argument π that cancels out the extra terms and the verifier will check that

$$e(c, \prod_{j\in[n]} g_{j(n+1)}) = e(d, \prod_{j\in[n]} g_{\rho(j)(n+2)-j}) e(g, \pi).$$

The prover also gives a restriction argument $\hat{\pi}, \dot{\pi}$ such that the verifier is guaranteed that π does not contain any $x^{n+2}, \ldots, x^{n(n+2)}$ terms. Soundness now follows from the verification equation giving us $\sum_{i\in[n]} a_i x^{i(n+2)} = \sum_{i\in[n]} b_i x^{\rho(i)(n+2)}$ when π is free of $x^{n+2}, \ldots, x^{n(n+2)}$ terms.

Statement: Commitments $c, d \in G$ and permutation $\rho \in S_n$.
Prover's witness: Openings $r, a_1, \ldots, a_n \in \mathbb{Z}_p$ and $s, b_1, \ldots, b_n \in \mathbb{Z}_p$ so

$$c = g^r \prod_{i\in[n]} g_i^{a_i} \quad \text{and} \quad d = g^s \prod_{i\in[n]} g_i^{b_i} \quad \text{and} \quad \forall i \in [n] : b_i = a_{\rho(i)}.$$

Argument: Compute the argument as

$$\pi = \prod_{j\in[n]} g_{j(n+1)}^r g_{\rho(j)(n+2)-j}^{-s} \prod_{i\in[n]} \prod_{j\in[n]\setminus\{i\}} g_{j(n+1)+i}^{a_i} g_{\rho(j)(n+2)+i-j}^{-b_i}$$

$$\hat{\pi} = \prod_{j\in[n]} \hat{g}_{j(n+1)}^r \hat{g}_{\rho(j)(n+2)-j}^{-s} \prod_{i\in[n]} \prod_{j\in[n]\setminus\{i\}} \hat{g}_{j(n+1)+i}^{a_i} \hat{g}_{\rho(j)(n+2)+i-j}^{-b_i}$$

$$\dot{\pi} = \prod_{j\in[n]} \dot{h}_{j(n+1)}^r \dot{h}_{\rho(j)(n+2)-j}^{-s} \prod_{i\in[n]} \prod_{j\in[n]\setminus\{i\}} \dot{h}_{j(n+1)+i}^{a_i} \dot{h}_{\rho(j)(n+2)+i-j}^{-b_i}$$

Verification: Output 1 if and only if $e(g, \hat{\pi}) = e(\pi, \hat{g})$, $e(g, \dot{\pi}) = e(\pi, \dot{h})$ and $e(c, \prod_{j\in[n]} g_{j(n+1)}) = e(d, \prod_{j\in[n]} g_{\rho(j)(n+2)-j}) e(g, \pi)$.

Theorem 4. *The permutation argument has perfect completeness and perfect witness-indistinguishability. If the q-CPDH assumption holds, a non-uniform probabilistic polynomial time adversary has negligible chance of outputting a permutation ρ, commitments (c, d) and an acceptable argument $(\pi, \hat{\pi}, \dot{\pi})$ with corresponding openings of the commitments and the argument such that for some $i \in [n]$ we have $b_i \neq a_{\rho(i)}$.*

The proof can be found in the full paper.

8 Constant Size NIZK Argument for Circuit Satisfiability

We will now give an NIZK argument for the satisfiability of a NAND-gate circuit C, which consists of a constant number of group elements but has a large common reference string. Let a be the output wire of the circuit and add an extra self-looping NAND gate $a = \neg(a \wedge b)$ to force a to be true. This reduces the

satisfiability problem to demonstrating that there is a truth-value assignment to the wires such that C is internally consistent with all the NAND-gates. In the following let the value -1 correspond to false and $+1$ correspond to true. We now give the full NIZK argument outlined in the introduction.

CRS: Generate common reference string $\sigma = (ck, \tilde{\sigma}, \bar{\sigma}, \dot{\sigma})$ with $n = 2N$.

Statement: A circuit C with N NAND-gates, where we want to prove the wires can be assigned values such that the circuit is internally consistent.

Witness: $2N$ input values $a_1, \ldots, a_N, b_1, \ldots, b_N \in \{-1, 1\}$ for the N gates that are consistent with the wires in the circuit and respect the NAND-gates. Define $u_1, \ldots, u_N$ to be values of the output wires and let $r_1, \ldots, r_N$ be the remaining values in $(a_1, \ldots, a_N, b_1, \ldots, b_N)$ (either inputs to the circuit or duplicates of NAND-gate output wires appearing multiple times as inputs to other NAND-gates).

Argument:

1. Make restricted commitment $(c_{a\|b}, \hat{c}_{a\|b}, \tilde{c}_{a\|b})$ to $(a_1, \ldots, a_N, b_1, \ldots, b_N)$.
2. Make restricted commitment $(d_{a\|b}, \hat{d}_{a\|b}, \bar{d}_{a\|b})$ to $(a_1, \ldots, a_N, b_1, \ldots, b_N)$.
3. Make restricted commitment $(c_{b\|a}, \hat{c}_{b\|a}, \tilde{c}_{b\|a})$ to $(b_1, \ldots, b_N, a_1, \ldots, a_N)$.
4. Make restricted commitment $(c_{b\|0}, \hat{c}_{b\|0}, \tilde{c}_{b\|0})$ to $(b_1, \ldots, b_N, 0, \ldots, 0)$.
5. Make restricted commitment $(c_{u\|r}, \hat{c}_{u\|r}, \tilde{c}_{u\|r})$ to $(u_1, \ldots, u_N, r_1, \ldots, r_N)$.
6. Make restricted comm. $(c_{-u\|0}, \hat{c}_{-u\|0}, \tilde{c}_{-u\|0})$ to $(-u_1, \ldots, -u_N, 0, \ldots, 0)$.
7. Show that $c_{a\|b}$ and $d_{a\|b}$ contain the same values by giving a product argument for $c_{a\|b}$ containing the entry-wise product of the values in $\prod_{i=1}^{2N} g_i$ (a commitment to $(1, \ldots, 1, 1, \ldots, 1)$) and $d_{a\|b}$.
8. Show that $a_1, \ldots, a_N, b_1, \ldots, b_N \in \{-1, 1\}$ by giving a product argument for $\prod_{i=1}^{2N} g_i$ (a commitment to $(1, \ldots, 1, 1, \ldots, 1)$) containing the entry-wise product of the values in $c_{a\|b}$ and $d_{a\|b}$.
9. Show that the values are internally consistent with the wires. The values $a_{i_1}, \ldots, a_{i_\ell}, b_{j_1}, \ldots, b_{j_m}$ may for instance all correspond to the same wire. Pick a permutation $\rho \in S_{2N}$ such that it contains cycles of the form $i_1 \to i_2 \to \ldots \to i_\ell \to j_1 + N \to j_2 + N \to \ldots \to j_m + N \to i_1$ for all sets of values corresponding to the same wire. Give a permutation argument for $c_{a\|b}$ containing the ρ-permutation of the values in $c_{a\|b}$. For each set corresponding to the same wire, this shows $a_{i_2} = a_{i_1}, \ldots, b_{j_1} = a_{i_\ell}, \ldots, b_{j_m} = b_{j_{m-1}}$ so the values must be the same.
10. Give a permutation argument for $c_{u\|r}$ and $c_{a\|b}$ showing that the outputs values $(u_1, \ldots, u_n)$ are consistent with the input values $(a_1, \ldots, a_N, b_1, \ldots, b_N)$. (The $(r_1, \ldots, r_N)$ values are the remaining N values in $(a_1, \ldots, a_N, b_1, \ldots, b_N)$ that correspond to duplicates of an output wire or input wires to the circuit.
11. Give a permutation argument for $c_{b\|a}$ containing the swap of the values in $c_{a\|b}$.
12. Give a product argument for $c_{b\|0}$ containing the entry-wise product of the values in $c_{b\|a}$ and $\prod_{j=1}^{N} g_{j(n+1)}$ (contains $(1, \ldots, 1, 0, \ldots, 0)$).
13. Give a product argument for $c_{-u\|0}$ containing the entry-wise product of the values in $c_{u\|r}$ and $\prod_{j=1}^{N} g_{j(n+1)}^{-1}$ (contains $(-1, \ldots, -1, 0, \ldots, 0)$).

14. Show the NAND-gates are respected by giving a product argument for $c_{-u\|0}$ containing the entry-wise product of the values in $c_{b\|0}$ and $d_{a\|b}$.

The argument consists of the 6 knowledge commitments with corresponding restriction arguments, the 5 product arguments and the 3 permutation arguments given above. The total size is 42 group elements.

Verification: Accept the argument if and only if the 6 knowledge commitments are well-formed, their corresponding restriction arguments are acceptable, the 5 product arguments are acceptable and the 3 permutation arguments are acceptable.

Theorem 5. *The NIZK argument for circuit satisfiability is perfectly complete and perfectly zero-knowledge. If the q-PKE and q-CPDH assumptions hold with* $q = (4N^2 + 6N - 2)$*, then the NIZK argument is computationally sound.*

The proof can be found in the full paper.

Arithmetic circuits. It is possible to adjust our NIZK argument to handle arithmetic circuits consisting of addition and multiplications gates over $\mathbb{Z}_p$. The commitment scheme is homomorphic so if we multiply two commitments we get the sum of their values, which can be used to handle the addition gates. The multiplication gates can be handled with our product arguments.

9 Reducing the Common Reference String

In the last section, we constructed constant size NIZK arguments. The common reference string, however, grows quadratically in the size of the circuit. If the NIZK argument is only used a few times the cost of setting up the common reference string may be prohibitive. In this section, we will outline how to reduce the size of the common reference string in return for increasing the size of the argument. If the circuit has $2N = n^d$ wires for some constant $d \geq 1$ we get a common reference string with $O(n^2)$ group elements and an NIZK argument with $O(n^{d-1})$ group elements. If we choose $d = 3$, the combined size of the CRS and the NIZK argument is $O(N^{2/3})$ group elements making both components sub-linear in the circuit size.

The idea is to reduce the common reference string and let each commitment hold fewer values. If we have a circuit with n^d wires and a common reference string of size $q = n^2 + 3n - 2 = O(n^2)$, the set $\tilde{S}$ will permit the commitment of n elements at a time. Each commitment is a constant number of group elements, but now we use n^{d-1} commitments to commit to all the $2N = n^d$ input values to the gates. The product and permutation arguments are also of constant size, but they only work on commitments to n values. If we look at our NIZK argument, the product argument can be used on each of the n^{d-1} triples of commitments containing n values each so there is no problem here. The permutation argument is not useful though, because we need to permute $2N = n^d$ committed values spread across n^{d-1} commitments. The goal in this section is to build a permutation argument for two n^{d-1}-tuples of commitments to a total of $2N = n^d$ values each. The permutation argument consists of $O(n^{d-1})$ group elements and uses the existing CRS consisting of $O(n^2)$ group elements.

9.1 Permutation Argument Spanning Multiple Commitments

Consider two sets of n commitments $c_1, \ldots, c_n, d_1, \ldots, d_n$ to values $a_{11}, \ldots, a_{nn}$ and $b_{11}, \ldots, b_{nn}$. We will use a Clos-network [12] to give an argument for the two sets of committed values being permutations of each other for a publicly known permutation $\rho \in S_{n^2}$. The idea in a Clos network is to build large permutations from smaller permutations. Consider a permutation $\rho \in S_{n^2}$. First we divide the elements into n blocks of n elements and permute the elements within each block. Next, we distribute the elements in each block evenly on n other blocks giving us a new set of n blocks each containing one element from each of the previous blocks. We permute the elements in each block again. Once again, we distribute the elements in each block evenly on n new blocks. Finally, we permute the elements within the last blocks to get the elements permuted in the desired order. The permutations in the Clos network vary depending on ρ, whereas the distributions between blocks are fixed and independent of ρ.

To give a permutation argument for $\{c_i\}_{i\in[n]}, \{d\}_{i\in[n]}$ containing the same values permuted according to $\rho \in S_{n^2}$ the prover builds a Clos-network for the permutation ρ. She constructs 4 sets of n intermediate commitments $\{c'_i\}_{i\in[n]}, \{v_i\}_{i\in[n]}, \{v'_i\}_{i\in[n]}, \{d'_i\}_{i\in[n]}$ together with arguments of knowledge and restriction arguments. Each commitment contains a block of n values in the middle stages of the Clos network. She uses the permutation argument from Section 7 to show that for all $i \in [n]$ the pairs of commitments $(c_i, c'_i), (d_i, d'_i)$ and (v_i, v'_i) contain the same elements in permuted order as dictated by $\rho \in S_{n^2}$. The remaining problem is to give an argument for having dispersed the values between $\{c'_i\}_{i\in[n]}$ and $\{v_j\}_{j\in[n]}$ such that for each c'_i the values have been dispersed to n different v_j's and to give a dispersion argument for having spread the values in $\{v'_i\}_{i\in[n]}$ to $\{d'_j\}_{j\in[n]}$ such that for each v'_i the n committed values have been dispersed to n different d'_js. We present a dispersion argument in Section 9.2, which uses the existing CRS consisting of $O(n^2)$ group elements and has an argument size of $O(n)$ group elements. Counting the cost of commitments, within-block permutation arguments and the dispersion arguments, we get a total size of $O(n)$ group elements for proving that two sets of n commitments to n values each are related by a publicly known permutation $\rho \in S_{n^2}$.

Once we have a permutation argument for n^2 values spread over n commitments, we can recursively get permutation arguments for larger permutations. The cost for a permutation of n^d elements spread over two sets of n^{d-1} commitments is $O(n^{d-1})$ group elements for any constant d.

9.2 Dispersion Argument

Consider a matrix of n^2 values $a_{11}, \ldots, a_{nn}$. We can view commitments $c_1, \ldots, c_n$ given by $c_j = g^{r_j} \prod_{i\in[n]} g_i^{a_{ij}}$ as commitments to the columns of the matrix. Similarly, we can view $d_1, \ldots, d_n$ given by $d_i = g^{s_i} \prod_{i\in[n]} g_{j(n+1)}^{a_{ij}}$ as commitments to the rows of the matrix. We give an argument for demonstrating that $c_1, \ldots, c_n$ and $d_1, \ldots, d_n$ contain respectively the columns and the rows of the same $n \times n$

matrix. This means that for each c_j the n committed values have been distributed to n different commitments $d_1, \ldots, d_n$.

To get some intuition for the construction consider first the simple case where all the randomizers are 0. We then have

$$\prod_{j\in[n]} e(c_j, g_{j(n+1)}) = \prod_{i\in[n]} (g_i, d_i).$$

Taking discrete logarithms on both sides of the equation we get

$$\sum_{j\in[n]}\sum_{i\in[n]} a_{ij}x^{j(n+1)+i} = \sum_{i\in[n]}\sum_{j\in[n]} b_{ij}x^{j(n+1)+i},$$

which by Lemma 1 implies $a_{ij} = b_{ij}$ for all $i, j \in [n]$. Due to the randomizers this verification equation will not hold in general though. The prover therefore constructs an argument $(\pi_L, \pi_R, \hat{\pi}_L, \hat{\pi}_R, \bar{\pi}_L, \tilde{\pi}_R)$ consisting of six group elements such that the cross-terms arising from the randomizers cancel out.

Statement: Commitments $c_1, \ldots, c_n, d_1, \ldots, d_n \in G$.
Prover's witness: Openings $r_1, \ldots, r_n, a_{11}, \ldots, a_{nn}, s_1, \ldots, s_n, b_{11}, \ldots, b_{nn}$

$$\forall i, j \in [n]: \quad c_j = g^{r_j}\prod_{i\in[n]} g_i^{a_{ij}} \qquad d_i = g^{s_i}\prod_{j\in[n]} g_{j(n+1)}^{b_{ij}} \qquad a_{ij} = b_{ij}.$$

Argument: Pick $t \leftarrow \mathbb{Z}_p$ at random and compute the argument $(\pi_L, \pi_R, \hat{\pi}_L, \hat{\pi}_R, \bar{\pi}_L, \tilde{\pi}_R)$ as

$$\pi_L = g^t \prod_{j\in[n]} g_{j(n+1)}^{-r_j} \qquad \pi_R = g^t \prod_{i\in[n]} g_i^{-s_i}$$

$$\hat{\pi}_L = \hat{g}^t \prod_{j\in[n]} \hat{g}_{j(n+1)}^{-r_j} \qquad \hat{\pi}_R = \hat{g}^t \prod_{i\in[n]} \hat{g}_i^{-s_i}$$

$$\bar{\pi}_L = \bar{h}^t \prod_{j\in[n]} \bar{h}_{j(n+1)}^{-r_j} \qquad \tilde{\pi}_R = \tilde{h}^t \prod_{i\in[n]} \tilde{h}_i^{-s_i}$$

Verification: Output 1 if and only if

$$e(g, \hat{\pi}_R) = e(\pi_R, \hat{g}) \qquad e(g, \tilde{\pi}_R) = e(\pi_R, \tilde{h}) \qquad e(g, \hat{\pi}_L) = e(\pi_L, \hat{g})$$

$$e(g, \bar{\pi}_L) = e(\pi_L, \bar{h}) \qquad e(g, \pi_L)\prod_{j\in[n]} e(c_j, g_{j(n+1)}) = e(g, \pi_R)\prod_{i\in[n]} e(g_i, d_i).$$

Theorem 6. *The dispersion argument is perfectly complete and perfectly witness-indistinguishable. If the q-CPDH assumption holds, a non-uniform probabilistic polynomial time adversary has negligible chance of producing commitments $c_1, \ldots, c_n, d_1, \ldots, d_n$ and an accepting argument $(\pi_L, \pi_R, \hat{\pi}_L, \hat{\pi}_R, \bar{\pi}_L, \tilde{\pi}_R)$ with corresponding openings of the commitments and the argument such that $c_1, \ldots, c_n$ and $d_1, \ldots, d_n$ are commitments to two different matrices.*

We refer to the full paper for the proof.

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. Barak, B., Canetti, R., Nielsen, J.B., Pass, R.: Universally composable protocols with relaxed set-up assumptions. In: FOCS, pp. 186–195 (2004)
3. Bellare, M., Boldyreva, A., Palacio, A.: An uninstantiable random-oracle-model scheme for a hybrid encryption problem. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 171–188. Springer, Heidelberg (2004)
4. Bellare, M., Palacio, A.: Towards plaintext-aware public-key encryption without random oracles. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 48–62. Springer, Heidelberg (2004)
5. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: ACM CCS, pp. 62–73 (1993)
6. Blum, M., Feldman, P., Micali, S.: Non-interactive zero-knowledge and its applications. In: STOC, pp. 103–112 (1988)
7. Boyen, X., Waters, B.: Compact group signatures without random oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 427–444. Springer, Heidelberg (2006)
8. Boyen, X., Waters, B.: Full-domain subgroup hiding and constant-size group signatures. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 1–15. Springer, Heidelberg (2007)
9. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited. In: STOC, pp. 209–218 (1998)
10. Canetti, R., Goldreich, O., Halevi, S.: On the random-oracle methodology as applied to length-restricted signature schemes. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 40–57. Springer, Heidelberg (2004)
11. Chandran, N., Groth, J., Sahai, A.: Ring signatures of sub-linear size without random oracles. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 423–434. Springer, Heidelberg (2007)
12. Clos, C.: A study of non-blocking switching networks. Bell System Technical Journal 32(2), 406–424 (1953)
13. Damgård, I.: Non-interactive circuit based proofs and non-interactive perfect zero-knowledge with preprocessing. In: Rueppel, R.A. (ed.) EUROCRYPT 1992. LNCS, vol. 658, pp. 341–355. Springer, Heidelberg (1993)
14. De Santis, A., Di Crescenzo, G., Ostrovsky, R., Persiano, G., Sahai, A.: Robust non-interactive zero knowledge. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 566–598. Springer, Heidelberg (2001)
15. De Santis, A., Di Crescenzo, G., Persiano, G.: Randomness-optimal characterization of two NP proof systems. In: Rolim, J.D.P., Vadhan, S.P. (eds.) RANDOM 2002. LNCS, vol. 2483, pp. 179–193. Springer, Heidelberg (2002)
16. De Santis, A., Persiano, G.: Zero-knowledge proofs of knowledge without interaction. In: FOCS, pp. 427–436 (1992)
17. Dolev, D., Dwork, C., Naor, M.: Non-malleable cryptography. SIAM Journal of Computing 30(2), 391–437 (2000)
18. Dwork, C., Naor, M.: Zaps and their applications. In: FOCS, pp. 283–293 (2000)
19. Feige, U., Lapidot, D., Shamir, A.: Multiple non-interactive zero knowledge proofs under general assumptions. SIAM Journal of Computing 29(1), 1–28 (1999)
20. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: STOC, pp. 169–178 (2009)

21. Goldreich, O., Krawczyk, H.: On the composition of zero-knowledge proof systems. SIAM Journal of Computing 25(1), 169–192 (1996)
22. Goldreich, O., Oren, Y.: Definitions and properties of zero-knowledge proof systems. Journal of Cryptology 7(1), 1–32 (1994)
23. Goldwasser, S., Kalai, Y.T.: On the (in)security of the Fiat-Shamir paradigm. In: FOCS, pp. 102–113 (2003)
24. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proofs. SIAM Journal of Computing 18(1), 186–208 (1989)
25. Groth, J.: Simulation-sound NIZK proofs for a practical language and constant size group signatures. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 444–459. Springer, Heidelberg (2006)
26. 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)
27. Groth, J.: Short non-interactive zero-knowledge proofs. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 321–340. Springer, Heidelberg (2010)
28. Groth, J., Ostrovsky, R.: Cryptography in the multi-string model. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 323–341. Springer, Heidelberg (2007)
29. Groth, J., Ostrovsky, R., Sahai, A.: Non-interactive zaps and new techniques for NIZK. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 97–111. Springer, Heidelberg (2006)
30. Groth, J., Ostrovsky, R., Sahai, A.: Perfect non-interactive zero-knowledge for NP. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 339–358. Springer, Heidelberg (2006)
31. 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)
32. Kilian, J.: A note on efficient zero-knowledge proofs and arguments. In: STOC, pp. 723–732 (1992)
33. Kilian, J., Petrank, E.: An efficient noninteractive zero-knowledge proof system for NP with general assumptions. Journal of Cryptology 11(1), 1–27 (1998)
34. Maurer, U.M., 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)
35. Micali, S.: Computationally sound proofs. SIAM Journal of Computing 30(4), 1253–1298 (2000)
36. Naor, M.: On cryptographic assumptions and challenges. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 96–109. Springer, Heidelberg (2003)
37. 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)
38. Sahai, A.: Non-malleable non-interactive zero-knowledge and adaptive chosen-ciphertext security. In: FOCS, pp. 543–553 (2001)

Short Non-interactive Zero-Knowledge Proofs

Jens Groth*

University College London
j.groth@ucl.ac.uk

Abstract. We show that probabilistically checkable proofs can be used to shorten non-interactive zero-knowledge proofs. We obtain publicly verifiable non-interactive zero-knowledge proofs for circuit satisfiability with adaptive and unconditional soundness where the size grows quasi-linearly in the number of gates. The zero-knowledge property relies on the existence of trapdoor permutations, or it can be based on a specific number theoretic assumption related to factoring to get better efficiency. As an example of the latter, we suggest a non-interactive zero-knowledge proof for circuit satisfiability based on the Naccache-Stern cryptosystem consisting of a quasi-linear number of bits. This yields the shortest known non-interactive zero-knowledge proof for circuit satisfiability.

Keywords: Non-interactive zero-knowledge proofs, adaptive soundness, probabilistically checkable proofs, Naccache-Stern encryption.

1 Introduction

Zero-knowledge proofs introduced by Goldwasser, Micali and Rackoff [GMR89] are interactive protocols that enable a prover to convince a verifier about the truth of a statement without leaking any information but the fact that the statement is true. Blum, Feldman and Micali [BFM88] followed up by introducing non-interactive zero-knowledge (NIZK) proofs where the prover outputs just one message called a proof, which convinces the verifier of the truth of the statement. The central properties of zero-knowledge proofs and non-interactive zero-knowledge proofs are completeness, soundness and zero-knowledge.

Completeness: If the statement is true, the prover should be able to convince the verifier.

Soundness: A malicious prover should not be able to convince the verifier if the statement is false.

Zero-knowledge: A malicious verifier learns nothing except that the statement is true.

In this paper, we focus on the NP-complete language of circuit satisfiability, which is the most widely studied language in the context of non-interactive zero-knowledge proofs. The statement is a binary circuit C and a claim that there

* Supported by Engineering and Physical Sciences Research Council grant number EP/G013829/1.

M. Abe (Ed.): ASIACRYPT 2010, LNCS 6477, pp. 341–358, 2010.

exists an input, a witness w, such that the circuit outputs 1 when evaluated on w. The prover has the witness w as a private input, while the statement C is public and known both to the prover and the verifier.

Only languages in BPP have non-interactive zero-knowledge proofs in the plain model without any setup [Ore87, GO94, GK96]. Blum, Feldman and Micali [BFM88] therefore suggested the common reference string model, where the prover and the verifier have access to a trusted bit-string. The common reference string can for instance be generated by a trusted third party or by a set of parties executing a multi-party computation protocol. Groth and Ostrovsky [GO07] has as an alternative suggested NIZK proofs in the multi-string model, where many parties generate a random string and the security of the NIZK proof relies on a majority of the strings being honestly generated. In this paper, we work in the common random string model, where the common reference string is a trusted uniformly random bit-string.

NIZK proofs have many applications, ranging from early chosen-ciphertext secure public-key cryptosystems [DDN00, Sah01] to recent advanced signature schemes [BW06, CGS07]. They have therefore been studied carefully in the literature. Blum, Feldman and Micali [BFM88] proposed an NIZK proof for all of NP based on a number theoretic assumption related to factoring. Feige, Lapidot and Shamir [FLS99] gave an NIZK proof for all of NP based on the existence of trapdoor permutations. While these results established the existence of NIZK proofs based on general assumptions, other works [Dam92, DDP02, KP98] have aimed at reducing the complexity of NIZK proofs. Gentry's fully homomorphic cryptosystem based on lattices can reduce the complexity of an NIZK to grow linearly in the witness size as opposed to the circuit size [Gen09]. Groth, Ostrovsky and Sahai [GOS06b, GOS06a, Gro06, GS08] have constructed highly efficient NIZK proofs using techniques from pairing-based cryptography.

1.1 Our Contribution

We construct NIZK proofs for circuit satisfiability with a size that grows quasi-linearly in the size of the circuit. Our NIZK proofs have perfect completeness, statistical soundness, and computational zero-knowledge. The zero-knowledge property of the NIZK proofs can be based on trapdoor permutations or for higher efficiency on the semantic security of the Naccache-Stern cryptosystem based on higher residues [NS98].

The Naccache-Stern cryptosystem is based on a decisional assumption in RSA-type groups, which predates but is otherwise incomparable to the decisional assumptions used in pairing-based NIZK proofs. Surprisingly, the construction based on the Naccache-Stern cryptosystem has better asymptotic efficiency than the recent pairing-based NIZK proofs for circuit satisfiability [GOS06b, GOS06a, GS08] (although pairing-based NIZK proofs remain more efficient for practical purposes due to the large constants involved in our construction). With pairing group elements of size k_G and a circuit size that is polynomial in the security parameter k we get an asymptotic improvement over pairing-based NIZK proofs

Table 1. Comparison of NIZK proofs for security parameter k, circuit size $|C| = k^{O(1)}$, witness size $|w|$, trapdoor permutations over $\{0,1\}^{k_T}$, and pairing group size k_G (usually $k_G \approx k^3$ for 2^{-k} security [GPS08])

	CRS size	Proof size	Assumption
Kilian-Petrank [KP98]	$\omega(\|C\|k_T k \log k)$	$\omega(\|C\|k_T k \log k)$	trapdoor perm.
This work	$\|C\|k_T \log^c(k) + \text{poly(k)}$	$\|C\|k_T \log^c(k) + \text{poly}(k)$	trapdoor perm.
Gentry [Gen09]	$\text{poly}(k)$	$\|w\|\text{poly}(k)$	lattice-based
GOS [GOS06b]	$\Theta(k_G)$	$\Theta(\|C\|k_G)$	pairing-based
This work	$\|C\| \log^c(k) + \text{poly}(k)$	$\|C\| \log^c(k) + \text{poly}(k)$	Naccache-Stern

of a multiplicative factor $\frac{k_G}{\text{polylog}(k)}$. This brings the NIZK proof size within a polylog(k) factor of the size of the circuit itself.

In Table 1, we compare our NIZK proofs with the current state of the art NIZK proofs for circuit satisfiability based on respectively trapdoor permutations [KP98] and specific number theoretic assumptions [GOS06b, GOS06a]. All of these NIZK proofs have an efficient (probabilistic polynomial time) prover and they are all publicly verifiable.

Soundness and zero-knowledge can be adaptive or non-adaptive. In non-adaptive soundness and zero-knowledge, the statement to be proven is chosen independently of the common reference string. Usually, NIZK proofs are used in a context where the common reference string is publicly available though, and it is therefore unreasonable to assume the statement is independent of the common reference string.[1] Adaptive soundness and adaptive zero-knowledge refers to the more realistic setting, where NIZK proofs need to be sound and zero-knowledge even when the common reference string is publicly available and the statement may depend on the common reference string. Our NIZK proofs are both adaptively sound and adaptively zero-knowledge, and in Table 1 we have compared the schemes in the efficient prover, adaptive soundness setting.

1.2 New Techniques

PCPs in NIZK. Probabilistically checkable proofs (PCPs) [AS98, ALM+98, Din07] are proofs for a statement that can be verified by reading a few bits in the proof instead of reading the whole proof. A PCP for a circuit being satisfiable will be larger than the circuit itself; however, one only needs to read a few bits of the proof to get more than 50% chance of detecting an attempt to prove a false statement. By reading more bits, we can get exponentially small risk of wrongly being convinced by the PCP.

PCPs have been very useful in the context of zero-knowledge arguments. Kilian [Kil92] for instance suggested a sub-linear size zero-knowledge argument, where the prover commits to the bits of a PCP and the verifier asks the prover to reveal the content of a few of these commitments.

[1] We have a hard time thinking of any applications where non-adaptive soundness suffices, while non-adaptive zero-knowledge sometimes may be useful.

We use PCPs in a different way. In our NIZK proofs the prover will prove that all queries to the PCP have satisfactory answers. At first sight this may seem counterintuitive; the PCP will be larger than the statement itself and it is odd that increasing the statement size would help us in shortening the size of the NIZK proofs. Using a PCP for the statement, however, has the advantage that the verifier can grant the malicious prover a non-trivial chance of falsely answering some of the queries, as long as there are other queries where he will detect the attempt to cheat. This stands in contrast to traditional NIZK proofs, where the verifier needs high certainty for every single part of the statement being correct.

To illustrate our technique, consider an NIZK proof such as Kilian-Petrank [KP98]. They first reduce circuit satisfiability to 3SAT5; where each clause has three variables and each variable appears in at most 5 clauses. By choosing a trapdoor permutation and revealing hard-core bits related to the common reference string, the prover can demonstrate that each clause is satisfied. There is a risk of error though, and the prover therefore needs to repeat the proof many times for each clause to increase the soundness guarantee.

Our idea is to use a PCP in a pre-processing step before applying the techniques of Kilian and Petrank. The effect of the PCP (see Section 3) is to increase the gap between satisfiable and unsatisfiable statements. In a standard 3SAT5 statement there are unsatisfiable statements where all but one clause can be satisfied. With the PCP, however, we can ensure that either all clauses can be satisfied or alternatively a constant fraction of the clauses are unsatisfied. The advantage over Kilian and Petrank's NIZK proof is that now we have resilience towards errors in individual clauses. Even if a malicious prover succeeds in falsely creating NIZK proofs for some of the clauses being satisfied, we still get soundness as long as this only happens for a small constant fraction of clauses. We can therefore avoid the repetition of proofs that Kilian and Petrank needed.

Implementing a Hidden Random String. We construct our NIZK proofs in two steps. We use cryptographic techniques to convert the common reference string into a hidden string of random bits, where the prover may selectively disclose some of the bits and keep other bits secret. We then construct NIZK proofs that assume the existence of a string of hidden bits, where the prover may keep some of them secret and reveal others to the verifier.

Feige, Lapidot and Shamir [FLS99] suggested the following way of implementing the hidden bits model. When working with trapdoor permutations, we can interpret the common reference string as a string of images of the trapdoor permutation. The hidden random bits are hardcore bits of the pre-images. The prover may with the knowledge of the trapdoor learn all the hidden random bits. By revealing a pre-image to the trapdoor permutation, she can disclose the value of a particular hidden random bit. This is a costly approach, however, since we only get one hidden random bit per trapdoor permutation image. In general, the common reference string has to be a factor k_T larger than the hidden random string, where k_T is the size a trapdoor permutation value.

The second contribution of this paper is using the Naccache-Stern cryptosystem [NS98] to reduce the cost of implementing the hidden bits model. We interpret the common reference string as a series of ciphertexts, but with the Naccache-Stern cryptosystem each ciphertext may hold many hardcore bits. The message space is of the form $\mathbb{Z}_P$, where $P = \prod_{i=1}^{n} p_i$ is a product of small primes of size $\log k$. We will show that with the Naccache-Stern cryptosystem, it is possible to disclose the plaintext modulo p_i without revealing the rest of the plaintext. This means that we can have $\Omega(\frac{k_G}{\log k})$ hidden random bits in each ciphertext, giving a common reference string that is only a factor $O(\log k)$ larger than the hidden random string.

Combining PCPs and the Naccache-Stern cryptosystem, we get the asymptotically shortest known NIZK proofs for circuit satisfiability consisting of a quasi-linear number of bits.

1.3 Overview

We construct NIZK proofs for circuit satisfiability in three steps. In Section 3 we describe how a PCP can be used to convert the circuit into a Gap-3SAT5 formula, where either all clauses are satisfiable or alternatively there are at least a constant fraction of unsatisfiable formulae. In Section 4 we construct an NIZK proof in the hidden bits model, where it is assumed that the prover has access to a string of uniformly random bits and may reveal an arbitrary subset of these bits and their positions to the verifier. Finally, in Sections 5 and 6 we show how to implement the hidden bits model under the general assumption of the existence of trapdoor permutations and more efficiently under a concrete number theoretic assumption related to factoring. The two main contributions of the paper are the conceptual idea of using PCPs in a preprocessing step as described in Section 3 and the introduction of a new technique for efficiently implementing the hidden random bits model using the Naccache-Stern cryptosystem described in Section 6.

2 Preliminaries

NOTATION. Given two functions $f, g : \mathbb{N} \to [0, 1]$ we write $f(k) \approx g(k)$ when $|f(k) - g(k)| = O(k^{-c})$ for every constant $c > 0$. We say that f is *negligible* if $f(k) \approx 0$ and that f is *overwhelming* if $f(k) \approx 1$.

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 the set S. We will for convenience assume uniform random sampling from various types of sets is possible; there are easy ways to amend our protocols to the case where the sets are only sampleable with a distribution that is statistically close to uniform.

NIZK PROOFS. Let R be a polynomial time computable binary relation. For pairs $(C, w) \in R$ we call C the statement and w the witness. Let L be the NP-language consisting of statements with witnesses in R. In this paper, we will

focus on the case where the statements are circuits and L is the language of satisfiable circuits, i.e., where there exists an input w so $C(w) = 1$. The size of the NIZK proofs will depend on the size of the statement. We will write L_n for the language of satisfiable circuit consisting of n binary gates and write R_n for the corresponding relation.

An efficient-prover non-interactive proof for the relation R consists of three probabilistic polynomial time algorithms (K, P, V). K is the common reference string generator that takes the security parameter written in unary 1^k and an intended statement size n as input and outputs a common reference string σ of length $\Omega(k)$. P is the prover algorithm that takes as input the common reference string σ, a statement C and a witness w so $(x, w) \in R$ and outputs a proof π. V is the verifier algorithm that on a common reference string σ, a statement C and a proof π outputs 0 or 1. We interpret a verifier output of 0 as a rejection of the proof and a verifier output of 1 as an acceptance of the proof. We call (K, P, V) a non-interactive proof system for R it is complete and sound as described below.

Perfect completeness. Completeness means that a prover with a witness can convince the verifier. For all adversaries $\mathcal{A}$ and $n = k^{O(1)}$ we have

$$\Pr\left[\sigma \leftarrow K(1^k, n); (C, w) \leftarrow \mathcal{A}(\sigma); \pi \leftarrow P(\sigma, C, w) : V(\sigma, C, \pi) = 1 \text{ if } (C, w) \in R_n\right] = 1.$$

Statistical soundness. Soundness means that it is impossible to convince the verifier of a false statement. For all non-uniform polynomial time adversaries $\mathcal{A}$ and $n = k^{O(1)}$ we have

$$\Pr\left[\sigma \leftarrow K(1^k, n); (C, \pi) \leftarrow \mathcal{A}(\sigma) : C \notin L_n \text{ and } V(\sigma, C, \pi) = 1\right] \approx 0.$$

Computational zero-knowledge. A non-interactive argument (K, P, V) is computationally zero-knowledge if it is possible to simulate the proof of a true statement without knowing the witness. Formally, we require the existence of a probabilistic polynomial time simulator $S = (S_1, S_2)$. S_1 outputs a simulated common reference string σ and a simulation trapdoor τ. S_2 takes the simulation trapdoor and a statement as input and produces a simulated proof π. We require for all non-uniform polynomial time stateful interactive adversaries $\mathcal{A}$ and $n = k^{O(1)}$ that

$$\begin{aligned}&\Pr\left[\sigma \leftarrow K(1^k, n); (C, w) \leftarrow \mathcal{A}(\sigma); \pi \leftarrow P(\sigma, C, w) : (C, w) \in R_n \text{ and } \mathcal{A}(\pi) = 1\right]\\ \approx &\Pr\left[(\sigma, \tau) \leftarrow S_1(1^k, n); (C, w) \leftarrow \mathcal{A}(\sigma); \pi \leftarrow S_2(\tau, C) : (C, w) \in R_n \text{ and } \mathcal{A}(\pi) = 1\right].\end{aligned}$$

3 Preprocessing with Probabilistically Checkable Proofs

We start by giving a polynomial time reduction f from circuit satisfiability to Gap-3SAT5. The reduction f takes as its input a circuit with n binary gates and outputs a 3SAT formula with $N = n$ polylog n variables and $\frac{5}{3}N$ clauses. The 3SAT formula, will be such that each variable appears exactly 3 times as a positive literal and 2 times as a negative literal. If the input of f is a satisfiable circuit C, it will output a satisfiable 3SAT5 formula $\phi = f(C)$. If the circuit

C is unsatisfiable, the reduction f will output a formula $\phi = f(C)$ such that all assignments have at least αN unsatisfied clauses for some constant $\alpha > 0$. We also need a polynomial time witness-reduction f_w, which on input C, w such that $C(w) = 1$ outputs a satisfying assignment $f_w(C, w)$ for the 3SAT5 formula $\phi = f(C)$.

The first step in our reduction is to map the circuit C to a constraint graph $G(C)$ with the following properties. The vertices of the constraint graph G may be assigned values from a constant size alphabet Σ, but each edge between two vertices describes a constraint on the values that they may be assigned. When starting with a satisfiable circuit, the output is a satisfiable constraint graph. However, on an unsatisfiable circuit the output is an unsatisfiable constraint graph where any assignment violates at least a α_0-fraction of the constraints for some constant $\alpha_0 > 0$. The polynomial time assignment tester [DR04] given by Dinur [Din07] in her proof of the PCP theorem has the properties described above. Moreover, given a witness for the satisfiability of the circuit C, we may in polynomial time compute a satisfying assignment for the constraint graph $G(C)$. Dinur's most efficient assignment tester building on work by Ben-Sasson and Sudan [BSS08] outputs a constraint graph $G(C)$ with n polylog n vertices and edges.

Given a constraint graph G over a constant size alphabet Σ, we assign a constant number of binary variables to each vertex such that it is possible to represent any element from the alphabet Σ. Each constraint between two vertices is of constant size Σ^2 and we can therefore write out a constant size 3SAT formula describing the constraint. Taking the conjunction of all these 3SAT formulas, we reduce the constraint graph to a 3SAT formula with n polylog n variables and clauses. A satisfying assignment for the constraint graph gives us a satisfying assignment to the 3SAT formula. Since each vertex has a constant number of variables associated with it, and each constraint has a constant number of clauses associated with it, a constraint graph with a constant fraction α_0 of unsatisfiable constraints reduces to a 3SAT formula with a α_1 fraction of unsatisfiable clauses for some constant $\alpha_1 > 0$.

Finally, we reduce the 3SAT formula to a 3SAT5 formula where each variable appears in the clauses as exactly 5 literals and each clause has exactly 3 literals. First we copy clauses and variables so each clause has exactly 3 literals and each variable appears at least 3 times. Then the ℓ appearances of a variable as a positive literal x_i or a negative literal $\neg x_i$ are replaced with copies $x_{i1}, \ldots, x_{i\ell}$. For each copy we add 4 clauses for consistency in the truth value assignment with the predecessor $x_{i,j-1 \bmod \ell}$ and the successor $x_{i,j+1 \bmod \ell}$ according to whether the original variable appeared as a positive or negative literal. In these consistency clauses the copy appears twice as a negative literal and twice as a positive literal, so all copies appear as exactly 3 positive literals and 2 negative literals in the resulting 3SAT5 formula. This is a linear size reduction, so we end up with n polylog n variables and clauses. A satisfying assignment for the 3SAT formula gives us a satisfying assignment for the resulting 3SAT5 formula. A 3SAT formula with a constant fraction α_1 of unsatisfiable clauses, gives a 3SAT5 formula with a α fraction of unsatisfiable clauses for some constant $\alpha > 0$.

In summary, there is a pair of polynomial time algorithms (f, f_w) and a constant $\alpha > 0$ so:

Reduction: f takes input a circuit C with n gates and outputs a 3SAT5 formula $f(C)$ with $N = n \text{ poly log } n$ variables. Each variable appears as 3 positive literals and 2 negative literals, and each clause has exactly 3 literals. If C is satisfiable, then $f(C)$ is satisfiable. If C is unsatisfiable, then all assignments to the variables of $f(C)$ leave at least αN clauses unsatisfied.

Witness-preservation: f_w takes as input a circuit C with n gates and a witness for C being satisfiable and outputs a truth value assignment satisfying the 3SAT5 formula $f(C)$.

4 NIZK Proofs in the Hidden Bits Model

We will now give an NIZK proof in the hidden-bits model for Gap-3SAT5-satisfiability. The ideas in this section are quite similar to Kilian and Petrank [KP98], although our setting allows us to simplify their scheme.

Let L_N be the language of satisfiable 3SAT5 formulae with N variables and $\frac{5}{3}N$ clauses, where each variable appears as 3 positive literals and 2 negative literals. Let R_N be the corresponding relation of formulae and satisfying assignments. Further, define L_N^α as the language of formulae in L_N that have a truth-value assignment to the variables that leaves at most αN clauses unsatisfied. We will be interested in a hidden-bits NIZK proof $(\ell_H(N), P_H, V_H)$ for R with perfect completeness, $(\alpha, \epsilon_H(N))$-soundness, and perfect zero-knowledge as described below.

Perfect completeness. For all $N \in 3\mathbb{N}$ and all $(\phi, w) \in R_N$ we have

$$\Pr\left[\rho \leftarrow \{0,1\}^{\ell_H(N)}; (i_1, \ldots, i_t) \leftarrow P_H(\rho, \phi, w) : V_H(\phi, i_1, \rho_{i_1}, \ldots, i_t, \rho_{i_t}) = 1\right] = 1.$$

Statistical soundness. For all $N \in 3\mathbb{N}$ and all adversaries $\mathcal{A}$

$$\Pr\Big[\rho \leftarrow \{0,1\}^{\ell_H(N)}; (\phi, i_1, \ldots, i_t) \leftarrow \mathcal{A}(\rho) : \phi \notin L_N^\alpha \text{ and } V_H(\phi, i_1, \rho_{i_1}, \ldots, i_t, \rho_{i_t}) = 1\Big] \leq \epsilon_H(N).$$

Perfect zero-knowledge. There exists a probabilistic polynomial time simulator S_H such that for all $N \in 3\mathbb{N}$ and all $(\phi, w) \in R_N$ the distribution

$$\{\rho \leftarrow \{0,1\}^{\ell_H(N)}; (i_1, \ldots, i_t) \leftarrow P_H(\rho, \phi, w) : (i_1, \rho_{i_1}, \ldots, i_t, \rho_{i_t})\}$$

is identical to the distribution

$$\{(i_1, \rho_{i_1}, \ldots, i_t, \rho_{i_t}) \leftarrow S_H(\phi) : (i_1, \rho_{i_1}, \ldots, i_t, \rho_{i_t})\}.$$

4.1 Hidden-Bits NIZK Proof for Gap-3SAT5

Let ϕ be a 3SAT5 formula with N variables and $\frac{5}{3}N$ clauses, where each variable appears exactly thrice as a positive literal x_i and twice as a negative literal $\neg x_i$ in the clauses. The verifier has the promise that either there is an assignment to the variables so all clauses are satisfied, or all assignments of truth values to the variables lead to more than αN unsatisfied clauses. The prover has a satisfying assignment and wants to give an NIZK proof in the hidden bits model for ϕ being satisfiable.

We first sketch the NIZK proof and then afterwards explain the main ideas in the construction. There is some freedom in the choice of parameters; for concreteness we suggest $a = \lceil \frac{8}{\alpha} \rceil, b = \lceil \log N \rceil, \Delta = \lceil \frac{N}{\log N} \rceil$.

Statement: A 3SAT5 formula $\phi \in L_N$.

Prover's input: A string ρ of $6a2^{6a}(bN + \Delta)$ hidden bits. A truth-value assignment to the variables $x_1, \ldots, x_N$ so $\phi(x_1, \ldots, x_N) = 1$.

Proof:

1. Interpret the hidden bits as $6a2^{6a-1}(bN + \Delta)$ consecutive pairs of bits. Each pair of bits is interpreted as one of three possible characters according to the following scheme

$$00 = 0 \qquad 01 = W \qquad 10 = W \qquad 11 = 1.$$

   Later the prover may reveal one of the bits in a character. In a wildcard character W the prover can reveal either 0 or 1, whereas 0 can only be revealed as 0 and 1 can only be revealed as 1.
2. Interpret the characters as $2^{6a-1}(bN + \Delta)$ consecutive $6a$-character blocks. Call a block good if it has exactly $3a$ W-characters and they are either all in the first half of the block or all in the second half of the block. Otherwise, call the block bad.
3. A block has 2^{1-6a} chance of being good, so we expect on average $(bN + \Delta)$ good blocks. If the number of good blocks is outside the interval $[bN; bN + 2\Delta]$ reveal all hidden bits and halt.
4. Reveal to the verifier all the hidden bits associated with bad blocks.
5. Assign the first b good blocks to the first variable, etc., so each variable has b blocks assigned to it. The remaining good blocks will not be used.
6. Interpret each good block as a set of 6 consecutive a-character strings (see examples in Figure 1). Assign in the order of appearance, 5 of these a-character strings to the 5 appearances of their variable x_i in the clauses as follows. If the witness has $x_i = 1$, assign the 3 wildcard strings to the 3 appearances of x_i, and the first 2 0/1-strings to the 2 appearances of $\neg x_i$. If the witness has $x_i = 0$, assign the first 2 wildcard strings to the 2 appearances of $\neg x_i$ and the 3 0/1-strings to the 3 appearances of x_i. Taking all good blocks into account, each appearance of x_i or $\neg x_i$ has b a-character strings assigned to it.
7. Each clause has 3 literals, and each literal has a corresponding tuple of b a-character strings. Pick at random a literal for which the b a-character strings only contain wildcard characters. Such a literal must

exist since the clause is satisfied by the truth-value assignment. For the other two literals reveal a random bit in each character's bit pair, thereby revealing two ab-bit strings. In the remaining wildcard literal, reveal in each wildcard character one of the bits, such that the revealed ab-bit string is the exclusive-or of the two other ab-bit strings.

8. The proof consists of the revealed bits. If the number of good blocks is outside the interval $[bN; bN + 2\Delta]$ the proof reveals all hidden bits. Else, the proof reveals the hidden bits of the bad blocks and a $\frac{5}{12}$-fraction of the hidden bits in the first bN good blocks.

Verification:

1. If the proof reveals all hidden bits, return 1 if the number of good blocks is outside the range $[bN; bN + 2\Delta]$ and else return 0.
2. Verify that there are no good blocks among the blocks where all bits have been revealed.
3. Verify that there are at most $bN + 2\Delta$ blocks where some of the bits remain hidden. Associate the first bN blocks with the variables in the order of appearance.
4. Verify that in each of the bN blocks corresponding to variables, exactly 5 of the 6 a-character strings have one revealed bit in each character. Verify also that in each block either the last a-character string in the first half of the block, or the last a-character string in the second half of the block has no revealed bits. Based on this, each revealed a-bit string can be uniquely associated with a corresponding literal in a clause.
5. For each clause, verify that the exclusive-or of the two first ab-bit strings corresponding to the first 2 literals equals the ab-bit string corresponding to the third literal.
6. Return 1 if all verifications passed, else return 0.

In the first step, note that there is 50% chance that a character is a wildcard and 50% chance that it is a 0 or a 1. Later, the prover will open some of the characters by revealing one of the bits. Wildcards can be opened as 0 or 1, whereas 0 can only be opened as 0 and 1 can only be opened as 1. The prover sets up the strings so wildcards correspond to true literals and non-wildcards correspond to false literals. In satisfied clauses there is a true literal, which can be opened at will. This is what gives the prover with a satisfying assignment the power to convince the verifier. On the other hand, in an unsatisfied clause there will only be non-wildcard characters associated with the false literals, which will reduce the power of the prover and make it hard to convince the verifier of a false statement. Finally, for zero-knowledge we can set more of the characters to be wildcard characters. This will make it possible to simulate a proof without knowing a satisfying truth assignment for the statement.

We interpret the string of characters as blocks of $6a$ characters. There will be an expected number of $bN + \Delta$ good blocks. We can use Chernoff-bounds to see that there is high probability that most of the hidden blocks that have not been revealed are indeed good blocks. The point of sampling good blocks is that they represent a consistent view of a variable. All true literals are assigned wildcard strings, all false literals are assigned non-wildcard string.

$010\ldots101$	$100\ldots011$	$010\ldots110$	$WW\ldots W$	$WW\ldots W$	$WW\ldots W$

$WW\ldots W$	$WW\ldots W$	$WW\ldots W$	$010\ldots110$	$000\ldots111$	$100\ldots101$

Fig. 1. Two examples of good blocks

The important thing to note is that with wildcard string, the prover may open the true literals to any a-bit string. For the false literals, however, the prover is bound to a particular a-bit string. We require that in each clause, the prover should open 3 a-bit strings, such that they exclusive-or to 0. In clauses with a true literal this is easy to accomplish, since the prover may open the corresponding wildcard string to any a-bit string. This gives us completeness. In unsatisfied clauses, however, the prover has 3 fixed a-bit strings and the probability of their exclusive-or being 0 is 2^{-a}. For each unsatisfied clause, we therefore get a good chance of catching a cheating prover.

We have now described the main idea in the construction. The prover has some degrees of freedom in choosing the statement, taking advantage of a few bad blocks that may be camouflaged as good blocks, etc. However, by repeating the proof b times in parallel and using the fact that for unsatisfiable statement there is actually a constant fraction of unsatisfied clauses no matter what the truth assignment is, we can ensure that a cheating prover still has very small chance of convincing the verifier on a false statement.

Theorem 1. *For sufficiently large N the protocol given above is a hidden-bits NIZK proof for 3SAT5 with perfect completeness, $(\alpha, 2^{-\frac{N}{6\log^3 N}})$-soundness and perfect zero-knowledge with a hidden string of size $\ell_H(N) = O(N \log N)$.*

We refer to the full paper for a proof.

5 Implementing the Hidden Bits Proof with Trapdoor Permutations

Trapdoor permutations. We will now implement the hidden bits NIZK proof using trapdoor permutations. A trapdoor permutation is a triple of algorithms $(K_{\mathrm{T}}, F, F^{-1})$. K_{T} generates a public key pk, which we for convenience will assume has k bits, and a secret key sk for the trapdoor permutation. F_{pk} and F_{sk}^{-1} are efficiently computable permutations of k-bit strings, such that $F_{pk}(F_{sk}^{-1}(y)) = y$. We will assume it is hard to compute F_{sk}^{-1} without knowledge of sk. All trapdoor permutations can easily be converted into trapdoor permutations with a hardcore predicate [GL89] so we will assume the existence of a hardcore predicate B for the trapdoor permutation. If $y \leftarrow \{0,1\}^k$ is chosen uniformly at random then $B(F_{sk}^{-1}(y))$ is uniformly random in $\{0,1\}$ and given only (pk, y) it is computationally hard to decide $B(F_{sk}^{-1}(y))$.

Implementing the hidden bit string. To implement a hidden bit string with N' random bits, we generate a common reference string σ consisting of $k(4N' + 4\Delta')$ uniformly random bits. There is a range of choices of Δ', for the sake of concreteness let us say $\Delta' = \lceil (N')^{\frac{3}{4}} \rceil$. The prover picks a trapdoor permutation and interprets σ as $4N' + 4\Delta'$ images of the trapdoor permutation. This gives the prover $4N' + 4\Delta'$ secret hardcore bits. The prover can selectively open some of the hardcore bits by computing the corresponding preimages and giving them to the verifier. This idea first described in [FLS99] indicates how we can generate hidden random bits that the prover can see and selectively disclose to the verifier.

Our hidden bits proof has perfect zero-knowledge if the simulator can choose the hidden bits itself. Once a trapdoor permutation has been chosen we cannot alter the preimages though so we have not yet implemented the hidden bits model in the *adaptive* zero-knowledge sense. The problem is that the common reference string is chosen before the adversary picks the statement and therefore the simulator needs to get hidden bits out of the simulated common reference string that can be revealed as both 0 and 1 depending on what is needed in the simulation. Our solution is to interpret pairs of hardcore bits as hidden bits as follows:

$$00 = \mathbf{0} \qquad 01 = S \qquad 10 = S \qquad 11 = \mathbf{1}.$$

The prover reveals a hidden bit by revealing one of the two preimages associated with it. This means it is bound to open $\mathbf{0}$ as 0 and open $\mathbf{1}$ as 1, but it can open a soft bit S as either 0 or 1. In the zero-knowledge simulation, we will set up the common reference string such that all hidden bits are soft. When all hidden bits are soft, the zero-knowledge simulator can open them as it likes and simulate the hidden bits proof.

When half the hidden bits are soft we have to be careful to preserve soundness though. We therefore require that the prover reveals the preimages corresponding to soft hidden bits. On average the prover should reveal $N' + \Delta'$ soft hidden bits; and the verifier checks that at last N' soft hidden bits are revealed. This leaves the prover with approximately N' hidden bits, which mostly will be hard hidden bits which can only be opened as 0 or only be opened as 1. Soundness will now follow from the fact that most of the remaining hidden bits are uniformly random hard bits.

NIZK proof. We will now give the full NIZK proof for circuit satisfiability. The statement will be a circuit C and the prover will have a satisfying witness w so $C(w) = 1$. We have to be careful that the prover chooses a well-formed public key for the trapdoor permutation and will therefore use an NIZK proof $(\ell_{\text{well}}, P_{\text{well}}, V_{\text{well}})$ for well-formedness. This NIZK proof could for instance be Kilian and Petrank's original NIZK proof [KP98], which would have a cost of $\text{poly}(k)$ bits. Alternatively, we could assume the existence of certifiable trapdoor permutations where the well-formedness of the public key is directly verifiable. Or we could use Bellare and Yung's [BY92] method of sampling preimages to show that the public key describes a function close to a trapdoor permutation and then give a more careful probability analysis that deals with the small statistical

bias this might introduce in the hidden bits. We will in the following let $N' = O(N \log N) = n$ polylog(n) be the number of bits needed in the hidden bits model for circuits with n gates.

CRS: $\sigma = (\sigma_{1,1}, \ldots, \sigma_{2N'+2\Delta',2}, \sigma_{\text{well}}) \leftarrow \{0,1\}^{k(4N'+4\Delta')+\ell_{\text{well}}(k)}$.
Proof:

1. Generate keys for the trapdoor permutation $(pk, sk) \leftarrow K_{\text{T}}(1^k)$.
2. Compute an NIZK proof π_{well} for pk being a valid public trapdoor permutation key.
3. Compute the hardcore bits $h_{1,1}, h_{1,2}, \ldots, h_{2N'+2\Delta',1}, h_{2N'+2\Delta',2}$ as $h_{i,j} = B(F_{sk}^{-1}(\sigma_{i,j}))$.
4. If there are less than N' pairs or more than $N' + 2\Delta'$ pairs where $h_{i,1} = h_{i,2}$ return the proof $(pk, \pi_{\text{well}}, F_{sk}^{-1}(\sigma_{1,1}), \ldots, F_{sk}^{-1}(\sigma_{2N'+2\Delta',2})$ and halt.
5. For each pair $h_{i,1} \neq h_{i,2}$ include preimages $\pi_{i,1} = F_{sk}^{-1}(\sigma_{i,1})$ and $\pi_{i,2} = F_{sk}^{-1}(\sigma_{i,2})$ in the proof.
6. Let $\rho = (\rho_1, \ldots, \rho_{N'})$ be the values of the first N' remaining pairs of hardcore bits.
7. Run the hidden bit string proof on ρ to get $\pi_H \leftarrow P_H(\rho, f(C), f_w(w))$.
8. For all revealed bits ρ_i in the hidden bits proof π_H corresponding to hardcore bits $h_{j,1} = h_{j,2}$ choose at random to include either $\pi_{j,1} = F_{sk}^{-1}(\sigma_{j,1})$ or $\pi_{j,2} = F_{sk}^{-1}(\sigma_{j,2})$ in the proof.

The proof is of the form $(pk, \pi_{\text{well}}, \pi_{i_1,j_1}, \ldots, \pi_{i_t,j_t})$.
Verification:

1. Verify the NIZK proof π_{well} for pk being a correctly generated public trapdoor permutation key.
2. Verify the correctness of all the preimages $\sigma_{i,j} = F_{pk}(\pi_{i,j})$.
3. Compute the corresponding hardcore bits $h_{i,j} = B(\pi_{i,j})$.
4. If all hardcore bits have been revealed, verify that there are less than N' or more than $N' + 2\Delta'$ pairs $h_{i,1} = h_{i,2}$ and accept if all verifications have succeeded.
5. Verify that all revealed pairs of hardcore bits have $h_{i,1} \neq h_{i,2}$ and that there are between N' and $N' + 2\Delta'$ pairs left in which at most one hardcore bit has been revealed.
6. Interpret the remaining hardcore bits as indices and revealed bits as a hidden bits proof $(i_1, \rho_{i_1}, \ldots, i_t, \rho_{i_t})$ and accept if all verifications have succeeded and $V_H(f(C), i_1, \rho_{i_1}, \ldots, i_t, \rho_{i_t}) = 1$.

The construction leads to the following theorem that we prove in the full paper.

Theorem 2. *Assuming the existence of trapdoor permutations on $\{0,1\}^k$ with k-bit keys there is an NIZK proof for circuit satisfiability with perfect completeness, statistical soundness and computational zero-knowledge. The size of the common reference string and the NIZK proof is $|C|$ polylog $|C| \cdot k + \text{poly}(k)$ bits.*

6 Implementing the Hidden Bits Proof with Naccache-Stern Encryption

Naccache-Stern encryption. The Naccache-Stern cryptosystem based on higher residues [NS98] has message space $\mathbb{Z}_P$ where P is a product of small primes. We will show how to reveal the plaintext modulo a small prime factor p_i without revealing the rest of the plaintext. Interpreting even numbers as 0, odd numbers as 1, and $p_i - 1$ as "ignore" we get a uniform distribution of hardcore bits modulo p_i assuming the Naccache-Stern encryption is semantically secure. With Naccache-Stern's cryptosystem having constant expansion rate and each prime factor of the message space being of logarithmic size in the security parameter we can construct a hidden random bits implementation that is quasi-linear in the number of hidden bits.

In the Naccache-Stern cryptosystem the public key is of the form $pk = (M, P, g)$, where M is a k-bit RSA modulus, P is a product of small odd primes $p_1, \ldots, p_d$ so $\gcd(8P^2, \varphi(M)) = 4P$, and $g \in \mathbb{Z}_M^*$ is a group element with $\mathrm{ord}(g) = \frac{\varphi(M)}{4}$. The secret key is $sk = \varphi(M)$. Encrypting a message $m \in \mathbb{Z}_P$ with randomness $r \leftarrow \mathbb{Z}_M^*$ yields the ciphertext

$$c = g^m r^P \bmod M.$$

To decrypt a ciphertext c, compute $c^{\frac{\varphi(M)}{P}} = (g^{\frac{\varphi(M)}{P}})^m$ and use the Pohlig-Hellman algorithm for finding discrete logarithms in groups with a smooth order to compute $m \bmod P$.

The cryptographic assumption underlying our NIZK proof is that there is a probabilistic polynomial time key generator K_{NS} for generating Naccache-Stern keys (pk, sk) such that the cryptosystem is IND-CPA secure and the number of small prime factors in P is larger than $\beta \frac{k}{\log k}$ for some constant $\beta > 0$. We refer to Naccache and Stern [NS98] for concrete key generator suggestions and a proof that the resulting cryptosystem is IND-CPA secure under a computational intractability assumption related to higher residues.

Opening and simulating openings of hardcore bits. In the implementation of the Naccache-Stern cryptosystem, the prover will generate Naccache-Stern keys $pk = (M, P, g)$ and $sk = \varphi(M)$. The random string is interpreted as a series of k-bit integers where those outside $\mathbb{Z}_M^*$ are ignored. An integer in $\mathbb{Z}_M^*$ can be interpreted as a ciphertext encrypting some message $m \bmod P$ where $P = \prod_{i=1}^{t} p_i$. Since there are $d = \lceil \frac{\beta k}{\log k} \rceil$ prime factors in P, this gives the prover d residues $\{m \bmod p_i\}_{i=1}^{d}$, each of which is translated into a hardcore bit. The prover will use the first N' hardcore bits as the hidden bit string and since she gets $\Theta(\frac{k}{\log k})$ bits per element in $\mathbb{Z}_M^*$ she only looses a logarithmic factor in implementing the hidden bit string.

The key observation needed for using Naccache-Stern encryption in this way is that the prover may verifiably disclose $m_i = m \bmod p_i$ without revealing the other parts of the message. Consider a particular k-bit block $c \in \mathbb{Z}_M^*$, which the prover can decrypt to get the plaintext $m \in \mathbb{Z}_P$. All $c \in \mathbb{Z}_M^*$ are valid ciphertexts but there are P different $r \in \mathbb{Z}_M^*$ so $c = r^P g^m$ so we will for notational

convenience fix an arbitrary such r in the following. To prove $m_i = m \bmod p_i$ is indeed part of the plaintext the prover gives a proof π satisfying

$$\pi^P = (cg^{-m_i})^{\frac{P}{p_i}}.$$

Raising both sides to the power $\frac{\phi(M)}{P}$ shows

$$1 = (\pi^P)^{\frac{\phi(M)}{P}} = (r^P g^{m-m_i})^{\frac{P}{p_i}\frac{\phi(M)}{P}} = (g^{\frac{\phi(M)}{p_i}})^{m-m_i}$$

telling the verifier that $m_i = m \bmod p_i$ since $P|\mathrm{ord}(g)$. The prover with the secret key $\phi(M)$ can compute a random π satisfying the equation by choosing $s \in \mathbb{Z}_M^*$ at random and setting

$$\pi = (cg^{-m_i})^{(P^{-1} \bmod \frac{\varphi(M)}{P})\frac{P}{p_i}} s^{\frac{\phi(M)}{P}}.$$

In the NIZK proof, we will generalize this idea to verifiably disclose $m \bmod P_I$ for arbitrary $P_I = \prod_{i \in I} p_i$. This makes it possible for the prover to reveal many values $\{m \bmod p_i\}_{i \in I}$ simultaneously.

There is a little variation in how many hardcore bits the prover gets out of a common reference string since not all k-bit integers will belong to $\mathbb{Z}_M^*$ and some hardcore bits are ignored but we can use Chernoff bounds to get a good estimate of how many hardcore bits the prover can extract and tune the proof accordingly. Since the verifier obtains proofs π for the correctness of the opened hardcore bits the soundness of the hidden bit proof system implies soundness of the full NIZK proof for circuit satisfiability.

The zero-knowledge property will come from using a different type of public key. Instead of using g that has order $\frac{\phi(M)}{4}$ the simulator will pick g with order $\frac{\phi(M)}{4P}$. As we shall see in the security proof, the semantic security of the Naccache-Stern cryptosystem implies that the two types of public keys are computationally indistinguishable. With the latter choice of public key $\mathrm{ord}(g) = \frac{\phi(M)}{4P}$ we can write $g = (g')^P$ and now a ciphertext is no longer binding since $c = r^P g^m = r^P (g')^{mP} = (r(g')^{m-m'})^P g^{m'}$ is at the same time an "encryption" of m and m'. The simulator sets up the common reference string so it contains ciphertexts that can be opened to any hardcore bits it chooses thereby allowing it to simulate the hidden bits proof.

NIZK proof based on Naccache-Stern encryption. We will now give the full NIZK proof for circuit satisfiability. The statement is a circuit C and the prover will have a satisfying witness w so $C(w) = 1$. Naccache-Stern keys are not directly verifiable, so we let $(\ell_{\mathrm{well}}, P_{\mathrm{well}}, V_{\mathrm{well}})$ be an NIZK proof system for well-formedness of a Naccache-Stern public key. This NIZK proof could for instance be Kilian and Petrank's original NIZK proof [KP98], which would have a cost of $\mathrm{poly}(k)$ bits. We will in the following let $N' = O(N \log N) = n\,\mathrm{polylog}(n)$ be the number of bits needed in the hidden bits model for circuits with n gates and let $\Delta' = \Theta((N')^{\frac{3}{4}})$. For notational simplicity, we will assume $d|N'$ and $\frac{N'+\Delta'}{\delta} \in \mathbb{Z}$, where $d = \lceil \frac{\beta k}{\log k} \rceil$ for a constant $\beta > 0$ and δ is defined in the protocol.

Common reference string: $\sigma = (\sigma_1, \ldots, \sigma_{\frac{3N'}{d}}, \sigma_{\text{well}}) \leftarrow \{0,1\}^{k\frac{3N'}{d}+\ell_{\text{well}}(k)}$.

Proof:

1. Generate Naccache-Stern keys $(pk, sk) = ((M, P, g), \phi(M)) \leftarrow K_{\text{NS}}(1^k)$ with $P = \prod_{i=1}^{d} p_i$.
2. Compute an NIZK proof π_{well} for the well-formedness of $pk = (M, P, g)$.
3. Define $\delta = d - \sum_{i=1}^{d} \frac{1}{p_i}$ and let $c_1, \ldots, c_{\frac{N'+\Delta'}{\delta}}$ be the first $\frac{N'+\Delta'}{\delta}$ of $\sigma_1, \ldots, \sigma_{\frac{3N'}{d}} \in \mathbb{Z}_M^*$.[2] If there are less than $\frac{N'+\Delta'}{\delta}$ of them return the proof $\pi = (pk, sk)$.
4. Decrypt $c_1, \ldots, c_{\frac{N'+\Delta'}{\delta}}$ to get plaintexts $m_1, \ldots, m_{\frac{N'+\Delta'}{\delta}}$. Define $m_{i,j} = m_j \bmod p_i$.
5. Define $h_{1,1}, \ldots, h_{d,\frac{N'+\Delta'}{\delta}}$ as $h_{i,j} = \perp$ if $m_{i,j} = -1$ and otherwise $h_{i,j} = 0$ if $m_{i,j}$ is even and $h_{i,j} = 1$ if $m_{i,j}$ is odd. If there are less than N' or more than $N' + 2\Delta'$ hardcore bits $h_{i,j} \in \{0,1\}$ return the proof $\pi = (pk, sk)$.
6. Define $\rho = (\rho_1, \ldots, \rho_{N'})$ as the first N' hardcore bits $h_{i,j}$.
7. Run the hidden bit string proof on ρ to get $\pi_H \leftarrow P_H(\rho, f(C), f_w(w))$.
8. Define $m_{i,j}$ as revealed if the hardcore bit $h_{i,j}$ is revealed in π_H or $h_{i,j} = \perp$.
9. Let for all j the set $I_j \subset \{1, \ldots, d\}$ be the indices i for which $m_{i,j}$ is revealed. Define $m_{I_j} = m_j \bmod P_{I_j}$ where $P_{I_j} = \prod_{i \in I_j} p_i$. Compute $\pi_j = (cg^{-m_{I_j}})^{(P^{-1} \bmod \frac{\phi(M)}{P})\frac{P}{P_{I_j}}} s_j^{\frac{\phi(M)}{P}}$ for a randomly chosen $s_j \leftarrow \mathbb{Z}_M^*$.

The proof is either $\pi = (pk, sk)$ or

$\pi = (pk, \pi_{\text{well}}, I_1, m_{I_1}, \pi_1, \ldots, I_{\frac{N'+\Delta'}{\delta}}, m_{I_{\frac{N'+\Delta'}{\delta}}}, \pi_{\frac{N'+\Delta'}{\delta}})$.

Verification:

1. If the proof is of the form $\pi = (pk, sk)$ accept it if and only if the key is well-formed (the secret key can be of a form so this can be verified) and there are less than $\frac{N'+\Delta'}{\delta}$ values in $\mathbb{Z}_M^*$ or the number of valid hardcore bits $h_{i,j} \in \{0,1\}$ is less than N' or higher than $N' + 2\Delta'$.
2. Verify the NIZK proof π_{well} for $pk = (M, P, g)$ being a correctly generated public Naccache-Stern key with d small odd primes $p_1, \ldots, p_d$.
3. Identify the first $\frac{N'+\Delta'}{\delta}$ values $c_1, \ldots, c_{\frac{N'+\Delta'}{\delta}} \in \mathbb{Z}_M^*$. Reject if there are less than $\frac{N'+\Delta'}{\delta}$ of them.
4. Verify the proofs $\pi_j^P = (cg^{-m_{I_j}})^{\frac{P}{P_{I_j}}} \bmod M$ and compute the hardcore bits $h_{i,j} \in \{0,1\}$ corresponding to $m_{I_1}, \ldots, m_{I_{\frac{N'+\Delta'}{\delta}}}$. Reject if the number of unopened hardcore bits plus opened valid hardcore bits $h_{i,j}$ is less than N' or more than $N' + 2\Delta'$.
5. Interpret the $h_{i,j} \in \{0,1\}$ as a hidden bits proof $(i_1, \rho_{i_1}, \ldots, i_t, \rho_{i_t})$. Accept if the verifications succeed and $V_H(f(C), i_1, \rho_{i_1}, \ldots, i_t, \rho_{i_t}) = 1$.

The construction gives us the following theorem that we prove in the full paper.

[2] We represent elements of $\mathbb{Z}_M^*$ as integers in the range $\{1, \ldots, M-1\}$.

Theorem 3. *Assuming the Naccache-Stern cryptosystem is IND-CPA secure, there is an NIZK proof for circuit satisfiability with perfect completeness, statistical soundness and computational zero-knowledge. The size of the common random string and the proof is* $|C|$ polylog $|C| + \text{poly}(k)$ *bits.*

7 Conclusion

We have suggested the shortest known NIZK proofs based on standard intractability assumptions. Based on trapdoor permutations we get an NIZK proof and common reference string of size $|C|k$ polylog$k$ bits (where we use that polylog$|C|$ = polylog$k$). This is a factor $\frac{k}{\text{polylog}k}$ improvement over Kilian and Petrank's construction [KP98].

Based on a specific number-theoretic assumption related to factoring, we get a very efficient implementation of a hidden bit string and an even shorter NIZK proof with a complexity of $|C|$ polylog$k$ bits. This is asymptotically a factor $\frac{k^3}{\text{polylog}k}$ more efficient than the pairing-based constructions by Groth, Ostrovsky and Sahai [GOS06b, GOS06a] (assuming the group elements have size $\frac{k^3}{\text{polylog}k}$) although it remains an open problem to reduce the polylogarithmic factor to make our construction practical.

References

[ALM+98] Arora, S., Lund, C., Motwani, R., Sudan, M., Szegedy, M.: Proof verification and the hardness of approximation problems. Journal of the ACM 45(3), 501–555 (1998)

[AS98] Arora, S., Safra, S.: Probabilistic checking of proofs: a new characterization of NP. Journal of the ACM 45(1), 70–122 (1998)

[BFM88] Blum, M., Feldman, P., Micali, S.: Non-interactive zero-knowledge and its applications. In: STOC, pp. 103–112 (1988)

[BSS08] Ben-Sasson, E., Sudan, M.: Short pcps with polylog query complexity. SIAM Journal of Computing 38(2), 551–607 (2008)

[BW06] Boyen, X., Waters, B.: Compact group signatures without random oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 427–444. Springer, Heidelberg (2006)

[BY92] Bellare, M., Yung, M.: Certifying cryptographic tools: The case of trapdoor permutations. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 442–460. Springer, Heidelberg (1993)

[CGS07] Chandran, N., Groth, J., Sahai, A.: Ring signatures of sub-linear size without random oracles. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 423–434. Springer, Heidelberg (2007)

[Dam92] Damgård, I.: Non-interactive circuit based proofs and non-interactive perfect zero-knowledge with preprocessing. In: Rueppel, R.A. (ed.) EUROCRYPT 1992. LNCS, vol. 658, pp. 341–355. Springer, Heidelberg (1993)

[DDN00] Dolev, D., Dwork, C., Naor, M.: Non-malleable cryptography. SIAM Journal of Computing 30(2), 391–437 (2000)

[DDP02] De Santis, A., Di Crescenzo, G., Persiano, G.: Randomness-optimal characterization of two NP proof systems. In: Rolim, J.D.P., Vadhan, S.P. (eds.) RANDOM 2002. LNCS, vol. 2483, pp. 179–193. Springer, Heidelberg (2002)

[Din07] Dinur, I.: The PCP theorem by gap amplification. Journal of the ACM 54(3) (2007)

[DR04] Dinur, I., Reingold, O.: Assignment testers: Towards a combinatorial proof of the pcp-theorem. In: FOCS, pp. 155–164 (2004)

[FLS99] Feige, U., Lapidot, D., Shamir, A.: Multiple non-interactive zero knowledge proofs under general assumptions. SIAM Journal of Computing 29(1), 1–28 (1999)

[Gen09] Gentry, C.: Fully homomorphic encryption using ideal lattices. In: STOC, pp. 169–178 (2009)

[GK96] Goldreich, O., Krawczyk, H.: On the composition of zero-knowledge proof systems. SIAM Journal of Computing 25(1), 169–192 (1996)

[GL89] Goldreich, O., Levin, L.A.: A hard-core predicate for all one-way functions. In: STOC, pp. 25–32 (1989)

[GMR89] Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proofs. SIAM Journal of Computing 18(1), 186–208 (1989)

[GO94] Goldreich, O., Oren, Y.: Definitions and properties of zero-knowledge proof systems. Journal of Cryptology 7(1), 1–32 (1994)

[GO07] Groth, J., Ostrovsky, R.: Cryptography in the multi-string model. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 323–341. Springer, Heidelberg (2007)

[GOS06a] Groth, J., Ostrovsky, R., Sahai, A.: Non-interactive zaps and new techniques for NIZK. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 97–111. Springer, Heidelberg (2006)

[GOS06b] Groth, J., Ostrovsky, R., Sahai, A.: Perfect non-interactive zero-knowledge for NP. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 339–358. Springer, Heidelberg (2006)

[GPS08] Galbraith, S.D., Paterson, K.G., Smart, N.P.: Pairings for cryptographers. Discrete Applied Mathematics 156(16), 3113–3121 (2008)

[Gro06] Groth, J.: Simulation-sound NIZK proofs for a practical language and constant size group signatures. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 444–459. Springer, Heidelberg (2006)

[GS08] 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)

[Kil92] Kilian, J.: A note on efficient zero-knowledge proofs and arguments. In: STOC, pp. 723–732 (1992)

[KP98] Kilian, J., Petrank, E.: An efficient noninteractive zero-knowledge proof system for NP with general assumptions. Journal of Cryptology 11(1), 1–27 (1998)

[NS98] Naccache, D., Stern, J.: A new public key cryptosystem based on higher residues. In: ACM CCS, pp. 59–66 (1998)

[Ore87] Oren, Y.: On the cunning power of cheating verifiers: Some observations about zero knowledge proofs. In: FOCS, pp. 462–471 (1987)

[Sah01] Sahai, A.: Non-malleable non-interactive zero-knowledge and adaptive chosen-ciphertext security. In: FOCS, pp. 543–553 (2001)

Optimistic Concurrent Zero Knowledge

Alon Rosen[1] and abhi shelat[2]

[1] IDC Herzileya
alon.rosen@idc.ac.il
[2] U. of Virginia
shelat@cs.virginia.edu

Abstract. We design cryptographic protocols that recognize best case (optimistic) situations and exploit them. As a case study, we present a new concurrent zero-knowledge protocol that is expected to require only a small constant number of rounds in practice. To prove that our protocol is secure, we identify a weak property of concurrent schedules—called *footer-freeness*—that suffices for efficient simulation.

Keywords: concurrent zero-knowledge, rational, optimistic.

1 Introduction

Cryptographic protocols anticipate *worst-case* behavior and therefore often contain complicated provisions that are meant solely to handle them. Such provisions can be expensive and counter-intuitive.

To circumvent these side-effects but still construct protocols that are secure against worst-case behavior, this paper proposes to use an *optimistic* technique for building protocols that is inspired by work on Byzantine agreement. The aim is to design protocols that can recognize the best cases and optimize for them, even in the midst of the protocol execution.

Optimism has been employed by researchers in distributed computing (e.g. the (Fast) Paxos algorithm [Lam05]) and fair exchange [ASW98]; the novelty of this work is to exploit *optimism* for the problem of concurrent zero-knowledge. Optimistic protocols make no attempt to improve worse-case performance. In fact doing so would require overcoming a lower bound argument in the case of zero-knowledge. Nonetheless, the optimistic cases that we exploit are common and meaningful to discuss.

1.1 Concurrent Zero-Knowledge

When many instances of a stand-alone zero-knowledge protocol are executed at the same time, the combination of all runs may leak information *about the theorem*. The standard methodology for arguing that a protocol transcript "does not leak information" is to exhibit a simulator algorithm that is able to produce transcripts that are indistinguishable from actual transcripts of protocol executions. Dwork, Naor and Sahai [DNS98] observed that in a concurrent zero-knowledge setting, a malicious verifier who controls the schedule of protocol messages can induce a schedule for which a

M. Abe (Ed.): ASIACRYPT 2010, LNCS 6477, pp. 359–376, 2010.

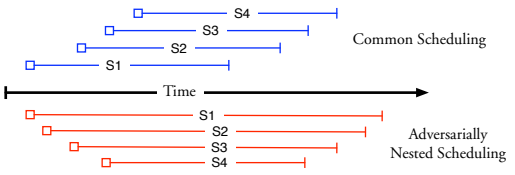


Fig. 1. Illustration of an "average-case" schedule, and an adversarial one

"naive" simulation algorithm will require exponential time (and thus the execution may leak information). An example of such a scheduling of messages is given in 1. In the bottom (red) schedule, the verifier has "nested" many executions of the zero-knowledge protocol. This type of scheduling is a *concurrency attack* on the zero-knowledge property of the original protocol and it captures the fundamental problem with designing efficient concurrently secure zero-knowledge protocols.

To address the concurrency attack, Dwork, Naor and Sahai [DNS98] proposed a timing model assumption and a protocol that limits the amount of nesting that can occur in an adversarial scheduling. Their protocol was an argument system; Goldreich [Gol02] later showed that proof systems can also be constructed in such a model. Pass, Tseng, and Venkitasubramaniam [PTV10] present an eye-for-an-eye solution in the timing model that reduces the overall delay of the protocol. Other protocols that handle concurrency attacks have been obtained by introducing different setup assumptions [DS98, Dam99, CGGM00] such as a common reference string or a PKI.

Richardson and Kilian [RK99] constructed the first concurrent zero-knowledge argument system in the standard model without extra setup assumptions. Kilian and Petrank [KP01] introduced a simulation technique which led to simpler and cleaner analysis and fewer rounds. Finally, the work of Prabhakaran, Rosen and Sahai [PRS02] (PRS) further simplified and improved the analysis of the Kilian and Petrank protocol to obtain a protocol with $\omega(\log n)$ rounds. This round complexity is close to optimal in the standard model because without any set-up assumptions, Canetti, Kilian, Petrank and Rosen [CKPR01, CKPR02] show that concurrent zero-knowledge argument systems for non-trivial languages using a "black-box" simulator require at least $\Omega(\log n/\log\log n)$ number of communication rounds. In order to show this lower bound, they rely on a framework proposed by Kilian, Petrank and Rackoff [KPR98], with further improvements from Rosen [Ros00], and present a specific malicious verifier and a particularly difficult schedule of messages. Recently, Pandey et al. [PPS+08] have proposed new precise concurrent zero-knowledge proofs with similar round complexity.

It has been a long-standing open problem to build communication-efficient concurrent zero-knowledge protocols. To circumvent the lower bound on the round complexity from [CKPR01, CKPR02], prior work (1) introduces additional trust assumptions [DNS98, DS98, Dam99, CGGM00], (2) relaxes the definition of security to allow quasi-polynomial time simulation [Pas03, PS04, PV08], or (3) employs a more complicated and powerful non-black-box simulation technique [Bar01] and restricted the number of concurrent sessions. The latter technique also relies on complex tools and techniques that require NP-reductions.

Overview of slots. Before detailing our approach, let us review the idea employed by the protocols from [RK99, KP01, PRS02, PTV10] to defend against concurrent attacks. In the first phase of the protocol, the verifier creates an irrelevant secret (such as a commitment to a string) and then (repeatedly) proves in zero-knowledge to the prover that it knows the secret. Each block of protocol messages during which the verifier proves knowledge of this secret is called a "slot." In the second phase, the Prover proves that it either knows the witness to the original theorem or that it knows the verifier's secret using a witness indistinguishable protocol. Prior work [RK99, KP01, PRS02, Ros06] proves that if the first phase has enough slots, then a simulation strategy can be devised such that for *any schedule of messages*, the simulator can successfully extract a witness from the verifier's proof and then use that witness in the second phase.

Optimistic Defense. We propose an optimistic defense against concurrency attacks in which we relax the requirements from prior work and specifically [CKPR01] that (1) each protocol session involves an independent prover who does not know anything about the other protocol instances and (2) each protocol execution has exactly the same (fixed) number of rounds. Doing so allows us to build protocols that *optimistically avoid* the worst-case schedules used in the lower bounds.

When one server handles many concurrent requests, the server knows the exact schedule of messages. The work of Persiano and Visconti [PV05] also exploits this relaxation by using a Prover who counts the total number of bytes sent in all sessions[1].

We believe this to be a reasonable and practical relaxation. In many applications of zero-knowledge proofs, for example, the prover will be the same party (some server), and it will have the opportunity to share state between protocol sessions. In particular, servers on the internet routinely keep track of the various protocol session statistics such as the total number of protocol executions that run at a given time. Operating systems which make quality of service guarantees also inspect different protocol instances in order to throttle connections. While the original motivation of the concurrent session model in which the Prover instances run independently of one another was to simplify implementation of systems, there is no fundamental implementation reason that prevents sharing the global scheduling information among the Prover algorithms in different protocol sessions. (Of course, requiring the Verifiers to coordinate their sessions would be unrealistic, since the Verifiers may be separate parties.)

In our model, each session of the protocol may require a different number of communication rounds. This relaxation allows us to instruct the prover to handle schedules which are *easier to simulate* differently than schedules which are more difficult. In contrast, in typical cryptographic protocols, each execution of the protocol has a fixed number of messages and each successful invocation usually requires exactly the same number.

Our protocol can "short circuit" the normal protocol when it is clear that such a shortcut preserves the security properties. To the best of our knowledge, this idea has not been applied in the context of a security guarantee such as zero-knowledge.

[1] In contrast to this work, their solution uses non-blackbox simulation techniques.

1.2 Our Protocol

The idea behind our fast track protocol is to discourage the verifier V^* from nesting sessions within the slots of other sessions by penalizing a verifier whose slots have nested sessions. The penalty will be to gradually add more slots to the protocol until either enough slots with no (or few) nestings occur, or a pre-specified bound on the number of rounds is reached.

The basis for our protocol is the $c\mathcal{ZK}$ protocol by Prabhakharan, Rosen and Sahai [PRS02] which uses statistically hiding commitments [NY89, DPP93] and Blum's 3-message protocol for Hamiltonicity [Blu86]. The only difference between our protocol and the PRS protocol is that it contains a special provision for exiting the "preamble" stage. Early exits are "approved" by the prover, provided that there is a slot in the current session that does not have any other session footers within it. Assuming that verifiers answer quickly, it is expected that the number of nested sessions within slots is generally small, optimistically resulting in an empty slot and thus in straightforward simulation.

Verifiers have incentive to answer fast since the longer they delay their answer, the more likely they are to have nested sessions (from some other verifier) within the slot that they are currently executing. Once the slot has a nested session within it, early exit is postponed to future rounds, and another slot is added to the protocol's execution. This process continues until $k = \omega(\log n)$ slots have been performed, in which case the Hamiltonicity proof takes place and the protocol terminates.

At the expense of a more involved analysis, one should be able to replace the PRS protocol with any other instantiation of a $c\mathcal{ZK}$ protocol that follows the RK "multi slot" paradigm, and obtain analogous results. One attractive instantiation would be the DDH-based $c\mathcal{ZK}$ protocols of Micciancio and Petrank [MP03]. These protocols admit fairly efficient implementations, and are thus a good match for our optimistic approach, whose primary objective is increased efficiency.

But worst-case schedules still require many rounds! Note that worse-case schedules still require the same number of rounds as PRS. Such an argument applies to any optimistic protocol, such as the Fast Paxos or Fair exchange protocols as well. The *worst-case* schedule, however, may be *rare* and *avoidable* by incentivized verifiers.

Comparison with Other Proposals. In Appendix A, we compare our approach to other simple proposals and to the timing model proposed by [DNS98].

2 Optimistic Rational Concurrency

Let $\langle P, V \rangle$ be an interactive proof (resp. argument) for a language L, and consider a single *concurrent adversary* (verifier) V^* that, given input $x \in L$, interacts an unbounded number of times with P (each with common input x) without any restrictions over the scheduling of its messages.

Formally, use the standard model for concurrency in the timing model put forth by Dwork, Naor, and Sahai [DNS98]. The adversary V^* takes as input the prover's partial conversation transcript that includes the times on the provers local clock when each message was sent or received by the prover. The adversary's output is either a

tuple $(recv, V, \alpha, t)$, indicating that P receives message α from V at local time t or $(send, V, t)$, indicating that P must send the next message to V at time t on P's local clock. In both cases, the time t that adversary chooses must be greater than all the times given in the input transcript (i.e., the adversary cannot rewind P), the session with V must be well-formed, and α must be in the protocol's "message space" (i.e. standard well-formedness conditions apply). If these conditions are not met, the transcript is discarded.

The *transcript* of a concurrent interaction consists of the common input x, followed by the sequence of prover and verifier messages exchanged during the interaction. We denote by $\text{view}^P_{V^*}(x)$ a random variable describing the content of the random tape of V^* and the conversation transcript between P and V^* as described above.

Definition 1 (Concurrent Zero-Knowledge). *Let $\langle P, V\rangle$ be an interactive proof system for a language L. We say that $\langle P, V\rangle$ is* concurrent zero-knowledge*, if for every probabilistic strict polynomial-time concurrent adversary V^* there exists a probabilistic polynomial-time algorithm S_{V^*} such that the ensembles $\{\text{view}^P_{V^*}(x)\}_{x\in L}$ and $\{S_{V^*}(x)\}_{x\in L}$ are computationally indistinguishable.*

Discussion. There may be other verifiers that are also interacting with P at the same time as V^*. In prior work, these sessions are ignored because either the monolithic adversary V^* can incorporate these sessions if they can be used to cheat, or because these extra sessions are completely independent of V^*'s view.

In our case, however, these extra sessions by honest verifiers are not completely independent of V^*'s view.[2] In the protocol we suggest, for example, a verifier will learn when one of its slot is not footer-free, and therefore it will learn the presence of another session. This is not necessarily the case with other concurrent ZK protocols such as PRS because the number of rounds in those protocols are not related to the schedule of messages. However, the aim for a zero-knowledge protocol in a networked setting is to ensure that no information about the witness w for instance x is leaked; we feel that it is reasonable for a protocol to leak network timing information because such information is typically leaked by the underlying network (or by timing or side channels).

To model this, we give V^* full control over the timing of *all network messages* including the Prover's messages and the timing of the messages from the honest verifier sessions that are not controlled by V^*. Although this is syntactically the same formal model with a single V^* as in prior work, there is a subtle difference. Our protocol and its simulator essentially guarantees that "a verifier V' who controls a subset of the sessions learns no more through interaction with P than a malicious verifier V^* who controls all network traffic, and such a verifier learns no more than the polynomial time simulator who does not have the witness."

Notation. We use the symbols $(\text{V}0), (\text{P}1), (\text{V}1), \ldots, (\text{P}j), (\text{V}j)$ to denote the messages in the *preamble*; these messages are completely independent of the common input and they serve to enable a successful simulation in the concurrent setting.

Every round (*slot*) in the preamble (i.e., every $(\text{P}j), (\text{V}j)$ pair) is viewed as a "rewinding opportunity." Successfully rewinding even one slot in the preamble is suf-

[2] We thank the anonymous reviewer for pointing out this subtle distinction.

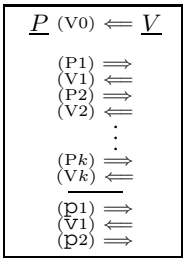


Fig. 2. A k-round preamble

ficient in order to cheat arbitrarily in the actual proof (messages $(\mathtt{p1}), (\mathtt{v1}), (\mathtt{v2})$) and thus complete the simulation.

One problem faced by a $c\mathcal{ZK}$ simulator is that rewinding a specific session may result in loss of work done for other sessions, and therefore require the simulator to do the same amount of work again. This will happen whenever the rewound slot contains other sessions "nested" within it.

For example, if a slot of session B contains the $(\mathrm{V0})$ message of session A within it, rewinding this slot will cause all simulation work done for session A to be lost. This is because the simulation of a session A hinges on the simulator "extracting" specific values that have been committed to by the verifier in message $(\mathrm{V0})$ of this session. Rewinding past the $(\mathrm{V0})$ message of A could alter the history of interaction up to this message and may result in a modification of its contents (rendering the extracted values irrelevant).

The simulator must invest work in session A whenever session A's preamble completes before the end of the slot of session B. In such a case, reaching the end of session A's preamble without having extracted the value committed to in message $(\mathrm{V0})$ of session A may prevent the simulator to proceed beyond the end of this preamble (since the malicious verifier may refuse to continue if is not convinced in the validity of the statement being proved in session A). Failure to proceed beyond the end of the session A preamble translates directly to failure to rewind the session B slot within which this preamble is nested.

Definition 2 (Nested Footer). *Slot j of session B is said to have a* nested footer *of session A within it if session A's $(\mathrm{V}k)$ message occurs between messages $(\mathrm{P}j), (\mathrm{V}j)$ of session B. A slot is said to be* footer free *if it has no nested footer.*

Avoiding nested footers enables the completion of the slot between messages $(\mathrm{P}j)$ and $(\mathrm{V}j)$ of session B without having to first invest work in simulating session A (implying that there is no risk to lose and thus redo this work as a result of rewinding). This observation will be crucial to the analysis of the footer-free version of our protocol.

Two simulation strategies. Currently, there are two known approaches for concurrent simulation. The first simulation strategy *adaptively* looks for slots that do not have many sessions with nested headers within them, and this is where it focuses its attempts to rewind the interaction with the verifier [RK99]. The second simulation strategy is different in that it performs a sequence of rewinds *obliviously* of the actual scheduling of the messages [KP01, PRS02].

The main advantage of the second approach over the first one is that it is known to guarantee correct "worst case" simulation using fewer slots ($\tilde{O}(\log n)$ vs. $O(n^{\epsilon})$ for every $\epsilon > 0$). However, being oblivious to the actual schedule, it does not seem suitable for taking advantage of lack of nested headers and/or footers within a slot. As we demonstrate in this paper, by adaptively identifying good places for multiple rewindings, the second approach can be tailored to work in our optimistic setting.

To the best of our knowledge, the idea of taking advantage of the lack of nested footers for the sake of improved concurrent simulation is new. As we argue in the paper, lack of nested footers within one slot is a fairly weak constraint on the schedule, and may be enforced using a variety of realistic mechanisms.

The protocol. Our protocol will have the prover monitor the scheduling of messages, and identify footer-free slots on the fly; once such a slot is identified, there is no need to keep adding slots to the execution of that specific session, so the protocol moves on to the execution of the actual constant-round $\mathcal{ZK}$ protocol.

Common Input: $x \in \{0,1\}^n$, security param n, max. # rounds param $k = \omega(\log n)$.
Prover's Input: a witness w such that $R_L(x,w) = 1$
Stage 1:
$P \rightarrow V$ (P0): Send first message of perfectly hiding commitment $\mathbf{Com}$.
$V \rightarrow P$ (V0): Using the commitment $\mathbf{Com}$, commit to random $\sigma, \{\sigma^0_{i,j}\}^k_{i,j=1}$, $\{\sigma^1_{i,j}\}^k_{i,j=1}$ such that $\sigma^0_{ij} \oplus \sigma^1_{ij} = \sigma$ for all i,j.
Slot j:
$P \rightarrow V$ (Pj)**:** Send a random challenge $r_i = r_{1,j}, \ldots, r_{k,j}$
$V \rightarrow P$ (Vj)**:** Upon receiving a message r_i, decommit to $\sigma^{r_{1,j}}_{1,j}, \ldots, \sigma^{r_{k,j}}_{k,j}$
$P \rightarrow V$**:** If any of the decommitments fails verification, abort.
If slot j is *footer free* **or** $j = k$ move to stage 2.
If slot j is **not** *footer free* **and** $j \leq k$ move to slot $j+1$.
Stage 2: P and V engage in Blum's 3-round Hamiltonicity protocol using challenge σ:
1. $P \rightarrow V$ (p1): Use witness to produce first message of Ham protocol
2. $V \rightarrow P$ (v1): Decommit to σ and to $\{\sigma^{1-r_{i,j}}_{ij}\}^k_{i,j=1}$.
3. $P \rightarrow V$ (p2): If decommitments are valid and $\sigma^0_{i,j} \oplus \sigma^1_{i,j} = \sigma$ for all i,j, answer σ with third message of Ham protocol. Otherwise abort.

Fig. 3. Fast-track concurrency

Completeness and soundness of Protocol 3 are inherited from the PRS protocol, and in particular follow from Proposition 4.3.2 in [Ros06]. We now turn to demonstrating the $c\mathcal{ZK}$ property.

2.1 The Simulator

We exhibit the $c\mathcal{ZK}$ property using a black-box simulator S. Let V^* be a concurrent adversary verifier. S will rewind the interaction with V^* and examine its input/output behavior. The rewinding strategy of the simulator is specified by a SOLVE procedure whose goal is to supply the simulator with V^*'s "challenges" before reaching stage

2 in the protocol. This is done by rewinding the interaction with V^* while trying to achieve two "different" answers to some $(\mathrm{P}j)$ message. *We refrain from specifying the way stage 2 messages are handled* and focus only on stage 1 messages. For standard details on how to handle stage 2 messages see [Ros06].

The timing of the rewinds performed by the SOLVE procedure depends on the number of stage 1 verifier messages received so far and on the size of the schedule. However, whenever it encounters a situation in which a slot of a given session is footer-free, the SOLVE procedure (adaptively) assumes that this is its only chance to solve that session and performs (an expected polynomial number of) extra rewinds in order to make sure that the slot is successfully rewound. The number of extra rewinds is not determined in advance, and is induced by the analysis of a constant round $\mathcal{ZK}$ protocol for $\mathcal{NP}$ by Rosen [Ros04].

At a high level, the SOLVE procedure splits the first stage messages it is about to explore into two halves and invokes itself recursively twice for each half (completing the two runs of the first half before proceeding to the two runs of the second half). At the top level of the recursion, the messages that are about to be explored consist of the entire schedule, whereas at the bottom level the procedure explores only a single message (and as we said may do so multiple times, depending on whether the recursive call corresponds to a message-free slot). The solve procedure always outputs the sequence of "first explored" messages.

The input to the SOLVE procedure consists of a triplet $(\ell, \mathsf{hist}, \mathcal{T})$. The parameter ℓ corresponds to the total number of verifier messages, the string hist consists of the messages in the "first visited" history of interaction, and $\mathcal{T}$ is a table containing the contents of all the messages explored so far. The messages stored in $\mathcal{T}$ are used in order to determine σ according to answers $(\mathrm{V}j)$ to different $(\mathrm{P}j)$. They are kept relevant by constantly keeping track of the sessions that are rewound past their initial commitment. That is, whenever the SOLVE procedure rewinds past the $(\mathrm{V}0)$ message of a session, all messages belonging to this session are deleted from $\mathcal{T}$.

The analysis takes advantage of the fact that no rewound slot contains a footer, building on the assumption that footer-freeness is an event of non-negligible probability (as otherwise it is assumed not to have occurred to begin with). By repeatedly rewinding, the simulator is likely to run into a footer-free situation again, which means that it will not get stuck on that rewinding. This will enable it to successfully complete the rewinding attempt, and to solve the corresponding session (thus avoiding getting stuck on sessions that have strictly less than k slots).

2.2 Analysis of the Simulator

To show that the simulator S succeeds we will need to argue that: (1) S runs in polynomial time, (2) conditioned on the success of the SOLVE procedure, the output of S is indistinguishable from a concurrent interaction between P and V^*, and (3) for every session $i \in \{1, \ldots, m\}$, whenever session i reaches the second stage in the protocol, the simulator will have obtained the value of σ in this session if required (i.e. did not get *stuck*) with overwhelming probability. Once (3) is established, we may apply a union bound over the i's and conclude that SOLVE fails with only negligible probability. We focus on (1) and (3).

Procedure SOLVE(ℓ, hist, $\mathcal{T}$):
Bottom level ($\ell = 1$):

1. For each $s \in \{1, \ldots, m\}$, if the initial commitment, (V0), of session s does not appear in hist, delete all session s messages from $\mathcal{T}$.
2. Run $\beta \leftarrow V^*(\mathsf{hist}, \mathrm{p})$. If β is of the form $(recv, V, \alpha, t)$, the continue to the next step. Else if it is $(send, V, t)$, then uniformly choose a first stage prover message p, append it to the transcript at time t, and repeat this step. If t or α are invalid, then halt the simulation and output the current transcript.
3. Let
   - $(\mathrm{p}_1, \mathrm{v}_1, \ldots, \mathrm{p}_t, \mathrm{v}_t) = (\mathsf{hist}, \mathrm{p}, \mathrm{v})$
   - i be the session number to which v corresponds,
4. If there exists a pair of indices (a, b) such that $a \in [t]$ and $b = t$ for which:
   - $\mathrm{v}_b \neq$ ABORT,
   - both v_b and p_a belong to session i.
   - the slot between messages p_a and v_b is footer-free.

   then pick one such (a, b) and rewind interaction to message p_a until
   - $\mathrm{v}_b \neq$ ABORT,
   - both v_b and p_a belong to session i.
   - the slot between messages p_a and v_b is footer-free.
5. Store the messages gathered in the rewindings along with p, v in $\mathcal{T}$
6. output $\mathcal{T}$, (p, v).

Recursive step ($\ell > 1$):

1. Set $\mathcal{T}_1$, $(\mathrm{p}_1, \mathrm{v}_1, \ldots, \mathrm{p}_{\ell/2}, \mathrm{v}_{\ell/2}) \leftarrow$SOLVE($\ell/2$, hist, $\mathcal{T}$).
2. Set $\mathcal{T}_2$, $(\tilde{\mathrm{p}}_1, \tilde{\mathrm{v}}_1, \ldots, \tilde{\mathrm{p}}_{\ell/2}, \tilde{\mathrm{v}}_{\ell/2}) \leftarrow$ SOLVE($\ell/2$, hist, $\mathcal{T}_1$).
3. Set $\mathcal{T}_3$, $(\mathrm{p}_{\ell/2+1}, \mathrm{v}_{\ell/2+1}, \ldots, \mathrm{p}_\ell, \mathrm{v}_\ell) \leftarrow$SOLVE($\ell/2$, $(\mathsf{hist}, \mathrm{p}_1, \mathrm{v}_1, \ldots, \mathrm{p}_{\ell/2}, \mathrm{v}_{\ell/2})$, $\mathcal{T}_2$).
4. Set $\mathcal{T}_4$, $(\tilde{\mathrm{p}}_{\ell/2+1}, \tilde{\mathrm{v}}_{\ell/2+1}, \ldots, \tilde{\mathrm{p}}_\ell, \tilde{\mathrm{v}}_\ell) \leftarrow$SOLVE($\ell/2$, $(\mathsf{hist}, \mathrm{p}_1, \mathrm{v}_1, \ldots, \mathrm{p}_{\ell/2}, \mathrm{v}_{\ell/2})$, $\mathcal{T}_3$).
5. Output $\mathcal{T}_4$, $(\mathrm{p}_1, \mathrm{v}_1, \ldots, \mathrm{p}_\ell, \mathrm{v}_\ell)$.

Fig. 4. The SOLVE procedure

Lemma 1. *For every $m = \mathrm{poly}(n)$, S_m runs in (expected) polynomial-time in n.*

Proof. We analyze the work invested at any given invocation of level $\ell = 1$. For any $G \in HC$, for any choice of hist, p, and of $a, b \in \{1, \ldots, t\}$ where $a \leq b$, let $\zeta_{a,b} = \zeta_{a,b}(G, \mathsf{hist}, \mathrm{p}, \mathrm{v})$ denote the probability that: (1) the verifier V^* does not send ABORT in message v_b, (2) both v_b and p_a belong to session i, (3) the slot between p_a and v_b is footer-free, (4) none of the v_j's correspond to message (V0) of session i, and (5) none of the p_j's correspond to message (p1) of session i. Let $\zeta'_{a,b}$ denote the probability that (1), (2) and (3) occur. The probabilities $\zeta_{a,b}$ and $\zeta'_{a,b}$ are taken over the random choices of the invocations of the SOLVE procedure. It can be seen that $\zeta'_{a,b} \geq \zeta_{a,b}$.

Using this notation, a pair (a, b) satisfying conditions (1)-(5) occurs with probability $\zeta_{a,b}$ and the SOLVE procedure is expected to repeat the loop in step 4 for at most $1/\zeta'_{a,b}$ times (since the condition in Step 4 is satisfied in each one of the rewinds with probability $\zeta'_{a,b}$, independently of other rewinds). For $i \in \{1, 2, 3, 4, 5, 6\}$, let $p_i(\cdot)$ be a polynomial bound on the work required in order to perform Step i in level $\ell = 1$ of the recursion (where in step 4, $p_4(\cdot) = p_{4,a,b}(\cdot)$ counts the number of steps required to

perform one rewinding). By linearity of expectation, and because the total number t of pairs of messages (and hence pairs $(a,b) \in R$) in the history of level $\ell = 1$ is at most $m \cdot (k+1)$ (recall that k is the maximal number of rounds in the protocol), the expected time required to execute level $\ell = 1$ of the recursion is upper bounded by:

$$\begin{aligned} & p_1(n) + p_2(n) + p_3(n) + \sum_{(a,b):a \leq b} \zeta_{a,b} \cdot \frac{1}{\zeta'_{a,b}} \cdot p_4(n) + p_5(n) + p_6(n) \\ & \leq p_1(n) + p_2(n) + p_3(n) + m \cdot (k+1) \cdot p_4(n) + p_5(n) + p_6(n) \\ & = \mathrm{poly}(n) \end{aligned}$$

Since each invocation of the SOLVE procedure with parameter $\ell > 1$ involves four recursive invocations of the SOLVE procedure with parameter $\ell/2$, we have that the expected work $W(\ell)$, that is invested by the SOLVE procedure in order to handle ℓ (first stage) verifier messages satisfies:

$$W(\ell) \leq \begin{cases} \mathrm{poly}(n) & \text{If } \ell = 1 \\ 4 \cdot W(\ell/2) & \text{If } \ell > 1 \end{cases} \tag{1}$$

Since the total number of first stage verifier messages in the m sessions of the concurrent schedule equals $m \cdot (k+1)$, the total expected running time of the simulation process (which consists of a single invocation of the SOLVE procedure with parameter $m \cdot (k+1)$) equals $W(m \cdot (k+1))$. By linearity of expectation we get that the expected value of $W(m \cdot (k+1))$ is upper bounded by:

$$4^{\log_2(m \cdot (k+1)) - \log_2 c} \cdot \mathrm{poly}(n) = \left(\frac{m \cdot (k+1)}{c} \right)^2 \cdot \mathrm{poly}(n) = \mathrm{poly}(n)$$

We now turn to show that for every $G \in HC$, the simulator's output distribution is computationally indistinguishable from V^*'s view of interactions with the honest prover P. Specifically,

Lemma 2. *The ensemble* $\{S_m^{V^*}(G)\}_{G \in HC}$ *is computationally indistinguishable from the ensemble* $\{\mathrm{view}_{V^*}^P(G)\}_{G \in HC}$.

Indistinguishability of the simulator's output from V^*'s view (of $m = \mathrm{poly}(n)$ concurrent interactions with P) is shown assuming that the simulator does not get "stuck" during its execution. Since the simulator S will get "stuck" only with negligible probability (see Lemma 3 below), indistinguishability will immediately follow.

The proof actually considers a "hybrid" simulator that on input $G = (V, E) \in HC$ obtains a directed Hamiltonian Cycle $C \subset E$ in G (as auxiliary input) and uses it in order to produce real prover messages whenever it reaches the second stage of the protocol. Specifically, whenever it reaches the second stage of session $s \in \{1, \ldots, m\}$, the hybrid simulator inspects the $\mathcal{T}$ table and checks whether it has managed to solve session s (thus being able to convince V^* in the section stage of session s). If it has not managed to solve session s, the hybrid simulator outputs $\perp$ and halts. Otherwise, the hybrid simulator follows the prescribed prover strategy and generates prover messages for the second stage of the session (by using the cycle C it possesses). The key for proving the above lies in the following two properties:

- First stage messages output by S are (almost) *identically* distributed to first stage messages sent by P. This property is proved based on the definition of the simulator's actions.
- Second stage messages output by S are *computationally indistinguishable* from second stage messages sent by P. This property is proved based on the special zero-knowledge property of Blum's Hamiltonicity protocol.

We now turn to argue that the hybrid simulator does not get stuck.

Lemma 3. *Let $\alpha : N \to N$ be any super-constant function, let $k(n) = \alpha(n) \cdot \log n$, and consider any instantiation of Protocol 3 with parameter $k = k(n)$. Then for any $i \in \{1, \ldots, m\}$ the probability of the hybrid simulator getting "stuck" on session i during the simulation is negligible.*

Proof. The SOLVE procedure is said to get *stuck* on session i if it reaches the second stage of session i and the following events occur: (1) the history of the interaction so far does not contain an ABORT message in session i, and (2) the table $\mathcal{T}$ does not contain two verifier messages $(\mathrm{V}j)$ and $(\mathrm{V}j)'$ that are replies to two *different* prover messages $(\mathrm{P}j)$ and $(\mathrm{P}j)'$. Note that if the history of the interaction does contain an ABORT message in session i then it is not necessary to obtain σ.

Consider any event in which the SOLVE procedure reaches the second stage of session i, and let hist denote the history of the interaction with which the second stage is reached. By definition of the solve procedure hist contains the messages first visited by the SOLVE procedure.

As before, we divide the analysis into two cases. In the first case, the number of slots in session i as they appear in hist is precisely k. The key for analyzing this case lies the fact that the SOLVE procedure as defined in this paper behaves identically to the SOLVE procedure described in [PRS02], except that in the bottom levels of the recursion the former may potentially perform more rewindings than the latter (but *never less*). This means that whenever the PRS variant of the SOLVE procedure manages to obtain the relevant value of σ then so does our variant. By the [PRS02] analysis, we know that as long as the number, k, of slots is super logarithmic, the PRS variant of the SOLVE procedure fails to obtain σ with negligible probability. Thus, the probability of getting stuck on session i in our case is negligible as well.

In the second case, the number of slots in session i as they appear in hist is strictly less than k. By definition of our protocol, this can happen only if there exists a slot in the history of the interaction that is footer-free.

Claim: *Suppose that the number of slots in session i is strictly less than k. Then, the schedule of messages as it appears in* hist *contains a slot in the history of the interaction that is footer free.*

Consider now any invocation of a bottom level of the recursion in which a footer free slot j of session i appears amongst messages $(\mathrm{p}_1, \mathrm{v}_1, \ldots, \mathrm{p}_t, \mathrm{v}_t) = (\mathsf{hist}, \mathrm{p}, \mathrm{v})$. Let $\mathrm{p}_a = (\mathrm{P}j), \mathrm{v}_b = (\mathrm{V}j)$ be those messages. By definition of the SOLVE procedure, the first messages generated in the visit will appear in hist. Let $\mathrm{p} = (\mathrm{P}j), \mathrm{v} = (\mathrm{V}j)$ be those messages. The simulator will get stuck if and only if: (1) hist does not contain an ABORT message in session i (and in particular if $\mathrm{v}_b \neq \texttt{ABORT}$), and (2) the table $\mathcal{T}$

does not contain two verifier messages $\mathrm{v}_b = (\mathrm{V}j)$ and $\mathrm{v}_{b'} = (\mathrm{V}j)'$ that are replies to two *different* prover messages $\mathrm{p}_a = (\mathrm{P}j)$ and $\mathrm{p}_{a'} = (\mathrm{P}j)'$. Since p_a and v_b belong to the same session, then if condition (1) is satisfied we have that the following three rewinding conditions hold:

- $\mathrm{v}_b \neq \texttt{ABORT}$,
- both v_b and p_a belong to session i, and
- the slot between messages p_a and v_b is footer-free,

This in particular means that the SOLVE procedure will rewind the interaction in Step 4, sending random p's until it finds another pair $\mathrm{p}'_a = (\mathrm{P}j)', \mathrm{v}' = (\mathrm{V}j)'$ in session i so that $\mathrm{p}_a \neq \mathrm{p}'_a$.

We next show that the probability of getting stuck (over random choices of $\mathrm{p}'_a = r \in \{0,1\}^k$ in the visit to the bottom level of the recursion) is precisely $1/2^k$. Since k is super-logarithmic it will immediately follow that the probability that the simulator gets stuck is negligible.

The key observation for the analysis is that, in the event that the slot between messages p_a and v_b is footer free, it will ultimately be possible to successfully perform the rewinding and reach some $\mathrm{v}' = (\mathrm{V}j)'$ message, *without having to "re-solve" a different session that is nested within the* j^{th} *slot of session* i. In other words, conditioned on the event of slot j being "footer-free" again (and $(\mathrm{V}j)'$ not being equal to ABORT), the rewinding will go through smoothly (since the simulation cannot get stuck on another session during that specific rewinding attempt).

For any $G \in HC$, and for any choice of hist, let $\zeta_{i,a,b} = \zeta_{i,a,b}(G, \mathsf{hist})$ denote the probability that: (1) message v_b corresponds to a $(\mathrm{V}j)$ message that is not equal to ABORT, (2) both v_b and p_a belong to session i, and (3) the slot between messages p_a and v_b is footer-free. The probability $\zeta_{i,a,b}$ is taken over the random choices of p_a. Using this notation, the SOLVE procedure proceeds to Step 4 with probability $\zeta_{i,a,b}$ (note that the condition in Step 4 is satisfied in each one of the rewinds with probability $\zeta_{i,a,b}$, independently of other rewinds). We would like to bound the probability that S gets stuck (we denote the event of the simulation getting stuck by having S output $\perp$).

Let S^A denote the simulator's execution with black box access to a machine A, let $\widetilde{V}^* = \widetilde{V}^*(\mathrm{p}_1, \mathrm{v}_1, \ldots, \mathrm{p}_{a-1}, \mathrm{v}_{a-1})$ denote the "residual" strategy of V^* when messages $\langle \mathrm{p}_1, \mathrm{v}_1, \ldots, \mathrm{p}_a \rangle$ are fixed (i.e., $\widetilde{V}^*(G, r) \stackrel{\text{def}}{=} V^*(G, r; \mathrm{p}_1, \mathrm{v}_1, \ldots, \mathrm{p}_{a-1}, \mathrm{v}_{a-1})$), let the phrase "$S$ rewinds in Step (4)" represent the event in which the three rewinding conditions from above hold, and let $\zeta_{i,a,b}$ be as above (in other words, the probability with which the "S rewinds in Step (4)" event holds). We then have:

$$
\begin{aligned}
&\Pr_r\left[S^{\widetilde{V}^*}(G, C) = \perp\right] \\
&= \Pr_r\left[S^{\widetilde{V}^*}(G, C) = \perp \;\middle|\; S \text{ rewinds in Step (4)}\right] \cdot \Pr_r\left[S \text{ rewinds in Step (4)}\right] \quad (2) \\
&= \Pr_r\left[S^{\widetilde{V}^*}(G, C) = \perp \;\middle|\; S \text{ rewinds in Step (4)}\right] \cdot \zeta_{i,a,b} \\
&= \Pr_r\left[\mathrm{p} = \mathrm{p}_t\right] \cdot \zeta_{i,a,b} \quad (3)
\end{aligned}
$$

Now, since p_a and p'_a are uniformly and independently chosen in $\{0,1\}^k$, and since the number of $r \in \{0,1\}^k$ for which $\widetilde{V}^*(G,r)$ is not equal to ABORT is precisely $2^k \cdot \zeta_{i,a,b}$, then it holds that $\Pr[\mathrm{p}_a = \mathrm{p}'_a] = 1/(2^k \cdot \zeta_{i,a,b})$. Using Eq. 3 we infer that:

$$\Pr_r\left[S^{\widetilde{V}^*}(G,C) = \perp\right] \;=\; \frac{1}{2^k \cdot \zeta_{i,a,b}} \cdot \zeta_{i,a,b} \;=\; \frac{1}{2^k}$$

as required.

Empirical Study. Here we provide some cursory evidence that the type of adversarial nesting which causes problems with concurrent simulation do not generally occur when verifiers are independently sending their protocol messages without delaying.

We performed a cursory empirical study of the webserver traffic at our University webserver. We analyzed roughly 122681 TCP sessions (syn-to-fin flows) served by our department webserver over a period of 16 hours; each session consisted of a SYN from a to our webserver, a SYN from the webserver to a, a FIN from a to the webserver, and a final FIN from the webserver to a such that the entire flow corresponded to a request and an error message response served by the webserver. We considered error messages because they are not input/output bound and therefore require roughly the same server processing time. The (4-flow) message pattern corresponds to a 1-slot preamble for our ZK protocol. From this experiment, we counted 26579 nested sessions. In other words, roughly 79% of the sessions were message-free, and would therefore only require 1 slot in our simplest optimistic protocol. (Of the remaining 21%, we cannot determine whether they would have required a second slot given the data set.) Moreover, this small data set reflected a high level of concurrency: there were 57161 instances when one session overlapped another session.

Acknowledgements. We thank the anonymous reviewers and Vinod Vaikuntanathan for helpful comments concerning our definition of security, and in particular about the possibility of our protocols leaking information about the presence of other concurrent sessions (as discussed in the beginning of Section 2).

References

[ASW98] Asokan, N., Shoup, V., Waidner, M.: Optimistic fair exchange of digital signatures. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 591–606. Springer, Heidelberg (1998)

[Bar01] Barak, B.: How to go beyond the black-box simulation barrier. In: Proc. 42nd IEEE Symposium on Foundations of Computer Science (FOCS), pp. 106–115 (2001)

[Blu86] Blum, M.: How to prove a theorem so no one can claim it. In: Proc. of The International Congress of Mathematicians, pp. 1444–1451 (1986)

[Can06] Canetti, R.: Security and composition of cryptographic protocols: A tutorial. Cryptology ePrint Archive, Report 2006/465 (2006)

[CGGM00] Canetti, R., Goldreich, O., Goldwasser, S., Micali, S.: Resettable zeroknowledge. In: Proc. 32nd Annual ACM Symposium on Theory of Computing (STOC), pp. 235–244. ACM Press, New York (2000)

[CKP01] Cohen, T., Kilian, J., Petrank, E.: Responsive round complexity and concurrent zero-knowledge. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 422–441. Springer, Heidelberg (2001)

[CKPR01] Canetti, R., Kilian, J., Petrank, E., Rosen, A.: Black-box concurrent zeroknowledge requires omega(log n) rounds. In: STOC 2001, pp. 570–579 (2001)

[CKPR02] Canetti, R., Kilian, J., Petrank, E., Rosen, A.: Black-box concurrent zero-knowledge requires (almost) logarithmically many rounds. SIAM J. Comput. 32(1), 1–47 (2002)

[Dam99] Damgard, I.: Concurrent zero-knowledge is easy in practice. Available online at Theory of Cryptography Library (June 1999)

[DNS98] Dwork, C., Naor, M., Sahai, A.: Concurrent zero knowledge. In: Proc. 30th Annual ACM Symposium on Theory of Computing, STOC (1998)

[DPP93] Damgäard, I., Pedersen, T., Pfitzmann, B.: On the existence of statistically hiding bit commitment schemes and fail-stop signatures. In: Crypto 1993, pp. 250–265 (1993)

[DS98] Dwork, C., Sahai, A.: Concurrrent zero-knowledge: Reducing the need for timing constraints. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 105–120. Springer, Heidelberg (1998)

[Gol02] Goldreich, O.: Concurrent zero-knowledge with timing, revisited. In: STOC 2002, pp. 332–340 (2002)

[KP01] Kilian, J., Petrank, E.: Concurrent and resettable zero-knowledge in polylogarithm rounds. In: Proc. 33rd Annual ACM Symposium on Theory of Computing (STOC), pp. 560–569 (2001)

[KPR98] Kilian, J., Petrank, E., Rackoff, C.: Lower bounds for zero knowledge on the internet. In: FOCS 1998, pp. 484–492. IEEE, Los Alamitos (1998)

[Lam05] Lamport, L.: Fast paxos. Technical Report MSR-TR-2005-112, Microsoft Research (July 2005)

[MP03] Micciancio, D., Petrank, E.: Simulatable commitments and efficient concurrent zero-knowledge. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 140–159. Springer, Heidelberg (2003)

[NY89] Naor, M., Yung, M.: Universal one-way hash functions and their cryptographic applications. In: STOC 1989, pp. 33–43 (1989)

[Pas03] 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)

[PPS+08] Pandey, O., Pass, R., Sahai, A., Tseng, W.-L.D., Venkitasubramaniam, M.: Precise concurrent zero knowledge. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 397–414. Springer, Heidelberg (2008)

[PRS02] Prabhakaran, M., Rosen, A., Sahai, A.: Concurrent zero-knowledge with logarithmic round complexity. In: FOCS 2002, pp. 366–375 (2002)

[PS04] Prabhakaran, M., Sahai, A.: New notions of security: achieving universal composability without trusted setup. In: Symposium on Theory of Computing (STOC), pp. 242–251 (2004)

[PTV10] Pass, R., Tseng, W.-L.D., Venkitasubramaniam, M.: Eye for an eye: Efficient concurrent zero-knowledge in the timing model. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 518–534. Springer, Heidelberg (2010)

[PV05] Persiano, G., Visconti, I.: Single-prover concurrent zero knowledge in almost constant rounds. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 228–240. Springer, Heidelberg (2005)

[PV08] Pass, R., Venkitasubramaniam, M.: On constant-round concurrent zero-knowledge. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 553–570. Springer, Heidelberg (2008)

[RK99] Richardson, R., Kilian, J.: On the concurrent composition of zero-knowledge proofs. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 311–326. Springer, Heidelberg (1999)

[Ros00] Rosen, A.: A note on the round complexity of concurrent zero-knowledge. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 451–468. Springer, Heidelberg (2000)

[Ros04] Rosen, A.: A note on constant round zero knowledge proofs for np. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 191–202. Springer, Heidelberg (2004)

[Ros06] Rosen, A.: Concurrent Zero-Knowledge. Series on Information Security and Cryptography. Springer, Heidelberg (2006)

A Comparison with Other Approaches

Consider an alternative Prover strategy that we denote *reset-on-nesting*:

> For some fixed constant C, send a "reset" message to any protocol sessions that have more than C nested sessions that begin and end within a slot. When a slot is reset, the verifier starts the protocol from the beginning.

The security proof for schedules with an upper-bounded number of nestings is straightforward. Moreover, only one slot is needed in this "reset-on-nesting" strategy for the security proof. Unfortunately, the reset-on-nesting idea has two major problems. First is an issue of completeness: it is possible for an honest Prover, and an honest *but very slow* Verifier to repeatedly fail in successfully completing a protocol.

Definition 3 (Completeness). *A concurrent protocol $\Pi = (P_1, \ldots, P_n)$ is* complete*, when for any schedule of concurrently executing sessions, and for every execution between honest parties $P_1, \ldots, P_n$, every P_i eventually* HALT*s and outputs $(1, z)$ (to indicate success).*

A second more troubling problem is one of *intentional starvation*: a malicious Prover may indefinitely postpone a proof by claiming the session has become too nested. An honest verifier has no way to audit the schedule of messages received by the Prover, and thus no recourse but to restart the protocol (which may fail again for the same reason). Even with auditing, the malicious prover may create a fictitious verifier instance and intentionally schedule this verifier so as to create nested sessions in the the honest verifier's slots. Thus, even an "honestly recorded" transcript of all of the Prover's messages could be justifiably used to starve the honest verifier.

Accountable Aborting versus fail. To be sure, a malicious prover may ABORT a protocol for many reasons; but this event is fundamentally different than the postponement attack discussed above: An ABORT is an admission of guilt by the malicious prover; a postponement attack is an accusation by the Prover of malice on the part of the Verifier!

Borrowing terminology from the distributed algorithms community, we state the concept of starvation-free protocols below. As mentioned, the solution in this paper is a starvation-free protocol, while the reset-on-nesting protocol is not.

Definition 4 (Starvation-Free Protocol). *A starvation-free concurrent protocol $\Pi = (P_1, \ldots, P_n)$ is one that guarantees that for any adversary P_i^*, and for any schedule of*

messages of concurrently executing sessions, every honest party $P_j, j \neq i$ interacting with P_i^* eventually HALT*s with output* $(1, z)$ *or* ABORT*s with output* $(0, z)$.

(Note, that the z is arbitrary protocol-specific output, i.e., it could be $f(x, y)$ in the case of two-party secure function evaluation.)

A reset-on-nesting protocol cannot be both complete and starvation-free. Either the protocol requires the Verifier to tolerate an infinite number of resets (in order to satisfy completeness)—in which case it is not starvation free—or it requires the Verifier to upper-bound the number of messages it tolerates before ABORT and output 0 (in order to satisfy starvation free-ness)—in which case it is not complete.

For this reason, we prefer our optimistic model to the reset-on-nesting protocol.

A.1 Comparison with the Timing Model

The timing model adds a notion of time on the standard communication model by (a) giving each party a *local* clock, (b) having all parties share a global bound $\rho \geq 1$ on the relative rates of the different clocks (i.e., clock drift), and (c) having all parties share a global bound Δ on the message-delivery time (which includes the time for local computation to receive and prepare messages). Protocols in the timing model can TIMEOUT messages that have not arrived in time Δ, and DELAY outgoing messages by a delay period that is also at least as big as Δ.

Prior work [DNS98, Gol02, PTV10] in this model employ the TIMEOUT and DELAY operations. Protocols in this model have two disadvantages: first, every protocol execution is forced to run for worst-case time $c \cdot \Delta$ even if the parties involved can communicate quickly. Transmission delays to some parts of the internet can be measured in fractions of a day, and so for completeness, Δ would have to be reasonably large. Whereas our protocol allows fast participants to complete interactions "as fast as the network allows," the timing protocols of [DNS98, Gol02] require all sessions to run in time related to *worst-case* network delays. Conceptually, our protocol handles more diverse schedules, whereas the timing model protocols use timing to ensure "roughly parallel" composition.

The work of [PTV10] reduces the required delay so some small constant $c < 1$. This protocol is major practical improvement to the timing model; however, it too must delay the verifier by some multiplicative penalty of the time it takes for the verifier to respond, and it requires 3 slots. For example, every session must run at least twice as slow (their penalty function is a parameter and can be $\omega(1)$ in some cases also) "as the network allows" and each verifier must still complete multiple slots (whereas in optimistic cases, only 1 slot is required).

New problems of Accountability. Unfortunately, any setting of Δ introduces the second more subtle problem with TIMEOUT: much like intentional starvation discussed above, a malicious Prover can send a TIMEOUT to a Verifier to avoid having to abort a session that it cannot complete. The verifier has no way to "contest" this timeout. As we will argue, such a use of TIMEOUTs introduces a new way for a malicious prover to cheat that is not possible in the standard model.

The basis for this problem is that clocks in the timing model must be *local* and unauthenticatable to allow the simulator to rewinding the verifier (or prover in the case of a

proof of knowledge). If a local clock can be authenticated, then a malicious verifier V^* algorithm could refuse to answer any message that is too old according to its clock and this would eliminate the possibility of rewinding. As a result, local clock timestamps that appear in transcripts or communication tapes can be forged by any party; it is not possible for a third party to verify such a timestamp.

This leads to the problem that transcripts that arise from the following two cases are indistinguishable which removes any *accountability for aborting*:

1. A malicious Prover algorithm receives a message from an honest Verifier, waits for time $\Delta + \epsilon$, and then sends TIMEOUT.
2. A malicious Verifier algorithm delays sending a message for time $\Delta + \epsilon$, and then sends its message. The honest Prover, consequently sends a TIMEOUT messsage.

Let us compare this situation to the standard model in which—say—messages can be authenticated. (Notice that messages *can* be authenticated and still allow rewinding.) Of course, a malicious prover can always abort a protocol by either sending an incorrect message or refusing to send any message. Both cases, however, are fundamentally different then the ability for a malicious prover to send a TIMEOUT.

When the Prover sends a bogus message, the verifier has proof (via the authenticated message) that the Prover *cannot* supply a proof of the statement, and the prover is therefore accountable for the abort. In fact, the second case is the same. As described by Canetti [Can06], "not sending a message" in the standard model is handled by an explicit halt which is proof that the Prover has failed.

> Protocol participants are modeled as strict polynomial-time interactive Turing machines; one machine sends a message to another by writing on the recipient's "communication tape." Message delivery is not guaranteed. However, when one machine executes a HALT operation, the other party in a protocol execution is informed of the HALT via this communication tape. This modeling guarantees that one party is not inadvertently left waiting for a message that will never arrive.[3] It is important to note that the standard model does not have any notion of "time" except for steps of computation.

Thus, the standard model makes it possible to determine which of the two parties cheated in an interaction. In contrast, the timing model with TIMEOUTS allows a malicious Prover to be unaccountable for its cheating. While we acknowledge that such a difference may be purely theoretical, it is nonetheless conceptually troubling.

Comparison with Responsive Round Complexity. Cohen, Kilian and Petrank propose the notion of responsive round complexity [CKP01]. A protocol is said to have *responsive round complexity* m with party A if it can guarantee that if A responds to every message of the protocol in at most time t, then the overall communication delay of the protocol execution is $m \cdot t$. The idea behind the protocol in that paper is the following:

[3] In particular, this mechanism is how a malicious party that does not send a message is modeled—since the party must be strict polynomial-time, if it refuses to send a message, we assume it runs a computation, eventually HALTS and then the recipient learns that the other party has aborted.

Every verifier is assigned a time bin T. If a verifier V delays a message by time $t < T$, then the prover delays the response to V by time $2T$. If a verifier delays a message by time $t > T$, then the verifier is moved into time bin $2T$, and the verifier must restart the protocol from the beginning.

Slow verifiers are penalized. In particular, this Prover strategy clumps verifiers into "time buckets" such that all verifiers in the same bucket act in a roughly parallel manner. The analysis then used in the timing model work can be applied.

Overall, the goal of this notion is to assure that a party that always responds quickly has a stronger guarantee on the communication time of the overall protocol. Similarly, our work also attempts to improve the communication time for Verifiers that respond quickly. However, the protocols in [CKP01] still have at least $\omega(\log n)$ slots in the best case when the verifiers respond quickly. (In other words, their protocol guarantees response round complexity of $O(\log n)$ whereas our protocol can use only 1 slot when the Verifier responds quickly.)

Buffering Sessions. Another idea is for the Prover to buffer sessions so that each one starts only after the previous session finishes. Buffering sessions, i.e. serializing them, eliminates the benefits of having multiple sessions run safely at the same time.

Denial of Service. A malicious verifier can "force" the protocol to require just as many rounds as the current best fixed-round $c\mathcal{ZK}$ protocol. This is not a denial-of-service attack because the malicious verifier can only force the same round complexity that the best current protocols achieve—thus, the optimistic approach is never worse than PRS. Moreover, in every additional round forced by a bad schedule, V^* is required to communicate and compute more than the Prover.

Handling Server Farms. Our optimistic approach requires the Prover to know the global schedule of Verifier messages. Very large systems, however, are usually built on *clusters* of servers instead of a single machine. Our optimistic approach can be made to work on clusters using a consensus protocol to share schedules among the servers. Since all servers belong to the same entity and are connected through internal fast links, the consensus protocol would work "in the best case" (as opposed to the Byzantine case) for most sessions. In other words, our protocol is viable even after counting the overhead to make all prover machines agree on a schedule of verifier requests. To be sure, many very large systems in existence today require even more complicated consensus on the order of requests that they serve. For example, consider distributed database systems (sometimes distributed over tens of thousands of machines), social network sites, and some distributed file systems that are implemented across thousands of machines.

Faster Fully Homomorphic Encryption

Damien Stehlé[1] and Ron Steinfeld[2]

[1] CNRS, Laboratoire LIP (U. Lyon, CNRS, ENS de Lyon, INRIA, UCBL),
46 Allée d'Italie, 69364 Lyon Cedex 07, France
damien.stehle@gmail.com
http://perso.ens-lyon.fr/damien.stehle
[2] Centre for Advanced Computing - Algorithms and Cryptography,
Department of Computing, Macquarie University, NSW 2109, Australia
ron.steinfeld@mq.edu.au
http://www.ics.mq.edu.au/~rons/

Abstract. We describe two improvements to Gentry's fully homomorphic scheme based on ideal lattices and its analysis: we provide a more aggressive analysis of one of the hardness assumptions (the one related to the Sparse Subset Sum Problem) and we introduce a probabilistic decryption algorithm that can be implemented with an algebraic circuit of low multiplicative degree. Combined together, these improvements lead to a faster fully homomorphic scheme, with a $\widetilde{O}(\lambda^{3.5})$ bit complexity per elementary binary add/mult gate, where λ is the security parameter. These improvements also apply to the fully homomorphic schemes of Smart and Vercauteren [PKC'2010] and van Dijk et al. [Eurocrypt'2010].

Keywords: fully homomorphic encryption, ideal lattices, SSSP.

1 Introduction

A homomorphic encryption scheme allows any party to publicly transform a collection of ciphertexts for some plaintexts $\pi_1, \ldots, \pi_n$ into a ciphertext for some function/circuit $f(\pi_1, \ldots, \pi_n)$ of the plaintexts, without the party knowing the plaintexts themselves. Such schemes are well known to be useful for constructing privacy-preserving protocols, for example as required in 'cloud computing' applications: a user can store encrypted data on a server, and allow the server to process the encrypted data without revealing the data to the server. For over 30 years, all known homomorphic encryption schemes supported only a limited set of functions f, which restricted their applicability. The theoretical problem of constructing a *fully* homomorphic encryption scheme supporting arbitrary functions f, was only recently solved by the breakthrough work of Gentry [9]. More recently, two further fully homomorphic schemes were presented [26,5], following Gentry's framework. The underlying tool behind all these schemes is the use of *Euclidean lattices*, which have previously proved powerful for devising many cryptographic primitives (see, e.g., [21] for a recent survey).

A central aspect of Gentry's fully homomorphic scheme (and the subsequent schemes) is the *ciphertext refreshing* `Recrypt` operation. The ciphertexts in

M. Abe (Ed.): ASIACRYPT 2010, LNCS 6477, pp. 377–394, 2010.

Gentry's scheme contain a random 'noise' component that grows in size as the ciphertext is processed to homomorphically evaluate a function f on its plaintext. Once the noise size in the ciphertext exceeds a certain threshold, the ciphertext can no longer be decrypted correctly. This limits the number of homomorphic operations that can be performed. To get around this limitation, the `Recrypt` operation allows to 'refresh' a ciphertext, i.e., given a ciphertext ψ for some plaintext π, to compute a new ciphertext ψ' for π (possibly for a different key), but such that the size of the noise in ψ' is smaller than the size of the noise in ψ. By periodically refreshing the ciphertext (e.g., after computing each gate in f), one can then evaluate arbitrarily large circuits f.

The `Recrypt` operation is implemented by evaluating the decryption circuit of the encryption scheme homomorphically, given 'fresh' (low noise) ciphertexts for the bits of the ciphertext to be refreshed and the scheme's secret key. This homomorphic computation of the decryption circuit must of course be possible *without* any ciphertext refreshing, a condition referred to as *bootstrappability*. Thus, the complexity (in particular circuit depth, or multiplicative degree) of the scheme's decryption circuit is of fundamental importance to the feasibility and complexity of the fully homomorphic scheme. Unfortunately, the relatively high complexity of the decryption circuit in the schemes [9,26,5], together with the tension between the bootstrappability condition and the security of the underlying hard problems, implies the need for large parameters and leads to resulting encryption schemes of high bit-complexity.

Our Contributions. We present improvements to Gentry's fully homomorphic scheme [9] and its analysis, that reduce its complexity. Overall, letting λ be the security parameter (i.e., all known attacks against the scheme take time $\geq 2^{\lambda}$), we obtain a $\widetilde{O}(\lambda^{3.5})$ bit complexity for refreshing a ciphertext corresponding to a 1-bit plaintext. This is the cost per gate of the fully homomorphic scheme. To compare with, Gentry [8, Ch. 12] claims a $\widetilde{O}(\lambda^{6})$ bound, although the proof is incomplete[1].

The improved complexity stems from two sources. First, we give a more aggressive security analysis of the Sparse Subset Sum Problem (SSSP) against lattice attacks, compared to the analysis given in [9]. The SSSP, along with the Ideal lattice Bounded Distance Decoding (BDD) problem, are the two hard problems underlying the security of Gentry's fully homomorphic scheme. In his security analysis of BDD, Gentry uses the best known complexity bound for the approximate shortest vector problem (SVP) in lattices, but in analyzing SSSP, Gentry assumes the availability of an exact SVP oracle. Our new finer analysis of SSSP takes into account the complexity of approximate SVP, making it more consistent with the assumption underlying the analysis of the BDD problem, and leads to smaller parameter choices. Second, we relax the definition of fully homomorphic encryption to allow for a negligible but non-zero probability

[1] This bound is claimed to hold for the scheme after Optimizations 1 and 2 of [8, Se. 12.3], but the analysis does not include the cost of the ciphertext expansion nor details which decryption circuit is applied homomorphically. For instance, the decryption circuit from [5, Le. 6.3] is too costly to derive the bound. These gaps can be filled using Section 6.2 of the present article, and the bound $\widetilde{O}(\lambda^{6})$ indeed holds.

of decryption error. We then show that, thanks to the randomness underlying Gentry's 'SplitKey' key generation for his squashed decryption algorithm (i.e., the decryption algorithm of the bootstrappable scheme), if one allows a negligible decryption error probability, then the rounding precision used in representing the ciphertext components can be roughly halved, compared to the precision in [9] which guarantees zero error probability. The reduced ciphertext precision allows us to decrease the degree of the decryption circuit. We concentrate on Gentry's scheme [9], but our improvements apply equally well to the other related schemes [26,5].

NOTATION. Vectors will be denoted in bold. If $\boldsymbol{x} \in \mathbb{R}^n$, then $\|\boldsymbol{x}\|$ denotes the Euclidean norm of $\boldsymbol{x}$. We make use of the Landau notations $O(\cdot)$, $\widetilde{O}(\cdot)$, $\omega(\cdot)$, $\Omega(\cdot)$, $\widetilde{\Omega}(\cdot)$, $\Theta(\cdot)$, $\widetilde{\Theta}(\cdot)$. If n grows to infinity, we say that a function $f(n)$ is negligible if it is asymptotically $\leq n^{-c}$ for any $c > 0$. If X is a random variable, $E[X]$ denotes its mean and $\Pr[X = x]$ denotes the probability of the event "$X = x$". We say that a sequence of events E_n holds with overwhelming probability if $\Pr[\neg E_n] \leq f(n)$ for a negligible function f. We will use the following variant of the Hoeffding bound [13].

Lemma 1.1. *Let $X_1, \ldots, X_t$ denote independent random variables with mean μ, where $X_i \in [a_i, b_i]$ for some $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^t$. Let $X = \sum_i X_i$. Then:*

$$\forall k \geq 0 : \ \Pr\left[|X - t\mu| \geq k\right] \leq 2 \cdot \exp(-2k^2/\|\boldsymbol{b} - \boldsymbol{a}\|^2).$$

Remark. Due to space limitations, some contents of the article are only given in the appendices of the full version, which is available on the authors' webpages. These include: a sketch of Gentry's bootstrapping transformation [9], adapted to handle decryption errors; a proof that an ideal sampled from Gentry's distribution [11] is of prime determinant with overwhelming probability, when the considered ring is $\mathbb{Z}[x]/(x^{2^k} + 1)$; the proofs of Lemmata 3.2 and 3.3; and the application of our improvements to other fully homomorphic encryption schemes.

2 Reminders

For a detailed introduction to the computational aspects of lattices, we refer to [20]. The article [10] provides an intuitive description of Gentry's fully homomorphic scheme.

2.1 Euclidean Lattices

An n-dimensional lattice L is the set of all integer linear combinations of some linearly independent vectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n \in \mathbb{Z}^n$, i.e., $L = \sum \mathbb{Z}\boldsymbol{b}_i$. The $\boldsymbol{b}_i$'s are called a basis of L. A basis $B = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n) \in \mathbb{Z}^{n \times n}$ is said to be in Hermite Normal Form (HNF) if $b_{i,j} = 0$ for $i > j$ and $0 \leq b_{i,j} < b_{i,i}$ otherwise. The HNF of a lattice is unique and can be computed in polynomial time given any basis, which arguably makes it a worst-case basis [19]. To a basis $B = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n) \in \mathbb{Z}^{n \times n}$ for lattice L, we associate the fundamental parallelepiped $\mathcal{P}(B) = \{\boldsymbol{v} = \sum_i y_i \cdot \boldsymbol{b}_i :$

$y_i \in (-1/2, 1/2]\}$. For a vector $\boldsymbol{v} \in \mathbb{R}^n$, we denote by $\boldsymbol{v} \bmod B$ the unique vector $\boldsymbol{v}' \in \mathcal{P}(B)$ such that $\boldsymbol{v} - \boldsymbol{v}' \in L$. Note that $\boldsymbol{v}' = \boldsymbol{v} - B\lfloor B^{-1}\boldsymbol{v} \rceil$, where $\lfloor \cdot \rceil$ rounds the coefficients to the nearest integers (upwards in case of a real that is equally distant to two consecutive integers).

The minimum $\lambda_1(L)$ is the norm of any shortest non-zero vector in L. More generally, the ith minimum $\lambda_i(L)$ is the radius of the smallest ball containing i linearly independent lattice vectors. We define the lattice *amplitude* as the ratio $\lambda_n(L)/\lambda_1(L)$. We now define two parametrized families of algorithmic problems that are central for Euclidean lattices. Let $\gamma \geq 1$ be a function of the dimension. The γ-SVP (for Shortest Vector Problem) computational problem consists in finding a vector $\boldsymbol{b} \in L$ such that $0 < \|\boldsymbol{b}\| \leq \gamma\lambda_1(L)$, given as input an arbitrary basis for L. The γ-BDD (for Bounded Distance Decoding) computational problem consists in finding a vector $\boldsymbol{b} \in L$ closest to $\boldsymbol{t}$ given as inputs an arbitrary basis for L and a target vector $\boldsymbol{t}$ whose distance to L is $\leq \gamma^{-1}\lambda_1(L)$. Solving γ-SVP and γ-BDD are in general computationally hard problems. The best algorithms for solving them for $\gamma = 1$ ([14,22]) run in time exponential with respect to the dimension. On the other hand, the smallest γ one can achieve in polynomial time is exponential, up to poly-logarithmic factors in the exponent ([17,24,1]). For intermediate γ, the best strategy is the hierarchical reduction of [24], and leads to the following conjecture.

Lattice 'Rule of Thumb' Conjecture. There exist absolute constants $c_1, c_2 > 1$ such that for any λ and any dimension n, for any n-dimensional lattice with amplitude $\leq \gamma/c_2$, one cannot solve γ-SVP (resp. γ-BDD) in time smaller than 2^λ, with $\gamma = c_1^{n/\lambda}$.

Let us discuss the conjecture. One often considers the lattice gap $\frac{\lambda_2}{\lambda_1}$. If $\frac{\lambda_2}{\lambda_1} > \gamma$, then γ-SVP is equivalent to γ'-SVP for any $\gamma' < \frac{\lambda_2}{\lambda_1}$: a γ'-SVP solver is guaranteed to output a multiple of a shortest vector, from which solving SVP is easy. Similarly, if $\frac{\lambda_2}{\lambda_1} = O(1)$ but $\frac{\lambda_3}{\lambda_2} > \gamma$, then lattice reduction will return a basis whose first two vectors span a sublattice containing vectors reaching λ_1 and λ_2: SVP can then be solved by 2-dimensional reduction. This explains why we consider $\frac{\lambda_n}{\lambda_1}$ rather than the more standard $\frac{\lambda_2}{\lambda_1}$. Note that for most common lattices, there is no a priori reason to expect λ_n to be significantly larger than λ_2. Finally, when $\frac{\lambda_n}{\lambda_1} \leq \gamma$, the complexity of γ-SVP does not seem to depend on $\frac{\lambda_n}{\lambda_1}$. The experimental results in [7] seem to be consistent with this conjecture.

Algorithmic improvements have been proposed (e.g., [6,16]), but they have only led to better constants, without changing the overall framework. The conjecture seems to hold even if one considers quantum computers [18]. We will consider it for two families of lattices: no algorithm is known to perform non-negligibly better for them than for general lattices.

For a lattice L, we define $\det L$ as $|\det B|$ for any basis B. Minkowski's theorem provides a link between the minimum and the determinant.

Theorem 2.1 ([4, III.2.2]). *Let L be an n-dimensional lattice and V be a compact convex set that is symmetric about the origin. Let $m \geq 1$ be an integer. If $\mathrm{vol}(V) \geq m2^n \det(L)$, then V contains at least m non-zero pairs of points $\pm\mathbf{b}$ of L.*

2.2 Ideal Lattices

Let $f \in \mathbb{Z}[x]$ a monic degree n irreducible polynomial. Let R denote the polynomial ring $\mathbb{Z}[x]/f$. Let I be an (integral) ideal of R, i.e., a subset of R that is closed under addition, and multiplication by arbitrary elements of R. By mapping polynomials to the vectors of their coefficients, we see that the ideal I corresponds to a sublattice of $\mathbb{Z}^n$: we can thus view I as both a lattice and an ideal. An *ideal lattice* for f is a sublattice of $\mathbb{Z}^n$ that corresponds to an ideal $I \subseteq \mathbb{Z}[x]/f$. In the following, an ideal lattice will implicitly refer to an f-ideal lattice. For $v \in R$ we denote by $\|v\|$ its Euclidean norm (as a vector). We define a multiplicative expansion factor $\gamma_\times(R)$ for the ring R by $\gamma_\times(R) = \max_{u,v \in R} \frac{\|u \times v\|}{\|u\| \cdot \|v\|}$. A typical choice is $f = x^n + 1$ with n a power of 2, for which $\gamma_\times(R) = \sqrt{n}$ (see [9, Th. 9]).

Two ideals I and J of R are said coprime if $I + J = R$, where $I + J = \{i + j : i \in I, j \in J\}$. An ideal I is said prime of degree 1 if $\det(I)$ is prime. For an ideal J of R, we define $J^{-1} = \{\boldsymbol{v} \in \mathbb{Q}[x]/f : \forall \boldsymbol{u} \in J, \boldsymbol{u} \times \boldsymbol{v} \in R\}$. This is a fractional ideal of R, and $J^{-1} \subseteq \frac{1}{\det J} R$ (since $(\det J) \cdot R \subseteq J$). If $f = x^n + 1$ with n a power of 2, then R is the ring of integers of the $(2n)$th cyclotomic field and $J^{-1} \times J = R$ for any integral ideal J (the product of two ideals I_1 and I_2 being the ideal generated by all products $i_1 \cdot i_2$ with $i_1 \in I_1$ and $i_2 \in I_2$). An ideal I is said principal if it is generated by a single element $r \in I$, and then we write $I = (r)$. We define $\mathrm{rot}_f(r) \in \mathbb{Q}^{n \times n}$ as the basis of I consisting of the $x^k r(x) \bmod f$'s, for $k \in [0, n-1]$.

If I is an ideal lattice for $f = x^n + 1$, then we have $\lambda_1(I) \geq \det(I)^{1/n}$: an easy way to prove it is to notice that the rotations $x^k v$ of any shortest non-zero vector v form a basis of a full-rank sublattice of I, and to use the inequalities $\lambda_1(I)^n = \prod_k \|x^k v\| \geq \det((v)) \geq \det I$.

2.3 Homomorphic Encryption

In this section, we review definitions related to homomorphic encryption. Our definitions are based on [9,8], but we slightly relax the definition of decryption correctness, to allow a negligible probability of error. This is crucial for our probabilistic improvement to Gentry's `Recrypt` algorithm.

Definition 2.1. *A homomorphic encryption scheme* Hom *consists of four algorithms:*

- KeyGen*: Given security parameter λ, returns a secret key sk and a public key pk.*
- Enc*: Given plaintext $\pi \in \{0,1\}$ and public key pk, returns ciphertext ψ.*
- Dec*: Given ciphertext ψ and secret key sk, returns plaintext π.*
- Eval*: Given public key pk, a t-input circuit C (consisting of addition and multiplication gates modulo 2), and a tuple of ciphertexts $(\psi_1, \ldots, \psi_t)$ (corresponding to the t input bits of C), returns a ciphertext ψ (corresponding to the output bit of C).*

Hom *is said* correct *for a family* $\mathcal{C}$ *of circuits with* $\leq t = \mathcal{P}oly(\lambda)$ *input bits if for any* $C \in \mathcal{C}$ *and input bits* $(\pi_i)_{i \leq t}$*, the following holds with overwhelming probability over the randomness of* KeyGen *and* Enc*:*

$$\mathsf{Dec}(sk, \mathsf{Eval}(pk, C, (\psi_1, \ldots, \psi_t))) = C(\pi_1, \ldots, \pi_t),$$

where $(sk, pk) = \mathsf{KeyGen}(\lambda)$ *and* $\psi_i = \mathsf{Enc}(pk, \pi_i)$ *for* $i = 1, \ldots, t$.

Hom *is said* compact *if for any circuit* C *with* $\leq t = \mathcal{P}oly(\lambda)$ *input bits, the bit-size of the ciphertext* $\mathsf{Eval}(pk, C, (\psi_1, \ldots, \psi_t))$ *is bounded by a fixed polynomial* $b(\lambda)$.

Gentry [9] defined the powerful notion of a *bootstrappable* homomorphic encryption scheme: one that can homomorphically evaluate a decryption of two ciphertexts followed by one gate applied to the decrypted values. We also relax this notion to allow decryption errors.

Definition 2.2. *Let* $\mathsf{Hom} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ *denote a homomorphic encryption scheme. We define two circuits:*

- $\mathsf{Dec\text{-}Add}$*: Takes as inputs a secret key* sk *and two ciphertexts* ψ_1, ψ_2*, and computes* $\mathsf{Dec}(sk, \psi_1) + \mathsf{Dec}(sk, \psi_2) \bmod 2$.
- $\mathsf{Dec\text{-}Mult}$*: Takes as inputs a secret key* sk *and two ciphertexts* ψ_1, ψ_2*, and computes* $\mathsf{Dec}(sk, \psi_1) \times \mathsf{Dec}(sk, \psi_2) \bmod 2$.

Hom *is said* bootstrappable *if it is correct for* $\mathsf{Dec\text{-}Add}$ *and* $\mathsf{Dec\text{-}Mult}$.

Gentry discovered that a bootstrappable homomorphic encryption can be used to homomorphically evaluate arbitrary circuits. More precisely, he proved the following result (adapted to allow for decryption error). The construction is sketched in the full version.

Theorem 2.2 ([9, Se. 2]). *Given a bootstrappable homomorphic encryption scheme* Hom*, and parameter* $d = \mathcal{P}oly(\lambda)$*, it is possible to construct another homomorphic encryption scheme* $\mathsf{Hom}^{(d)}$ *that is compact and correct for all circuits of size* $\mathcal{P}oly(\lambda)$*. Furthermore, if the scheme* Hom *is semantically secure, then so is the scheme* $\mathsf{Hom}^{(d)}$.

3 Summary of Gentry's Fully Homomorphic Scheme

We now review Gentry's fully homomorphic encryption scheme [9,8].

3.1 The Somewhat Homomorphic Scheme

We first recall Gentry's somewhat homomorphic encryption scheme (see [8, Se. 5.2 and Ch. 7]) which supports a limited number of multiplications. It is the basis for the bootstrappable scheme presented in Subsection 3.3. The scheme, described in Figure 1, produces ciphertexts in the ring $R = \mathbb{Z}[x]/f$ for a suitable

irreducible degree n monic polynomial f. In this paper, we will assume $f = x^n+1$ with n a power of 2. Here n is a function of the security parameter λ.

The key generation procedure generates two coprime ideals I and J of R. The ideal I has basis B_I. To simplify the scheme (and optimize its efficiency), a convenient choice, which we assume in this paper, is to take $I = (2)$: Reduction of v modulo I corresponds to reducing the coefficients of the vector/polynomial v modulo 2. The ideal J is generated by an algorithm $\mathsf{IdealGen}$, that given (λ, n), generates a 'good' secret basis B_J^{sk} (consisting of short, nearly orthogonal vectors) and computes its HNF to obtain a 'bad' public basis B_J^{pk}. Suggestions for concrete implementations of $\mathsf{IdealGen}$ are given in [8, Se. 7.6], [11] and [26]. To obtain the $\widetilde{O}(\lambda^{3.5})$ bit complexity bound, we will assume that J is a degree 1 prime ideal, which is the case with the implementation of [26] and is also the case with probability exponentially close to 1 for the distribution considered in [11] (see full version). Associated with $\mathsf{IdealGen}$ is a parameter r_{Dec}, which is a lower bound on the radius of the largest origin-centered ball which is contained inside $\mathcal{P}(B_J^{sk})$. In all cases we have $r_{Dec} \geq \lambda_1(J)/\mathcal{P}oly(n)$ (see, e.g., [8, Le. 7.6.2]). Using Babai's rounding-off algorithm [1] with B_J^{sk}, the decryptor can recover the point of J closest to any target vector within distance r_{Dec} of J (see [8, Le. 7.6.1]).

- $\mathsf{KeyGen}(\lambda)$**:** Run $\mathsf{IdealGen}(\lambda, n)$ to generate secret/public bases (B_J^{sk}, B_J^{pk}) for ideal J such that $\mathcal{P}(B_J^{sk})$ contains an origin-centered ball of radius $r_{Dec} \approx \lambda_1(J)$. Return public key $pk = B_J^{pk}$ and secret key $sk = B_J^{sk}$.
- $\mathsf{Enc}(pk, \pi)$**:** Given plaintext $\pi \in \{0,1\}$ and public key pk, run $\mathsf{Samp}(I, \pi)$ to get $\pi' \in \pi + I$ with $\|\pi'\| \leq r_{Enc}$. Return ciphertext $\psi = \pi' \bmod B_J^{pk}$.
- $\mathsf{Dec}(sk, \psi)$**:** Given ciphertext ψ and secret key sk, returns $\pi = (\psi \bmod B_J^{sk}) \bmod I$.
- $\mathsf{Eval}(pk, C, (\psi_1, \ldots, \psi_t))$**:** Given public key pk, circuit C and ciphertexts $\psi_1, \ldots, \psi_t$, for each add or multiply gate in C, perform a $+$ or $\times$ operation in R mod B_J^{pk}, respectively, on the corresponding ciphertexts. Return the ciphertext ψ corresponding to the output of C.

Fig. 1. Gentry's Somewhat Homomorphic Encryption Scheme SomHom

The plaintext space is a subset of $\mathcal{P}(I)$, that we assume to be $\{0,1\}$. The encryption algorithm uses a sampling algorithm Samp, which given $(B_I, \boldsymbol{x})$ for a vector $x \in R$, samples a 'short' vector in the coset $x + I$. Concrete implementations of Samp are given in [8, Se. 7.5 and 14.1]. Associated with Samp is a parameter r_{Enc}, which is a (possibly probabilistic) bound on the norms of vectors output by Samp. For both implementations, one can set $r_{Enc} = \mathcal{P}oly(n)$. To encrypt a message π, a sample $\pi + i$ from the coset $\pi + I$ is generated, and the result is reduced modulo the public basis B_J^{pk}: $\psi = \pi + i \bmod B_J^{pk}$. It is assumed that $r_{Enc} < r_{Dec}$. Therefore, by reducing ψ modulo the secret basis B_J^{sk} one can recover $\pi + i$, and then plaintext π can be recovered by reducing modulo B_I.

Homomorphic addition and multiplication of the encrypted plaintexts π_1, π_2 modulo B_I are supported by performing addition and multiplication respectively in the ring R on the corresponding ciphertexts modulo B_J^{pk}. Namely,

for $\psi_1 = \pi_1 + i_1 \bmod B_J^{pk}$, $\psi_2 = \pi_2 + i_2 \bmod B_J^{pk}$ with $i_1, i_2 \in I$, we have $\psi_1 + \psi_2 \bmod B_J^{pk} \in (\pi_1 + \pi_2) + I$ and $\psi_1 \times \psi_2 \bmod B_J^{pk} \in (\pi_1 \times \pi_2) + I \bmod B_J^{pk}$. However, for ensuring correct decryption of these new ciphertexts, we need that $\|(\pi_1 + i_1) + (\pi_2 + i_2)\|, \|(\pi_1 + i_1) \times (\pi_2 + i_2)\| \leq r_{Dec}$. This limits the degree of polynomials that can be evaluated homomorphically. Note that our choice for J implies that a ciphertext reduced modulo B_J^{pk} is simply an integer modulo $\det(J)$ and thus homomorphic evaluations modulo B_J^{pk} reduce to integer arithmetic modulo $\det(J)$ (such as in [26]).

3.2 A Tweaked Somewhat Homomorphic Scheme

Gentry [8, Ch. 8] introduced tweaks to SomHom to simplify the decryption algorithm towards constructing a fully homomorphic scheme. The tweaked scheme SomHom′ differs from the original scheme in the key generation and decryption algorithm, as detailed in Figure 2.

- $\mathsf{KeyGen}'(\lambda)$**:** Run $\mathsf{KeyGen}(\lambda)$ to obtain (B_J^{sk}, B_J^{pk}). From B_J^{sk}, compute a vector $\boldsymbol{v}_J^{sk} \in J^{-1}$ such that $\mathcal{P}(\mathrm{rot}_f(\boldsymbol{v}_J^{sk})^{-1})$ contains a ball of radius $r'_{Dec} = \frac{r_{Dec}}{8\sqrt{2}n^{2.5}}$ (see [8, Le. 8.3.1]). Return public key $pk = B_J^{pk}$ and secret key $sk = B_J^{sk}$.
- $\mathsf{Dec}'(sk, \psi)$**:** Given ciphertext ψ and secret key sk, return $\pi = \psi - \lfloor \boldsymbol{v}_J^{sk} \times \psi \rceil \bmod I$.

Fig. 2. Algorithms of the Tweaked Somewhat Homomorphic Encryption Scheme SomHom′ that differ from those of SomHom

Gentry showed the following on the correctness of Dec′.

Lemma 3.1 (Adapted from [8, Le. 8.3.1 and 8.4.2]). *A ciphertext $\psi = \pi + i \bmod B_J^{pk}$ with $\|\pi + i\| \leq r'_{Dec}$ is correctly decrypted to π by* Dec′*. Moreover, if $\|\pi + i\| \leq r'_{Dec}$, then each coefficient of $\boldsymbol{v}_J^{sk} \times \psi$ is within $1/8$ of an integer.*

Let C be a mod 2 circuit consisting of add and multiply gates with two inputs and one output. We let $g(C)$ denote the generalized circuit obtained from C by replacing the add and multiply gates mod 2 by the $+$ and $\times$ operations of the ring R, respectively. We say that circuit C is *permitted*, if for any set of inputs $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t$ to $g(C)$ with $\|\boldsymbol{x}_k\| \leq r_{Enc}$ for $k = 1, \ldots, t$, we have $\|g(C)(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t)\| \leq r'_{Dec}$. A permitted circuit which is evaluated homomorphically on encryptions of plaintexts $\pi_1, \ldots, \pi_t$ will yield a ciphertext $\psi = g(C)(\pi_1 + i_1, \ldots, \pi_t + i_t) \bmod B_J^{pk}$ that correctly decrypts to $C(\pi_1, \ldots, \pi_t)$, and such that the coefficients of $\boldsymbol{v}_J^{sk} \times \psi$ are within $1/8$ of an integer. As in [5, Le 3.4], we characterize the permitted circuits by the maximal degree of the polynomial evaluated by the circuit. Note that Gentry [9,8] considers the circuit depth, which is less flexible.

Lemma 3.2. *Let C be a mod 2 circuit, and $g(C)$ denote the corresponding generalized circuit over R, evaluating $h \in \mathbb{Z}[x_1, \ldots, x_t]$ of (total) degree d. The circuit C is permitted if $\gamma_\times^{d-1} \|h\|_1 r_{Enc}^d \leq r'_{Dec}$. In particular, assuming that h has coefficients in $\{0, 1\}$, the circuit C is permitted if d satisfies*

$$d \leq \frac{\log r'_{Dec}}{\log(r_{Enc} \cdot \gamma_\times \cdot (t+1))}.$$

Remark. The polynomial h referred to above is the one evaluated by the *generalized* circuit $g(C)$. For arbitrary circuits C mod 2, the polynomial h may differ from the polynomial h' evaluated by the circuit C mod 2; in particular, the polynomial h may have non-binary integer coefficients, and some may be multiples of 2. However, for circuits C for which h has binary coefficients (the condition in the lemma), we have $h = h'$ (this condition on h is also needed, but is not explicitly stated in [5]).

3.3 Gentry's Squashed Bootstrappable Scheme

To make it bootstrappable, Gentry [8, Ch. 10] modified SomHom' by 'squashing' the decryption circuit. He moved some of the decryption computation to the encryption stage, by providing additional information in the public key. This results in the bootstrappable scheme SqHom described in Figure 3. The scheme introduces three new integer parameters $(p, \gamma_{set}, \gamma_{sub})$. Note that we incorporated Optimization 2 from [8, Ch. 12], which is made possible thanks to the choice $I = (2)$.

- $\mathsf{KeyGen}''(\lambda)$**:**
  - Run KeyGen' to get B_J^{pk} and $\boldsymbol{v}_J^{sk}$.
  - Generate a uniform γ_{set}-bit vector $\boldsymbol{s} = (s_1, \ldots, s_{\gamma_{set}})$ with Hamming weight γ_{sub} and $s_{\gamma_{set}} = 1$.
  - Generate $\boldsymbol{t}_1, \ldots, \boldsymbol{t}_{\gamma_{set}-1}$ uniformly and independently from J^{-1} mod B_I. Compute $\boldsymbol{t}_{\gamma_{set}} = \boldsymbol{v}_J^{sk} - \sum_{k<\gamma_{set}} s_k \boldsymbol{t}_k$.
  - Return $sk = \boldsymbol{s}$ and $pk = (B_J^{pk}; \boldsymbol{t}_1, \ldots, \boldsymbol{t}_{\gamma_{set}})$.
- $\mathsf{Enc}''(pk, \pi)$**:** Run Enc of SomHom' to generate ciphertext ψ. For $k = 1, \ldots, \gamma_{set}$, compute c_k on $p+1$ bits (1 bit before the binary point, and p bits after) such that $|c_k - [\boldsymbol{t}_k \times \boldsymbol{\psi}]_0 \bmod 2| \leq 2^{-p}$, where $[g]_0$ denotes the constant coefficient of the polynomial $g \in R$. Return ciphertext $(\psi; c_1, \ldots, c_{\gamma_{set}})$.
- $\mathsf{Dec}''(sk, (\psi; c_1, \ldots, c_{\gamma_{set}}))$: Given expanded ciphertext $(\psi; c_1, \ldots, c_{\gamma_{set}})$ and secret key sk, return $\pi = [\psi]_0 - \lfloor \sum_k s_k c_k \rceil \bmod 2$.
- Eval''**:** Same as for SomHom' (while recomputing the c_k's, like in algorithm Enc'').

Fig. 3. Algorithms of the Squashed Scheme SqHom

Note that $\sum_k s_k c_k \approx \sum_k s_k [\boldsymbol{t}_k \times \boldsymbol{\psi}]_0 = ([(\sum_k s_k \boldsymbol{t}_k) \times \boldsymbol{\psi}]_0) = [\boldsymbol{v}_J^{sk} \times \boldsymbol{\psi}]_0$, modulo 2. Hence, in terms of decryption correctness, SqHom differs from SomHom' only due to the rounding errors. The following lemma provides a sufficient precision p (see also [5, Le. 6.1]). In Section 5, we will show that p can be almost halved, using a probabilistic error analysis.

Lemma 3.3. *If $p \geq 3 + \log_2 \gamma_{sub}$, a ciphertext $(\psi; c_1, \ldots, c_{\gamma_{set}})$ of* SqHom *with $\psi = \pi + i \bmod B_J^{pk}$ and $\|\pi + i\| \leq r'_{Dec}$ is correctly decrypted by the decryption algorithm* Dec''*, and $\sum_k s_k c_k$ is within $1/4$ of an integer.*

For bootstrappability, we need to be able to implement the augmented decryption circuits Dec-Add and Dec-Mult with circuit degrees smaller than the degree capacity of the scheme. This is summarized in the following, in terms of the size γ_{sub} of the hidden subset in the secret key.

Theorem 3.1 (Adapted from [5, Th. 6.2]). *Assuming that $\sum_k s_k c_k$ is within $1/4$ of an integer, the augmented decryption circuits* Dec-Add *and* Dec-Mult *for scheme* SqHom *with precision parameter p can be evaluated by circuits of degrees $\leq \gamma_{sub} \cdot 2^9 p^{1.71}$.*

Proof. To decrypt ψ, we have to compute $\pi = [\psi]_0 - \lfloor \sum_k s_k c_k \rceil \bmod 2$. We proceed as follows:

1- Compute $a_k = s_k \cdot c_k$ for $k = 1, \ldots, \gamma_{set}$.
2- Let $a_{k,0}.a_{k,1} \ldots a_{k,p}$ be the bit representation of a_k. To sum the a_k's:
  2.1- For $j = 0, \ldots, p$, compute W_j, the Hamming weight of the bit vector $(a_{0,j}, \ldots, a_{\gamma_{set},j})$.
  2.2- Compute $\pi = [\psi]_0 - \sum_{j \leq p} W_j \cdot 2^{-j} \bmod 2$.

Note that because only γ_{sub} of the a_k's are non-zero, each Hamming weight W_j is at most γ_{sub} and hence its binary representation has at most $\lceil \log_2(\gamma_{sub} + 1) \rceil$ bits. Step 1 requires a single multiplication mod 2 for each output bit, hence has degree 2. For Step 2.1, we use the following.

Lemma 3.4 (Adapted from [5, Le. 6.3]). *Let $(\sigma_1, \ldots, \sigma_t)$ be a binary vector, and $W = W_n \ldots W_0$ be the binary representation of its Hamming weight. Then for any k, the bit W_k can be expressed as a the evaluation in the σ_j's of an integer polynomial of degree exactly 2^k.*

We conclude that Step 2.1 can be computed by a circuit of degree $2^{\lceil \log_2(\gamma_{sub}+1) \rceil} \leq 2\gamma_{sub}$. Using the '3-for-2' trick [15], van Dijk et al. [5] show that Step 2.2 can be done with a circuit of degree $\leq 2^{\lceil \log_{3/2}(p+1) \rceil + 4} \leq 2^6 p^{1.71}$. The total degree of the decryption circuit is thus $\leq \gamma_{sub} \cdot 2^8 p^{1.71}$, and hence that of Dec-Add (resp. Dec-Mult) is $\leq \gamma_{sub} \cdot 2^9 p^{1.71}$. □

Combining Theorem 3.1 with Lemmata 3.2 and 3.3, we get:

Corollary 3.1. *The scheme* SqHom *is bootstrappable as long as*

$$\gamma_{sub} \cdot 2^9 \log^{1.71}(\gamma_{sub} + 4) \leq \frac{\log r'_{Dec}}{\log(r_{Enc} \cdot \gamma_\times \cdot (t+1))}.$$

4 A Less Pessimistic Hardness Analysis of the SSSP

The semantic security of Gentry's schemes SomHom and SomHom′ relies on the hardness of a bounded distance decoding problem. As explained in Section 2, this hardness assumption is asymptotically well understood (with the lattice reduction 'rule of thumb' conjecture). When converted into the bootstrappable

scheme SqHom, another hardness assumption is added, namely that of the so-called SplitKey distinguishing problem. To be precise, a semantic attack against SqHom either leads to an efficient ideal lattice BDD algorithm or to an efficient algorithm for the SplitKey distinguishing problem (see [9, Th. 10]). In [9, Th. 11.1.3], the following Sparse Vector Subset Sum Problem (SVSSP) is shown to reduce to the SplitKey distinguishing problem.

Definition 4.1 (SVSSP$_{\gamma_{sub},\gamma_{set}}$). *Let γ_{sub} and γ_{set} be functions of the hardness parameter λ. Let J be as generated by* KeyGen*, and B_{IJ} be the HNF of ideal IJ. The decisional SVSSP is as follows: Distinguish between $(\boldsymbol{a}_1, \ldots, \boldsymbol{a}_{\gamma_{set}})$ chosen uniformly in $R \cap \mathcal{P}(B_{IJ})$ and the same but conditioned on the existence of a vector $\boldsymbol{s} \in \{0,1\}^{\gamma_{set}}$ of Hamming weight γ_{sub} with $\sum_k s_k \boldsymbol{a}_k = 0 \bmod IJ$.*

For our choice $I = (2)$, we have $B_{IJ} = 2B_J^{pk}$, where B_J^{pk} is the HNF of J. In the following, we use $q = \det(B_{IJ}) = 2^n \det(J)$. A simple birthday paradox attack runs in time $\approx \binom{\gamma_{set}}{\gamma_{sub}}^{1/2}$. To achieve 2^λ hardness, we require that $\gamma_{sub} = \Omega(\lambda)$ and $\gamma_{set} \geq 2\gamma_{sub}$. We now analyze another attack, based on lattice reduction. Consider the lattice

$$L = \left\{ \boldsymbol{x} \in \mathbb{Z}^{\gamma_{set}} : \sum_{k \leq \gamma_{set}} x_k \cdot \boldsymbol{a}_k = 0 \mod IJ \right\}.$$

Since $q\mathbb{Z}^{\gamma_{set}} \subseteq L$, we have $\dim L = \gamma_{set}$. Furthermore, we have $\det L = |\mathbb{Z}^{\gamma_{set}}/L| = |\phi(\mathbb{Z}^{\gamma_{set}})| \leq \det(B_{IJ}) = q$, where $\phi : \mathbb{Z}^{\gamma_{set}} \to \mathbb{Z}^n/IJ$ is the map $\boldsymbol{x} \mapsto \sum_k x_k \boldsymbol{a}_k \bmod IJ$. Also, the existence of the solution vector $\boldsymbol{s}$ implies that $1 \leq \lambda_1(L) \leq \sqrt{\gamma_{sub}}$.

Suppose we are limited to a computational power of 2^λ. The lattice reduction 'rule of thumb' conjecture suggests that we cannot find vectors in L of norms $\leq U := c_1^{\frac{\gamma_{set}}{\lambda}}$, assuming that $\frac{\lambda_{\gamma_{set}}(L)}{\lambda_1(L)} \leq U/c_2$. Apart from the unusual smallness of the lattice minimum, there is no reason to expect the remaining $\lambda_i(L)$'s to vary significantly: the lattice gap $\frac{\lambda_2(L)}{\lambda_1(L)}$ and the lattice amplitude $\frac{\lambda_{\gamma_{set}}(L)}{\lambda_1(L)}$ should be similar. Now, there are $\leq m := U\sqrt{\gamma_{sub}}$ pairs of non-zero multiples $\pm k \cdot \mathbf{s}$ with norm $\leq U \cdot \lambda_1(L) \leq U\sqrt{\gamma_{sub}}$. At the same time, Minkowski's theorem (Theorem 2.1) asserts that there are far more lattice vectors of norm $\leq U/c_2$.

Lemma 4.1. *Assuming that $\frac{\pi^{\frac{\gamma_{set}}{2}}}{\Gamma\left(\frac{\gamma_{set}+2}{2}\right)} \cdot (U/c_2)^{\gamma_{set}} \geq (2^\lambda m) \cdot 2^{\gamma_{set}} \cdot q$, we have $|L \cap \mathcal{B}(\mathbf{0}, U/c_2)| \geq 2^\lambda m$.*

Note that if the condition in Lemma 4.1 holds, then for any $\lambda \geq 1$, the ball of radius $U\lambda_1(L) \geq U/c_2$ contains more than m pairs of non-zero points of L, so the lattice gap $\frac{\lambda_2(L)}{\lambda_1(L)}$ must be $\leq U/c_2$.

It seems reasonable to assume that the lattice points that are not multiples of $\mathbf{s}$ do not provide information towards solving SVSSP. Also, we heuristically expect lattice reduction to return one of these relevant vectors with probability $\approx 2^{-\lambda}$ if they constitute a fraction $2^{-\lambda}$ of the total number of lattice vectors of norm

$\leq U$. Under these assumptions, if the computational effort of lattice reduction is limited to 2^λ and if we wish to bound the likelihood of finding a relevant vector by $2^{-\lambda}$, it seems sufficient to set the parameters so that:

$$c_1^{\frac{\gamma_{set}^2}{\lambda}} \geq 2^\lambda \cdot \gamma_{set}^{\Omega(\gamma_{set})} \cdot q.$$

As $\gamma_{set} = \widetilde{\Omega}(\lambda)$, the above is implied by $\frac{\gamma_{set}^2}{\lambda} = \widetilde{\Omega}(\log q)$. Note that this condition is less restrictive than the corresponding one used in [9,26,5] (i.e., $\gamma_{set} = \Omega(\log q)$).

Remark. In algorithm KeyGen$''$, the SVSSP instances satisfy $s_{\gamma_{set}} = 1$. This does not result in any security reduction, as an attacker can guess an i such that $s_i = 1$ and then permute indices i and γ_{set}.

Remark. Our analysis differs in two ways from the one from [9] relying on [23]: for consistency with the hardness analysis of the ideal BDD, we consider an approximate SVP solver rather than an exact SVP solver; secondly, we do not consider the 'replay' attack from [23] (which would lead to larger involved constants), as contrarily to the case of server-aided RSA, only one instance of the SSSP is given.

5 Improved Ciphertext Refreshing Algorithm

As explained in the proof of Theorem 3.1, the main component in the degree of the decryption algorithm comes from the addition of the rationals $s_k c_k = [s_k \boldsymbol{t}_k \times \boldsymbol{\psi}]_0 \bmod 2$. This accounts for degree γ_{sub}, and all other components of degree are negligible compared to this one.

Recall that $\boldsymbol{t}_1, \ldots, \boldsymbol{t}_{\gamma_{set}-1}$, and hence also $[\boldsymbol{t}_1 \times \boldsymbol{\psi}]_0 \bmod 2, \ldots, [\boldsymbol{t}_{\gamma_{set}-1} \times \boldsymbol{\psi}]_0 \bmod 2$'s are chosen *independently with identical distribution* (iid), and that $\boldsymbol{t}_{\gamma_{set}} = \boldsymbol{v}_J^{sk} - \sum_{k<\gamma_{set}} s_k \boldsymbol{t}_k \bmod 2$. We are to exploit the iid-ness of the first $\boldsymbol{t}_i$'s to obtain a sufficient precision p that is essentially half of that of Section 3.3. This will have the effect of taking the square root of the decryption circuit degree.

5.1 Using Less Precision

We first sum the $s_k[\boldsymbol{t}_k \times \boldsymbol{\psi}]_0$'s for $k < \gamma_{set}$, since they are iid, and then we add the remaining $c_{\gamma_{set}}$. The first sum will be represented on 6 bits (1 bit before the point and 5 bits after) and we will ensure that it is within $1/16$ of $\sum_{k<\gamma_{set}} s_k[\boldsymbol{t}_k \times \boldsymbol{\psi}]_0 \bmod 2$, with high probability. We take $c_{\gamma_{set}}$ within distance $1/16$ of $[\boldsymbol{t}_{\gamma_{set}-1} \times \boldsymbol{\psi}]_0 \bmod 2$ and represent it on 6 bits. The last sum will provide a result within distance $1/8$ of $\sum_{k\leq\gamma_{set}} s_k[\boldsymbol{t}_k \times \boldsymbol{\psi}]_0 \bmod 2$, and can be done with a circuit of constant degree. Using Lemma 3.1, we obtain that the result is within $1/4$ of an integer.

We now concentrate on the first sum. Let the c_k's be fixed-point approximations to the $[\boldsymbol{t}_k \times \boldsymbol{\psi}]_0$'s, with some precision p. We have $\varepsilon_k \leq 2^{-p}$ with $\varepsilon_k = c_k - [\boldsymbol{t}_k \times \boldsymbol{\psi}]_0$. As the c_k's for $k < \gamma_{set}$ are iid, so are the ε_k's, $k < \gamma_{set}$. Also,

we will ensure that $E[\varepsilon_k] = 0$ for any $k < \gamma_{set}$. The following lemma leads to a probabilistic error bound for the sum of the c_k's.

Lemma 5.1. *Let $\varepsilon_1, \ldots, \varepsilon_t$ be iid variables with values in $[-\varepsilon, \varepsilon]$ and such that $E[\varepsilon_k] = 0$ for all k. Then $|\sum_{k \le t} \varepsilon_k| > \sqrt{t}\varepsilon \cdot \omega(\sqrt{\log \lambda})$ with probability negligibly small with respect to λ.*

Proof. We apply Hoeffding's inequality to the ε_k's. We have $\Pr[|\sum \varepsilon_k| \ge x] \le \exp(-x^2/(2t\varepsilon^2))$, for any $x > 0$. We take $x = \sqrt{t}\varepsilon \cdot \omega(\sqrt{\log \lambda})$. □

We use this lemma with $\varepsilon = 2^{-p}$ and $t = \gamma_{sub} - 1$ (i.e., the number of non-zero $s_k\varepsilon_k$'s for $k < \gamma_{sub}$). It indicates that taking $p = \frac{1}{2}\log_2 \gamma_{sub} + \omega(\log\log\lambda)$ suffices to ensure that with probability negligibly close to 1 we have $|\sum_{k<\gamma_{set}} s_k(c_k - [\boldsymbol{t}_k \times \boldsymbol{\psi}]_0) \bmod 2| \le 1/32$. Truncating the result to 5 bits after the binary point cannot add more than an error of $1/32$.

5.2 Expliciting the Computation of the c_k's in Enc″

In order to be able to apply Lemma 5.1, we have to ensure that $E[\varepsilon_k] = 0$ for any $k < \gamma_{set}$. To guarantee the latter and that this computation enjoys a limited complexity bound, the c_k's need to be computed carefully.

We are given $\boldsymbol{t}_k$ and $\boldsymbol{\psi}$, and wish to compute a $(1+p)$-bit approximation c_k to $[\boldsymbol{t}_k \times \boldsymbol{\psi}]_0 \bmod 2$. As J is a degree 1 prime ideal, vector $\boldsymbol{\psi}$ is in fact an integer modulo $\det(J)$. We are thus interested in computing $[\boldsymbol{t}_k]_0 \cdot \psi$ modulo 2. We explicit this computation in Figure 4.

Inputs: Vectors $\boldsymbol{t}_k$ and $\boldsymbol{\psi}$, and precision p.
Output: A precision $(1+p)$ real $c_k \in [-1, 1]$ with $|c_k - ([\boldsymbol{t}_k \times \boldsymbol{\psi}]_0 \bmod 2)| \le 2^{-p}$.
1. $p' := \log_2 \det(J) + p + 1$;
2. Compute the closest precision $(1+p')$ number $\bar{t}_k \in [-1, 1]$ to $[\boldsymbol{t}_k]_0$.
3. Compute $c'_k := \bar{t}_k \psi$ exactly.
4. Reduce c'_k modulo 2, while preserving its sign (the result belongs to $[-1, 1]$).
5. Round c'_k to the closest precision $(1+p)$ number $c_k \in [-1, 1]$.

Fig. 4. Computing coefficient c_k for algorithm Enc″

Lemma 5.2. *The algorithm of Figure 4 is correct. Furthermore, if the vector $\boldsymbol{t}_k$ is chosen uniformly in $J^{-1} \bmod 2$ with uniformly random choice of sign when a coordinate of $\boldsymbol{t}_k$ belongs to $\{-1, 1\}$, then $E[\varepsilon_k] = 0$, where $\varepsilon_k = c_k - ([\boldsymbol{t}_k \times \boldsymbol{\psi}]_0 \bmod 2)$.*

Proof. At Step 2 of the algorithm, we have $|\bar{t}_k - [\boldsymbol{t}_k]_0| \le 2^{-p'-1}$. As ψ is exact and belongs to $[0, \det J)$, we have $|\bar{t}_k\psi - [\boldsymbol{t}_k]_0\psi| \le 2^{-p'-1}\det(J) \le 2^{-p-1}$. Thus, at Step 3, we have $|c'_k - [\boldsymbol{t}_k \times \boldsymbol{\psi}]_0| \le 2^{-p-1}$. The rounding of Step 5 leads to $|c_k - ([\boldsymbol{t}_k \times \boldsymbol{\psi}]_0 \bmod 2)| \le 2^{-p-1} + 2^{-p-1} = 2^{-p}$.

To prove the second statement, we use the symmetry of the distribution of $\boldsymbol{t}_k$. It implies that $E[[\boldsymbol{t}_k \times \boldsymbol{\psi}]_0 \bmod 2] = 0$. We now use the same property to show that $E[c_k] = 0$. At Step 2, changing $\boldsymbol{t}_k$ into $-\boldsymbol{t}_k$ has the effect of changing $\bar{t}_k$ into $-\bar{t}_k$. This implies that at Step 3, changing $\boldsymbol{t}_k$ into $-\boldsymbol{t}_k$ has the effect of changing c'_k into $-c'_k$. Due to the symmetry of the rounding to nearest, this carries over to c_k and ε_k at Step 5. □

Note that the choice of rounding to nearest is not benign: the above proof strongly relies on the symmetry of the rounding with respect to 0.

5.3 Decreasing the Decryption Circuit Depth

We now want to compute $\sum_{k<\gamma_{set}} s_k c_k \bmod 2$, where the c_k's are fixed-point reals with precision $p = \frac{1}{2}\log_2 \gamma_{sub} + \omega(\log\log\lambda)$. Instead of computing the Hamming weights W_j for $j \in \{0,\ldots,p\}$ as in the proof of Theorem 3.1, we compute only the bits $W_{j,\ell}$ (for $0 \leq \ell \leq \lceil \log_2 \gamma_{sub} \rceil$) that are going to contribute to $\sum_{k<\gamma_{set}} s_k c_k \bmod 2$: the most significant bits are rendered useless by the reduction modulo 2. Most interestingly, these unnecessary most significant bits were the ones requiring the higher degree circuits to evaluate. More precisely, we have:

$$\sum_{k<\gamma_{set}} s_k c_k = \sum_{j=0}^{p} \sum_{\ell=0}^{\lceil \log_2 \gamma_{sub} \rceil} W_{j,\ell} 2^{-j+\ell} = \sum_{j=0}^{p} \sum_{\ell=0}^{j+1} W_{j,\ell} 2^{-j+\ell} \bmod 2.$$

Lemma 3.4 now implies that the desired sum mod 2 can be computed correctly with probability negligibly close to 1 with respect to λ, by evaluating an arithmetic circuit of size $\mathcal{P}oly(\gamma_{sub})$ corresponding to a polynomial of degree exactly $2^{p+1} = \sqrt{\gamma_{sub}} \cdot \omega(\sqrt{\log\lambda})$. Overall, we get:

Theorem 5.1. *The scheme* SqHom *is bootstrappable as long as*

$$\sqrt{\gamma_{sub}} \cdot \omega(\sqrt{\log\lambda}) \leq \frac{\log r'_{Dec}}{\log(r_{Enc} \cdot \gamma_{\times} \cdot (t+1))}.$$

6 Asymptotic Efficiency

We now use the improvements described in the two previous sections to derive bounds for the complexity of Gentry's fully homomorphic scheme.

6.1 Optimizing the Parameters in Gentry's Scheme

The table below summarizes and compares the conditions for Gentry's scheme to be 2^{λ}-secure and correct. The semantic security of SomHom′ is related to the hardness of γ-BDD for $\gamma = r'_{Dec}/r_{Enc}$. Recall that $r'_{Dec} = \lambda_1(J)/\mathcal{P}oly(n)$. Recall also that J is an ideal lattice, and thus we have $\lambda_1(J) \geq \det(J)^{1/n} = q^{1/n}/2$ (where q is the SVSSP determinant of Section 4). As a consequence, it suffices

to ensure that γ-BDD is hard to solve for $\gamma = q^{1/n}/(r_{Enc}\mathcal{P}oly(n))$. We use the lattice reduction 'rule of thumb' to derive a sufficient condition. As the encryptor is limited to polynomial-time algorithms, we can safely assume that $n = \mathcal{P}oly(\lambda)$. Also, since $f = x^n + 1$, we have $\gamma_\times = \sqrt{n}$. Finally, by choosing $r_{Enc} = \mathcal{P}oly(\lambda)$, the ciphertexts have sufficient entropy to prevent any exhaustive search.

Condition	[9]	This article
BDD resistant to lattice attacks	$\frac{q^{1/n}}{\mathcal{P}oly(\lambda)} \leq c_1^{n/\lambda}$	
SSSP resistant to birthday paradox	$\binom{\gamma_{set}}{\gamma_{sub}}^{1/2} \geq 2^\lambda$	
SSSP resistant to lattice attacks	$\gamma_{set} = \widetilde{\Omega}(\log q)$	$\frac{\gamma_{set}^2}{\lambda} = \widetilde{\Omega}(\log q)$
Bootstrappability achieved	$\gamma_{sub} \leq \frac{\log(q^{1/n})}{\Theta(\log \lambda)}$	$\sqrt{\gamma_{sub}} \leq \frac{\log(q^{1/n})}{\mathcal{P}oly(\log \lambda)}$

To fulfill these conditions, we set $\gamma_{sub} = \Theta(\lambda)$, $n = \widetilde{\Theta}(\lambda^{1.5})$, $\log q = \widetilde{\Theta}(\lambda^2)$ and $\gamma_{set} = \Theta(\lambda^{1.5})$. In [8, Ch. 12], these values were $\gamma_{sub} \approx \lambda$, $n \approx \lambda^2$, $\log q \approx \lambda^3$ and $\gamma_{set} \approx \lambda^3$ respectively.

6.2 Bit Complexity

The `Recrypt` procedure consists in expanding the ciphertext ψ as described in algorithm Enc'' of SqHom, encrypting the bits of the expanded ciphertext with the new public key pk_2, and then applying algorithm Dec'' homomorphically, using the encrypted ciphertext bits and the encrypted secret key sk_1 (under pk_2). We also consider the cost of homomorphically evaluating an elementary add/mult gate.

Let us first bound the cost of computing the c_k's in Enc'', calling γ_{set} times the algorithm from Figure 4. First, note that Steps 1 and 2 should not be done within Enc'', but at the key generation time, i.e., in KeyGen''. Note that during the third step of KeyGen'', one should also pay attention to perform the reduction modulo (2) such that the assumption of Lemma 5.2 holds. The quantity c'_k obtained at Step 3 of the algorithm from Figure 4 is encoded on $O(\log q)$ bits, and its computation can be performed in $\widetilde{O}(\log q)$ bit operations, using fast integer arithmetic [25]. The costs of Steps 4 and 5 are negligible. Overall, the computation of the c_k's in Enc'' can be done in $\widetilde{O}(\gamma_{set} \log q) = \widetilde{O}(\lambda^{3.5})$ bit operations.

The secret key is made of $\gamma_{set} = \Theta(\lambda^{1.5})$ bits. The bit-length of the encrypted secret key is $\gamma_{set} \log q = \widetilde{O}(\lambda^{3.5})$. To encrypt the bits of the c_k's under pk_2, we use $\mathsf{Samp} = 0$, as explained in [8, Re. 4.1.1], i.e., we consider as encrypted values the bits themselves.

Let us now explain how algorithm Dec'' is implemented. We concentrate on the most expensive part, i.e., the (homomorphic) computations of $O(\log \gamma_{sub}) = \widetilde{O}(1)$ Hamming weights of vectors in $\{0,1\}^{\gamma_{set}}$. Let $(\alpha_1, \ldots, \alpha_{\gamma_{set}})$ be such a vector. As explained in [9, Le. 5] (which relies on [2, Le. 11]), it suffices to compute the developed form of the polynomial $\prod_{k \leq \gamma_{set}}(x - \alpha_k)$. Recall that in Section 5 we showed that we are interested in only a few coefficients of the result, corresponding to monomials of degrees $\widetilde{O}(\sqrt{\gamma_{sub}})$. For the sake of simplicity (and with a negligible cost increase), we compute the full developed form anyway, and then throw away the spurious coefficients. Our circuit here differs from those of [26,5] and [8, Ch. 9]

as we use fast polynomial multiplications and a tree-based construction instead of school-book multiplications and Horner's method, to lower the overall asymptotic complexity. Note that the circuit is over the integers, and evaluates an integer polynomial whose coefficients of interest have small multiplicative degrees in the inputs. We compute the developed form of $\prod_{k \leq \gamma_{set}} (x - \alpha_k)$ with a binary tree:

- At level 0, we have the linear factors $(x - \alpha_k)$.
- At level i, we have $\gamma_{set}/2^i$ polynomials of degree 2^i that are the products of the linear factors corresponding to their binary subtrees.
- A father of two nodes is obtained by multiplying his two sons, with a quasi-linear time multiplication for polynomials over rings that uses only ring operations [3].

The size of each circuit that allows to move from sons at level $i-1$ to father at level i is $\widetilde{O}(2^i)$. The overall number of add/mult integer gates is therefore $\widetilde{O}(\gamma_{set})$. While evaluating this circuit homomorphically, each gate corresponds to an add/mult modulo B_J^{pk}, i.e., thanks to our choice for J, to an add/mult of two integers modulo $\det(J)$, whose bit-length is $O(\log q)$. The overall complexity of Dec'' is $\widetilde{O}(\gamma_{set} \log q) = \widetilde{O}(\lambda^{3.5})$.

To summarize, `Recrypt` for 1 plaintext bit costs $\widetilde{O}(\lambda^{3.5})$ bit operations (compared to the bound $\widetilde{O}(\lambda^6)$ claimed in [8, Ch. 12]). And the cost of homomorphically evaluating an elementary add/mult gate is also $\widetilde{O}(\lambda^{3.5})$. The secret $\boldsymbol{s}$ and the public key $(B_J^{pk}; \bar{\boldsymbol{t}}_1, \ldots, \bar{\boldsymbol{t}}_{\gamma_{set}})$ are respectively encoded on $\gamma_{set} = \Theta(\lambda^{1.5})$ and $\widetilde{O}(n \log q + \gamma_{set} \log q) = \widetilde{O}(\lambda^{3.5})$ bits.

7 Open Problems

It would be interesting to relax our assumptions $f = x^n + 1$ and $I = (2)$, in case other choices prove interesting (see the full version for $I = (2, x+1)$). An important question is to assess the practical impact of our results (see [26,12] for implementations of Gentry's scheme). At the end of [8, Se. 12.3], Gentry suggests using non-independent `SplitKey` vectors $\boldsymbol{t}_i$ to lower the costs. The idea is to encode n vectors $\boldsymbol{t}_{i,j} = x^j \boldsymbol{t}_i \bmod x^n + 1$ using only $\boldsymbol{t}_i$. This leads to a faster amortized cost per plaintext bit using the plaintext domain $\mathbb{Z}_2[x]/f(x)$. However, it is not clear how to homomorphically decrypt with such a variant, as one is now restricted to more complex circuit gates than addition and multiplication modulo 2.

Acknowledgments. We thank Ali Akhavi, Guillaume Hanrot, Steven Galbraith, Craig Gentry and Paul Zimmermann for helpful discussions. We also thank the anonymous reviewers, for pointing out an important error in Section 4. The first author was partly supported by the LaRedA ANR grant, and the second author by a Macquarie University Research Fellowship (MQRF) and ARC Discovery Grant DP0987734.

References

1. Babai, L.: On Lovász' lattice reduction and the nearest lattice point problem. Combinatorica 6, 1–13 (1986)

2. Boyar, J., Peralta, R., Pochuev, D.: On the multiplicative complexity of boolean functions over the basis $(\wedge, \oplus, 1)$. Theoret. Comput. Sci. 235(1), 43–57 (2000)
3. Cantor, D.G., Kaltofen, E.: On fast multiplication of polynomials over arbitrary algebras. Acta Inf. 28(7), 693–701 (1991)
4. Cassels, J.W.S.: An Introduction to the Geometry of Numbers, 2nd edn. Springer, Heidelberg (1971)
5. 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)
6. Gama, N., Nguyen, P.Q.: Finding short lattice vectors within Mordell's inequality. In: Proc. of STOC, pp. 207–216. ACM, New York (2008)
7. Gama, N., Nguyen, P.Q.: Predicting lattice reduction. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 31–51. Springer, Heidelberg (2008)
8. Gentry, C.: A fully homomorphic encryption scheme. PhD thesis, Stanford University (2009), http://crypto.stanford.edu/craig (manuscript)
9. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proc. of STOC, pp. 169–178. ACM, New York (2009)
10. Gentry, C.: Computing arbitrary functions of encrypted data. Communications of the ACM 53(3), 97–105 (2010)
11. 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)
12. Gentry, C., Halevi, S.: Implementing Gentry's fully-homomorphic encryption scheme. Preliminary version (August 5, 2010), https://researcher.ibm.com/researcher/files/us-shaih/fhe-implementation.pdf
13. Hoeffding, W.: Probability inequalities for sums of bounded random variables. J. Amer. Statist. Assoc. 58(301), 13–30 (1963)
14. Kannan, R.: Improved algorithms for integer programming and related lattice problems. In: Proc. of STOC, pp. 99–108. ACM, New York (1983)
15. Karp, R.M.: A survey of parallel algorithms for shared-memory machines. Technical report (1988)
16. Klein, P.N.: Finding the closest lattice vector when it's unusually close. In: Proc. of SODA, pp. 937–941. ACM, New York (2000)
17. Lenstra, A.K., Lenstra Jr., H.W., Lovász, L.: Factoring polynomials with rational coefficients. Math. Ann. 261, 515–534 (1982)
18. Ludwig, C.: A faster lattice reduction method using quantum search. In: Ibaraki, T., Katoh, N., Ono, H. (eds.) ISAAC 2003. LNCS, vol. 2906, pp. 199–208. Springer, Heidelberg (2003)
19. Micciancio, D.: Improving lattice-based cryptosystems using the Hermite normal form. In: Silverman, J.H. (ed.) CALC 2001. LNCS, vol. 2146, pp. 126–145. Springer, Heidelberg (2001)
20. Micciancio, D., Goldwasser, S.: Complexity of lattice problems: a cryptographic perspective. Kluwer Academic Press, Dordrecht (2002)
21. Micciancio, D., Regev, O.: Lattice-based Cryptography. In: Post-Quantum Cryptography. Springer, Heidelberg (2008)
22. Micciancio, D., Voulgaris, P.: A deterministic single exponential time algorithm for most lattice problems based on Voronoi cell computations. In: Proc. of STOC, pp. 351–358. ACM, New York (2010)

23. Nguyen, P.Q., Shparlinski, I.: On the insecurity of a server-aided RSA protocol. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 21–35. Springer, Heidelberg (2001)
24. Schnorr, C.P.: A hierarchy of polynomial lattice basis reduction algorithms. Theoret. Comput. Sci. 53, 201–224 (1987)
25. Schönhage, A., Strassen, V.: Schnelle Multiplikation grosser Zahlen. Computing 7, 281–292 (1971)
26. 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)

A Smaller Keys

In [8, Se. 4.3], Gentry suggests to re-use the same key-pair for all levels of the fully homomorphic scheme derived from Theorem 2.2. This allows one to significantly decrease the key-sizes of the boostrapped fully homomorphic scheme. This strategy can be proved secure if the underlying bootstrappable homomorphic encryption scheme is assumed or known to be KDM-secure [8, Th. 4.3.2]. Our lower-degree decryption may fail with non-negligible probability after the first refreshing of a ciphertext, as our technique does not handle the non-independence of the ciphertext and the secret key. To circumvent this issue, we randomize the ciphertext to waive its possible non-independence with the secret key. Note that this technique is similar in flavor to Gentry's modified scheme providing circuit privacy [9, Se. 7].

Consider algorithm Enc'' of SqHom. The condition required for the probabilistic technique described in Section 5 to work is that the ciphertext $\boldsymbol{\psi} = \pi + \boldsymbol{r} \bmod B_J^{pk}$ (where $\boldsymbol{r} \in (2)$ and $\|\boldsymbol{r}\| \leq r'_{Dec}$) is independent of the $\boldsymbol{t}_i$'s. This fact, together with the iid-ness of the $\boldsymbol{t}_i$'s, implies that the rounding errors ε_i in computing the c_i's, are iid, as required to apply Hoeffding's bound. In the key-reuse application, the internal randomness $\boldsymbol{r}$ of $\boldsymbol{\psi}$ may depend on the $\boldsymbol{t}_i$'s (due to a previous refreshing). To circumvent this, we randomize the ciphertext $\boldsymbol{\psi} = \pi + \boldsymbol{r} \bmod B_J^{pk}$ into another ciphertext $\boldsymbol{\psi'} = \pi + \boldsymbol{r'} \bmod B_J^{pk}$ for the same message π but with internal randomness $\boldsymbol{r'} \in (2)$ which is almost independent of the $\boldsymbol{t}_i$'s. More precisely, given the $\boldsymbol{t}_i$'s, the distribution of $\boldsymbol{r'}$ is within negligible statistical distance from the ($\boldsymbol{t}_i$-independent) distribution $2U$, where U is the uniform distribution on the origin-centered ball of radius r'_{Dec}/ρ with ρ any negligible function of λ such that $\log \rho = \widetilde{O}(1)$ (e.g., $\rho = \lambda^{-\log \lambda}$).

We compute $\boldsymbol{\psi'}$ by adding to $\boldsymbol{\psi}$ an encryption of 0 with sufficiently large randomness compared to the randomness in $\boldsymbol{\psi}$, i.e., we set $\boldsymbol{\psi'} = \boldsymbol{\psi} + \boldsymbol{\zeta} \bmod B_J^{pk}$, where $\boldsymbol{\zeta}$ is sampled from $2U$. If we replace the decryption radius r'_{Dec} by $r''_{Dec} = \frac{r'_{Dec}}{1+2/\rho}$ in Lemma 3.2, then the correctness of the scheme is preserved, as $\boldsymbol{\psi}$ and $\boldsymbol{\psi'}$ both decode to the same plaintext via algorithm Dec'. This has a negligible effect for the asymptotic efficiency (see Section 6.1). Assume that $\boldsymbol{\psi} = \pi + \boldsymbol{r} \bmod B_J^{pk}$ with $\|\boldsymbol{r}\| \leq r'_{Dec}$. Let us consider the statistical distance between the distributions $\boldsymbol{r} + 2U$ and $2U$. As a ball of radius $r'_{Dec}/\rho - r'_{Dec}$ is contained in the intersection of the two balls of radius r'_{Dec}/ρ corresponding to U and $\boldsymbol{r} + U$, we obtain that the statistical distance under scope is at most $n \cdot \rho$, and hence negligible.

A Group Signature Scheme from Lattice Assumptions

S. Dov Gordon[1,⋆], Jonathan Katz[1,⋆⋆], and Vinod Vaikuntanathan[2,*]

[1] Dept. of Computer Science, University of Maryland
{gordon,jkatz}@cs.umd.edu
[2] Microsoft Research, Redmond
vinod@microsoft.com

Abstract. Group signature schemes allow users to sign messages on behalf of a group while (1) maintaining *anonymity* (within that group) with respect to an outside observer, yet (2) ensuring *traceability* of a signer (by the group manager) when needed. In this work we give the first construction of a group signature scheme based on lattices (more precisely, the *learning with errors* assumption), in the random oracle model. Towards our goal, we construct a new algorithm for sampling a basis for an orthogonal lattice, together with a trapdoor, that may be of independent interest.

1 Introduction

Group signature schemes [16] allow users to sign messages on behalf of a group administered by some manager. The group is initialized by having the group manager generate master public and secret keys; upon admission to the group, a user is given a personal secret key that is derived from the master secret key by the manager. A member of the group can sign a message using their personal secret key, enabling anyone who knows the master public key to verify that *some* group member signed the message. Roughly, group signatures are required to satisfy two seemingly contradictory requirements: given some legitimate group signature σ, the group manager should be able to determine which member of the group issued σ (*traceability*), but no one other than the group manager should be able to determine any information about the signer (*anonymity*). Group signatures have proven to be a popular primitive, and since their introduction several constructions have been proposed both with random oracles [5,6,13,10,14,22] and without [8,9,4,11,12,21].

While there exist constructions of group signature schemes based on trapdoor permutations [8,9], such schemes serve only as proofs of feasibility and are far from practical. On the other hand, practical schemes are based on a relatively small set of assumptions: namely, the strong RSA assumption [5,6,13,22] and various assumptions related to groups having an associated bilinear map [10,14,4,11,12,21].

⋆ Work done while at IBM Research.

⋆⋆ Work done in part while at IBM Research, and supported by NSF grants #0627306 and #0716651.

M. Abe (Ed.): ASIACRYPT 2010, LNCS 6477, pp. 395–412, 2010.

In this work we show the first construction of a group signature scheme from assumptions related to *lattices*. The use of lattice-based assumptions in cryptography has seen a flurry of activity in recent years. In part, this is due to a general desire to expand the set of assumptions on which cryptosystems can be based (i.e., beyond the standard set of assumptions related to the hardness of factoring and solving the discrete logarithm problem). Relying on lattice-based assumptions offers several concrete advantages as well: such assumptions are appealing because of the known worst-case/average-case connections between lattice problems, and also because lattice problems are currently immune to quantum attacks. Even restricting to classical attacks, the best-known algorithms for solving several lattice problems require exponential time (in contrast to the sub-exponential algorithms known, e.g., for factoring). Finally, relying on lattices can potentially yield efficient constructions because the basic lattice operations manipulate relatively small numbers and are inherently parallelizable.

While our resulting construction is less efficient than existing schemes based on number-theoretic assumptions, our construction is significantly more efficient than the generic approaches of [8,9] that rely on NIZK proofs based on a Karp reduction to some NP-complete language. (Peikert and Vaikuntanathan [26] construct NIZK proofs for specific lattice problems, however their results are not directly applicable to our work.)

1.1 Our Techniques

Our construction combines ideas from several different works, tying these together using a new technical tool described below. At a high level, our group signature scheme follows a template similar (but not identical) to that of Bellare et al. [8]. The master public key in our scheme includes a public key pk_E for a public-key encryption scheme, along with n signature verification keys $pk_1, \ldots, pk_N$. The personal secret key given to the ith group member is sk_i, the signing key corresponding to pk_i. To sign a message M, the group member (1) signs M using sk_i; (2) encrypts the resulting signature using pk_E; and then (3) provides a NIZK proof of well-formedness (namely, that the given ciphertext encrypts a signature on M relative to one of the pk_i). This implies anonymity (since no one other than the group manager knows the decryption key sk_E corresponding to pk_E), yet ensures traceability because the group manager can decrypt the ciphertext that is included as part of any valid group signature.

To instantiate this approach using lattice-based assumptions, we need to identify candidate signature and encryption schemes along with an appropriate NIZK proof system. While constructions of the former based on lattices are known, we do not currently have constructions of NIZK for all of NP from lattice-based assumptions and we will therefore have to tailor our scheme so that it can rely on (efficient) NIZK proofs for some *specific* language. This is explained in more detail in what follows.

For the underlying signature scheme we use the GPV signature scheme [19] that works roughly as follows. The public key is a basis $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ for a random lattice. To sign a message M, the signer uses a trapdoor $\mathbf{T}$ to find a "short" vector

$\mathbf{e} \in \mathbb{Z}^m$ with $\mathbf{Ae} = H(M)$ (where H is a hash function modeled as a random oracle). Under suitable assumptions, finding such a short vector $\mathbf{e}$ without the trapdoor is hard.

We encrypt the resulting signature using what can be viewed as a non-standard variant of the Regev encryption scheme [27]. Given a matrix $\mathbf{B} \in \mathbb{Z}_q^{n \times m}$ (viewed as a public key), we encrypt $\mathbf{e} \in \mathbb{Z}^m$ by choosing a random vector $\mathbf{s} \in \mathbb{Z}_q^n$ and outputting the ciphertext $\mathbf{z} = \mathbf{B}^T\mathbf{s} + \mathbf{e}$. Effectively, $\mathbf{e}$ here is being used as the noise in an instance of the "learning with errors" (LWE) problem [27]. Before going further, we stress that this "encryption scheme" is *not* semantically secure. However, it turns out that we need something much weaker than semantic security in order to prove anonymity of our scheme; roughly, all we need is that the encryption of a uniformly random $\mathbf{e} \in \mathbb{Z}_q^m$ is computationally indistinguishable from the encryption of a vector $\mathbf{e}$ chosen from a certain discrete Gaussian distribution. We defer further discussion to Section 3.

As described thus far, our group signature scheme would have a master public key consisting of verification keys $\mathbf{A}_1, \ldots, \mathbf{A}_N$ along with an encryption key $\mathbf{B}$; a signature would include $\mathbf{z} = \mathbf{B}^T\mathbf{s} + \mathbf{e}$, where $\mathbf{e}$ is such that $\mathbf{A}_i\mathbf{e} = H(M)$ for some i, along with a proof of well-formedness of the ciphertext $\mathbf{z}$. Constructing the proof of well-formedness turns out to be the most difficult aspect of our work, and it will be useful to modify our scheme a bit in order to help construct this proof. (In doing so, we also rely on specific properties of the GPV signature scheme.) We change our scheme as follows: Now, the master public key contains N verification keys $\mathbf{A}_1, \ldots, \mathbf{A}_N$ (as before) and also N encryption keys $\mathbf{B}_1, \ldots, \mathbf{B}_N$. To sign a message M, user i computes a real signature $\mathbf{e}_i$ (using the trapdoor associated with $\mathbf{A}_i$) and "pseudo-signatures" $\mathbf{e}_j$ for all $j \neq i$. Each "pseudo-signature" $\mathbf{e}_j$ has the property that $\mathbf{A}_j\,\mathbf{e}_j = H(M)$, however $\mathbf{e}_j$ *is not short* (and thus not a valid signature). All the $\{\mathbf{e}_j\}_{j=1}^N$ are then encrypted as before, with each $\mathbf{e}_j$ being encrypted using $\mathbf{B}_j$ to give a ciphertext $\mathbf{z}_j$. We then have the signer provide a proof that (1) each $\mathbf{z}_j$ encrypts a correct pseudo-signature with respect to $\mathbf{A}_j$, and (2) at least one of these pseudo-signatures is in fact *short* (and hence a valid signature). Further details are given next.

To provide a way for the signer to prove that every ciphertext $\mathbf{z}_j$ encrypts a pseudo-signature, we develop a new technical tool that we believe to be of independent interest: a way to sample a basis for an *orthogonal lattice* with its associated trapdoor.[1] Specifically, we show a technique that, given a matrix $\mathbf{B}$, generates $(\mathbf{A}, \mathbf{T})$ such that $\mathbf{AB}^T = 0 \pmod q$ and $\mathbf{T}$ is still a "good trapdoor" (in the sense required for GPV signatures) for $\mathbf{A}$. If we use matrices $\{\mathbf{A}_i\}$ generated in this way as verification keys in the group signature scheme described earlier, then it is possible to verify that a given ciphertext $\mathbf{z}_j$ encrypts a pseudo-signature with respect to $\mathbf{A}_j$ by checking whether $\mathbf{A}_j\mathbf{z}_j \stackrel{?}{=} H(M)$. This works because

$$\mathbf{A}_j\mathbf{z}_j = \mathbf{A}_j \cdot \left(\mathbf{B}_j^T\mathbf{s}_j + \mathbf{e}_j\right) = \mathbf{A}_j\mathbf{e}_j = H(M)$$

by construction.

[1] For our definition of an orthogonal lattice, see Section 2.

The only thing that remains is to provide a proof that at least one of the $\mathbf{z}_j$ encrypts a vector $\mathbf{e}_j$ that is also *short*. This translates to proving that at least one of the vectors $\mathbf{z}_j = \mathbf{B}_j^T \mathbf{s}_j + \mathbf{e}_j$ is "close to" the lattice generated by the columns of $\mathbf{B}_j^T$. This can be done using the (statistical) zero-knowledge protocol demonstrated by Micciancio and Vadhan [23], coupled with standard techniques [17,18] for making the proof witness indistinguishable and noninteractive in the random oracle model.

In essence, we obtain our efficiency gain by coupling together the encryption and the signature components so that the NIZK proof system we need to use is for a very simple language.

1.2 Outline of the Paper

We introduce some notation and review the necessary background on lattices in Section 2. For the reader who is already familiar with lattices, we highlight the following aspects of our treatment that are new to this work:

- In Section 2.2 (cf. Lemma 1) and in the rest of the paper, we consider the LWE problem under a non-standard error distribution. Fortunately, a recent result of Peikert [25] demonstrates that the hardness of the LWE problem under this distribution is implied by standard hardness results.
- In Section 2.4 we describe a technique for sampling a basis for an *orthogonal* lattice and its associated trapdoor.

We turn to group signatures in Section 3. We review the standard definitions of security for group signature schemes in Section 3.1, describe our construction in Section 3.2, and prove anonymity and traceability in Sections 3.3 and 3.4.

2 Preliminaries on Lattices

Throughout, we use n for the security parameter; other parameters are taken to be functions of n. When we say "statistically close" we mean "within statistical difference negligible in n."

We review some basic properties of lattices as used in prior work. This section is included mainly to fix notation and ideas, and we refer to the original papers (cited below) for further exposition.

We use bold lower-case letters (e.g., $\mathbf{x}$) to denote vectors, and bold upper-case letters (e.g., $\mathbf{B}$) to denote matrices. (Our vectors are always column vectors.) We let $||\mathbf{x}||$ denote the Euclidean (i.e., ℓ_2) norm of the vector $\mathbf{x}$, and let $||\mathbf{B}||$ denote the maximum of the Euclidean norms of the columns of $\mathbf{B}$; i.e., if $\mathbf{B} = (\mathbf{b}_1|\cdots|\mathbf{b}_n)$ then $||\mathbf{B}|| \stackrel{\text{def}}{=} \max_i ||\mathbf{b}_i||$. We let $\widetilde{\mathbf{B}} = (\widetilde{\mathbf{b}}_1|\cdots|\widetilde{\mathbf{b}}_n)$ denote the Gram-Schmidt orthogonalization of $\mathbf{B}$, defined iteratively in the following way: $\widetilde{\mathbf{b}}_1 = \mathbf{b}_1$, and for each $i = 2, \ldots, n$, $\widetilde{\mathbf{s}}_i$ is the component of $\mathbf{s}_i$ orthogonal to $\mathsf{span}(\mathbf{s}_1, \ldots, \mathbf{s}_{i-1})$. If $x \in \mathbb{R}$, then $\lfloor x \rceil$ denotes the rounding of x to the nearest integer.

For q an integer, $\mathbb{Z}_q$ denotes the standard group of integers modulo q. We will extend modular arithmetic to the reals in the obvious way: for example, for $q \in \mathbb{Z}^+$ and $x \in \mathbb{R}$ we use $x \bmod q$ to represent the unique real number $y \in [0, q)$ such that $x - y$ is an integer multiple of q. Finally, we define a notion of distance between elements in $\mathbb{Z}_q$ in the natural way: given $x, y \in \mathbb{Z}_q$, their distance is defined by mapping $(x - y) \bmod q$ to the set of integers $\{-\lfloor q/2 \rfloor, \ldots, \lfloor q/2 \rfloor\}$ and then taking the absolute value of the result.

Fixing q and given a matrix $\mathbf{B} \in \mathbb{Z}_q^{n \times m}$, we define the m-dimensional lattice $\mathcal{L}(\mathbf{B}^T)$ as $\mathcal{L}(\mathbf{B}^T) \stackrel{\text{def}}{=} \{y \in \mathbb{Z}^m \mid y \equiv \mathbf{B}^T\mathbf{s} \bmod q \text{ for some } \mathbf{s} \in \mathbb{Z}^n\}$. We define the *orthogonal lattice* $\Lambda^\perp(\mathbf{B})$ as $\Lambda^\perp(\mathbf{B}) \stackrel{\text{def}}{=} \{\mathbf{w} \in \mathbb{Z}^m \mid \mathbf{B} \cdot \mathbf{w} = 0 \bmod q\}$. (Note that the notion of an orthogonal lattice is defined differently in some previous work.) Finally, for a vector $\mathbf{z} \in \mathbb{Z}_q^m$ we define

$$\mathsf{dist}(\mathcal{L}(\mathbf{B}^T), \mathbf{z}) \stackrel{\text{def}}{=} \min_{\mathbf{s} \in \mathbb{Z}_q^n} ||\mathbf{B}^T\mathbf{s} - \mathbf{z}||.$$

In other words, $\mathsf{dist}(\mathcal{L}(\mathbf{B}^T), \mathbf{z})$ is the distance of $\mathbf{z}$ from the lattice spanned by the columns of $\mathbf{B}^T$.

2.1 Gaussian Error Distributions

The one-dimensional (continuous) Gaussian distribution over $\mathbb{R}$, parameterized by $s \in \mathbb{R}^+$, is defined by the density function

$$\forall x \in \mathbb{R}: \qquad D_s(x) = 1/s \cdot \exp(-\pi(x/s)^2).$$

In this work we always use a *truncated* Gaussian, i.e., the Gaussian distribution D_s whose support is restricted to numbers $x \in \mathbb{R}$ such that $|x| < s \cdot \omega(\sqrt{\log n})$. The truncated and non-truncated distributions are statistically close, and we drop the word "truncated" from now on. The m-dimensional continuous Gaussian distribution is defined in a similar way, by the density function $D_s(\mathbf{x}) = 1/s^m \cdot \exp(-\pi(||\mathbf{x}||/s)^2)$. Finally, we denote by $D_{s,\mathbf{c}}$ the m-dimensional continuous Gaussian distribution centered at the point $\mathbf{c} \in \mathbb{R}^m$. i.e., $D_{s,\mathbf{c}}(\mathbf{x}) = 1/s^m \cdot \exp(-\pi(||\mathbf{x} - \mathbf{c}||/s)^2)$.

Let $\Lambda \subseteq \mathbb{Z}^m$ be a lattice. The *discrete Gaussian distribution* $D_{\Lambda,s,\mathbf{c}}$ is the m-dimensional Gaussian distribution centered at $\mathbf{c}$, but with support restricted to the lattice Λ. (We write $D_{\Lambda,s}$ as shorthand for $D_{\Lambda,s,\mathbf{0}}$.) Formally, the density function of the discrete Gaussian distribution is defined as

$$\forall \mathbf{x} \in \Lambda: \qquad D_{\Lambda,s,\mathbf{c}}(\mathbf{x}) = \frac{D_{s,\mathbf{c}}(\mathbf{x})}{\sum_{\mathbf{y} \in \Lambda} D_{s,\mathbf{c}}(\mathbf{y})}.$$

Gentry et al. [19] show that given a basis $\mathbf{B}$ for Λ, this distribution can be sampled efficiently (to within negligible statistical distance) for $s \geq ||\mathbf{B}|| \cdot \omega(\sqrt{\log n})$.

2.2 The Learning with Errors Problem

The "learning with errors" (LWE) problem was introduced by Regev [27] as a generalization of the "learning parity with noise" problem. We describe the problem in a form suitable for our applications in this paper.

Fix a positive integer n, integers $m \geq n$ and $q \geq 2$, a vector $\mathbf{s} \in \mathbb{Z}_q^n$, and a probability distribution χ on the interval $[0, q)^m$. Define the following two distributions over $\mathbb{Z}_q^{n \times m} \times [0, q)^m$:

- $\mathsf{LWE}_{m,q,\chi}(\mathbf{s})$ is the distribution obtained by choosing uniform $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, sampling $\mathbf{e} \leftarrow \chi$, and outputting $(\mathbf{A}, \mathbf{A}^T\mathbf{s} + \mathbf{e} \bmod q)$.
- $U_{m,q}$ is the distribution obtained by choosing uniform $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and uniform $\mathbf{y} \in [0, q)^m$, and outputting $(\mathbf{A}, \mathbf{y})$.

The decisional variant of the LWE problem (relative to the distribution χ) can be stated informally as the problem of distinguishing between $U_{m,q}$ and $\mathsf{LWE}_{m,q,\chi}(\mathbf{s})$ for a uniform $\mathbf{s}$. Formally, for m, q, and χ that may depend on n (viewed now as a security parameter) we say the $\mathsf{LWE}_{m,q,\chi}$ *problem is hard* if the following is negligible for any probabilistic polynomial-time algorithm D:

$$\big|\Pr[\mathbf{s} \leftarrow \mathbb{Z}_q^n; (\mathbf{A}, \mathbf{y}) \leftarrow \mathsf{LWE}_{m,q,\chi}(\mathbf{s}) : D(\mathbf{A}, \mathbf{y}) = 1] - \Pr[(\mathbf{A}, \mathbf{y}) \leftarrow U_{m,q} : D(\mathbf{A}, \mathbf{y}) = 1]\big| .$$

A standard setting for the LWE problem considers the error distribution Ψ_α^m over $[0, q)^m$ defined as follows: Sample m numbers $\eta_1, \ldots, \eta_m \leftarrow D_\alpha$, let $e_i := q \cdot \eta_i \pmod q$, and output $\mathbf{e} := (e_1, \ldots, e_m)^T$. We write $\mathsf{LWE}_{m,q,\alpha}(\mathbf{s})$ as an abbreviation for $\mathsf{LWE}_{m,q,\Psi_\alpha^m}(\mathbf{s})$.

Evidence for the hardness of the $\mathsf{LWE}_{m,q,\alpha}$ problem comes from a result of Regev [27], who gave a *quantum* reduction from approximating certain lattice problems to within a factor of $\widetilde{O}(n/\alpha)$ on n-dimensional lattices in the worst case to solving $\mathsf{LWE}_{m,q,\alpha}$, subject to the condition that $\alpha \cdot q > 2\sqrt{n}$. Recently, Peikert [24] gave a *classical* reduction for similar parameters. For our purposes, we note that the $\mathsf{LWE}_{m,q,\alpha}$ problem is believed to be hard — given the state-of-the-art in lattice algorithms — for any $m, q = \mathsf{poly}(n)$ and $\alpha = 1/\mathsf{poly}(n)$ (subject to the above condition).

A second error distribution for the LWE problem[2] — and one that we will use in this paper — is the discrete Gaussian distribution $D_{\mathbb{Z}^m,s} \pmod q$. Although this distribution may seem similar to a discretized (rounded) version of Ψ_α^m, these distributions are statistically *far* from each other and thus we cannot immediately conclude anything about the hardness of the LWE problem with respect to one distribution from hardness of the LWE problem with respect to the other. Fortunately, a recent result of Peikert [25] can be used to show that hardness of the LWE problem with respect to error distribution $D_{\mathbb{Z}^m,\alpha \cdot q \cdot \sqrt{2}}$ is implied by hardness of the LWE problem with respect to error distribution Ψ_α^m. We write $\widehat{\mathsf{LWE}}_{m,q,\alpha q\sqrt{2}}$ as an abbreviation for $\mathsf{LWE}_{m,q,D_{\mathbb{Z}^m,\alpha q\sqrt{2}}}$

Lemma 1. *For any α, hardness of the $\mathsf{LWE}_{m,q,\alpha}$ problem implies hardness of the $\widehat{\mathsf{LWE}}_{m,q,\alpha q\sqrt{2}}$ problem.*

[2] When using a discrete error distribution χ over $\mathbb{Z}_q^m$ (rather than a continuous distribution over $[0, q)^m$), the LWE problem is to distinguish $\mathsf{LWE}_{m,q,\chi}$ from the uniform distribution over $\mathbb{Z}_q^{n \times m} \times \mathbb{Z}_q^m$ (rather than $\mathbb{Z}_q^{n \times m} \times [0, q)^m$).

Proof. We show an efficient transformation T that takes as input $(\mathbf{A}, \mathbf{y}) \in \mathbb{Z}_q^{n\times m} \times [0,q)^m$ and has the following properties:

- If $(\mathbf{A}, \mathbf{y})$ is uniform over $\mathbb{Z}_q^{n\times m} \times [0,q)^m$ then the output $T(\mathbf{A}, \mathbf{y})$ is uniform over $\mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^m$.
- If $(\mathbf{A}, \mathbf{y})$ is distributed according to $\mathsf{LWE}_{m,q,\alpha}(\mathbf{s})$ then $T(\mathbf{A}, \mathbf{y})$ is distributed according to $\widehat{\mathsf{LWE}}_{m,q,\alpha q\sqrt{2}}(\mathbf{s})$.

The lemma follows immediately from these two properties.

The transformation T works as follows. Given $(\mathbf{A}, \mathbf{y})$, it samples a vector $\mathbf{w} \leftarrow D_{\mathbb{Z}^m-\mathbf{y},\alpha q}$ and outputs the pair $(\mathbf{A}, \mathbf{y} + \mathbf{w} \pmod q)$.

First, say $(\mathbf{A}, \mathbf{y})$ is distributed uniformly over $\mathbb{Z}_q^{n\times m} \times [0,q)^m$. Note that $\mathbf{y}+\mathbf{w}$ is always an integer, and the distribution $\mathbf{w} \leftarrow D_{\mathbb{Z}^m-\mathbf{y},\alpha q}$ depends only on the fractional part of each entry in $\mathbf{y}$. It follows that $(\mathbf{A}, \mathbf{y}+\mathbf{w} \pmod q)$ is distributed uniformly over $\mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^m$.

On the other hand, say $\mathbf{y} = \mathbf{A}^T\mathbf{s} + \mathbf{e} \pmod q$ where $\mathbf{e} \sim \Psi_\alpha^m$. Since we have $\mathbf{A}^T\mathbf{s} \in \mathbb{Z}^m$, sampling $\mathbf{w} \sim D_{\mathbb{Z}^m-\mathbf{y},\alpha q} \pmod q$ is equivalent to sampling $\mathbf{w} \sim D_{\mathbb{Z}^m-\mathbf{e},\alpha q} \pmod q$. A recent theorem of Peikert [25, Theorem 3.1] shows that the following two processes produce statistically close distributions:

- Sampling $\mathbf{e} \sim \Psi_\alpha^m$ and then setting $\mathbf{e}' = \mathbf{e} + D_{\mathbb{Z}^m-\mathbf{e},\alpha q} \pmod q$;
- Sampling $\mathbf{e}' \sim D_{\mathbb{Z}^m,\alpha q\sqrt{2}} \pmod q$.

We conclude that the output $T(\mathbf{A}, \mathbf{y}) = \big(\mathbf{A}, \mathbf{A}^T\mathbf{s} + (\mathbf{e} + \mathbf{w}) \pmod q\big)$ is distributed according to $\widehat{\mathsf{LWE}}_{m,q,\alpha q\sqrt{2}}(\mathbf{s})$.

2.3 Trapdoor Functions and the GPV Signature Scheme

Ajtai [2] and Alwen and Peikert [3] show algorithms that generate an almost uniform matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ together with a "trapdoor" matrix $\mathbf{T} \in \mathbb{Z}^{m\times m}$ satisfying the following conditions:

Lemma 2 ([3]). *There is a probabilistic polynomial-time algorithm* $\mathsf{TrapSamp}$ *that, on input* $1^n, 1^m, q$ *with* $q \geq 2$ *and* $m \geq 8n \log q$*, outputs matrices* $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ *and* $\mathbf{T} \in \mathbb{Z}^{m\times m}$ *such that:*

- *The distribution on* $\mathbf{A}$ *as output by* $\mathsf{TrapSamp}$ *is statistically close to uniform over* $\mathbb{Z}_q^{n\times m}$*,*
- *the columns of* $\mathbf{T}$ *form a basis of the lattice* $\Lambda^\perp(\mathbf{A})$*, implying in particular* $\mathbf{A} \cdot \mathbf{T} = 0 \pmod q$*,*
- $\|\mathbf{T}\| = O(n \log q)$ *and* $\|\widetilde{\mathbf{T}}\| \leq C \cdot \sqrt{n \log q}$*, for some absolute constant* $C < 40$*.*

Given an "LWE instance" $(\mathbf{A}, \mathbf{y} = \mathbf{A}^T\mathbf{s} + \mathbf{e})$ for a "short" vector $\mathbf{e}$, knowledge of $\mathbf{T}$ can be used to recover $\mathbf{s}$. Specifically, if $||\mathbf{T}|| < L$ and $\mathbf{e}$ is drawn from Ψ_α^m for $\alpha \leq 1/(L \cdot \omega(\sqrt{\log n}))$, then $\mathbf{s}$ can be easily recovered. This is done by first computing

$$\begin{aligned}\mathbf{T}^T\mathbf{y} \pmod q = \mathbf{T}^T(\mathbf{A}^T\mathbf{s} + \mathbf{e}) \pmod q &= (\mathbf{A}\mathbf{T})^T\mathbf{s} + \mathbf{T}^T\mathbf{e} \pmod q \\ &= \mathbf{T}^T\mathbf{e} \pmod q.\end{aligned}$$

Since both $\mathbf{T}$ and $\mathbf{e}$ contain only "small" entries, each entry of the vector $\mathbf{T}^T\mathbf{e}$ is smaller than q and thus $\mathbf{T}^T\mathbf{e} \pmod q$ is equal to $\mathbf{T}^T\mathbf{e}$ (over the integers). Multiplying by $(\mathbf{T}^T)^{-1}$ thus gives $\mathbf{e}$, after which it is easy to recover $\mathbf{s}$.

The GPV signature scheme. Gentry, Peikert, and Vaikuntanathan [19] showed how to use the trapdoor sampling procedure described above to construct a one-way preimage-sampleable function. This can then be turned into a digital signature scheme using an "FDH-like" construction [7]. (See [19] for a formal definition of preimage-sampleable functions and the construction of the signature scheme.) Here, we describe how the preimage-sampleable function works.

Take $q = \mathsf{poly}(n)$, $m \geq 8n \log q$, and $s \geq C \cdot \sqrt{n \log q} \cdot \omega(\sqrt{\log n})$ (where the constant C is from Lemma 2). The one-way preimage-sampleable function is defined by the following algorithms:

- $\mathsf{GPVGen}(1^n)$ runs $\mathsf{TrapSamp}(1^n, 1^m, q)$ to obtain $(\mathbf{A}, \mathbf{T})$. The matrix $\mathbf{A}$ (and q) defines the function $f_{\mathbf{A}}(\mathbf{e}) = \mathbf{A}\mathbf{e} \pmod q$, with domain $\{\mathbf{e} \in \mathbb{Z}^m : ||\mathbf{e}|| \leq s\sqrt{m}\}$ and range $\mathbb{Z}_q^n$. Hardness of inversion is with respect to the distribution $D_{\mathbb{Z}^m,s}$ over the domain.
- The trapdoor inversion algorithm $\mathsf{GPVInvert}(\mathbf{A}, \mathbf{T}, s, \mathbf{u})$ samples from $f_{\mathbf{A}}^{-1}(\mathbf{u})$ as follows: first, it computes (using standard linear algebra) an arbitrary $\mathbf{t} \in \mathbb{Z}^m$ such that $\mathbf{A}\mathbf{t} = \mathbf{u} \pmod q$ (except for a negligible fraction of $\mathbf{A}$, such a $\mathbf{t}$ always exists). Then it samples and outputs $\mathbf{e} \leftarrow D_{\Lambda^\perp(\mathbf{A})+\mathbf{t},s}$.

The above function is one-way if GapSVP_γ is hard on the worst-case for polynomial approximation factor γ [1].

2.4 Sampling an Orthogonal Lattice with Trapdoor

We show a variant of the trapdoor sampling algorithm described in Lemma 2. In our variant, the algorithm is additionally given a matrix $\mathbf{B} \in \mathbb{Z}_q^{n \times m}$ and (informally) should output a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ with an associated trapdoor $\mathbf{T} \in \mathbb{Z}^{m \times m}$ with the additional requirement that the rows of $\mathbf{A}$ are orthogonal (over $\mathbb{Z}_q$) to the rows of $\mathbf{B}$. In other words, we require that $\mathbf{A}\mathbf{B}^T = \mathbf{0} \pmod q$.

Overview of the construction. The basic idea is as follows. Write $\mathbf{B}$ as

$$\mathbf{B}^T = \begin{pmatrix} \mathbf{B}_1 \\ \mathbf{B}_2 \end{pmatrix},$$

with $\mathbf{B}_2$ a square, invertible matrix of dimension $n \times n$. We then generate an orthogonal matrix $\mathbf{A} = [\mathbf{A}_1 \mid \mathbf{A}_2]$ in two steps. We generate the first component $\mathbf{A}_1$ using the $\mathsf{TrapSamp}$ protocol. Recall, this returns a matrix that is statistically close to uniform, along with an associated trapdoor $\mathbf{T}_1$. Once we have chosen $\mathbf{A}_1$ the second component $\mathbf{A}_2$ is constrained to a fixed value by the requirement that $\mathbf{A}\mathbf{B}^T = \mathbf{0} \pmod q$; we thus generate $\mathbf{A}_2$ by solving the linear equations that define this constraint.

All that remains is to find a trapdoor $\mathbf{T}$ such that the columns of $\mathbf{T}$ are short and $\mathbf{A} \cdot \mathbf{T} = \mathbf{0}$. Here we rely on the recent basis delegation techniques of Cash

et al. [15], which allows us to "extend" the basis $\mathbf{T}_1$ into a larger basis $\mathbf{T}$ for $\Lambda^{\perp}(\mathbf{A})$ as desired. The details follow.

Lemma 3. *There is a probabilistic polynomial-time algorithm* OrthoSamp *that on input* $1^n, 1^m, q$ *(with* $q \geq 2$ *and* $m \geq n + 8n \log q$*) and a matrix* $\mathbf{B} \in \mathbb{Z}_q^{n \times m}$ *whose columns span* $\mathbb{Z}_q^n$*, outputs matrices* $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ *and* $\mathbf{T} \in \mathbb{Z}^{m \times m}$ *such that:*

- $\mathbf{A}\mathbf{B}^T = \mathbf{0} \pmod q$*. Moreover, the distribution on* $\mathbf{A}$ *is statistically close to uniform over* $\mathbb{Z}_q^{n \times m}$*, subject to this condition,*
- *the columns of* $\mathbf{T}$ *form a basis of the lattice* $\Lambda^{\perp}(\mathbf{A})$*, implying in particular* $\mathbf{A} \cdot \mathbf{T} = 0 \pmod q$*,*
- *Furthermore, each column* $\mathbf{t}_i$ *of* $\mathbf{T}$ *is distributed (independently) according to* $D_{\Lambda^{\perp}(\mathbf{A}),s}$*, where* $s = C \cdot \sqrt{n \log q} \cdot \omega(\sqrt{\log m})$ *and* C *is the constant from Lemma 2.*

Proof. Let $m_1 = 8n \log q$ and $m_2 = n$. Write

$$\mathbf{B}^T = \begin{pmatrix} \mathbf{B}_1 \\ \mathbf{B}_2 \end{pmatrix},$$

where $\mathbf{B}_1 \in \mathbb{Z}_q^{m_1 \times n}$ and $\mathbf{B}_2 \in \mathbb{Z}_q^{m_2 \times n}$. Furthermore, we require that the square matrix $\mathbf{B}_2$ has full-rank, i.e., its rows span $\mathbb{Z}_q^n$ (such a decomposition of $\mathbf{B}$ into $\mathbf{B}_1$ and $\mathbf{B}_2$ can always be found since the rows of $\mathbf{B}^T$ span $\mathbb{Z}_q^n$).

The algorithm OrthoSamp works as follows:

1. Compute $(\mathbf{A}_1, \mathbf{T}_1) \leftarrow \mathsf{TrapSamp}(1^n, 1^{m_1}, q)$. Let $\mathbf{A}_2 \in \mathbb{Z}_q^{n \times m_2}$ be a uniformly random matrix satisfying

$$\mathbf{A}_2\mathbf{B}_2 = -\mathbf{A}_1\mathbf{B}_1 \pmod q$$

   Since $\mathbf{B}_2$ is invertible over $\mathbb{Z}_q$ by construction, $\mathbf{A}_2$ can be computed as $-\mathbf{A}_1\mathbf{B}_1\mathbf{B}_2^{-1} \pmod q$. If the columns of $\mathbf{A}_1$ do not span $\mathbb{Z}_q^n$, output $\perp$. This occurs only with negligible probability.
2. Extend the basis $\mathbf{T}_1$ into basis $\mathbf{T}' \in \mathbb{Z}_q^{m \times m}$ for $\Lambda^{\perp}(\mathbf{A})$ using the technique of Cash et al. [15, Lemma 3.2]. We present their technique for completeness. Let $\mathbf{T}'$ be of the form

$$\mathbf{T}' = \begin{pmatrix} \mathbf{T}_1 & \mathbf{W} \\ \mathbf{0} & \mathbf{I} \end{pmatrix}$$

   where $\mathbf{W} \in \mathbb{Z}_q^{m_1 \times m_2}$ is an arbitrary matrix satisfying $\mathbf{A}_1\mathbf{W} = -\mathbf{A}_2$, and $\mathbf{I} \in \mathbb{Z}_q^{m_2 \times m_2}$ is the identity matrix. (Note that $\mathbf{W}$ exists by the assumption that the columns of $\mathbf{A}_1$ span $\mathbb{Z}_q^n$.)
3. Randomize the basis $\mathbf{T}'$ into a "random basis" $\mathbf{T}$. This is done by running the RandBasis algorithm of [15, Lemma 3.3] on $\mathbf{T}'$ using parameter $s = ||\widetilde{\mathbf{T}'}|| \cdot \omega(\sqrt{\log m})$. Output $\mathbf{A} = [\mathbf{A}_1 \mid \mathbf{A}_2]$ and $\mathbf{T}$.

We now verify that this algorithm satisfies the required properties. First observe that

$$\mathbf{A}\mathbf{B}^T = \mathbf{A}_1\mathbf{B}_1 + \mathbf{A}_2\mathbf{B}_2 = \mathbf{A}_1\mathbf{B}_1 - \mathbf{A}_1\mathbf{B}_1 = \mathbf{0} \pmod q.$$

The claim regarding the distribution of $\mathbf{A}$ follows directly from the construction. We also have

$$\mathbf{A}\cdot\mathbf{T}' = [\mathbf{A}_1|\mathbf{A}_2]\cdot\begin{pmatrix}\mathbf{T}_1 & \mathbf{W}\\ \mathbf{0} & \mathbf{I}\end{pmatrix} = [\,\mathbf{A}_1\mathbf{T}_1 + \mathbf{A}_2\mathbf{0} \mid \mathbf{A}_1\mathbf{W} + \mathbf{A}_2\,] = \mathbf{0} \pmod q,$$

where the final equality holds because $\mathbf{A}_1\mathbf{T}_1 = \mathbf{0}$ by the properties of $\mathsf{TrapSamp}$, and $\mathbf{A}_1\mathbf{W} = -\mathbf{A}_2$ by construction. Thus, $\mathbf{T}'$ is a basis for $\Lambda^\perp(\mathbf{A})$. Finally, since $\mathbf{T}$ is the result of running $\mathsf{RandBasis}$ on $\mathbf{T}'$, $\mathbf{T}$ is also a basis for $\Lambda^\perp(\mathbf{A})$.

Finally, from the work of Cash et al. [15], we know that $||\widetilde{\mathbf{T}'}|| \leq ||\widetilde{\mathbf{T}_1}|| = O(\sqrt{n\log q})$. Thus, by the property of the $\mathsf{RandBasis}$ algorithm from [15], each column of $\mathbf{T}$ is independently distributed according to $D_{\Lambda^\perp(\mathbf{A}),s}$ where $s = C\cdot\sqrt{n\log q}\cdot\omega(\sqrt{\log m})$.

The following corollary follows from the above construction, and will be used in the security proof of our signature scheme.

Corollary 1. *The distributions*

$$\left\{\mathbf{B}\leftarrow\mathbb{Z}_q^{n\times m}; (\mathbf{A},\mathbf{T})\leftarrow\mathsf{OrthoSamp}(1^n,1^m,q,\mathbf{B}) : (\mathbf{A},\mathbf{T},\mathbf{B})\right\}$$

and

$$\begin{aligned}\{(\mathbf{A},\mathbf{T}')\leftarrow\mathsf{TrapSamp}(1^n,1^m,q); \mathbf{T}\leftarrow\mathsf{RandBasis}(\mathbf{T}');\\ (\mathbf{B},\mathbf{S})\leftarrow\mathsf{OrthoSamp}(1^n,1^m,q,\mathbf{A}) : (\mathbf{A},\mathbf{T},\mathbf{B})\}\end{aligned}$$

are statistically close.

2.5 Efficient NIWI Proofs for Lattice Problems

Let $\mathbf{B}_1,\ldots,\mathbf{B}_N\in\mathbb{Z}_q^{n\times m}$ and $\mathbf{z}_1,\ldots,\mathbf{z}_N\in\mathbb{Z}_q^m$. In this section we briefly describe how it is possible to construct a noninteractive witness-indistinguishable (NIWI) proof (in the random oracle model) for the gap language $L_{s,\gamma} = (L_{YES}, L_{NO})$ defined by:

$$L_{YES} = \left\{\begin{pmatrix}\mathbf{B}_1,\ldots,\mathbf{B}_N\\ \mathbf{z}_1,\ldots,\mathbf{z}_N\end{pmatrix}\;\middle|\;\exists\mathbf{s}\in\mathbb{Z}_q^n \text{ and } i\in[N]\;:\;||\mathbf{z}_i-\mathbf{B}_i^T\mathbf{s}||\leq s\sqrt{m}\right\}$$

$$L_{NO} = \left\{\begin{pmatrix}\mathbf{B}_1,\ldots,\mathbf{B}_N\\ \mathbf{z}_1,\ldots,\mathbf{z}_N\end{pmatrix}\;\middle|\;\forall\mathbf{s}\in\mathbb{Z}_q^n \text{ and } i\in[N]\;:\;||\mathbf{z}_i-\mathbf{B}_i^T\mathbf{s}||>\gamma\cdot s\sqrt{m}\right\}.$$

Here, L_{YES} is a collection of N points at least one of which is close to the corresponding lattice, and L_{NO} is a collection of N points all of which are far from the corresponding lattices.

Our starting point is an (interactive) witness-indistinguishable (WI) proof system for the gap version of the *closest vector problem*, i.e., for the language $L'_\gamma = \{L'_{YES}, L'_{NO}\}$ [20,23]:

$$L'_{YES} = \left\{(\mathbf{B}, \mathbf{z}, t) \mid \exists \mathbf{s} \,:\, ||\mathbf{z} - \mathbf{B}^T\mathbf{s}|| \leq t\right\}.$$

$$L'_{NO} = \left\{(\mathbf{B}, \mathbf{z}, t) \mid \forall \mathbf{s} \,:\, ||\mathbf{z} - \mathbf{B}^T\mathbf{s}|| > \gamma \cdot t\right\}.$$

Our language $L_{s,\gamma}$ can be described as the OR of several instance of L'_γ; that is,

$$\begin{pmatrix} \mathbf{B}_1, \ldots, \mathbf{B}_N \\ \mathbf{z}_1, \ldots, \mathbf{z}_N \end{pmatrix} \in L_{YES} \Leftrightarrow \bigvee_i \left((\mathbf{B}_i, \mathbf{z}_i, s\sqrt{m}) \in L'_{YES}\right).$$

$$\begin{pmatrix} \mathbf{B}_1, \ldots, \mathbf{B}_N \\ \mathbf{z}_1, \ldots, \mathbf{z}_N \end{pmatrix} \in L_{NO} \Leftrightarrow \bigwedge_i \left((\mathbf{B}_i, \mathbf{z}_i, s\sqrt{m}) \in L'_{NO}\right).$$

We can thus use the techniques of Cramer, Damgård, and Schoenmakers [17] to obtain an interactive WI proof for $L_{s,\gamma}$ with negligible soundness error. Using the Fiat-Shamir transformation [18], the resulting protocol can be made non-interactive in the random oracle model.

We remark that for our application we only require soundness (and do not require the proof system to be a proof of knowledge) and witness indistinguishability (rather than zero knowledge). The observations in this section are summarized in the following lemma.

Lemma 4. *Let $\gamma \geq O(\sqrt{m/\log m})$. Then there is a noninteractive witness-indistinguishable proof system for the language $L_{s,\gamma}$ in the random oracle model, where the length of the proof is $O(mnN \log q)$ bits.*

3 A Group Signature Scheme Based on Lattices

3.1 Definitions

We adopt the definition of group signature schemes from the work of Bellare, Micciancio, and Warinschi [8], with the relaxation suggested by Boneh, Boyen, and Shacham [10] (and considered also in, e.g., [11]). Formally, a group signature scheme $\mathcal{GS} = (\mathsf{G.KeyGen}, \mathsf{G.Sign}, \mathsf{G.Vrfy}, \mathsf{G.Open})$ is a collection of four polynomial-time algorithms defined as follows.

- The *group key-generation algorithm* $\mathsf{G.KeyGen}(1^n, 1^N)$ is a randomized algorithm that takes a security parameter 1^n and the group size 1^N as input, and outputs $(\mathsf{PK}, \mathsf{TK}, \mathbf{gsk})$, where PK is the group public key, TK is the group manager's tracing key, and $\mathbf{gsk}$ is a vector of N signing keys with $\mathsf{gsk}[i]$ being the signing key given to the i^{th} group member.
- The *group signature algorithm* $\mathsf{G.Sign}(\mathsf{gsk}[i], M)$ is a randomized algorithm that takes as input a secret signing key $\mathsf{gsk}[i]$ and a message M, and outputs a signature σ.

- The *group signature verification algorithm* $\mathsf{G.Vrfy}(\mathsf{PK}, M, \sigma)$ is a deterministic algorithm that takes as input the group public key PK, a message M, and a signature σ, and outputs either 1 or 0 (signifying accept or reject, respectively).
- The *opening algorithm* $\mathsf{G.Open}(\mathsf{TK}, M, \sigma)$ is a deterministic algorithm that takes as input the tracing key TK, a message M, and a signature σ, and outputs an identity $i \in [N]$.

The basic consistency requirements of a group signature scheme are that an honest signature generated by a group member should be accepted as correct, and must be traceable to the group member who issued it. That is, for any $(\mathsf{PK}, \mathsf{TK}, \mathbf{gsk})$ output by $\mathsf{G.KeyGen}(1^n, 1^N)$, any M, and any $i \in [N]$, if $\sigma \leftarrow \mathsf{G.Sign}(\mathsf{gsk}[i], M)$ then

$$\mathsf{G.Vrfy}(\mathsf{PK}, M, \sigma) = 1 \text{ and } \mathsf{G.Open}(\mathsf{TK}, M, \sigma) = i,$$

except with negligible probability over the entire experiment.

Group signature schemes are also required to satisfy two basic security properties: *anonymity* and *traceability*. Anonymity means that without the tracing key it should be infeasible to determine which group member issued a particular signature (even given all the signing keys). Bellare et al. [8] defined a "CCA-version" of this notion, where the adversary is given access to a tracing oracle. Following [10] we use a "CPA-version" of anonymity where such oracle access is not given.

Definition 1. *A group signature scheme* $\mathcal{GS} = (\mathsf{G.KeyGen}, \mathsf{G.Sign}, \mathsf{G.Vrfy}, \mathsf{G.Open})$ *is* anonymous *if for all polynomials* $N(\cdot)$ *and all probabilistic polynomial-time adversaries* $\mathcal{A}$*, the advantage of* $\mathcal{A}$ *in the following experiment is negligible in* n*:*

1. *Compute* $(\mathsf{PK}, \mathsf{TK}, \mathbf{gsk}) \leftarrow \mathsf{G.KeyGen}(1^n, 1^N)$ *and give* $(\mathsf{PK}, \mathbf{gsk})$ *to* $\mathcal{A}$*.*
2. $\mathcal{A}$ *outputs two identities* $i_0, i_1 \in [N]$*, along with a message* M*. A random bit* b *is chosen, and* $\mathcal{A}$ *is given* $\mathsf{G.Sign}(\mathsf{gsk}[i_b], M)$*. Finally,* $\mathcal{A}$ *outputs a bit* b'*.*

$\mathcal{A}$ *succeeds (denoted* Succ*) if* $b' = b$*, and the advantage of* $\mathcal{A}$ *is* $\left|\Pr[\mathsf{Succ}] - \frac{1}{2}\right|$*.*

Traceability means that it should be infeasible for an adversary who corrupts some set of users $\mathcal{C}$ to output a valid signature that cannot be traced to some member of $\mathcal{C}$.

Definition 2. *A group signature scheme* $\mathcal{GS} = (\mathsf{G.KeyGen}, \mathsf{G.Sign}, \mathsf{G.Vrfy}, \mathsf{G.Open})$ *is* traceable *if for all polynomials* $N(\cdot)$ *and all probabilistic polynomial-time adversaries* $\mathcal{A}$*, the success probability of* $\mathcal{A}$ *in the following experiment is negligible in* n*:*

1. *Compute* $(\mathsf{PK}, \mathsf{TK}, \mathbf{gsk}) \leftarrow \mathsf{G.KeyGen}(1^n, 1^N)$ *and give* PK *and* TK *to* $\mathcal{A}$*.*
2. $\mathcal{A}$ *may query the following oracles adaptively and in any order:*
   - *A* Corrupt *oracle that on input* $i \in [N]$ *returns* $\mathsf{gsk}[i]$*.*
   - *A* Sign *oracle that on input* i, M *outputs* $\mathsf{G.Sign}(\mathsf{gsk}[i], M)$*.*

Let $\mathcal{C}$ be the set of identities queried to Corrupt.

3. *At some point, $\mathcal{A}$ outputs a message M and a signature σ.*

$\mathcal{A}$ succeeds if (1) $\mathsf{G.Vrfy}(\mathsf{PK}, M, \sigma) = 1$ *and (2)* $\mathsf{Sign}(i, M)$ *was never queried for $i \notin \mathcal{C}$, yet (3)* $\mathsf{G.Open}(\mathsf{TK}, M, \sigma) \notin \mathcal{C}$.

3.2 Our Construction

We let n be the security parameter, $q = \mathsf{poly}(n), m \geq 8n \log q$ and $s \geq C\sqrt{n \log q} \cdot \omega(\sqrt{\log m})$ be parameters of the system. We let $H : \{0,1\}^* \to \mathbb{Z}_q^n$ be a hash function, to be modeled as a random oracle.

$\mathsf{G.KeyGen}(1^n, 1^N)$: First compute $(\mathbf{B}_1, \mathbf{S}_1), \ldots, (\mathbf{B}_N, \mathbf{S}_N) \leftarrow \mathsf{TrapSamp}(1^n, 1^m, q)$ and then, for $1 \leq i \leq N$, compute $(\mathbf{A}_i, \mathbf{T}_i) \leftarrow \mathsf{OrthoSamp}(1^n, 1^m, q, \mathbf{B}_i)$. Output $\mathsf{PK} = \left((\mathbf{A}_i, \mathbf{B}_i)_{i=1}^N\right)$ as the public key, $\mathsf{TK} = \left(\mathbf{S}_i\right)_{i=1}^N$ as the tracing key, and $\mathsf{gsk} = (\mathbf{T}_i)_{i=1}^N$ as the users' signing keys.

$\mathsf{G.Sign}(\mathsf{gsk}[j], M)$: To sign message M using secret key $\mathsf{gsk}[j] = \mathbf{T}_j$, choose random $r \leftarrow \{0,1\}^n$, set $\bar{M} = M \| r$, and then compute $\mathbf{h}_i = H(\bar{M} \| i)$ for $1 \leq i \leq N$. Then:

- Compute $\mathbf{e}_j \leftarrow \mathsf{GPVInvert}(\mathbf{A}_j, \mathbf{T}_j, s, \mathbf{h}_j)$.
- For $i \neq j$, choose $\mathbf{e}_i \in \mathbb{Z}_q^m$ uniformly subject to the condition that $\mathbf{A}_i \mathbf{e}_i = \mathbf{h}_i \pmod q$.

For all i, sample $\mathbf{s}_i \leftarrow \mathbb{Z}_q^n$ and compute $\mathbf{z}_i = \mathbf{B}_i^T \mathbf{s}_i + \mathbf{e}_i \pmod q \in \mathbb{Z}_q^m$. Finally, construct an NIWI proof π for the gap language $L_{s,\gamma}$ as discussed in Section 2.5 (and using the witness $(\mathbf{s}_i, i)$). Output the signature $(r, \mathbf{z}_1, \ldots, \mathbf{z}_N, \pi)$.

$\mathsf{G.Vrfy}(\mathsf{PK}, M, \sigma)$: Parse the signature as $(r, \mathbf{z}_1, \ldots, \mathbf{z}_N, \pi)$ and set $\bar{M} = M \| r$. Output 1 iff the proof π is correct, and $\mathbf{A}_i \mathbf{z}_i = H(\bar{M} \| i) \pmod q$ for all i.

$\mathsf{G.Open}(\mathsf{TK}, M, \sigma)$: Parse the signature as $(r, \mathbf{z}_1, \ldots, \mathbf{z}_N, \pi)$. Using the $\{\mathbf{S}_i\}$, output the smallest index i for which[3] $\mathsf{dist}(\mathcal{L}(\mathbf{B}_i^T), \mathbf{z}_i) \leq s\sqrt{m}$.

We first check correctness. Let $(r, \mathbf{z}_1, \ldots, \mathbf{z}_N, \pi)$ be a signature produced by an honest signer. It is clear that π is a valid proof. Moreover, for any i we have

$$\mathbf{A}_i \mathbf{z}_i = \mathbf{A}_i(\mathbf{B}_i^T \mathbf{s}_i + \mathbf{e}_i) = \mathbf{A}_i \mathbf{e}_i = H(\bar{M} \| i) \pmod q,$$

and so verification succeeds. Correctness of the opening algorithm follows easily.

Theorem 1. *Let m, q, and s be as described above. If $\mathsf{LWE}_{m,q,\alpha}$ is hard for $\alpha = s/(q\sqrt{2})$, and the proof system used is witness indistinguishable, then the group signature scheme described above is anonymous. If GapSVP_γ is hard for $\gamma = O(n \log^4 n)$, then the group signature scheme described above is traceable.*

[3] Soundness of the proof system ensures that if σ is valid, then some such i exists except with negligible probability.

We note that for values of s as described above, the hardness of $\mathsf{LWE}_{m,q,\alpha}$ is implied by the difficulty of finding a quantum algorithm for approximating $\mathsf{GapSVP}_{\hat{\gamma}}$ for $\hat{\gamma} = \widetilde{O}(n/\alpha)$ [27], so our entire scheme can be based on the difficulty of finding a quantum algorithm for GapSVP.

We prove anonymity in Section 3.3 and traceability in Section 3.4.

3.3 Anonymity

Fix $N = \mathsf{poly}(n)$ and let $\mathcal{A}$ be a PPT adversary attacking the group signature scheme in the sense of Definition 1. Let G_0 denote the experiment of Definition 1 with $b = 0$, and let G_1 be the same experiment with $b = 1$. We consider a sequence of experiments $\mathsf{G}_0, \mathsf{G}'_0, \mathsf{G}'_1, \mathsf{G}_1$ and show that each experiment is indistinguishable from the one preceding it. This implies anonymity.

We review G_0 as applied to our group signature scheme. First, the key-generation algorithm $\mathsf{G.KeyGen}(1^n, 1^N)$ is run and $\mathcal{A}$ is given the public key $\mathsf{PK} = \left((\mathbf{A}_i, \mathbf{B}_i)_{i=1}^N\right)$ and the users' secret keys $\mathsf{gsk} = (\mathbf{T}_i)_{i=1}^N$, where each $\mathbf{B}_i$ is statistically close to uniform and $(\mathbf{A}_i, \mathbf{T}_i) \leftarrow \mathsf{OrthoSamp}(1^n, 1^m, q, \mathbf{B}_i)$. (The tracing key TK is irrelevant in the CPA-version of the anonymity experiment that we are considering.) Next, $\mathcal{A}$ outputs i_0, i_1, M, and is given a signature of user i_0 on M, computed as follows. Let $\mathbf{h}_i = H(M \| r \| i)$, for a random $r \in \{0,1\}^n$. Then $\mathbf{e}_{i_0}$ is computed as $\mathbf{e}_{i_0} \leftarrow \mathsf{GPVInvert}(\mathbf{A}_{i_0}, \mathbf{T}_{i_0}, s, \mathbf{h}_{i_0})$, whereas $\mathbf{e}_i$ (for $i \neq i_0$) is chosen uniformly subject to the condition that $\mathbf{A}_i \mathbf{e}_i = \mathbf{h}_i \pmod q$. Then, for all $i \in [N]$, choose random $\mathbf{s}_i \leftarrow \mathbb{Z}_q^n$ and compute $\mathbf{z}_i = \mathbf{B}_i^T \mathbf{s}_i + \mathbf{e}_i$. Finally, a proof π is generated and $\mathcal{A}$ is given the signature $(r, \mathbf{z}_1, \ldots, \mathbf{z}_N, \pi)$.

In G'_0 we introduce the following modification with respect to G_0: when generating the signature, we now compute $\mathbf{e}_{i_0} \leftarrow \mathsf{GPVInvert}(\mathbf{A}_{i_0}, \mathbf{T}_{i_0}, s, \mathbf{h}_{i_0})$ and $\mathbf{e}_{i_1} \leftarrow \mathsf{GPVInvert}(\mathbf{A}_{i_1}, \mathbf{T}_{i_1}, s, \mathbf{h}_{i_1})$. (For $j \notin \{i_0, i_1\}$, the value $\mathbf{e}_j$ is computed as before.)

Claim. If the $\mathsf{LWE}_{m,q,\alpha}$ problem is hard, then G_0 and G'_0 are computationally indistinguishable.

Proof. Recall (cf. Lemma 1) that hardness of the $\mathsf{LWE}_{m,q,\alpha}$ problem implies hardness of the $\widehat{\mathsf{LWE}}_{m,q,\alpha q\sqrt{2}}$ problem. We use $\mathcal{A}$ to construct a PPT algorithm $\mathcal{D}$ for the $\widehat{\mathsf{LWE}}_{m,q,\alpha q\sqrt{2}}$ problem. $\mathcal{D}$ is given as input $(\mathbf{B}, \mathbf{y}) \in \mathbb{Z}_q^{n \times m} \times \mathbb{Z}_q^m$, where $\mathbf{B}$ is uniform and $\mathbf{y}$ is either uniform or equal to $\mathbf{B}^T\mathbf{s} + \mathbf{e}$ for $\mathbf{e} \sim D_{\mathbb{Z}^m, \alpha q\sqrt{2}}$.

$\mathcal{D}$ first chooses a random index $i^* \leftarrow [N]$ and sets $\mathbf{B}_{i^*} = \mathbf{B}$. For all $i \neq i^*$, it chooses $\mathbf{B}_i$ uniformly at random. Then, for $1 \leq i \leq N$ algorithm $\mathcal{D}$ computes $(\mathbf{A}_i, \mathbf{T}_i) \leftarrow \mathsf{OrthoSamp}(1^n, 1^m, q, \mathbf{B}_i)$. It gives $\mathsf{PK} = \left((\mathbf{A}_i, \mathbf{B}_i)_{i=1}^N\right)$ and $\mathsf{gsk} = (\mathbf{T}_i)_{i=1}^N$ to $\mathcal{A}$. All H-queries of $\mathcal{A}$ are answered with random elements from the appropriate domain.

Eventually $\mathcal{A}$ outputs two identities $i_0, i_1 \in [N]$ along with a message M. If $i^* \neq i_1$ then $\mathcal{D}$ outputs a random bit and aborts. Otherwise, $\mathcal{D}$ creates a signature

by choosing random $r \in \{0,1\}^n$ and fixing[4] $\mathbf{h}_{i_1} \stackrel{\text{def}}{=} H(M\|r\|i_1) = A_{i_1}\mathbf{y}$. (The value $\mathbf{h}_i = H(M\|r\|i)$ for $i \neq i_1$ is chosen uniformly.) Then $\mathcal{D}$ computes $\mathbf{e}_{i_0} \leftarrow \mathsf{GPVInvert}(\mathbf{A}_{i_0}, \mathbf{T}_{i_0}, s, \mathbf{h}_{i_0})$ and, for $i \notin \{i_0, i_1\}$, chooses $\mathbf{e}_i$ uniformly subject to the condition that $\mathbf{A}_i\mathbf{e}_i = \mathbf{h}_i \pmod q$. ($\mathcal{D}$ does not explicitly compute any value $\mathbf{e}_{i_1}$.) For $i \neq i_1$, the ciphertext $\mathbf{z}_i$ is computed as in G_0 and G'_0. However, $\mathcal{D}$ sets $\mathbf{z}_{i_1} = \mathbf{y}$.

Let $\mathcal{D}_{\text{rand}}$ denote the above experiment when $\mathcal{D}$'s input $\mathbf{y}$ is uniformly distributed. We claim that $\mathcal{A}$'s view in $\mathcal{D}_{\text{rand}}$ is statistically close to its view in G_0. Indeed:

- In G_0 we have $\mathbf{h}_{i_1}$ chosen uniformly in $\mathbb{Z}_q^n$; then $\mathbf{e}_{i_1}$ is chosen uniformly subject to $\mathbf{A}_{i_1}\mathbf{e}_{i_1} = \mathbf{h}_{i_1}$; and finally $\mathbf{z}_{i_1} = \mathbf{B}_{i_1}^T\mathbf{s}_{i_1} + \mathbf{e}_{i_1}$.
- In $\mathcal{D}_{\text{rand}}$ we have $\mathbf{z}_{i_1} = \mathbf{y} = \mathbf{B}_{i_1}^T\mathbf{s}_{i_1} + \mathbf{e}_{i_1}$ for $\mathbf{e}_{i_1}$ chosen uniformly in $\mathbb{Z}_q^m$; then $\mathbf{h}_{i_1} = \mathbf{A}_{i_1}\mathbf{e}_{i_1}$.

To see that these are statistically close, we demonstrate that the choice of $\mathbf{e}_{i_1}$ in G_0 is statistically close to uniform over $\mathbb{Z}_q^m$. We view $\mathbf{A}$ as a function from $\mathbb{Z}_q^m \to \mathbb{Z}_q^n$, and note that this function is regular. Furthermore, since the columns of $\mathbf{A}$ generate all of $\mathbb{Z}_q^n$ with all but negligible probability (over the choice of $\mathbf{A}$), our randomly chosen $\mathbf{h}$ is uniform over the image of $\mathbf{A}$. For a regular function, choosing a uniform element from the image, followed by a uniform element from its pre-image, is equivalent to choosing a uniform element from the domain, as is done in $\mathcal{D}_{\text{rand}}$.

On the other hand, let $\mathcal{D}_{\text{LWE}}$ denote the above experiment when $\mathcal{D}$'s input $\mathbf{y}$ is distributed according to $\mathbf{y} = \mathbf{B}^T\mathbf{s} + \mathbf{e}$ for $\mathbf{e} \sim D_{\mathbb{Z}^m,\alpha q\sqrt{2}}$. We claim that $\mathcal{A}$'s view in $\mathcal{D}_{LWE}$ is statistically close to its view in G'_0. Indeed:

- In experiment G'_0 we have $\mathbf{h}_{i_1}$ chosen uniformly in $\mathbb{Z}_q^n$. Next, we compute $\mathbf{e}_{i_1} \leftarrow \mathsf{GPVInvert}(\mathbf{A}_{i_1}, \mathbf{T}_{i_1}, s, \mathbf{h}_{i_1})$; and finally $\mathbf{z}_{i_1} = \mathbf{B}_{i_1}^T\mathbf{s}_{i_1} + \mathbf{e}_{i_1}$.
- In $\mathcal{D}_{LWE}$ we have $\mathbf{z}_{i_1} = \mathbf{y} = \mathbf{B}_{i_1}^T\mathbf{s}_{i_1} + \mathbf{e}_{i_1}$ for $\mathbf{e}_{i_1} \sim D_{\mathbb{Z}^m,\alpha\cdot q\cdot\sqrt{2}}$; then $\mathbf{h}_{i_1} = \mathbf{A}_{i_1}\mathbf{e}_{i_1}$.

The above are easily seen to be statistically close for our choice of parameters, again using the results of [19]. Since the probability that $\mathcal{D}$ does not abort is $1/N$, and its decision to abort is independent of $\mathcal{A}$'s success, the proof is complete.

The rest of the proof of anonymity is straightforward, and so we merely provide a sketch. Experiment G'_1 is identical to G'_0 with the exception that the proof π is now computed using the witness $(\mathbf{s}_{i_1}, i_1)$ rather than $(\mathbf{s}_{i_0}, i_0)$. Witness indistinguishability of the proof system implies that G'_1 and G'_0 are computationally indistinguishable.

Computational indistinguishability of G'_1 and G_1 (the experiment from Definition 1 with $b = 1$) can be proved exactly as in the proof o the previous claim.

3.4 Traceability

Fix $N = \mathsf{poly}(n)$ and let $\mathcal{A}$ be a PPT adversary attacking the group signature scheme in the sense of Definition 2. We construct a PPT forger $\mathcal{F}$ for the GPV

[4] Note that, except with negligible probability, $H(M\|r\|i_1)$ has not been queried thus far.

signature scheme [19] (in the random oracle model) whose success probability is polynomially related to that of $\mathcal{A}$. Since the GPV signature scheme is secure assuming hardness of the GapSVP_γ problem, this completes the proof.

We first observe that we may, without loss of generality, assume that $\mathcal{A}$ never corrupts *all* users in $[N]$ because $\mathcal{A}$ can succeed with only negligible probability in this case. (Given a valid signature $(r, \mathbf{z}_1, \ldots, \mathbf{z}_N, \pi)$, soundness of the proof system implies that $\mathsf{G.Open}$ outputs some $i \in [N]$ except with negligible probability.) We will assume this in what follows.

$\mathcal{F}$ is given a public key $\mathbf{A}$ for the GPV signature scheme, and begins by choosing a random index $i^* \in [N]$ and setting $\mathbf{A}_{i^*} = \mathbf{A}$. Next, it computes the values $(\mathbf{B}_{i^*}, \mathbf{S}_{i^*}) \leftarrow \mathsf{OrthoSamp}(1^n, 1^m, q, \mathbf{A}_{i^*})$. For all the remaining indices $i \neq i^*$, the forger computes the values $(\mathbf{B}_i, \mathbf{S}_i) \leftarrow \mathsf{TrapSamp}(1^n, 1^m, q)$ and $(\mathbf{A}_i, \mathbf{T}_i) \leftarrow \mathsf{OrthoSamp}(1^n, 1^m, q, \mathbf{B}_i)$ exactly as in the legitimate key-generation algorithm. After this, $\mathcal{F}$ gives $\mathsf{PK} = (\mathbf{A}_i, \mathbf{B}_i)_{i=1}^N$ and $\mathsf{TK} = (\mathbf{S}_i)_{i=1}^N$ to $\mathcal{A}$. We note that by Corollary 1, the distribution of these keys is statistically close to the distribution that is expected by the adversary.

$\mathcal{F}$ answers random oracle queries of $\mathcal{A}$ by simply passing these queries to its own random oracle. $\mathcal{F}$ responds to the other queries of $\mathcal{A}$ as follows:

- $\mathsf{Corrupt}(i)$: if $i \neq i^*$ then $\mathcal{F}$ gives $\mathbf{T}_i$ to $\mathcal{A}$, while if $i = i^*$ then $\mathcal{F}$ aborts.
- $\mathsf{Sign}(i, M)$: If $i \neq i^*$ then $\mathcal{F}$ computes the signature using $\mathbf{T}_i$ and the honest signing algorithm. If $i = i^*$, then:
  1. $\mathcal{F}$ chooses random $r \in \{0,1\}^n$ and queries its own signing oracle on the message $M \| r \| i^*$. It receives in return a signature $\mathbf{e}_{i^*}$.
  2. The remainder of the signature is computed using the honest signing algorithm. (Note that computation of $\mathbf{e}_{i^*}$ the only aspect of signing that relies on the secret key of user i^*.)

Let $\mathcal{C}$ denote the set of identities that $\mathcal{A}$ has queried to $\mathsf{Corrupt}$. (Recall that if $\mathcal{F}$ has not aborted, then $i^* \notin \mathcal{C}$.) At some point $\mathcal{A}$ outputs a message M and signature $\sigma = (r, \mathbf{z}_1, \ldots, \mathbf{z}_N, \pi)$. Assume $\mathsf{G.Vrfy}(\mathsf{PK}, M, \sigma) = 1$, and that $\mathsf{Sign}(i, M)$ was never queried for $i \notin \mathcal{C}$. Since $\mathcal{F}$ has the tracing key TK, it can compute $j \leftarrow \mathsf{G.Open}(\mathsf{TK}, M, \sigma)$. If $j \neq i^*$ then $\mathcal{F}$ aborts. Otherwise, $\mathcal{F}$ does:

1. Use $\mathbf{S}_{i^*}$ to recover $\mathbf{e}_{i^*}$ such that
   - $||\mathbf{e}_{i^*}||_\infty \leq s\sqrt{m}$, and
   - $\mathbf{z}_{i^*} - \mathbf{e}_{i^*} \in \mathcal{L}(\mathbf{B}_{i^*}^T)$.
2. Output the forgery $(M \| r \| i^*, \mathbf{e}_{i^*})$.

Let ϵ denote the probability with which $\mathcal{A}$ succeeds in the experiment of Definition 2. It is easy to see that $\mathcal{F}$ aborts with probability at most[5] $(N-1)/N$ and, conditioned on not aborting, the view of $\mathcal{A}$ when run as a sub-routine by $\mathcal{F}$ is statistically close to its view in the experiment described in Definition 2. Thus, with probability at least ϵ/N it holds that $\mathcal{A}$ outputs (M, σ) with

[5] Actually, $\mathcal{F}$ aborts with probability at most $(N-1)/N + \mathsf{negl}(n)$, where the negligible term arises from the possibility that $\mathcal{A}$ violates soundness of the proof system. We ignore this for simplicity.

$\mathsf{G.Vrfy}(\mathsf{PK}, M, \sigma) = 1$ and $\mathsf{G.Open}(\mathsf{TK}, M, \sigma) = i^*$, and where $\mathcal{A}$ never queried $\mathsf{Sign}(i^*, M)$. We show that whenever this occurs, then $\mathcal{F}$ outputs a valid forgery (except with negligible probability).

Fix (M, σ) such that the above hold, and let $\sigma = (r, \mathbf{z}_1, \ldots, \mathbf{z}_N, \pi)$. Since $\mathsf{G.Open}(\mathsf{TK}, M, \sigma) = i^*$, this implies that $\mathcal{F}$ will indeed be able to recover $\mathbf{e}_{i^*}$ such that (1) $||\mathbf{e}_{i^*}||_\infty \leq s\sqrt{m}$ and (2) $\mathbf{z}_{i^*} - \mathbf{e}_{i^*} \in \mathcal{L}(\mathbf{B}_{i^*}^T)$. Moreover, since $\mathsf{G.Vrfy}(\mathsf{PK}, M, \sigma) = 1$ we have $\mathbf{A}_{i^*}\mathbf{z}_{i^*} = H(M\|r\|i^*)$; since $\mathbf{A}_{i^*}(\mathbf{z}_{i^*} - \mathbf{e}_{i^*}) = \mathbf{0}$ this means $\mathbf{A}_{i^*}\mathbf{e}_{i^*} = H(M\|r\|i^*)$. Thus $\mathbf{e}_{i^*}$ is a valid GPV signature on the message $M\|r\|i^*$. Since $\mathcal{A}$ never queried $\mathsf{Sign}(i^*, M)$, we know that $\mathcal{F}$ never queried its own signing oracle for a signature on $M\|r\|i^*$. It follows that the output of $\mathcal{F}$ is indeed a valid forgery.

Acknowledgments. We thank Chris Peikert for pointing out that the results from [25] can be used to prove Lemma 1, and the anonymous reviewers of Asiacrypt 2010 for helpful comments.

References

1. Ajtai, M.: Generating hard instances of lattice problems. In: 28th Annual ACM Symp. on Theory of Computing (STOC), pp. 99–108. ACM Press, New York (1996)
2. Ajtai, M.: Generating hard instances of the short basis problem. In: Wiedermann, J., Van Emde Boas, P., Nielsen, M. (eds.) ICALP 1999. LNCS, vol. 1644, pp. 1–9. Springer, Heidelberg (1999)
3. Alwen, J., Peikert, C.: Generating shorter bases for hard random lattices. In: STACS. Dagstuhl Seminar Proceedings, vol. 09001, pp. 75–86. Schloss Dagstuhl (2009), http://drops.dagstuhl.de/portals/STACS09/
4. Ateniese, G., Camenisch, J., Hohenberger, S., de Medeiros, B.: Practical group signatures without random oracles. Cryptology ePrint Archive, report 2005/385 (2005)
5. Ateniese, G., Camenisch, J., Joye, M., Tsudik, G.: A practical and provably secure coalition-resistant group signature scheme. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 255–270. Springer, Heidelberg (2000)
6. Ateniese, G., Song, D.X., Tsudik, G.: Quasi-efficient revocation in group signatures. In: Blaze, M. (ed.) FC 2002. LNCS, vol. 2357, pp. 183–197. Springer, Heidelberg (2003)
7. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: 1st ACM Conference on Computer and Communications Security, pp. 62–73. ACM Press, New York (1993)
8. Bellare, M., Micciancio, D., Warinschi, B.: Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 614–629. Springer, Heidelberg (2003)
9. Bellare, M., Shi, H., Zhang, C.: Foundations of group signatures: The case of dynamic groups. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 136–153. Springer, Heidelberg (2005)
10. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)

11. Boyen, X., Waters, B.: Compact group signatures without random oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 427–444. Springer, Heidelberg (2006)
12. Boyen, X., Waters, B.: Full-domain subgroup hiding and constant-size group signatures. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 1–15. Springer, Heidelberg (2007)
13. 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)
14. Camenisch, J., Lysyanskaya, A.: Signature schemes and anonymous credentials from bilinear maps. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 56–72. Springer, Heidelberg (2004)
15. 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)
16. Chaum, D., van Heyst, E.: Group signatures. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 257–265. Springer, Heidelberg (1991)
17. Cramer, R., Damgård, I., Schoenmakers, B.: Proofs 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)
18. 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)
19. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: 40thAnnual ACM Symposium on Theory of Computing (STOC), pp. 197–206. ACM Press, New York (2008)
20. Goldreich, O., Goldwasser, S.: On the limits of nonapproximability of lattice problems. Journal of Computer and System Sciences 60(3), 540–563 (2000)
21. Groth, J.: Fully anonymous group signatures without random oracles. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 164–180. Springer, Heidelberg (2007)
22. Kiayias, A., Yung, M.: Group signatures with efficient concurrent join. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 198–214. Springer, Heidelberg (2005)
23. Micciancio, D., Vadhan, S.: Statistical zero-knowledge proofs with efficient provers: Lattice problems and more. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 282–298. Springer, Heidelberg (2003)
24. Peikert, C.: Public-key cryptosystems from the worst-case shortest vector problem. In: 41st Annual ACM Symposium on Theory of Computing (STOC), pp. 333–342. ACM Press, New York (2009)
25. Peikert, C.: An efficient and parallel gaussian sampler for lattices. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 80–97. Springer, Heidelberg (2010)
26. Peikert, C., Vaikuntanathan, V.: Noninteractive statistical zero-knowledge proofs for lattice problems. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 536–553. Springer, Heidelberg (2008)
27. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: 37th Annual ACM Symposium on Theory of Computing (STOC), pp. 84–93. ACM Press, New York (2005)

Lattice-Based Blind Signatures

Markus Rückert*

Technische Universität Darmstadt
Department of Computer Science
Cryptography and Computeralgebra, Germany
markus.rueckert@cased.de

Abstract. Blind signatures (BS), introduced by Chaum, have become a cornerstone in privacy-oriented cryptography. Using hard lattice problems, such as the shortest vector problem, as the basis of security has advantages over using the factoring or discrete logarithm problems. For instance, lattice operations are more efficient than modular exponentiation and lattice problems remain hard for quantum and subexponential-time adversaries. Generally speaking, BS allow a signer to sign a message without seeing it, while retaining a certain amount of control over the process. In particular, the signer can control the number of issued signatures. For the receiver of the signature, this process provides perfect anonymity, e.g., his spendings remain anonymous when using BS for electronic money.

We provide a positive answer to the question of whether it is possible to implement BS based on lattice problems. More precisely, we show how to turn Lyubashevsky's identification scheme into a BS scheme, which has almost the same efficiency and security in the random oracle model. In particular, it offers quasi-linear complexity, statistical blindness, and its unforgeability is based on the hardness of worst-case lattice problems with an approximation factor of $\widetilde{\mathcal{O}}(n^5)$ in dimension n. Moreover, it is the first blind signature scheme that supports leakage-resilience, tolerating leakage of a $(1-o(1))$ fraction of the secret key in a model that is inspired by Katz and Vaikuntanathan.

Keywords: Blind signatures, post-quantum, lattices, provable security, leakage resilience.

1 Introduction

Since Chaum proposed his idea of blind signatures [Cha82], it has become an important primitive for anonymous Internet banking, e-voting (e.g., [RHOAGZ07]), as well as for oblivious transfer [CNS07]. These applications will retain their importance in both, near and far future. As for the near future, we are convinced that current factoring and discrete logarithm based instantiations are efficient and secure. *But for how long?*

Today, when building provably secure cryptographic schemes, one also has to anticipate emerging technologies that may lead to new attacks. This is why we

* This work was supported by CASED (www.cased.de).

M. Abe (Ed.): ASIACRYPT 2010, LNCS 6477, pp. 413–430, 2010.

typically try to use the mildest possible assumptions. Let us consider the example of quantum computers as a metaphor for these future developments. In the quantum-age, the cryptographic assumptions change with the leap in computing power that quantum computers will provide. There are only a few cryptographic assumptions that are conjectured to be *post-quantum*, i.e., they are considered to withstand quantum computer attacks. One of those assumptions is the hardness of finding short vectors in a lattice. Even for today, there are benefits when building cryptography upon hard lattice problems because, unlike factoring, they withstand subexponential attacks and the best known algorithms, e.g., [AKS01], have an exponential complexity in the lattice dimension. Furthermore, lattice problems typically allow a worst-case to average-case reduction that goes back to Ajtai [Ajt96]. It states that a randomly chosen instance of a certain lattice problem is at least as hard as the worst-case instance of a related lattice problem. Thus, choosing secure keys is easy. This reduction was later on adapted to work with ideal lattices by Lyubashevsky and Micciancio [LM06] because ideal lattices offer a compact public-key representation and very efficient operations at the expense of a slightly stronger assumption.

The security model, mainly influenced by Juels, Luby, and Ostrovsky [JLO97] as well as Pointcheval and Stern [PS00], requires blind signature schemes to satisfy blindness and one-more unforgeability. Blindness states that the signer must not obtain any information on the signed messages and one-more unforgeability means that an adversary cannot obtain more signatures than there were interactions with the signer.

Our Contribution. We construct the first lattice-based blind signature scheme. It is inspired by Lyubashevsky's ID scheme [Lyu08] in combination with the Fiat-Shamir paradigm [FS86]. It is unconditionally blind, selective-failure blind [CNS07], and one-more unforgeable in the random oracle model [BR93] if standard lattice problems in ideal lattices [LM06] are hard in the worst-case. With its four moves it is quite efficient. All operations have quasi-linear complexity and all keys and signatures require a quasi-linear amount of storage bits, with respect to the main parameter n. Moreover, it is leakage resilient according to a model inspired by Katz and Vaikuntanathan [KV09]. Let L be the bit-length of the secret key. Our scheme remains secure, even if the adversary obtains $L(1 - o(1))$ bits of the secret key via arbitrary side channels. This brings the security model closer to reality, where the adversary may obtain information about the secret key, e.g, via (remote) timing attacks or by having physical access to the signing device. When applied in e-voting or e-cash schemes, such a resilience also helps against insider attacks and may improve the trust that we are willing to grant these schemes. Another application of our construction is identity-based blind signatures, when combined with [Rüc10].

Our scheme is also the first leakage resilient blind signature scheme and our results in this respect are applicable to Lyubashevsky's ID and signature schemes [Lyu08, Lyu09]. It may be possible to use an analogue of Pointcheval and Stern's approach [PS00] to turn the leakage resilient variants [KV09, ADW09] of the Okamoto-Schnorr signature scheme [Sch91, Oka92] into blind signature schemes.

Table 1. Comparison of RSA, Okamoto-Schnorr, and our blind signature scheme

Scheme	Secure until	Security (bits)	Moves	KeyGen	Sign	Verify
RSA-1229	2012	Current (76)	2	95 ms	16 ms	5 ms
RSA-3313	2050	Medium (102)	2	1250 ms	46 ms	6 ms
RSA-15424	2282	Future (256)	2	251849 ms	2134 ms	20 ms
OS-1229	2012	Current (76)	3	16 ms	64 ms	24 ms
OS-3313	2050	Medium (102)	3	46 ms	184 ms	69 ms
OS-15424	2282	Future (256)	3	2134 ms	8536 ms	3201 ms
Section 3 ($n = 1024$)	2012	Current (76)	4	37 ms	220 ms	33 ms
Section 3 ($n = 2048$)	2050	Medium (102)	4	52 ms	283 ms	57 ms
Section 3 ($n = 8192$)	2282	Future (256)	4	305 ms	1175 ms	320 ms

The table compares our scheme with RSA and Okamoto-Schnorr for various moduli according to [Len05] (Current, Medium) and [ECR10] (Future). The bitlengths can be computed on `www.keylength.com`. For our blind signature scheme, we propose three *optimized* parameter sets for the same security levels based on [RS10], which provides a framework for choosing secure parameters for lattice-based cryptography. Note that the parameters for RSA and OS do *not* take potential quantum-computer attacks into account. All timings are averaged over 1000 random instances.

However, it is unclear whether this will actually work and whether it will be efficient.

Table 1 compares RSA and Okamoto-Schnorr (OS) blind signatures with our construction in terms of computational cost. For all schemes, we propose parameter sets for current, medium, and future security levels. We believe that RSA is a good basis for comparison because it is easy to understand and very efficient as signing only involves two modular exponentiations and verification can be done in a single one (small exponent). We do *not* count multiplications. As observed in [BNPS03], the security of the RSA blind signature scheme is based on a specially tailored interactive assumption that is stronger than the original RSA assumption [BMV08]. Taking all this into account, the timings observed for RSA provide an optimistic lower bound for current practical and secure schemes. The timings for OS are expected timings based on the number of modular exponentiations, *not* counting multiplications. We include OS because it follows the typical 3-move structure and is based on a standard assumption. It is therefore closer to our protocol. The timings were obtained on an AMD Opteron CPU, running at 2.3 GHz. For RSA and OS, we have used OpenSSL 0.9.8g, which is supposed to be very efficient. For our blind signature schemes, we did a straightforward implementation, which certainly leaves room for improvements. Here, the timings reflect the *full* scheme.

From Table 1, we clearly see that our scheme benefits from its quasi-linear complexity, especially in higher levels of security. In addition, for our scheme, we

can have various trade-offs between signature size and speed. For more details, refer to the full version [Rüc08]. There, we also show how to optimize the key and signature sizes, which are typically large in lattice-based constructions.

We believe that our work is an important contribution because the previous efficient constructions, such as [Cha82, PS97, PS00, Abe01, BNPS03, CKW04, Oka06], have one thing in common: they are built upon classic number theoretic assumptions, like the hardness of factoring large integers or computing discrete logarithms. The more recent approaches, e.g., by Boldyreva [Bol03] or Okamoto [Oka06], tend to use pairings that yield very elegant constructions. They, however, are again based on the discrete logarithm problem in this specific setting. None of the above schemes remains secure in the presence of reasonably large quantum computers, where both factoring and computing discrete logarithms become easy due to the seminal work of Shor [Sho97].

Main Obstacles. For every blind signature scheme, one has to overcome three basic obstacles. The scheme needs to be blind, one-more unforgeable, and at the same time complete. Blindness and unforgeability are already somewhat orthogonal because granting the user too much power to ensure blindness harms unforgeability and vice-versa. Since working with lattices, we do not have access to a cyclic group structure as in schemes that are based on the DDH or DL assumptions. There, blindness is typically easier to achieve by multiplying the message with a random group element. The result is again a random group element.

In lattices, we need to emulate this over an infinite structure via a filtering technique that is inspired by [Lyu08]. However, this technique introduces a completeness defect that even affects the interaction of an honest user with an honest signer. Thus, the protocol may need to be restarted. We show how this technique can be refined to allow a time-memory trade-off, reducing the number of expected restarts at the expense of only slightly larger signatures. When addressing this defect, we need additional means to ensure blindness over repetitions of the protocol. Our solution involves a statistically hiding commitment.

Similarly, the completeness defect has implications with respect to unforgeability as the user may claim that the protocol has failed, whereas it was indeed successful. Here, we extend the typical three-move structure to a four-move structure where the user needs to demonstrate that he or she could not obtain a valid signature. Such a last move, from user to signer, is highly unusual for blind signature schemes. We solve this issue by designing a special proof of failure and by employing a computationally binding commitment scheme.

All these issues, and the additional leakage resilience, need to be addressed simultaneously as they are interconnected. This leads to an intricate process of correctly setting up the numerous parameters and sets for our scheme in Table 2.

RSA-style Blind Signatures. One might think that RSA-style (hash $\rightarrow$ blind $\rightarrow$ invert $\rightarrow$ unblind) lattice-based blind signatures can be implemented using the preimage sampleable trapdoor function $f : D \subset \mathbb{Z}^m \rightarrow \mathbb{Z}_q^n$ from [GPV08]. If certain lattice problems are hard, it is hard to sample preimages from D (small norm) unless one knows short vectors x such that $f(x) = 0$. The user

would hash the message M using a full-domain hash $h \leftarrow \mathsf{H}(M)$ and blind using $M^* \leftarrow h + f(\beta)$ for $\beta \in D$. The signer would *sample* from $f^{-1}(M^*) \cap D$ and return the result σ^*. The function is compressing, so there are no unique preimages. Using β and the fact that f is linear, the user can compute $\sigma \leftarrow \sigma^* - \beta$, which passes verification: $f(\sigma) = f(\sigma^*) - f(\beta) = \mathsf{H}(M^*)$. For the proof, one would rely on an interactive "one-more" trapdoor inversion assumption akin to [BNPS03]. However, the adversary must never obtain a non-zero $x \in D$ such that $f(x) = 0$ because this would imply learning a piece of the secret key. Unfortunately, such an attack is easy: take $u \in D$ and send $M^* = f(u)$ to the signer, who returns σ^*. Now, $x = u - \sigma^*$ is small and $f(x) = 0$. Also, $x \neq 0$ with high probability because there are many preimages of $f(u)$.

Organization. After a brief preliminaries section, we propose our blind signature scheme in Section 3. There, we also provide a detailed analysis, including completeness, blindness, one-more unforgeability, and leakage resilience. The full version of the paper is [Rüc08]. There, we discuss how to choose practical parameters and prove all supporting lemmas for the theorems in Section 3.

2 Preliminaries

With n, we always denote the security parameter. The joint execution of two algorithms $\mathcal{A}$ and $\mathcal{B}$ in an interactive protocol with private inputs x to $\mathcal{A}$ and y to $\mathcal{B}$ is written as $(a, b) \leftarrow \langle \mathcal{A}(x), \mathcal{B}(y) \rangle$. The private outputs are a for $\mathcal{A}$ and b for $\mathcal{B}$. Accordingly, $\langle \mathcal{A}(x), \mathcal{B}(y) \rangle^k$ means that the interaction can take place up to k times. The statement $x \leftarrow_\$ X$ means that x is chosen uniformly at random from the finite set X. Recall that the statistical distance of two random variables X, Y over a discrete domain D is defined as $\Delta(X, Y) = 1/2 \sum_{a \in D} |\operatorname{Prob}[X = a] - \operatorname{Prob}[Y = a]|$. A function is negligible if it vanishes faster than $1/p(n)$ for any polynomial p. All logarithms are base 2 and we identify $\{1, \ldots, k\}$ with $[k]$.

We recall the definitions of blind signatures and commitments. Afterwards, we briefly recall some facts from lattice theory.

2.1 Blind Signatures

A blind signature scheme BS consists of three algorithms $(\mathsf{Kg}, \mathsf{Sign}, \mathsf{Vf})$, where Sign is an interactive protocol between a signer $\mathcal{S}$ and a user $\mathcal{U}$. The specification is as follows.

Key Generation. $\mathsf{Kg}(1^n)$ outputs a private signing key sk and a public verification key pk.

Signature Protocol. $\mathsf{Sign}(\mathsf{sk}, M)$ describes the joint execution of $\mathcal{S}$ and $\mathcal{U}$. The private output of $\mathcal{S}$ is a view $\mathcal{V}$ and the private output of $\mathcal{U}$ is a signature $\mathbf{s}$ on the message $M \in \mathcal{M}$ with message space $\mathcal{M}$ under sk. Thus, we write $(\mathcal{V}, \mathbf{s}) \leftarrow \langle \mathcal{S}(\mathsf{sk}), \mathcal{U}(\mathsf{pk}, M) \rangle$.

Signature Verification. The algorithm $\mathsf{Vf}(\mathsf{pk}, \mathbf{s}, M)$ outputs 1 if $\mathbf{s}$ is a valid signature on M under pk and otherwise 0.

Completeness is defined as with digital signature schemes, i.e., every honestly created signature for honestly created keys and for any messages $M \in \mathcal{M}$ has to be valid under this key. Views are interpreted as random variables, whose output is generated by subsequent executions of the respective protocol. Two views $\mathcal{V}_1$ and $\mathcal{V}_2$ are considered equal if they cannot be distinguished by any computationally unbounded algorithm with noticeable probability.

As for security, blind signatures have to satisfy two properties: blindness and one-more unforgeability [JLO97, PS00]. The notion of blindness is defined in an experiment $\mathsf{Exp}^{\mathsf{blind}}_{\mathcal{S}^*,\mathsf{BS}}$, where the adversarial signer $\mathcal{S}^*$ works in three modes. In mode *find*, it chooses two messages M_0, M_1 and interacts with two users in mode *issue*. Depending on a coin flip b, the first (second) user obtains a blind signature for M_b (M_{1-b}). After seeing the unblinded signatures in the original order, with respect to M_0, M_1, the signer has to guess the bit b in mode *guess*. If either of the user algorithms fails in outputting a valid signature, the signer is merely notified of the failure and does not get any signature. Below, we deal with aborts as an extension. Also note that we allow the adversary to keep a state that is fed back in subsequent calls. A scheme BS is (t, δ)-blind, if there is no adversary $\mathcal{S}^*$, running in time at most t, that wins the above experiment with advantage at least δ, where the advantage is defined as $\mathsf{Adv}^{\mathsf{blind}}_{\mathcal{S}^*,\mathsf{BS}} = \left|\mathrm{Prob}\left[\mathsf{Exp}^{\mathsf{blind}}_{\mathcal{S}^*,\mathsf{BS}}(n) = 1\right] - \frac{1}{2}\right|$. A scheme is *statistically* blind if the it is (∞, δ)-blind for a negligible δ. The second security property, one-more unforgeability, ensures that each completed interaction between signer and user yields at most one signature. It is formalized in the experiment $\mathsf{Exp}^{\mathsf{omf}}_{\mathcal{U}^*,\mathsf{BS}}$, where an adversarial user tries to output $\jmath$ valid signatures after $\ell < \jmath$ completed interactions with an honest signer. $\mathcal{H}$ is a family of random oracles.

A signature scheme BS is $(t, q_{\mathsf{Sign}}, q_{\mathsf{H}}, \delta)$-one-more unforgeable if there is no adversary $\mathcal{A}$, running in time at most t, making at most q_{Sign} signature queries and at most q_{H} hash oracle queries, that wins the above experiment with probability at least δ.

2.2 Extensions

We consider three extensions to the above security model for blind signatures: one deals with user aborts, the second with dishonestly chosen keys, and the third with leakage resilience.

Security Under Aborts. Blindness in the previous subsection does not cover the case where the protocol is aborted prematurely. There is the strengthened notion of selective failure blindness [CNS07], where the malicious signer may choose either M_0 or M_1 according to some secret distribution that makes the protocol fail. Preventing this generically is easy as was shown by Fischlin and Schröder in [FS09]. In the course of the discussion of our construction, we argue that it already is blind in this sense.

Adversely-chosen Keys. Consider the blindness experiment in [ANN06]. Instead of having the experiment select pk, sk, we can let the signer output pk.

Blindness may be harder to achieve in this setting. However, our construction remains blind in this stronger model as the proof does not exploit specifics about the key.

Leakage Resilience. Resilience to key leakage is a way to ensure security against side-channel attacks. In [KV09], Katz and Vaikuntanathan give a nice overview of past developments and the evolution of leakage resilience for authenticity and secrecy. Obviously, we are interested in authenticity in the special case of blind signatures. We model key leakage in the unforgeability experiment by adding a leakage oracle $\mathsf{Leak}(\cdot)$ to $\mathsf{Exp}^{\mathsf{omf}}_{\mathcal{U}^*,\mathsf{BS}}$. The adversary can adaptively query Leak with a series of functions f_i, $i \in \{1,\ldots,\kappa\}$, and receives $f_i(\mathsf{sk})$. The only restriction is that $\sum_{i=1}^{n} |f_i(\mathsf{sk})| < \lambda(|\mathsf{sk}|)$, where the function λ determines the amount of leakage that we are willing to tolerate. Notice that the signer's key does not have to evolve over time and its secret state consists of the secret key only. Furthermore, observe that this extension is only sensible as long as $\lambda(|\mathsf{sk}|) < \min\{|\mathsf{sk}|, |\mathbf{s}|\}$, where $|\cdot|$ denotes bit-length and $\mathbf{s}$ is a signature. Otherwise, the adversary could easily obtain the entire secret key or a signature of its choice. See the full version [Rüc08] for the experiment. To demonstrate leakage resilience, one has to show that the conditional min-entropy $H_\infty(\mathsf{sk}|\mathsf{Leak}(\mathsf{sk})) = \min_{\mathsf{sk}'}\{-\log(\mathrm{Prob}\left[\mathsf{sk} = \mathsf{sk}'|\mathsf{Leak}(\mathsf{sk})\right])\}$ of the secret key is still sufficiently large to prove security.

2.3 Commitments

Commitments typically work in two phases. First, one party publishes a commitment $C = \mathsf{com}(M;r) \in \{0,1\}^n$, $r \leftarrow_\$ \{0,1\}^n$, to a message $M \in \{0,1\}^*$ without revealing any information about it. This is the "hiding" property of the commitment scheme. In the second phase, the party can prove that C actually corresponds to M by revealing r. It is important that no algorithm can find a second message M' and randomness r' such that $C = \mathsf{com}(M';r')$, i.e., break the "binding" property. As usual, these properties are defined for families of such commitment functions. A scheme is (t,δ)-hiding (-binding) if there is no algorithm running in time at most t that can break the hiding (binding) property with probability at least δ. Both properties can be satisfied computationally or unconditionally but there is no scheme that is unconditionally hiding *and* unconditionally binding [Gol04].

For our scheme, we assume a statistically $\delta^{(h)}_{\mathsf{com}}$-hiding and computationally $(t_{\mathsf{com}}, \delta^{(b)}_{\mathsf{com}})$-binding commitment scheme. As we are interested in fully lattice-based schemes, we would like to point out that such commitment schemes can be built upon hard lattice problems [KTX08] but in practice, one rather uses cryptographic hash functions as a message authentication code. For example, using a lattice-based hash function [ADL+08].

2.4 Lattices

A lattice in $\mathbb{R}^n$ is a discrete set $\Lambda = \{\sum_{i=1}^{d} x_i\,\mathbf{b}_i \,|\, x_i \in \mathbb{Z}\}$, where $\mathbf{b}_1,\ldots,\mathbf{b}_d$ are linearly independent over $\mathbb{R}$. The matrix $\mathbf{B} = [\mathbf{b}_1,\ldots,\mathbf{b}_d]$ is a basis of the

lattice Λ and we write $\Lambda = \Lambda(\mathbf{B})$. The dimension of the lattice is d. The main computational problem in lattices is the shortest vector problem (SVP), where an algorithm is given a description, a basis, of a lattice Λ and is supposed to find the shortest vector $\mathbf{v} \in \Lambda \setminus \{\mathbf{0}\}$ with respect to a certain ℓ_p norm (up to an approximation factor). More precisely, find a vector $\mathbf{v} \in \Lambda \setminus \{\mathbf{0}\}$, such that $\|\mathbf{v}\|_p \leq \gamma \|\mathbf{w}\|_p$ for all $\mathbf{w} \in \Lambda \setminus \{\mathbf{0}\}$ for a fixed approximation factor $\gamma \geq 1$.

In this work, we are interested in a special family of lattices related to ideals in the ring $\mathbf{R} = \mathbb{Z}_q[X]/\langle \mathbf{g} \rangle$, where q is prime and $\mathbb{Z}_q = \{-(q-1)/2, \ldots, (q-1)/2\}$. We focus on $\mathbf{g} = X^n + 1$ and $n =$ "power of two" for efficiency reasons but it may be replaced by any irreducible polynomial over $\mathbb{Z}$. Then, our scheme and the analysis become only slightly more involved. We identify $\mathbf{f} \in \mathbf{R}$ with its coefficient vector $\mathbf{f} = (f_0, \ldots, f_{n-1}) \in \mathbb{Z}_q^n$. Furthermore, we denote elements of the $\mathbf{R}$-module $\mathbf{R}^m$ with $\hat{\mathbf{a}} = (\mathbf{a_0}, \ldots, \mathbf{a_{m-1}})$ or directly with $(a_0, \ldots, a_{mn-1}) \in \mathbb{Z}_q^{mn}$. Consequently, we define $\|\mathbf{f}\|_\infty = \|(f_0, \ldots, f_{n-1})\|_\infty$. The norm on $\mathbf{R}$ is a slight abuse of notation, but it will only be used if $\mathbf{f}$ has small coefficients over $\mathbb{Z}$. A lattice corresponds to an ideal $I \subset \mathbf{R}$ if and only if every lattice vector is the coefficient vector of a polynomial in I. The SVP problem easily translates to ideal lattices, where we call it ideal-SVP (ISVP).

The average-case hardness assumption for our construction relies on the problem of finding short vectors in the kernel of the family $\mathcal{H}(\mathbf{R}, m)$ of module homomorphisms $h_{\hat{\mathbf{a}} \in \mathbf{R}^m} : \mathbf{R}^m \to \mathbf{R}, \hat{\mathbf{x}} \mapsto \hat{\mathbf{a}} \circledast \hat{\mathbf{x}} = \sum_{i=0}^{m-1} \mathbf{a_i} \mathbf{x_i}$, when restricting the domain to $D' \subset \mathbf{R}$, i.e., restricting the coefficients in the input to $[-2d, 2d] \cap \mathbb{Z}$. This problem can be stated as the following collision problem [LM06].

Definition 1 (Collision Problem). *The collision problem $Col(\mathcal{H}(\mathbf{R}, m), D)$ asks to find a distinct pair $(\hat{\mathbf{x}}, \hat{\mathbf{x}}') \in D^m \times D^m$ such that $h(\hat{\mathbf{x}}) = h(\hat{\mathbf{x}}')$ for $h \leftarrow_\$ \mathcal{H}(\mathbf{R}, m)$.*

Obviously, the function is linear over $\mathbf{R}^m$, i.e., $h(\mathbf{a}(\hat{\mathbf{x}} + \hat{\mathbf{y}})) = \mathbf{a}(h(\hat{\mathbf{x}}) + h(\hat{\mathbf{y}}))$ for all $\mathbf{a} \in \mathbf{R}$, $\hat{\mathbf{x}}, \hat{\mathbf{y}} \in \mathbf{R}^m$. In addition, solving $Col(\mathcal{H}(\mathbf{R}, m), D)$ implies being able to solve ISVP^∞ in *every* lattice that corresponds to an ideal in $\mathbf{R}$ by the following theorem.

Theorem 1 (Worst-case to Average-case, Theorem 2 in [LM06]). *Let $D = \{\mathbf{f} \in \mathbf{R} : \|\mathbf{f}\|_\infty \leq d\}$, $m > \log(q)/\log(2d)$, and $q \geq 4dmn\sqrt{n}\log(n)$. An adversary $\mathcal{C}$ that solves the $Col(h, D)$ problem, i.e., finds distinct preimages $\hat{\mathbf{x}}, \hat{\mathbf{y}} \in D^m$ such that $h(\hat{\mathbf{x}}) = h(\hat{\mathbf{y}})$, can be used to solve ISVP^∞ with approximation factors $\gamma \geq 16dmn\log^2(n)$ in the worst case.*

3 Blind Signatures from Ideal Lattices

We construct a lattice-based blind signature scheme. It is secure in the random oracle model under a worst-case assumption in ideal lattices and its time *and* space complexity is quasi-optimal, $\widetilde{\mathcal{O}}(n)$.

The road map for this section is as follows: We describe the 4-move blind signature scheme BS. Then, we prove completeness, blindness, and one-more unforgeability. Proving completeness is non-trivial as we need to address an

Table 2. Parameters for the security parameter n

Parameter	Value	Asymptotics	Usage
n	power of 2	-	main security parameter
d_s	positive integer constant $< q/(4n)$	$\mathcal{O}(1)$	secret key size, unforgeability
D_s	$\{\mathbf{f} \in \mathbf{R} : \|\mathbf{f}\|_\infty \leq d_s\}$		set of secret keys
c_m	$> 1/\log(2d_s)$	$\widetilde{\mathcal{O}}(1)$	witness indistinguishability, leakage resilience
m	$\lfloor c_m \log(q) \rfloor + 1$	$\Omega(\log(n))$	worst-case to average-case reduction
D_ϵ	$\{\mathbf{f} \in \mathbf{R} : \|\mathbf{f}\|_\infty \leq 1 =: d_\epsilon\}$	$\mathcal{O}(1)$	hash output size
ϕ, ψ	positive integer constant ≥ 1	$\mathcal{O}(1)$	completeness, speed
D_α	$\{\mathbf{f} \in \mathbf{R} : \|\mathbf{f}\|_\infty \leq \psi n d_\epsilon =: d_\alpha\}$	$\mathcal{O}(n)$	blindness
D_{ϵ^*}	$\{\mathbf{f} \in \mathbf{R} : \|\mathbf{f}\|_\infty \leq d_\alpha - d_\epsilon =: d_{\epsilon^*}\}$	$\mathcal{O}(n)$	blindness
D_y	$\{\mathbf{f} \in \mathbf{R} : \|\mathbf{f}\|_\infty \leq \phi m n^2 d_s d_{\epsilon^*} =: d_y\}$	$\widetilde{\mathcal{O}}(n^3)$	witness indistinguishability
G_*	$\{\mathbf{f} \in \mathbf{R} : \|\mathbf{f}\|_\infty \leq d_y - n d_s d_{\epsilon^*} =: d_{G_*}\}$	$\widetilde{\mathcal{O}}(n^3)$	witness indistinguishability, completeness defect
D_β	$\{\mathbf{f} \in \mathbf{R} : \|\mathbf{f}\|_\infty \leq \phi m n d_{G_*} =: d_\beta\}$	$\widetilde{\mathcal{O}}(n^4)$	blindness
G	$\{\mathbf{f} \in \mathbf{R} : \|\mathbf{f}\|_\infty \leq d_\beta - d_{G_*} =: d_G\}$	$\widetilde{\mathcal{O}}(n^4)$	blindness, completeness defect
D	$\{\mathbf{f} \in \mathbf{R} : \|\mathbf{f}\|_\infty \leq d_{G_*} + d_\beta + n d_s d_\epsilon =: d_D\}$	$\widetilde{\mathcal{O}}(n^4)$	collisions under h
q	$\geq 4mn\sqrt{n}\log(n) d_D$, prime	$\widetilde{\Theta}(n^5\sqrt{n})$	worst-case to average-case reduction

The table defines all parameters and sets for our scheme. The sets are defined via a norm bound, for which we also state the asymptotic growth with respect to the security parameter n. The last column states the main usage for the individual parameter or set. Some sets introduce a completeness error to the scheme that can be reduced by increasing ϕ. Reducing this defect also significantly improves performance. All sets are subsets of the ring $\mathbf{R} = \mathbb{Z}_q[X]/(X^n + 1)$.

inevitable completeness defect. In the course of the discussion we show that it neither harms security nor efficiency. Afterwards, we prove that the scheme is statistically blind and that it is one-more unforgeable unless the collision problem $Col(\mathcal{H}(\mathbf{R}, m), D)$ is easy. In consequence, one-more unforgeability can be based on the worst-case hardness of the ISVP. After the main analysis, we prove that our scheme also supports leakage resilience.

Observe that the scheme requires lots of parameters that need to be carefully worked out. Their definition in Table 2 will be justified later in the analysis. We chose not to "unwind" the parameters d_s, d_ϵ, etc. because we need their relative size in the various lemmas below, making the proofs easier to understand. The asymptotics in the third column should help estimating their magnitude. The parameter d_ϵ is a constant 1 here but it can be increased if it is necessary to sign hash values of bit length $> n \log_2(3)$. The "usage" hint in the table points at the section, where they are most influential. As for selecting practical parameters, we refer the reader to the full version [Rüc08]. There, we propose secure parameter sets based on the analysis in [RS10]. The full version also includes a discussion on possible trade-offs for efficiency.

3.1 Our Construction

We construct our blind signature scheme $\mathsf{BS} = (\mathsf{Kg}, \mathsf{Sign}, \mathsf{Vf})$ as follows.

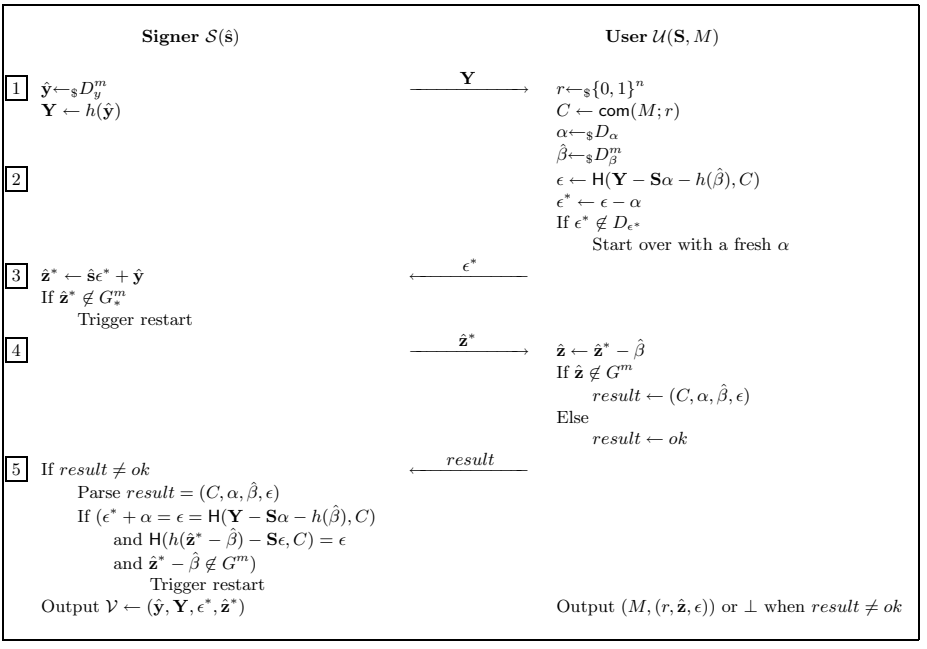


Fig. 1. Issue protocol of the blind signature scheme BS. All parameters and sets are defined in Table 2. Note that the signer implicitly verifies that the user's protocol messages come from the correct domains.

Key Generation. $\mathsf{BS.Kg}(1^n)$ selects a secret key $\hat{\mathbf{s}} \leftarrow_\$ D_s^m$, and a compression function $h \leftarrow_\$ \mathcal{H}(\mathbf{R}, m)$. Let $\mathcal{C}(1^n)$ be a commitment scheme, mapping $\{0,1\}^* \times \{0,1\}^n \to \{0,1\}^n$. The algorithm chooses a function $\mathsf{com} \leftarrow_\$ \mathcal{C}(1^n)$ and, in addition, selects $\mathsf{H} \leftarrow_\$ \mathcal{H}(1^n)$ mapping $\{0,1\}^* \to D_\epsilon \subset D$.

Then, it computes the public key $\mathbf{S} \leftarrow h(\hat{\mathbf{s}})$ and outputs $(\hat{\mathbf{s}}, \mathbf{S})$. For simplicity, we treat h, com, H, and the parameters in Table 2 as globally known and implicit inputs to all algorithms. However, each signer may choose them individually and include them in the public key.

Signature Protocol. The signature issue protocol for messages $M \in \{0,1\}^*$ is depicted in Figure 1. Eventually, the user outputs a message M and a signature $(r, \hat{\mathbf{z}}, \epsilon)$.

Notes: Upon a restart after Step 2, the user only selects a fresh $\alpha \leftarrow_\$ D_\alpha$ and repeats the operations that involve α. Whenever the signer triggers a restart, the user chooses a fresh r in order to make the protocol execution independent of the previous ones. Therefore, we omit values from previous runs in the signer's view. During Step 5, the signer can detect a cheating user that tries to trigger a restart, despite having received a valid signature. In this case, the signer can stop the protocol and assume that the user has obtained a valid signature.

Verification. $\mathsf{BS.Vf}(\mathbf{S}, (r, \hat{\mathbf{z}}, \epsilon), M)$ outputs 1 iff $\hat{\mathbf{z}} \in G^m$ and $\mathsf{H}(h(\hat{\mathbf{z}}) - \mathbf{S}\epsilon, \mathsf{com}(M; r)) = \epsilon$.

3.2 Analysis and Security

In this section, we analyze our blind signature scheme with regard to completeness, blindness, one-more unforgeability, and leakage resilience. For each aspect, we prove a main theorem. Supporting lemmas are stated before the theorems and proven in the full version [Rüc08].

Completeness. Completeness of BS is a non-trivial issue due to the eventual restarts and the many parameters involved. The next lemma ensures that the number of restarts is small, effectively constant.

Lemma 1. *Let* $k = \Omega(n)$, $\mathbf{a}, \mathbf{b} \in \mathbb{Z}^k$ *with* arbitrary $\mathbf{a} \in \{\mathbf{v} \in \mathbb{Z}^k : \|\mathbf{v}\|_\infty \leq A\}$ *and* random $\mathbf{b} \leftarrow_\$ \{\mathbf{v} \in \mathbb{Z}^k : \|\mathbf{v}\|_\infty \leq B\}$. *Given* $B \geq \phi k A$ *for* $\phi \in \mathbb{N}_{>0}$, *we have* $\operatorname{Prob}_{\mathbf{b}}[\|\mathbf{a} - \mathbf{b}\|_\infty \leq B - A] > \frac{1}{e^{1/\phi}} - o(1)$.

Theorem 2 (Completeness). *Let* $g(n) = \omega(\log^2(n))$. *The scheme* BS *is complete after at most* $g(n)$ *(or, an expected number of* $e^{2/\phi}$*) repetitions.*

See the full version [Rüc08] for the proof. There, we also argue that $\phi = 4$ is good choice to make the protocol more efficient in practice. Observe that in any case, all operations (including eventual restarts) in BS have $\widetilde{\mathcal{O}}(n)$ complexity and that private keys, public keys, and signatures have size $\widetilde{\mathcal{O}}(n)$.

Blindness. We prove that BS is statistically blind based on the observation that the signer only sees values that are independent of the message being signed. More precisely, the views generated by two different messages are indistinguishable. For this argument to work, we require a statistically hiding commitment scheme and carefully selected sets D_α, D_β, D_{ϵ^*}, and G. The following probabilistic lemma is crucial as it guarantees that the user's message after Step 2 and the final output are independent of the message. In the context of $\mathsf{Exp}^{\mathsf{blind}}_{\mathcal{S}^*,\mathsf{BS}}$, this establishes a form of witness indistinguishability w.r.t. the messages that are chosen by the malicious signer.

Lemma 2. *Let* $k \in \mathbb{N}$, $\mathbf{a}, \mathbf{a}', \mathbf{b} \in \mathbb{Z}^k$ *with* arbitrary $\mathbf{a}, \mathbf{a}' \in \{\mathbf{v} \in \mathbb{Z}^k : \|\mathbf{v}\|_\infty \leq A\}$, *a* random $\mathbf{b} \leftarrow_\$ \{\mathbf{v} \in \mathbb{Z}^k : \|\mathbf{v}\|_\infty \leq B\}$ *for* $B > A$. *We define the random variables* $\mathbf{c} \leftarrow \mathbf{a} - \mathbf{b}$ *and* $\mathbf{c}' \leftarrow \mathbf{a}' - \mathbf{b}$ *if* $\max\{\|\mathbf{a} - \mathbf{b}\|_\infty, \|\mathbf{a}' - \mathbf{b}\|_\infty\} \leq B - A$, *otherwise, we resample* $\mathbf{b}$. *Then,* $\Delta(\mathbf{c}, \mathbf{c}') = 0$.

The role of com is to ensure that the signer can only obtain negligible information from restarts. Notice that BS is perfectly blind ($(\infty, 0)$-blind) if the commitment scheme is perfect (0-hiding).

Theorem 3 (Blindness). BS *is* $(\infty, \delta^{(h)}_{\mathsf{com}})$*-blind if* com *is* $\delta^{(h)}_{\mathsf{com}}$*- hiding.*

Proof. As per experiment $\mathsf{Exp}^{\mathsf{blind}}_{\mathcal{S}^*,\mathsf{BS}}$, the adversarial signer outputs two messages M_0, M_1 and interacts with two users $\mathcal{U}(\mathbf{S}, M_b)$, $\mathcal{U}(\mathbf{S}, M_{1-b})$ after a secret coin flip $b \leftarrow \{0, 1\}$. We show that these users do not leak any information about their respective message.

Technically, we establish that all protocol messages and the output, when interpreted as random variables, are distributed independently of the message being signed. This involves an analysis of ϵ^*, $\hat{\mathbf{z}}$, and eventual restarts. As for ϵ and r we need not worry. They are chosen uniformly at random.

Distribution of ϵ^*. Let $\epsilon_b^*, \epsilon_{1-b}^*$ be the first protocol messages of $\mathcal{U}(\mathsf{pk}, M_b)$ resp. $\mathcal{U}(\mathsf{pk}, M_{1-b})$. They are in D_{ϵ^*} and they are both of the form $\epsilon - \alpha$ with $\epsilon \in D_\epsilon$ and $\alpha \leftarrow_\$ D_\alpha$. The statistical distance $\Delta(\epsilon_b^*, \epsilon_{1-b}^*)$ is 0 by Lemma 2 $(k = n, A = d_s, B = d_\alpha)$ because the coefficients in D_{ϵ^*} are bounded by $B - A = d_\alpha - d_s$.

Distribution of $\hat{\mathbf{z}}$. Let $\hat{\mathbf{z}}_0, \hat{\mathbf{z}}_1$ be part of the final output of $\mathcal{U}(\mathsf{pk}, M_0)$ resp. $\mathcal{U}(\mathsf{pk}, M_1)$. Both are of the form $\hat{\mathbf{z}}^* - \hat{\beta}$ for $\hat{\mathbf{z}}^* \in G_*^m$ and $\hat{\beta} \leftarrow_\$ D_\beta^m$. Furthermore, $\hat{\mathbf{z}}_0$ and $\hat{\mathbf{z}}_1$ are forced to be in G^m, having coefficients bounded by $d_\beta - d_{G_*}$. Hence, the statistical distance $\Delta(\hat{\mathbf{z}}_0, \hat{\mathbf{z}}_1)$ is 0 because of Lemma 2 $(k = mn, A = d_{G_*}, B = d_\beta)$.

Restarts. Observe that each protocol run is statistically independent of the previous runs by the statistical hiding property of the commitment com and because the user selects fresh $r, \alpha, \hat{\beta}$ after every restart. This is the reason why we inherit the statistical $\delta_{\mathsf{com}}^{(h)}$-hiding property to obtain $(\infty, \delta_{\mathsf{com}}^{(h)})$-blindness instead of perfect blindness. Finally, we need to argue about the restart after Step 4. The user sends $(C, \alpha, \hat{\beta}, \epsilon)$ to the signer. These information allow the verification of the signature with respect to C. The message is still statistically hidden by the hiding property of com because the user never reveals the decommitment r.

Hence, the protocol hides the to-be-signed message and subsequent runs of the protocol for the same message are statistically independent. □

Furthermore, our scheme already supports selective failure blindness as shown in [FS09] because we are signing commitments instead of the adversely chosen messages. Even the fourth move does not reveal any information about the message due to the hiding property of the commitment.

One-more Unforgeability. In this section, we show that BS is one-more unforgeable, provided that the collision problem $Col(\mathcal{H}(\mathbf{R}, m), D)$ is hard and the commitment scheme is binding. The main tool in the reduction is the Forking Lemma [PS00, BN06]. To simulate the environment, especially blind signature queries, for the attacker $\mathcal{A}$ in the unforgeability experiment, we require that there are at least two possible secret keys for each public key $\mathbf{S}$ (Lemma 3). Moreover, we need the signature protocol to be witness indistinguishable to prevent the attacker from learning the secret key (Lemma 4). The binding property of com is necessary to prevent an attacker from obtaining one signature that works for two messages by changing the message under the commitment. All other attackers output at least one signature that does not correspond to a completed interaction. Here, we apply the Forking Lemma to extract knowledge about the secret key that was used to compute the forgery. Using this knowledge the reduction can solve the collision problem. Finally, we need to deal with Step 5 in the

protocol. The adversary proves that it was unable to obtain a valid signature. We show that this is sufficient if Col is hard.

Since the function family $\mathcal{H}(\mathbf{R}, m)$ compresses the domain D_s^m, it is easy to show that all secret keys collide with at least one other secret key.

Lemma 3. *Let $h \in \mathcal{H}(\mathbf{R}, m)$. For every secret key $\hat{\mathbf{s}} \leftarrow_\$ D_s^m$, there is a second $\hat{\mathbf{s}}' \in D_s^m \setminus \{\hat{\mathbf{s}}\}$ with $h(\hat{\mathbf{s}}) = h(\hat{\mathbf{s}}')$ (with overwhelming probability).*

The next lemma establishes witness indistinguishability of the protocol. Witness indistinguishability ensures that the malicious verifier cannot distinguish whether the prover uses one of two possible secret keys $\hat{\mathbf{s}}, \hat{\mathbf{s}}' \in h^{-1}(\mathbf{S}) \cap D_s^m$. Basically, it can be interpreted as an application of Lemma 2 to $\hat{\mathbf{z}}^* = (\hat{\mathbf{s}}\epsilon^*) + \hat{\mathbf{y}} \in G_*^m$ with some further observations. The choice of $\hat{\mathbf{y}} \leftarrow_\$ D_y$ and the restriction "$\in G_*^m$" hide the first summand.

Lemma 4. *Let $h \in \mathcal{H}(\mathbf{R}, m)$ and $\mathbf{S} \in \mathbf{R}$. For any message M and any two secret keys $\hat{\mathbf{s}}, \hat{\mathbf{s}}' \in D_s^m$ with $h(\hat{\mathbf{s}}) = \mathbf{S} = h(\hat{\mathbf{s}}')$, the resulting protocol views $(\mathbf{Y}, \epsilon^*, \hat{\mathbf{z}}^*)$ and $(\mathbf{Y}', \epsilon^{*\prime}, \hat{\mathbf{z}}^{*\prime})$ are indistinguishable.*

Using lemmas 3 and 4, we can exploit witness indistinguishability to simulate all blind signature oracle queries with a secret key $\hat{\mathbf{s}}$ and at the same time expect the adversary to output a forgery that corresponds to a different secret key $\hat{\mathbf{s}}'$ with non-negligible probability or break the binding property of the commitment scheme. We apply the Forking Lemma to extract a solution to the $Col(\mathcal{H}(\mathbf{R}, m), D)$.

Theorem 4 (One-more unforgeability). *Let* Sig *be the signature oracle. Let T_{Sig} and T_{H} be the cost functions for simulating the oracles* Sig *and* H*, and let $c < 1$ be the probability for a restart in the protocol.* BS *is $(t, q_{\mathsf{Sign}}, q_{\mathsf{H}}, \delta)$-one-more unforgeable if* com *is $(t', \delta/2)$-binding and $Col(\mathcal{H}(\mathbf{R}, m), D)$ is $(t', \delta'/2)$-hard with $t' = t + q_{\mathsf{H}}^{q_{\mathsf{Sig}}}(q_{\mathsf{Sign}} T_{\mathsf{Sig}} + q_{\mathsf{H}} T_{\mathsf{H}})$ and non-negligible δ' if δ is non-negligible.*

The probability δ' depends on the number of issued signatures. It can be found at the end of the proof.

Proof. Towards contradiction, we assume that there exists a successful forger $\mathcal{A}$ against one-more unforgeability of BS with non-negligible probability δ. Using $\mathcal{A}$, we construct an algorithm $\mathcal{B}$, such that it either solves the collision problem or breaks the binding property of com.

Setup. $\mathcal{B}$ flips a coin $b \leftarrow_\$ \{0, 1\}$. For $b = 0$, it selects $h \leftarrow_\$ \mathcal{H}(\mathbf{R}, m)$. For $b = 1$, it gets the description of h as input. $\mathcal{B}$ initializes a list $L_{\mathsf{H}} \leftarrow \emptyset$ of query-hash pairs $(\mathbf{R} \times \{0, 1\}^*, D_\epsilon)$. It chooses $\hat{\mathbf{s}} \leftarrow_\$ D_s^m$ and sets $\mathbf{S} \leftarrow h(\hat{\mathbf{s}})$. Furthermore, it randomly pre-selects random oracle answers $\mathbf{h}_1, \ldots, \mathbf{h}_{q_{\mathsf{H}}} \leftarrow_\$ D_\epsilon$ and a random tape ρ. It runs $\mathcal{A}(\mathbf{S}; \rho)$ in a black-box simulation.

Random Oracle Queries. On input $(\mathbf{u}, C)$, $\mathcal{B}$ looks up $(\mathbf{u}, C)$ in L_{H}. If it finds corresponding hash value ϵ then it returns ϵ. Otherwise, $\mathcal{B}$ selects the first unused ϵ from the list $\mathbf{h}_1, \ldots, \mathbf{h}_{q_{\mathsf{H}}}$, stores $((\mathbf{u}, C), \epsilon)$ in L_{H}, and returns ϵ.

Blind Signature Queries. $\mathcal{B}$ acts according to the protocol in Figure 1.

Output. Eventually, $\mathcal{A}$ stops and outputs $(M_1,(r_1,\hat{\mathbf{z}}_1,\epsilon_1)),\ldots,(M_\jmath,(r_\jmath,\hat{\mathbf{z}}_\jmath,\epsilon_\jmath))$, $q_{\mathsf{Sign}}+1=\jmath$, for distinct messages. If $b=0$, the reduction looks for two pairs $(M_1^*,(r_1^*,\hat{\mathbf{z}}^*,\epsilon^*))$ and $(M_2^*\neq M_1^*,(r_2^*,\hat{\mathbf{z}}^*,\epsilon^*))$ and outputs $(M_1^*,r_1^*),(M_2^*,r_2^*)$ to break the binding property of com. If there is no such collision, $\mathcal{B}$ aborts. If $b=1$, the simulator $\mathcal{B}$ guesses an index $k\leftarrow_\$[\jmath]$ such that $h_\imath=\epsilon_k$ for some $\imath\in[q_{\mathsf{H}}]$. Then, $\mathcal{B}$ starts over, running $\mathcal{A}(\mathbf{S};\rho)$ with random oracle answers $\mathbf{h}_1,\ldots,\mathbf{h}_{\imath-1},\mathbf{h}'_\imath,\ldots,\mathbf{h}'_{q_{\mathsf{H}}}$ for a fresh set $\mathbf{h}'_\imath,\ldots,\mathbf{h}'_{q_{\mathsf{H}}}\leftarrow_\$ D_\epsilon$. Both $\mathcal{A}$ and $\mathcal{B}$ are run with the same random tape as in the first run. Among other values, $\mathcal{A}$ outputs $(M'_k,(r'_k,\hat{\mathbf{z}}'_k,\epsilon'_k))$ and $\mathcal{B}$ returns $(\hat{\mathbf{z}}_k-\hat{\mathbf{s}}\epsilon_k,\hat{\mathbf{z}}'_k-\hat{\mathbf{s}}\epsilon'_k)$ if $\epsilon'_k=\epsilon_k$ in an attempt to solve $Col(\mathcal{H}(\mathbf{R},m),D)$. If $\epsilon'_k\neq\epsilon_k$, the reduction retries at most $q_{\mathsf{H}}^\jmath$ times with a different random tape and random oracle.

Analysis. $\mathcal{A}$'s environment is perfectly simulated. Especially, restarts happen with the same probability as in the original protocol. For $b=0$, $\mathcal{B}$ $(t',\delta/2)$-breaks the binding property of com if $\mathcal{A}$ breaks the binding property of com to break one-more unforgeability.

For $b=1$, we assume that $\mathcal{A}$ breaks one-more unforgeability without attacking com. So, at least one of the output signatures is not obtained via an interaction. The probability that $\mathcal{B}$ guesses the index k of this signature correctly is at least $1/(q_{\mathsf{Sign}}+1)$. Observe that ϵ_k is a random oracle answer but with probability $1/|D_\epsilon|$. Furthermore, notice that with probability $1/2$, at least one of the re-runs of $\mathcal{A}$ yields the same map $\{(\imath,k):h_\imath=\epsilon_k\}$ as in the first run of $\mathcal{A}$. Thus, we consider the indices in both "interesting" replays to be constant.

Applying the Forking Lemma, we know that with probability $\delta_{\mathsf{frk}}\geq(1-c)(\delta-1/|D_\epsilon|)((\delta-1/|D_\epsilon|)/q_{\mathsf{H}}-1/|D_\epsilon|)$, $\mathcal{A}$ is again successful in the one-more unforgeability experiment and outputs $(M'_k,(r'_k,\hat{\mathbf{z}}'_k,\epsilon'_k))$ using the *same random oracle query* as in the first run. The additional $(1-c)$ factor takes a potential abort during the second run into account, which happen with probability at most c. Therefore, we know that $(h(\hat{\mathbf{z}}_k-\mathbf{S}\epsilon_k),\mathsf{com}(M_k;r_k))=(h(\hat{\mathbf{z}}'_k-\mathbf{S}\epsilon'_k),\mathsf{com}(M'_k;r'_k))$.

Now, we turn to solving the collision problem. We have to show that $\hat{\mathbf{z}}_k-\hat{\mathbf{s}}\epsilon_k\neq\hat{\mathbf{z}}'_k-\hat{\mathbf{s}}\epsilon'_k$ and $h(\hat{\mathbf{z}}_k-\hat{\mathbf{s}}\epsilon_k)=h(\hat{\mathbf{z}}'_k-\hat{\mathbf{s}}\epsilon'_k)$. The second requirement follows directly from the previous paragraph. The first is more involved. Here, it is important that the protocol is witness indistinguishable (Lemma 4), i.e., the adversary does not recognize whether we have used one of at least two possible $\hat{\mathbf{s}},\hat{\mathbf{s}}'$ (Lemma 3 with probability greater than $1/2$. Thus, with probability at least $1/2$ its output corresponds to $\hat{\mathbf{s}}'$. We show that either $\hat{\mathbf{z}}_k-\hat{\mathbf{s}}\epsilon_k\neq\hat{\mathbf{z}}'_k-\hat{\mathbf{s}}\epsilon'_k$ or $\hat{\mathbf{z}}_k-\hat{\mathbf{s}}'\epsilon_k\neq\hat{\mathbf{z}}'_k-\hat{\mathbf{s}}'\epsilon'_k$. Assuming both are equal, we subtract the equations and obtain $(\epsilon_k-\epsilon'_k)(\hat{\mathbf{s}}'-\hat{\mathbf{s}})=\mathbf{0}$. We know that $\epsilon_k-\epsilon'_k\neq\mathbf{0}$. Now, $\|(\epsilon_k-\epsilon'_k)(\hat{\mathbf{s}}'-\hat{\mathbf{s}})\|_\infty\leq 2d_sn<q/2$ because $\|\epsilon_k-\epsilon'_k\|_\infty\leq 2$ and $\|\hat{\mathbf{s}}'-\hat{\mathbf{s}}\|_\infty\leq 2d_s$. Thus, $(\epsilon_k-\epsilon'_k)(\hat{\mathbf{s}}'-\hat{\mathbf{s}})=\mathbf{0}$ over $\mathbb{Z}[X]/\langle X^n+1\rangle$, which is an integral domain. So, we have the contradiction $\hat{\mathbf{s}}'=\hat{\mathbf{s}}$ and a collision $(\hat{\mathbf{z}}_k-\hat{\mathbf{s}}\epsilon_k,\hat{\mathbf{z}}'_k-\hat{\mathbf{s}}\epsilon'_k)\in D\times D$. The success probability is at least $\delta_{\mathsf{col}}\geq 1/4\,\delta_{\mathsf{frk}}/(q_{\mathsf{Sign}}+1)$, which is non-negligible if δ is non-negligible.

Concerning restarts, we argue that the user cannot obtain a valid signature out of an aborted interaction without solving the collision problem. In order to

trigger an abort after Step 4, it outputs $result = (C, \alpha, \hat{\beta}, \epsilon)$ which, together with $\hat{\mathbf{z}}^*, \hat{\mathbf{y}}, \epsilon^*$, satisfies all abort criteria:

$$\epsilon^* + \alpha = \epsilon = \mathsf{H}(\mathbf{Y} - \mathbf{S}\alpha - h(\hat{\beta}), C) \tag{1}$$

$$\epsilon = \mathsf{H}(h(\hat{\mathbf{z}}^* - \hat{\beta}) - \mathbf{S}\epsilon, C) \tag{2}$$

$$\hat{\mathbf{z}}^* - \hat{\beta} \notin G^m \tag{3}$$

Assume that it also obtains a valid signature $(r', \hat{\mathbf{z}}', \epsilon')$ from this interaction. If $\epsilon = \epsilon'$, then $h(\hat{\mathbf{z}}^* - \hat{\beta}^* - \hat{\mathbf{s}}\epsilon) = h(\hat{\mathbf{z}}' - \hat{\mathbf{s}}\epsilon)$ by (2). If the arguments under h are equal, we have $\hat{\mathbf{z}}^* - \hat{\beta} \in G^m$ — a contradiction with (3). If the arguments are distinct, we have a collision in D because $\|\hat{\mathbf{z}}' - \hat{\mathbf{s}}\epsilon\|_\infty \leq d_G < d_D$ and $\left\|\hat{\mathbf{z}}^* - \hat{\beta}^* - \hat{\mathbf{s}}\epsilon\right\|_\infty \leq d_{G_*} + d_\beta + nd_s d_\epsilon = d_D$.

The adversary may succeed by hiding $\epsilon' \neq \epsilon$ in ϵ^*. But then, we necessarily have $\epsilon^* = \epsilon - \alpha = \epsilon' - \alpha'$ by (1) for an $\alpha \neq \alpha'$ and we know that $\alpha = \epsilon - \epsilon' + \alpha'$. So, the adversary had to be able to predict the output of H to compute α.

To conclude, the probability that we can extract a collision from a cheating user during an abort is at least $\delta_{\mathsf{abort}} \geq \delta\,(1 - 1/|D_\epsilon|)$, which is non-negligible if δ is non-negligible. Thus, the overall success probability of the reduction is $\delta' \geq \min(\delta_{\mathsf{col}}, \delta_{\mathsf{abort}})$ if the guess $b = 1$ was correct. □

Hence, we require that $q_{\mathsf{Sig}} = o(n)$ to be able to rely on the subexponential hardness of lattice problems. This constraint is an artifact of the proof technique as discussed in [PS00] and it is not at all unusual for efficient blind signature schemes. There, it was even required that $q_{\mathsf{Sig}} \leq (\log(n))^{\mathcal{O}(1)}$ because they needed a polynomial-time reduction. In consequence, in our reduction, we greatly benefit from the subexponential hardness of the underlying lattice problem. Alternatively, we believe that the running time of the reduction can be significantly reduced to being polynomial in q_{Sig} by using techniques due to Pointcheval [Poi98].

By Theorem 1, we get the following strong worst-case security guarantees.

Corollary 1. BS *is one-more unforgeable if solving* ISVP^∞ *is hard in the worst case for approximation factors* $\gamma \geq 16 d_D mn \log^2(n) = \widetilde{\mathcal{O}}(n^5)$ *in lattices that correspond to ideals in* $\mathbf{R}$.

Leakage Resilience. Using an additional restriction for one of the parameters, we can safely leak a $(1 - o(1))$ fraction of the secret key in the unforgeability experiment according to the definition in the full version [Rüc08]. Recall that $m = \lfloor c_m \log(q) \rfloor + 1$ for some $c_m = \widetilde{\mathcal{O}}(1)$. Thus, it is possible to choose c_m, say $\log(n)$, without loosing the scheme's quasi-optimal efficiency. The following theorem states that such a choice is sufficient to provide strong leakage resilience. The proof can be found in the full version [Rüc08].

Theorem 5 (Leakage Resilience). *Let* $c_m = \omega(1)$ *and let* $L := \log(|D_s^m|) = mn \log(2d_s + 1)$ *be the length of the secret key. The conditional min-entropy* H_∞ *of* $\hat{\mathbf{s}}$*, conditioned on* $\mathbf{S} = h(\hat{\mathbf{s}})$ *and a total secret-key leakage* $f(\hat{\mathbf{s}})$ *of* $\lambda = \delta L = (1 - o(1))L$ *bits, is positive with overwhelming probability.*

4 Conclusions

We have shown how to construct an efficient and provably secure blind signature scheme based on the hardness of worst-case lattice problems. Our scheme has four moves, offers quasi-optimal performance, and it is leakage resilient in an almost optimal sense. Therefore, we expect our construction to withstand even subexponential-time and quantum computer attacks, as well as limited side-channel attacks against the secret key.

Acknowledgments

The author thanks Özgür Dagdelen, Marc Fischlin, Tibor Jager, Vadim Lyubashevsky, Chris Peikert, Michael Schneider, and Dominique Schröder for reviewing parts of this work and for very helpful and encouraging discussions. He also thanks the ananymous reviewers of ASIACRYPT 2010 for their valuable input.

References

[Abe01] Abe, M.: A secure three-move blind signature scheme for polynomially many signatures. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 136–151. Springer, Heidelberg (2001)

[ADL+08] Arbitman, Y., Dogon, G., Lyubashevsky, V., Micciancio, D., Peikert, C., Rosen, A.: SWIFFTX: A proposal for the SHA-3 standard. In: The First SHA-3 Candidate Conference (2008)

[ADW09] Alwen, J., Dodis, Y., Wichs, D.: Leakage-resilient public-key cryptography in the bounded-retrieval model. In: Halevi, S. (ed.) Advances in Cryptology - CRYPTO 2009. LNCS, vol. 5677, pp. 36–54. Springer, Heidelberg (2009)

[Ajt96] Ajtai, M.: Generating hard instances of lattice problems (extended abstract). In: STOC, pp. 99–108. ACM, New York (1996)

[AKS01] Ajtai, M., Kumar, R., Sivakumar, D.: A sieve algorithm for the shortest lattice vector problem. In: STOC, pp. 601–610. ACM, New York (2001)

[ANN06] Abdalla, M., Namprempre, C., Neven, G.: On the (im)possibility of blind message authentication codes. In: Pointcheval, D. (ed.) CT-RSA 2006. LNCS, vol. 3860, pp. 262–279. Springer, Heidelberg (2006)

[BMV08] 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)

[BN06] Bellare, M., Neven, G.: Multi-signatures in the plain public-key model and a general forking lemma. In: Juels, A., Wright, R.N., De Capitani di Vimercati, S. (eds.) ACM Conference on Computer and Communications Security, pp. 390–399. ACM, New York (2006)

[BNPS03] Bellare, M., Namprempre, C., Pointcheval, D., Semanko, M.: The one-more-rsa-inversion problems and the security of chaum's blind signature scheme. J. Cryptology 16(3), 185–215 (2003)

[Bol03] Boldyreva, A.: Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In: Desmedt, Y. (ed.) PKC 2003. LNCS, vol. 2567, pp. 31–46. Springer, Heidelberg (2002)

[BR93] Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: CCS. ACM, New York (1993)

[Cha82] Chaum, D.: Blind signatures for untraceable payments. In: CRYPTO, pp. 199–203 (1982)

[CKW04] Camenisch, J., Koprowski, M., Warinschi, B.: Efficient blind signatures without random oracles. In: Blundo, C., Cimato, S. (eds.) SCN 2004. LNCS, vol. 3352, pp. 134–148. Springer, Heidelberg (2005)

[CNS07] Camenisch, J., Neven, G., Shelat, A.: Simulatable adaptive oblivious transfer. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 573–590. Springer, Heidelberg (2007)

[ECR10] ECRYPT2. Yearly report on algorithms and keysizes — report D.SPA.13 (2010), http://www.ecrypt.eu.org/documents/D.SPA.13.pdf

[FS86] 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)

[FS09] Fischlin, M., Schröder, D.: Security of blind signatures under aborts. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 297–316. Springer, Heidelberg (2009)

[Gol04] Goldreich, O.: The Foundations of Cryptography, vol. 1. Cambridge University Press, Cambridge (2004)

[GPV08] Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Ladner, R.E., Dwork, C. (eds.) STOC, pp. 197–206. ACM, New York (2008)

[JLO97] Juels, A., Luby, M., Ostrovsky, R.: Security of blind digital signatures (extended abstract). In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 150–164. Springer, Heidelberg (1997)

[KTX08] 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)

[KV09] 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)

[Len05] Lenstra, A.: The Handbook of Information Security. Key Lengths, ch. 14. Wiley, Chichester (2005), http://www.keylength.com/biblio/Handbook_of_Information_Security_-_Keylength.pdf

[LM06] Lyubashevsky, V., Micciancio, D.: Generalized compact knapsacks are collision resistant. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4052, pp. 144–155. Springer, Heidelberg (2006)

[Lyu08] Lyubashevsky, V.: Lattice-based identification schemes secure under active attacks. In: Cramer, R. (ed.) PKC 2008. LNCS, vol. 4939, pp. 162–179. Springer, Heidelberg (2008)

[Lyu09] 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)

[Mat09] Matsui, M. (ed.): ASIACRYPT 2009. LNCS, vol. 5912. Springer, Heidelberg (2009)

[Oka92] 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)

[Oka06] Okamoto, T.: Efficient blind and partially blind signatures without random oracles. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 80–99. Springer, Heidelberg (2006)

[Poi98] Pointcheval, D.: Strengthened security for blind signatures. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 391–405. Springer, Heidelberg (1998)

[PS97] Pointcheval, D., Stern, J.: New blind signatures equivalent to factorization (extended abstract). In: ACM Conference on Computer and Communications Security, pp. 92–99 (1997)

[PS00] Pointcheval, D., Stern, J.: Security arguments for digital signatures and blind signatures. J. Cryptology 13(3), 361–396 (2000)

[RHOAGZ07] Rodríguez-Henríquez, F., Ortiz-Arroyo, D., García-Zamora, C.: Yet another improvement over the mu-varadharajan e-voting protocol. Comput. Stand. Interfaces 29(4), 471–480 (2007)

[RS10] Rückert, M., Schneider, M.: Selecting secure parameters for lattice-based cryptography. Cryptology ePrint Archive, Report 2010/137 (2010), http://eprint.iacr.org/

[Rüc08] Rückert, M.: Lattice-based blind signatures. Cryptology ePrint Archive, Report 2008/322 (2008), http://eprint.iacr.org/

[Rüc10] Rückert, M.: Adaptively secure identity-based identification from lattices without random oracles. In: Garay, J.A., De Prisco, R. (eds.) SCN 2010. LNCS, vol. 6280, pp. 345–362. Springer, Heidelberg (2010), http://dblp.uni-trier.de/rec/bibtex/conf/scn/Ruckert10

[Sch91] Schnorr, C.P.: Efficient signature generation by smart cards. J. Cryptology 4, 161–174 (1991)

[Sho97] Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM J. Comput. 26(5), 1484–1509 (1997)

The Round Complexity of Verifiable Secret Sharing: The Statistical Case

Ranjit Kumaresan[1,⋆], Arpita Patra[2], and C. Pandu Rangan[2]

[1] Dept. of Computer Science
University of Maryland
ranjit@cs.umd.edu
[2] Dept. of Computer Science
IIT Madras
arpitapatra_10@yahoo.co.in, rangan@cs.iitm.ernet.in

Abstract. We consider the round complexity of a basic cryptographic task: *verifiable secret sharing* (VSS). This well-studied primitive provides a good "test case" for our understanding of round complexity in general; moreover, VSS is important in its own right as a central building block for, e.g., Byzantine agreement and secure multi-party computation.

The round complexity of *perfect* VSS was settled by Gennaro et al. (STOC 2001) and Fitzi et al. (TCC 2006). In a surprising result, Patra et al. (Crypto 2009) recently showed that if a negligible probability of error is allowed, the previous bounds no longer apply. We settle the key questions left open by their work, and in particular determine the *exact* round complexity of statistical VSS with optimal threshold. Let n denote the number of parties, at most t of whom are malicious. Their work showed that 2-round statistical VSS is impossible for $t \geq n/3$. We show that 3-round statistical VSS is possible iff $t < n/2$. We also give an *efficient* 4-round protocol for $t < n/2$.

1 Introduction

The round complexity of cryptographic protocols is a central measure of their efficiency, and has been the subject of intense study. In this work, we are interested in understanding the round complexity of *verifiable secret sharing* (VSS) [2]. Here, there is a *dealer* who shares a secret among a group of n parties, at most t of whom (possibly including the dealer) may be malicious. The requirements (roughly speaking) are that if the dealer is honest, then no information about the dealer's secret is revealed to the t malicious parties by the end of the *sharing phase*; nevertheless, by the end of the sharing phase even a dishonest dealer is irrevocably committed to *some* value that will be recovered by the honest parties in the *reconstruction phase*. Furthermore, if the dealer is honest then this committed value must be identical to the dealer's initial input.

We focus on *information-theoretic* VSS, where the security requirements are required to hold even when the malicious parties have unbounded computational

⋆ Supported by the U.S. DoD/ARO MURI program and NSF award #0627306.

M. Abe (Ed.): ASIACRYPT 2010, LNCS 6477, pp. 431–447, 2010.

power. Here, two different possibilities can be considered: either the security requirements hold *perfectly* (i.e., always), or the security requirements hold *statistically* but can possibly be violated with negligible probability. Assuming a broadcast channel, perfect VSS is possible if and only if $t < n/3$ [1,4], while statistical VSS is possible up to threshold $t < n/2$ [11].

The round complexity of perfect VSS has been extensively studied. For the case of optimal threshold (i.e., $t < n/3$), Gennaro et al. [6] showed that 3 rounds[1] are necessary and sufficient for perfect VSS, and gave an efficient 4-round protocol for the task. The 3-round VSS protocol by Gennaro et al. requires communication exponential in the number of players, but Fitzi et al. [5] later demonstrated that an *efficient* 3-round protocol is possible. Katz et al. [7] showed that perfect VSS could be achieved with optimal round complexity and, at the same time, optimal use of the broadcast channel.

The 3-round lower bound of Gennaro et al. was generally believed to apply also to the case of statistical VSS. It was therefore relatively surprising when Patra et al. [8] showed recently that *statistical* VSS could be realized in *two* rounds for $t < n/3$. The protocol of Patra et al. does not apply when $n/3 \leq t < n/2$, and finding a minimal-round protocol for the optimal security threshold was left open by their work. On the other hand, the work of Patra et al. proves that 2-round statistical VSS is impossible for $t \geq n/3$, which obviously applies to our setting as well.

Our results and organization of the paper. In this work we resolve the round complexity of statistical VSS with optimal threshold $t < n/2$. We show that 3-round statistical VSS is possible for any $t < n/2$. We also give an efficient 4-round protocol for $t < n/2$.

2 Model and Definitions

We consider the standard communication model where parties communicate in synchronous rounds using pairwise private and authenticated channels. We also assume a broadcast channel. (VSS is impossible for $t \geq n/3$ unless broadcast is assumed.) A broadcast channel allows any party to send the same message to all other parties — and all parties to be assured they have received identical messages — in a single round.

When we say a protocol tolerates t malicious parties, we always mean that it is secure against an adversary who may *adaptively* corrupt up to t parties during an execution of the protocol and coordinate the actions of these parties as they deviate from the protocol in an arbitrary manner. Parties not corrupted by the adversary are called *honest*. We always assume a *rushing* adversary; i.e., in any round the malicious parties receive the messages (including the broadcast messages) sent by the honest parties before deciding on their own messages.

[1] Following the accepted convention, the round complexity of VSS refers to that of the sharing phase.

In our protocol descriptions we assume without loss of generality that parties send properly formatted messages, as we may interpret an improper or missing message as some default message.

We let $\mathbb{F}$ denote a finite field and set $\kappa = \log|\mathbb{F}|$. We require the dealer's secret to lie in $\mathbb{F}$. In the case of statistical VSS, we allow *error* with probability at most $\varepsilon = 2^{-\Theta(\kappa)}$ and so κ can be treated as a security parameter. Note that the dealer's secret can be padded to lie in a larger field, if desired, to reduce the probability of error.

Definition 1. *A two-phase protocol for parties $\mathcal{P} = \{P_1, \ldots, P_n\}$, where a distinguished dealer $D \in \mathcal{P}$ holds initial input $s \in \mathbb{F}$, is a* $(1-\varepsilon)$-statistical VSS protocol tolerating t malicious parties *if the following conditions hold for any adversary controlling at most t parties:*

Privacy: *If the dealer is honest at the end of the first phase (the* sharing phase*), then at the end of this phase the joint view of the malicious parties is independent of the dealer's input s.*

Correctness/Commitment: *Each honest party P_i outputs a value s_i at the end of the second phase (the* reconstruction phase*). Except with probability at most ε, the following hold:*

1. *At the end of the sharing phase, the joint view of the honest parties defines a value s' such that $s_i = s'$ for every honest P_i.*
2. *If the dealer is honest throughout the execution, then $s' = s$.* ♢

Remark: Our definition of statistical VSS relaxes the *correctness/commitment* requirement, but not the *secrecy* requirement. This is the definition that has been considered previously in the literature, and is the definition that our protocols achieve.

3 A Multiple-Verifier Information Checking Protocol

Our protocols rely on what is known as an *information checking (sub)protocol* (ICP), a notion first introduced by Rabin and Ben-Or [11]. The traditional definition of an ICP [11,3] involves the dealer D, an intermediary INT, and a verifier $\mathcal{V}$. In an initial phase, the dealer gives a secret value $s \in \mathbb{F}$ to INT and some verification information (that reveals nothing about s) to $\mathcal{V}$. Later, INT gives s to $\mathcal{V}$ along with a "proof" that s is indeed the value that INT received initially from D.

The basic definition of ICP involves only a *single* verifier; Patra et al. [10,9], extend this definition to allow every party in the network to act as a verifier. Defining ICP in this way (i.e., enabling multiple verifiers) will be helpful when we use it as a black box in our VSS protocols. Formally, an *information checking protocol* (ICP) consists of three stages Distr, AuthVal, and RevealVal:

– $\mathsf{Distr}(D, INT, s)$ is initiated by D, using as input some value s. The algorithm generates some *authentication information* (which includes s itself) that is given to INT, as well as some *verification information* that is given to each of the verifiers.

- $\mathsf{AuthVal}(D, INT, s)$ is initiated by INT after receiving the authentication information from D. The information held by INT after this stage is called D's *IC-signature* and is denoted by $ICSIG(D, INT, s)$.
- $\mathsf{RevealVal}(D, INT, s)$ is a sub-protocol in which all messages are broadcast. Based on the broadcast messages, either $ICSIG(D, INT, s)$ is accepted or rejected by all honest verifiers (with high probability).

We require ICP to satisfy the following properties:

1. **Correctness 1** : If D and INT are honest, then every honest verifier accepts $ICSIG(D, INT, s)$ during $\mathsf{RevealVal}$.
2. **Correctness 2** : If INT is honest then at the end of $\mathsf{AuthVal}$, INT possesses an $ICSIG(D, INT, s)$, which will be accepted in $\mathsf{RevealVal}$ by each honest verifier, except with probability $2^{-\Omega(\kappa)}$.
3. **Correctness 3** : If D is honest then during $\mathsf{RevealVal}$, with probability at least $1 - 2^{-\Omega(\kappa)}$, $ICSIG(D, INT, s)$ revealed as some $s' \neq s$ by a corrupted INT will be rejected by *each* honest verifier.
4. **Secrecy:** If D and INT are honest, then till the end of $\mathsf{AuthVal}$, the adversary has no information about s.

3.1 An ICP Protocol

Here we reproduce a simplified version of the ICP protocol (from Patra et al., [10,9]) tolerating $t < n/2$ malicious parties, such that Distr requires one round and $\mathsf{AuthVal}$ and $\mathsf{RevealVal}$ require two rounds each. We omit the proofs due to space limitations.

$\mathsf{Distr}(D, INT, s)$:
Round 1:

1. D sends the following to INT:
   (a) A random degree-t polynomial $F(x)$ over $\mathbb{F}$, with $F(0) = s$. Let INT receive $F'(x)$ as the polynomial with $F'(0) = s'$. [2]
   (b) A random degree-t polynomial $R(x)$ over $\mathbb{F}$. Let INT receive $R(x)$ as a t-degree polynomial $R'(x)$.
2. D privately sends the following to each verifier P_i:
   (a) (α_i, v_i, r_i), where $\alpha_i \in \mathbb{F} \setminus \{0\}$ is random (all α_i's are distinct), $v_i = F(\alpha_i)$ and $r_i = R(\alpha_i)$.

$\mathsf{AuthVal}(D, INT, s)$:
Round 1: INT chooses a random $d \in \mathbb{F} \setminus \{0\}$ and broadcasts $(d, B(x))$ where $B(x) = dF'(x) + R'(x)$.

Round 2: D checks $dv_i + r_i \stackrel{?}{=} B(\alpha_i)$ for $i = 1, \ldots, n$. If D finds any inconsistency, he broadcasts $s^D = s$.

The polynomial $F'(x)$ (when D does not broadcast s^D in round 2 of $\mathsf{AuthVal}$) or s^D (broadcast by D in round 2 of $\mathsf{AuthVal}$) as held by INT is denoted by $ICSIG(D, INT, s)$.

[2] If INT is honest, then $F'(x) = F(x)$.

RevealVal(D, INT, s)**:**
Round 1: INT broadcasts $ICSIG(D, INT, s)$ (i.e., he reveals D's secret contained in $ICSIG(D, INT, s)$ as $s' = s^D$ or as $s' = F'(0)$).
Round 2: Verifier P_i broadcasts Accept if one of the following conditions holds. (Otherwise, P_i broadcasts Reject.)

1. $ICSIG(D, INT, s) = s'$, and $s' = s^D$.
2. $ICSIG(D, INT, s) = F'(x)$, and one of the following holds.
   1. *C1*: $v_i = F'(\alpha_i)$; OR
   2. *C2*: $B(\alpha_i) \neq dv_i + r_i$ ($B(x)$ was broadcasted by INT during AuthVal).

Local Computation (By Every Verifier): If at least $t + 1$ verifiers have broadcasted Accept during round 2 of RevealVal then accept $ICSIG(D, INT, s)$ and output s' or $F'(0)$ (depending on whether $ICSIG(D, INT, s)$ is s' or $F'(x)$). Else reject $ICSIG(D, INT, s)$.

In our protocols, we use $\mathsf{AuthVal}^{(1)}$, $\mathsf{AuthVal}^{(2)}$ to denote the first round and second round of AuthVal respectively. Similarly $\mathsf{RevealVal}^{(1)}$, $\mathsf{RevealVal}^{(2)}$ are used for RevealVal. By $\mathsf{ICP}_{\mathsf{sh}}(X, Y, s)$, we mean an execution $\mathsf{Distr}(X, Y, s)$ followed by $\mathsf{AuthVal}(X, Y, s)$. In order to make the presentation clearer, we sometimes use $\mathsf{ICP}_{\mathsf{rec}}(X, Y, s)$ in place of $\mathsf{RevealVal}(X, Y, s)$. Also, in an execution $\mathsf{ICP}_{\mathsf{sh}}(X, Y, s)$, we say that X *conflicts* with Y, if X had to broadcast correctional information in $\mathsf{AuthVal}^{(2)}(X, Y, s)$. Lastly we say that "$(F(x), d, B(x))$ is *consistent* with (α, v, r)" if at least one of the following holds:

1. $F(\alpha) = v$.
2. $B(\alpha) \neq dv + r$.

4 3-Round Statistical VSS with Optimal Resilience

In this section, we present a 3-round statistical VSS protocol with optimal resilience. Although the complexity of the protocol is exponential in terms of the number of parties, the protocol proves optimality of the lower bound from [8]. We also show an efficient 4-round statistical VSS protocol in Section 5.

In our 3-round VSS protocol, the dealer additively shares the secret s into $\binom{n-1}{t}$ shares. Loosely speaking, each of the $\binom{n-1}{t}$ shares correspond to a t-sized subset in $\mathcal{P} - \{D\}$. Then the dealer runs a "VSS-like" subprotocol to share s_m amongst the players in the t-sized subset $S_m \subseteq \mathcal{P} - \{D\}$. In the reconstruction phase, the shares corresponding to each subset are reconstructed first. These shares, in turn, are used to reconstruct the original secret s.

We begin by describing a subroutine that we call U-VSS.

4.1 *U*-VSS

The goal of the U-VSS sub-routine, is to achieve VSS-like functionality for a subset U (with $|U| = t$) of the player set $\mathcal{P}$. In particular, we want correctness and commitment property to hold as in the definition of VSS. However, the privacy requirement needs to met only when all players in $U \cup \{D\}$ are honest.

Informally, the 3 rounds of the U-VSS protocol can be described as follows:

- In Round 1, D sends the secret s to all players in U. Players in U exchange random pads with each other.
- In Round 2, each player in U authenticates his share (via $\mathsf{AuthVal}$). In addition he also broadcasts the secret masked with random pads received from other players in U. Players in U also authenticate random pads received from each other.
- In Round 3, D resolves conflicting broadcasts (if necessary, by broadcasting s to all players). Players finish authenticating their shares with D and their random pads with one another.

Unfortunately the U-VSS protocol described above does not guarantee commitment as such because players in U might (pretend to) have conflicts over random pads, thereby having an option to reveal different random pads in the reconstruction phase. To see this, consider the case when $n = 5$ and $t = 2$. Without loss of generality, let $U = \{P_2, P_3\}$. In round 3, P_2 might (or pretend to) be unhappy (i.e., the $\mathsf{AuthVal}^{(2)}$ check fails) with P_3's authentication of random pad r_{23} (sent by P_2 to P_3). This would result in P_2 broadcasting $F^{(2)}(x)$ and r_{23}. Similarly P_3 might (or pretend to) be unhappy with P_2 over r_{32}. Note that other players have no information about r_{23} and r_{32}. In this case, players in $\mathcal{P} - (U \cup \{D\})$ cannot distinguish (by the end of the sharing phase) between the following 3 cases:

1. (D and P_2 are dishonest.) P_2 broadcasted incorrect authentication information for r_{32} (thereby making P_3 unhappy over r_{32}) and pretends to be unhappy over P_3's broadcast related to r_{23}.
2. (D and P_3 are dishonest.) P_3 broadcasted incorrect authentication information for r_{23} (thereby making P_2 unhappy over r_{23}) and pretends to be unhappy over P_2's broadcast related to r_{32}.
3. (P_2 and P_3 are dishonest.) Both pretend to be unhappy over each other's broadcast related to random pads r_{23} and r_{32}.

Note that in Case (3), an honest D cannot detect any foul play by end of the 2^{nd} round. [3] If we are in Cases (1) or (2), then we have dishonest majority in $U \cup \{D\}$. Thus a dishonest D could take sides with either P_2's reveal or with P_3's reveal in the reconstruction phase. Depending on which player he supports, different secrets could be reconstructed. Note that the players in $\mathcal{P} - (U \cup \{D\})$ *may* not be able to tell whether P_2 or P_3 is honest and whose version of the secret they need to output.

However, in executions where there are no unresolved mutual conflicts, U-VSS does achieve the desired VSS properties. Looking back at the $n = 5, t = 2$ case, we motivate our definition of *mutual conflict* in the general case:

Definition 2. *A* mutual conflict *is said to exist in an execution of U-VSS if*

1. *Some P_i broadcasted $r_{ij}, F^{(i)}(x)$ for some P_j;* and

[3] If we allowed one more round, then Case (3) can be resolved in the following way. When any player broadcasts a "correction" value on a random pad, D will broadcast the secret s in the fourth round. With this modification, commitment can be achieved easily.

2. *P_j also broadcasted $r_{ji}, F^{(j)}(x)$;* and
3. *D did not broadcast s in round 3 of the sharing phase.* ♢

To begin with, we want our U-VSS protocol to satisfy the following weak property: If there is no *mutual conflict* in an execution of U-VSS, then:

- If all players in $U \cup \{D\}$ are honest, then no information about s is revealed to players in $\mathcal{P} - (U \cup \{D\})$ at the end of the sharing phase.
- If D is honest, then D is not discarded in the sharing phase. Also, if there is no mutual conflict then the value shared by D is reconstructed with high probability.
- There exists a value s', such that D is committed to s' at the end of the sharing phase. This s' is reconstructed in the reconstruction phase.

4.2 A Protocol for *U*-VSS

We present a protocol for U-VSS protocol which satisfies the above requirements.
Inputs: Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ denote the set of players and let $D = P_1$ be the dealer with input s. Let $U \subset \mathcal{P}$ be the target subset with $|U| = t$.
Sharing Phase:
Round 1:

1. Execute $\mathsf{ICP}_{\mathsf{sh}}(D, P_i, s)$. for every party P_i in the subset U. Let P_i receive s from D as $s^{(i)}$. Denote the polynomials used in $\mathsf{Distr}(D, P_i, s)$ by $F^{(i)}(x)$, $R^{(i)}(x)$ (both are random t-degree polynomials with $F^{(i)}(0) = s^{(i)}$).
2. For each pair (P_i, P_j) from subset U, party P_i picks a random value r_{ij} and executes $\mathsf{ICP}_{\mathsf{sh}}(P_i, P_j, r_{ij})$ for every $P_j \in U \cup \{D\}$. Let P_j receive r_{ij} from P_i as r'_{ij}.

Round 2: Each $P_i \in U \cup \{D\}$ broadcasts $a_{ij} := s^{(i)} + r_{ij}$ and $b_{ij} := s^{(i)} + r'_{ji}$ for every $P_j \in U \cup \{D\}$.
Round 3:

1. If for some $P_i, P_j \in U \cup \{D\}$, $a_{ij} \neq b_{ji}$ or $a_{ji} \neq b_{ij}$, then D broadcasts s.
2. If P_i conflicts with P_j, then he broadcasts $r_{ij}, F^{(i)}(x)$.

Local Computation: D is **discarded** if for some $P_i, P_j \in U \cup \{D\}$, $a_{ij} \neq b_{ji}$ or $a_{ji} \neq b_{ij}$, and D did not broadcast s.
Reconstruction Phase: If D broadcasted s in round 3 of the sharing phase, then each player P_i sets $s^{(i)} := s$ and outputs s and terminates.

If there is a *mutual conflict* then each player (in $\mathcal{P}$) outputs $\perp$ and the reconstruction phase terminates. Else,

1. Each $P_i \in U$ executes $\mathsf{ICP}_{\mathsf{rec}}(D, P_i, s)$ and each $P_j \in U \cup \{D\}$ executes $\mathsf{ICP}_{\mathsf{rec}}(P_i, P_j, r_{ij})$.
2. D broadcasts the secret s.

Local Computation: Construct GOOD in the following way: For $P_i \in U$, include P_i in GOOD if

1. P_i is successful in revealing $s^{(i)}$.
2. For each P_j that did not conflict with P_i, P_i is successful in revealing r'_{ji}.
3. For every r'_{ji} revealed by P_i in the previous step, $a_{ji} = s^{(i)} + r'_{ji}$ holds.
4. If r'_{ij} was successfully revealed by any P_j, $a_{ij} = s^{(i)} + r'_{ij}$ holds.

Compute s' as follows:

1. If GOOD is empty, then $s' := s$, where s is D's broadcast in Step 2.
2. Else pick any $P_i \in$ GOOD and assign $s' := s^{(i)}$.

Output s'.

4.3 Proofs

We show that the U-VSS protocol presented above satisfies the necessary requirements through a series of claims.

The following claim is proved by means of a standard argument. We omit the proof due to space limitations.

Claim 1. If all players in $U \cup \{D\}$ are honest, then no information about s is revealed to players in $\mathcal{P} - (U \cup \{D\})$ at the end of the sharing phase.

It is easy to see that an honest D is never discarded in the sharing phase.

Claim 2. If there is no mutual conflict then the value shared by honest D, say s, is reconstructed with high probability.

Proof. Since only the values held by $P_i \in$ GOOD are reconstructed, we need to argue that a dishonest P_i is contained in GOOD only if he reveals $s^{(i)} = s$. This is easily shown since when D is honest, by **Correctness 3**, every successful reveal is equal to s.

Claim 3. If D is not discarded, then for all honest P_i, $s^{(i)} = s'$ for some s'.

Proof. Assume that honest players $P_i, P_j \in U$ received shares $s^{(i)}, s^{(j)}$, with $s^{(i)} \neq s^{(j)}$. Then in round 2, a_{ij} is not equal to b_{ji}. Therefore, D has to broadcast s, otherwise he is discarded. Consequently every P_i sets $s^{(i)} := s'$ (see Local Computation).

The following claim can be easily verified.

Claim 4. If D is not discarded, and does not broadcast s in the sharing phase, then with high probability, all honest players in U are contained in GOOD.

Claim 5. If there is no mutual conflict, then there exists a value s' such that D is committed to s' at the end of the sharing phase. This s' is reconstructed in the reconstruction phase.

Proof. When D is honest, the claim follows from Claim 2. Assume D is dishonest. If D is discarded in the sharing phase, then the claim trivially holds. In the following, we assume that D is not discarded. Since D is dishonest and $U \cup \{D\}$

contains $(t+1)$ players, there exists an honest $P_j \in U$. From Claim 3, we have that all honest players received the same share $s' = s^{(j)}$ (P_j's share) from D. We now show that if there is no *mutual conflict*, then s' is reconstructed.

The idea is to show that any $P_i \in U$ is contained in GOOD only if he reveals $s^{(i)}$ as s'. This would prove the claim, since all honest players are already contained in GOOD (follows from Claim 4).

For the sake of reaching a contradiction, assume that $P_i \in U$ successfully reveals $s^{(i)} \neq s'$. We consider two cases:

Case 1: P_j did not conflict with P_i.
By **Correctness 3**, with high probability, P_i can successfully reveal r'_{ji} only as r_{ji}. Since P_j used r_{ji} to compute a_{ji}, it holds that $a_{ji} \neq s^{(i)} + r'_{ji}$ for $s^{(i)} \neq s'$. Hence in this case, P_i will not be included in GOOD.

Case 2: P_i did not conflict with P_i.
By **Correctness 2**, with very high probability, it holds that P_j successfully revealed r'_{ij} that he received. Since D is not discarded, $a_{ij} = b_{ji} = s' + r'_{ij}$. Observe that the condition "$a_{ij} = s^{(i)} + r'_{ij}$" will not be satisfied for $s^{(i)} \neq s'$. Hence in this case, P_i will not be included in GOOD.

The cases discussed above are sufficient since there are no *mutually conflicting* parties in U, i.e., we do not have to consider the case when both P_i and P_j broadcast the random pads which they had used.

4.4 Building Statistical VSS for $t < n/2$ from U-VSS

In the previous section we saw how U-VSS gives us the desired VSS properties when there is no *mutual conflict*. In this section, we'll develop techniques to cope up with executions in which there is *mutual conflict*. Let's first look at the $n = 5, t = 2$ case. There's a small trick that we can use to achieve commitment: First observe that a *mutual conflict* arises when at least 2 parties in $U \cup \{D\}$ are corrupted. Since $U = \{P_2, P_3\}$ and $t = 2$, all players in $\mathcal{P} - (U \cup \{D\})$ are honest. (For higher n, this is not the case, and hence the difficulty is amplified.) Since conflicting P_2, P_3 would have revealed their polynomials $F^{(2)}(x), F^{(3)}(x)$ (with $F^{(2)}(0) \neq F^{(3)}(0)$) respectively, the reveals for the set U is fixed. Since P_4 and P_5 are honest, the "check points" are also fixed! The key observation is that for an honest D (Case (3)), dishonest P_2, P_3 will not be able to guess the honest "check points" correctly. If D is honest then at least one of the revealed polynomials is not *consistent* with any of the honest "check points" except with negligible probability. So one of P_2, P_3's reveal will not be Accepted.

For general t, n, when we encounter a *mutual conflict* in an U-VSS execution, all players in $\mathcal{P} - (U \cup \{D\})$ are not necessarily honest. So instead of assigning a "check point" to each player, we assign a "check point" to each t-sized subset via an U-VSS protocol. In addition, to avoid the problems caused by *mutual conflicts*, only those U-VSS executions in which is no *mutual conflict* are used to generate the verification points in the reconstruction phase. The reason behind using U-VSS to share the "check points" is that now checking for Consistency is

made public (i.e., dishonest players can no longer arbitrarily broadcast Accept or Reject to force a favorable outcome). U-VSS executions with no *mutual conflict*, guarantee agreement over the revealed check points. This results in an agreement over which of the revealed polynomials are actually *consistent*. There might be two conflicting polynomials both of which satisfy all the check points. However at the end of the sharing phase, the outcome of the check for Consistency is fixed! If two conflicting polynomials do pass the Consistency test, then $\perp$ is reconstructed. Note that this does not violate the commitment property of VSS since whether $\perp$ is reconstructed is fixed at the end of the sharing phase. (We assume that $\perp$ represents a default element in $\mathbb{F}$). Also, dishonest players could possibly reveal incorrect polynomials in the reconstruction phase. We prove that our statistical VSS protocol is robust against such adversarial behavior.

4.5 A 3-Round Protocol for VSS

Inputs: Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ denote the set of players and let $D = P_1$ be the dealer with input s. Let $T \stackrel{\text{def}}{=} 2t - 1$.
Sharing Phase: D additively shares s into $s_1, \ldots, s_K$ where $s_1, \ldots, s_K$ are random subject to $s = s_1 + s_2 + \ldots + s_K$. The following U-VSS executions are run in parallel.

1. Iterate over all t-sized subsets S_m: Execute U-VSS(D, S_m, s_m).
2. For each player subset S_k of size t, D picks "check points" $(\alpha_k^{(m,i)}, v_k^{(m,i)} = F_m(\alpha_k^{(m,i)}), r_k^{(m,i)} = R_{m,i}(\alpha_k^{(m,i)}))$ and sends it to S_k (to check for the polynomials revealed by each $P_i \in S_m$). Execute U-VSS$(D, S_k, (\alpha_k^{(m,i)}, v_k^{(m,i)}, r_k^{(m,i)}))$ for all $P_i \in S_m$, and for every t-sized subset S_m.

Local Computation: D is **discarded** if at least one of the following hold:

1. D is discarded in some execution of U-VSS$(D, S_k, (\alpha_k^{(m,i)}, v_k^{(m,i)}, r_k^{(m,i)}))$.
2. D is discarded in some execution of U-VSS(D, S_m, s_m).

Reconstruction Phase: Let $\mathcal{B} \stackrel{\text{def}}{=} \{S_m \mid D \text{ broadcasted } s_m\}$. Let

$$\mathcal{A}_{m,i} \stackrel{\text{def}}{=} \{S_k \mid \text{There are no } \textit{mutual conflicts} \text{ in an execution of } U\text{-VSS}(D, S_k, (\alpha_k^{(m,i)}, v_k^{(m,i)}, r_k^{(m,i)}))\}$$

Reconstruction phase consists of the following 2 rounds:
Round 1: Iterate over all S_m, and every $P_i \in S_m$: Execute reconstruction phase of U-VSS(D, S_m, s_m), and U-VSS$(D, S_k, (\alpha_k^{(m,i)}, v_k^{(m,i)}, r_k^{(m,i)}))$ (for each $S_k \in \mathcal{A}_{m,i}$).
Round 2: Reveals started in round 1 are completed in this round. Also D broadcasts s_m for each S_m.
Local Computation: Let

$$\mathcal{C}_m \stackrel{\text{def}}{=} \{F_m^{(i)}(x) \mid P_i \in S_m \text{ broadcasted } F_m^{(i)}(x) \text{ and } \textit{mutually conflicted} \text{ with some } P_j \in S_m \text{ in the sharing phase}\}$$

All players reconstruct $\perp$ if for any S_m:

1. There is a player $P_i \in S_m$ with $F_m^{(i)}(x) \in \mathcal{C}_m$ and $(F_m^{(i)}(x), d_m^{(i)}, B_m^{(i)}(x))$ *consistent* with $(\alpha_k^{(m,i)}, v_k^{(m,i)}, r_k^{(m,i)})$, for all $S_k \in \mathcal{A}_{m,i}$; AND
2. There is a player P_j $(\neq P_i) \in S_m$ with $F_m^{(j)}(x) \in \mathcal{C}_m$, $F_m^{(i)}(0) \neq F_m^{(j)}(0)$ and $(F_m^{(j)}(x), d_m^{(j)}, B_m^{(j)}(x))$ *consistent* with $(\alpha_k^{(m,j)}, v_k^{(m,j)}, r_k^{(m,j)})$ for all $S_k \in \mathcal{A}_{m,j}$.

If $\perp$ is not reconstructed, then for each $S_m \notin \mathcal{B}$ construct $\mathsf{GOOD_m}$ in the following way: Include $P_i \in S_m$ in $\mathsf{GOOD_m}$ if

1. P_i is contained in GOOD corresponding to the execution U-VSS(D, S_m, s_m).
2. $(F_m^{(i)}(x), d_m^{(i)}, B_m^{(i)}(x))$ is *consistent* with $(\alpha_k^{(m,i)}, v_k^{(m,i)}, r_k^{(m,i)})$ for all $S_k \in \mathcal{A}_{m,i}$ (where $F_m^{(i)}(x), d_m^{(i)}, B_m^{(i)}(x)$ are internal variables in $\mathsf{ICP_{sh}}(D, P_i, s_m)$ corresponding to U-VSS(D, S_m, s_m) with $P_i \in S_m$). Let $s_m^{(i)} = F_m^{(i)}(0)$.

Compute s'_m (which is D's commitment to S_m) as follows:

1. For $S_m \in \mathcal{B}$, set s'_m to be the one broadcasted by D during round 3 of the sharing phase.
2. For $S_m \notin \mathcal{B}$, pick any $P_i \in \mathsf{GOOD_m}$ and set $s'_m = s_m^{(i)}$. If $\mathsf{GOOD_m}$ is empty, then $s'_m = s_m$, where s_m is D's broadcast in round 2 of reconstruction phase.

Reconstruct D's secret as $s' = \sum_{m=1}^{K} s'_m$.

4.6 Proof of Correctness for 3-Round-VSS

We now prove that 3-Round-VSS satisfies all the properties required of a statistical VSS protocol. Let $T \stackrel{\text{def}}{=} 2^t - 1$.

The following lemma is proved by means of a standard argument. We omit the proof due to space limitations.

Lemma 1. *(**Secrecy**) Protocol 3-round-VSS satisfies perfect secrecy.*

Lemma 2. *(**Correctness**) Protocol 3-Round-VSS satisfies $(1 - \varepsilon)$-correctness property.*

Proof. It is easy to see that an honest D is never discarded in the sharing phase. We now show that with high probability, $\perp$ is not reconstructed whenever D is honest.

The only possibility of $\perp$ getting reconstructed is when there exist two *mutually conflicting* players $P_i, P_j \in S_m$ (with $S_m \notin \mathcal{B}$) such that $(F_m^{(i)}(x), d_m^{(i)}, B_m^{(i)}(x))$, $(F_m^{(j)}(x), d_m^{(j)}, B_m^{(j)}(x))$ are *consistent* with $(\alpha_k^{(m,i)}, v_k^{(m,i)}, r_k^{(m,i)})$, $(\alpha_l^{(m,j)}, v_l^{(m,j)}, r_l^{(m,j)})$ (respectively) for all $S_k \in \mathcal{A}_{m,i}$ and $S_l \in \mathcal{A}_{m,j}$. Since D is honest, at least one of P_i, P_j has to be dishonest (otherwise they wouldn't conflict on random pads and broadcast their polynomials).

The key point is that there is at least one set, say $S_l (\neq S_m)$ which contains only honest players. Since all the players are honest, there is no *mutually conflicting* pair in S_l. As a result, $S_l \in \mathcal{A}_{m,i} \cap \mathcal{A}_{m,j}$. By Claim 1, no information

is revealed about $(\alpha_l^{(m,i)}, v_l^{(m,i)}, r_l^{(m,i)}), (\alpha_l^{(m,j)}, v_l^{(m,j)}, r_l^{(m,j)})$. Also the correct values $(\alpha_l^{(m,i)}, v_l^{(m,i)}, r_l^{(m,i)}), (\alpha_l^{(m,j)}, v_l^{(m,j)}, r_l^{(m,j)})$, as shared by D, are revealed in the reconstruction phase of the corresponding U-VSS protocols (follows from Claim 2). So if a dishonest player, say P_i is able to discard an honest D by revealing $F_m^{(i)}(x) \neq F_m(x)$, then he must have guessed $\alpha_l^{(m,i)}$ (follows from the proof of **Correctness 3**). This happens with negligible probability.

Given that $\perp$ is not reconstructed, a dishonest P_i revealing $F_m^{(i)}(x) \neq F_m(x)$ can be in $\mathsf{GOOD_m}$ only if he guessed $\alpha_l^{(m,i)}$ where S_l is the set of honest players (as described above). Again, this happens with negligible probability. Correctness follows immediately.

Claim 6. If a corrupted D is not discarded, then for every S_m, at least one honest player is contained in $\mathsf{GOOD_m}$ with very high probability.

Proof. First, let us fix an S_m. By Claim 5 (commitment property for U-VSS), we have that for every tuple $(\alpha_k^{(m,i)}, v_k^{(m,i)}, r_k^{(m,i)}) \in \mathcal{C}_m$, the exact tuple was held by (all) the honest player(s) in S_k. This essentially makes every verification "check point" behave as if it were possessed by an honest player. Now, from the proof of **Correctness 2** for ICP [4] , each honest player in S_m is *consistent* with "check points" in $\mathcal{C}_m$ with high probability $(1 - \frac{1}{|\mathbb{F}|-1})$.

Suppose there are k honest players in S_m. By the above argument, the claim can fail for a given S_m, only if it fails for each honest player in S_m. This happens with probability at most $\frac{1}{(|\mathbb{F}|-1)^k}$ [5]. Since there are $\binom{t+1}{k}\binom{t-1}{t-k}$ such S_m, the probability that the claim fails for any one such S_m is bounded by $\frac{t^{2k}}{|\mathbb{F}|^k}$. Summing over all k, we see that D can cause the claim to fail for any one S_m with probability at most $\frac{2t^2}{|\mathbb{F}|} = 2^{-\Theta(\kappa)}$. Hence the claim holds.

Lemma 3. ***(Commitment)*** *Protocol 3-Round-VSS satisfies* $(1-\varepsilon)$*-commitment property.*

Proof. For an honest D, the lemma follows from Lemma 2. In the following, we assume that D is dishonest. First we show that whether or not, $\perp$ is reconstructed, is fixed at the end of the sharing phase. Note that the polynomials in $\mathcal{C}_m$ are taken from the sharing phase. Also, the "check points" for these polynomials are fixed at the end of the sharing phase (by the commitment property of U-VSS proved in Claim 5). Therefore, the decision of whether $\perp$ is reconstructed, is fixed at the end of the sharing phase. Since $\perp \in \mathbb{F}$ (by our assumption), we achieve commitment even when $\perp$ is reconstructed.

[4] The proof is identical since in both cases we are dealing with a dishonest D and an honest intermediary. In both cases, the dealer wasn't unhappy with $\mathsf{AuthVal}^{(1)}(D, P_i, s)$, where s is the dealer's secret.

[5] We have used the fact that a corrupt D's ability to cause failure for a particular honest player is *independent* of his ability to cause failure for a different honest player. This is true because D can cause failure for an honest $P_i \in S_m$, only by guessing $d_m^{(i)}$ (follows from the proof of **Correctness 2**). A different honest player $P_j \in S_m$, chooses $d_m^{(j)}$ independent of $d_m^{(i)}$. Hence our argument is justified.

We prove commitment in the case when $\perp$ is not reconstructed. By Claim 6, we now need to prove that for each $S_m \notin \mathcal{B}$, the share held by honest player(s), say $s'_m = s_m^{(j)}$ for some honest P_j, will be reconstructed with high probability (Recall that, by Claim 3, all honest players in $U = S_m (\notin \mathcal{B})$ have the same share).

Let us assume (for the sake of reaching a contradiction) that some dishonest $P_i \in S_m$ successfully reveals some $s_m^{(i)} \neq s_m^{(j)}$. Let $a_{ij}^m, b_{ij}^m, r_{ij}^m$ be the internal variables used in U-VSS(D, S_m, s_m) with $P_i, P_j \in S_m$. We consider two cases:

Case 1: P_j did not broadcast r_{ji}^m in round 3 of the sharing phase.
By **Correctness 3**, with very high probability, P_i can successfully reveal $r_{ji}^{m\prime}$ only as r_{ji}^m. Since P_j computed $a_{ji}^m := s_m^{(j)} + r_{ji}^m$, it holds (with high probability) that $a_{ji}^m \neq s_m^{(i)} + r_{ij}^{m\prime}$ for $s_m^{(i)} \neq s_m^{(j)}$. Hence in this case, P_i will not be included in $\mathsf{GOOD_m}$.

Case 2: P_i did not broadcast r_{ij}^m in round 3 of the sharing phase.
By **Correctness 2**, with very high probability, it holds that P_j revealed $r_{ij}^{m\prime}$ as the random pad that he used in computing $b_{ji}^m := s_m^{(j)} + r_{ij}^{m\prime}$. Since $S_m \notin \mathcal{B}$, and since D is not discarded, we have $a_{ij}^m = b_{ji}^m$. Therefore, the condition "$a_{ij}^m = s_m^{(i)} + r_{ij}^{m\prime}$" will not be satisfied for any $s_m^{(i)} \neq s_m^{(j)}$. Hence in this case, P_i will not be included in $\mathsf{GOOD_m}$.

We do not have to consider the case when both P_i, P_j broadcasted the random pads which they had used (in round 3). This is because if some P_i revealed $F_m^{(i)}(x)$ (with $F_m^{(i)}(0) \neq s'_m$) *consistent* with the all the revealed "check points", then $\perp$ will be reconstructed. Hence an honest P_j's share (i.e., $s_m^{(j)} = s'_m$) is reconstructed always. Given this, commitment follows immediately.

The theorem follows from Lemmas 1, 2 and 3.

Theorem 1. *There exists a 3-round statistical VSS protocol tolerating $t < n/2$ malicious parties.*

5 Efficient 4-Round Statistical VSS with Optimal Resilience

We now design a 4-round sharing, 2-round reconstruction $(2t+1, t)$ statistical VSS with polynomial communication complexity. In the protocol, D selects a random symmetric bivariate polynomial $F(x, y)$ such that $F(0, 0) = s$ and sends $f_i(x)$ to P_i. At the end of the sharing phase, if D is not discarded then every honest P_i holds a degree t polynomial $f_i(x)$ such that for every pair of honest parties (P_i, P_j), $f_i(j) = f_j(i)$. This implies that if D is not discarded, then the $f_i(x)$ polynomials of the honest parties define a symmetric bivariate polynomial $F^H(x, y)$. Moreover in the protocol, it is ensured by using the properties of $ICSig$ that no corrupted P_i will be able to disclose $f'_i(x) \neq f_i(x)$ in the reconstruction phase. Hence irrespective of whether D is honest or corrupted, reconstruction of

$s = F^H(0,0)$ is enforced. To achieve all the properties of VSS, D gives $ICSig$ to individual parties, and concurrently every individual party gives $ICSig$ to every other party. The protocol is somewhat inspired by the VSS protocol of [3]. As the ICP proposed in [3] takes one round in Distr, 3 rounds in AuthVal and 2 rounds in RevealVal, the VSS of [3] takes at most eleven rounds in the sharing phase.

5.1 The Protocol

Inputs: The dealer has a secret s. Let D be the dealer and let $F(x,y)$ be a *symmetric* bivariate polynomial of degree t in each variable. Let $F(0,0) = s$.
Sharing Phase
Round 1: Let $f_i(x)$ be defined as $F(i,x)$. Let $r_{ij} \in_R \mathbb{F}$ for each P_i, P_j. Execute $\mathsf{ICP}_{\mathsf{sh}}(D, P_i, f_i(j))$, $\mathsf{ICP}_{\mathsf{sh}}(P_i, P_j, r_{ij})$ and $\mathsf{ICP}_{\mathsf{sh}}(P_i, D, r_{ij})$. Let the corresponding values received be $f'_i(j)$, r'_{ij} and r^D_{ij}.
Round 2:

1. P_i broadcasts $a_{ij} = f'_i(j) + r_{ij}$ and $b_{ij} = f'_i(j) + r'_{ji}$.
2. D broadcasts $a^D_{ij} = f_i(j) + r^D_{ij}$ and $b^D_{ij} = f_i(j) + r^D_{ji}$.
3. If P_i received $f'_i(x)$ which is not a polynomial of degree t, then P_i executes $\mathsf{ICP}_{\mathsf{rec}}(D, P_i, f'_i(j)$ for all j.

Round 3:

1. If D *conflicts* with P_i or $a_{ij} \neq a^D_{ij}$ or $a_{ij} = \perp$, then D broadcasts $f^D_i(x) = f_i(x)$ and executes $\mathsf{ICP}_{\mathsf{rec}}(P_i, D, r^D_{ik})$ and $\mathsf{ICP}_{\mathsf{rec}}(P_k, D, r^D_{ki})$ for all k.
2. If P_i *conflicts* with P_j or $a_{ij} \neq b_{ji}$ or $a_{ji} \neq b_{ij}$ or $a_{ij} \neq a^D_{ij}$ or $b_{ij} \neq b^D_{ij}$, then P_i executes $\mathsf{ICP}_{\mathsf{rec}}(D, P_i, f'_i(j))$ and $\mathsf{ICP}_{\mathsf{rec}}(P_j, P_i, r'_{ji})$.
3. If P_i *conflicts* with D, then he executes $\mathsf{ICP}_{\mathsf{rec}}(D, P_i, f'_i(k))$, for all k.

Round 4: Corresponding $\mathsf{ICP}_{\mathsf{rec}}$ executions are completed in this round.
Local Computation: D is **discarded** if for some P_i, P_j, at least one of the following does not hold:

1. $\{f'_i(k)\}_k$ lie on a t-degree polynomial.
2. $f^D_i(j) = f^D_j(i) = f'_i(j) = f'_j(i)$.
3. $a^D_{ij} = b^D_{ji} = f^D_j(i) + r^D_{ij}$.
4. All $\mathsf{ICP}_{\mathsf{rec}}(D, P_i, r^D_{ij})$ reveals were successful (i.e., at least $t+1$ accepts were broadcasted).

Reconstruction Phase: Every P_i executes (if they haven't already) $\mathsf{ICP}_{\mathsf{rec}}(D, P_i, f_i(j))$, $\mathsf{ICP}_{\mathsf{rec}}(P_j, P_i, r_{ji})$ for all P_j.
Local Computation: Let $P_i \in \mathsf{U}$ if D broadcasted $f^D_i(x)$. Construct Rec in the following way:

1. $P_i \in \mathsf{Rec}$ if $P_i \in \mathsf{U}$. In this case, define $f'_i(x) = f^D_i(x)$.
2. $P_i \in \mathsf{Rec}$ if he successfully executed $\mathsf{ICP}_{\mathsf{rec}}(D, P_i, f_i(j))$ for all j, and they lie on a t-degree polynomial.

Delete $P_i \notin \mathsf{U}$ from Rec if

1. P_i successfully revealed $f'_i(j)$ and $f'_i(j) \neq f_j^D(i)$ for some $P_j \in \mathsf{U}$.
2. P_j successfully revealed r'_{ij} and $f'_i(j) + r'_{ij} \neq a_{ij}$.
3. If for some P_j, P_j did not *conflict* with P_i and $b_{ij} - r'_{ji} \neq f'_i(j)$.

Reconstruct a symmetric bivariate polynomial $F'(x, y)$ of degree t from $\{f'_i(x)\}_{P_i \in \mathsf{Rec}}$. Output $s' = F'(0, 0)$.

5.2 Proofs

Note that in our 4-Round-VSS protocol, ICP properties **Correctness 1**, **Correctness 2**, **Correctness 3** hold for concurrent executions of $\mathrm{ICP}(P_i, P_j, r_{ij})$ and $\mathrm{ICP}(P_i, D, r_{ij})$. Also when D is honest, **Secrecy** holds for concurrent executions of $\mathrm{ICP}(P_i, P_j, r_{ij})$ and $\mathrm{ICP}(P_i, D, r_{ij})$.

The following lemma is proved by means of a standard argument.

Lemma 4. *(**Secrecy**) Protocol 4-round-VSS satisfies perfect secrecy.*

Claim 7. If D is not discarded and P_i is honest, then for every $P_j \in \mathsf{U}$, $f'_i(j) = f_j^D(i)$.

Proof. If $P_i \in \mathsf{U}$, then $f'_i(x) = f_i^D(x)$, and since D is not discarded, the claim holds. Now let $P_i \notin \mathsf{U}$. Recall that $P_j \in \mathsf{U}$ because D *conflicted* with P_j (over some value $f_j(k)$) OR because $a_{jk} \neq a_{jk}^D$ OR $a_{jk} = \perp$. As a result D reveals r_{ij} (Round 3 Step 1). Recall that $P_i \notin \mathsf{U}$. Therefore, w.h.p, his reveals are successful. Now there are two cases to consider. First, if P_i conflicts with D, then he reveals $f'_i(k)$ as well (Round 3 Step 3). If $f'_i(j) \neq f_j^D(i)$, then D is discarded (see Local Computation). On the other hand, if P_i did not conflict with D, then D has to reveal the correct value of r_{ij} (follows from **Correctness 3**), i.e. $r_{ij}^D = r_{ij}$. Since $P_i \notin \mathsf{U}$, we have $a_{ij}^D = a_{ij}$. Therefore, for an honest P_i, we have $a_{ij}^D - r_{ij}^D = a_{ij} - r_{ij} = f'_i(j)$. If $a_{ij}^D - r_{ij}^D \neq f_j^D(i)$, then D is discarded (see Local Computation). Therefore, $f'_i(j) = f_j^D(i)$.

Claim 8. If D is not discarded and P_i is honest, then $P_i \in \mathsf{Rec}$.

Proof. If $P_i \in \mathsf{U}$, then $P_i \in \mathsf{Rec}$ by construction. Honest $P_i \notin \mathsf{U}$ successfully reveals $f'_i(j))$ for all j. We now show that none of rules that delete P_i from Rec apply to an honest P_i.

1. By Claim 7, we have that for each $P_j \in \mathsf{U}$, $f'_i(j) = f_j^D(i)$.
2. Since revealed r'_{ij} is equal to r_{ij} w.h.p (by **Correctness 3**), $a_{ij} = f'_i(j) + r'_{ij}$.
3. If P_j did not conflict with P_i, then an honest P_i will be successful in revealing the pad r'_{ji} (by **Correctness 2**). Hence $b_{ij} - r'_{ji} = f'_i(j)$.

Claim 9. If D is not discarded, then $f'_i(j) = f'_j(i)$ for every honest P_i, P_j.

Proof. Recall that when $P_i \in \mathsf{U}$, $f'_i(x) = f_i^D(x)$. When both P_i and P_j are in U, then the claim follows directly. Now suppose $P_i, P_j \notin \mathsf{U}$. For honest P_i, P_j , if $f'_i(j) \neq f'_j(i)$, then $a_{ij} \neq b_{ji}$ and $a_{ji} \neq b_{ij}$. Consequently, P_i, P_j would

have successfully revealed $f'_i(j), f'_j(i)$ respectively (by **Correctness 2**). Since we assume that D is not discarded, the claim follows in this case too.

Lastly, consider the case when exactly one of P_i, P_j is contained in U. W.l.o.g, let $P_i \notin \mathsf{U}$,$P_j \in \mathsf{U}$. If $f'_i(j) \neq f_j^D(i)$, then P_i would have been deleted from Rec. But by Claim 8, we have honest $P_i \in \mathsf{Rec}$. Therefore, the claim must hold.

Recall that there are at least $t+1$ honest players, and by Claim 8 all of them are contained in Rec. By Claim 9, the shares of these honest players are consistent. The following claim is now easy to see:

Claim 10. If D is not discarded then all honest parties are consistent with an unique t-degree symmetric bivariate polynomial, say $F^H(x, y)$.

Claim 11. If D is not discarded and $P_i \in \mathsf{Rec}$, then $f'_i(x)$ is consistent with $F^H(x, y)$.

Proof. By Claim 7, for every $P_i \in \mathsf{U}$, $f_i^D(x)$ is consistent with all the honest players' shares. This implies that $f'_i(x)$ is consistent with $F^H(x, y)$.

Now let $P_i \notin \mathsf{U}$. Since $P_i \in \mathsf{Rec}$, we have $f'_i(j) = f_j^D(i)$ for every $P_j \in \mathsf{U}$ (otherwise, P_i is deleted from Rec). Therefore, if $f'_i(x)$ is inconsistent with $F^H(x, y)$, then $f'_i(j) \neq f'_j(i)$ must hold for some *honest* $P_j \notin \mathsf{U}$. If $a_{ij} \neq b_{ji}$ or $a_{ji} \neq b_{ij}$, then P_i, P_j would reveal $f'_i(j), f'_j(i)$ respectively. Since D was not discarded, we have $f'_i(j) = f'_j(i)$. For the rest of the proof, we assume $a_{ij} = b_{ji}$ and $a_{ji} = b_{ij}$.

If P_i had a conflict with P_j, then P_i reveals $f'_i(j)$. If P_j also had a conflict with P_i, then P_j would have revealed $f'_j(i)$. Since D was not discarded, we have $f'_i(j) = f'_j(i)$. On the other hand, if P_j did not have a conflict with P_i, then P_i would have to reveal $r'_{ji} = r_{ji}$ (follows from **Correctness 3**) Since P_j is honest, $b_{ij} - r_{ji} = f'_j(i)$. If $P_i \in \mathsf{Rec}$, then $b_{ij} - r'_{ji} = f'_i(j)$. Since $r'_{ji} = r_{ji}$, this shows that $f'_i(j) = f'_j(i)$. Hence $f'_i(x)$ is consistent with $F^H(x, y)$.

On the other hand if P_i did not have a conflict with P_j, an honest P_j would successfully reveal r'_{ij} . Since $a_{ij} = b_{ji} = f'_j(i) + r'_{ij}$, P_i would have to reveal $f'_i(x)$ such that $f'_i(j) = f'_j(i)$, otherwise $a_{ij} \neq f'_i(j) + r'_{ij}$, and P_i will be deleted from Rec.

Since $F^H(x, y)$ can be computed from the joint view of the honest parties at the end of the sharing phase, the following claim holds.

Claim 12. If D is not discarded, then $F^H(x, y)$ will be reconstructed in the reconstruction phase. Moreover, this $F^H(x, y)$ is fixed at the end of the sharing phase.

It is easy to see that an honest D is never disqualified. Given this, the next two lemmas follow directly from Claim 12, and the theorem follows from Lemmas 4, 5 and 6.

Lemma 5. ***(Correctness)*** *Protocol 4-Round-VSS satisfies* $(1 - \varepsilon)$*-correctness property.*

Lemma 6. *(**Strong Commitment**) Protocol 4-Round-VSS satisfies* $(1 - \varepsilon)$*-strong commitment property.*

Theorem 2. *There exists an efficient 4-round sharing, 2-round reconstruction* $(2t + 1, t)$ *statistical VSS protocol.*

Acknowledgements

We thank Jonathan Katz for fruitful collaboration during early stages of this research.

References

1. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation. In: 20th Annual ACM Symposium on Theory of Computing (STOC), pp. 1–10. ACM Press, New York (1988)
2. Chor, B., Goldwasser, S., Micali, S., Awerbuch, B.: Verifiable secret sharing and achieving simultaneity in the presence of faults. In: 26th Annual Symposium on Foundations of Computer Science (FOCS), pp. 383–395. IEEE, Los Alamitos (1985)
3. Cramer, R., Damgård, I., Dziembowski, S., Hirt, M., Rabin, T.: Efficient multiparty computations secure against an adaptive adversary. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 311–326. Springer, Heidelberg (1999)
4. Dolev, D., Dwork, C., Waarts, O., Yung, M.: Perfectly secure message transmission. Journal of the ACM 40(1), 17–47 (1993)
5. Fitzi, M., Garay, J.A., Gollakota, S., Pandu Rangan, C., Srinathan, K.: Round-optimal and efficient verifiable secret sharing. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 329–342. Springer, Heidelberg (2006)
6. Gennaro, R., Ishai, Y., Kushilevitz, E., Rabin, T.: The round complexity of verifiable secret sharing and secure multicast. In: 33rd Annual ACM Symposium on Theory of Computing (STOC), pp. 580–589. ACM Press, New York (2001)
7. Katz, J., Koo, C.-Y., Kumaresan, R.: Improving the round complexity of VSS in point-to-point networks. Information and Computation 207(8), 889–899 (2009)
8. Patra, A., Choudhary, A., Rabin, T., Rangan, C.P.: The round complexity of verifiable secret sharing revisited. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 487–504. Springer, Heidelberg (2009)
9. Patra, A., Choudhary, A., Pandu Rangan, C.: Round efficient unconditionally secure multiparty computation protocol. In: Chowdhury, D.R., Rijmen, V., Das, A. (eds.) INDOCRYPT 2008. LNCS, vol. 5365, pp. 185–199. Springer, Heidelberg (2008)
10. Patra, A., Choudhary, A., Pandu Rangan, C.: Simple and efficient asynchronous byzantine agreement with optimal resilience. In: Tirthapura, S., Alvisi, L. (eds.) PODC, pp. 92–101. ACM, New York (2009)
11. Rabin, T., Ben-Or, M.: Verifiable secret sharing and multiparty protocols with honest majority. In: 21st Annual ACM Symposium on Theory of Computing (STOC), pp. 73–85. ACM Press, New York (1989)

General Perfectly Secure Message Transmission Using Linear Codes

Qiushi Yang* and Yvo Desmedt**

Department of Computer Science, University College London, UK
{q.yang,y.desmedt}@cs.ucl.ac.uk

Abstract. We study perfectly secure message transmission (PSMT) from a sender S to a receiver R in the general adversary model. In this model, instead of being bounded by a threshold, the Byzantine adversary in a network is characterized by an adversary structure. By regarding monotone general access structures as linear codes, we introduce some new properties that allow us to design efficient PSMT protocols. We give a number of efficient PSMT protocols in both undirected and directed network graphs. These protocols comprehensively improve the transmission complexity of some previous results in this area. More significantly, as all of our protocols are executed in either 3 or 2 rounds, our result is the first, in the context of PSMT in the general adversary model, to have constant round complexity when using interaction.

Keywords: perfectly secure message transmission, adversary structure, linear codes, transmission complexity, round complexity.

1 Introduction

In most of the communication networks, a *sender* S and a *receiver* R are connected by unreliable and distrusted channels. The distrust of the channels is because of the assumption that there exists a *Byzantine adversary* who, with unbounded computational power, can control some nodes on these channels. The aim of *perfect secure message transmission* (PSMT) is to enable a secret message to be transmitted from S to R with perfect *privacy* and *reliability*. That is, the adversary should learn no information about the message, and the receiver R can output the message correctly.

Initial study by Dolev et al. [9] shows that PSMT is possible by applying secure transmission protocols. It assumes a threshold adversary who can control up to t nodes, and hence can control up to t channels. Extensive studies on the threshold model have been carried out ever since (e.g., [7,22,2,15]).

There are many other studies on a more general adversary model, which allow an adversary to control nodes in a less symmetric way. In many cases, using a

* The author would like to acknowledge financial support from UCL PhD Studentship.
** Part of this work was done while funded by EPSRC EP/C538285/1, by BT as BT Chair of Information Security, and by RCIS, AIST, Japan.

M. Abe (Ed.): ASIACRYPT 2010, LNCS 6477, pp. 448–465, 2010.

Table 1. PSMT in the general adversary model

	Network graph	RC	TC over 1	TC over ℓ
Kumar et al. [14]	undirected	$O(n)$	$O(hn^2)$	–
Desmedt et al. [8]	directed-1	1	$O(\|\mathcal{A}\|n)$	–
Yang-Desmedt [24]	directed-2	expo. in $\|\mathcal{A}\|$	expo. in $\|\mathcal{A}\|$	–
Our result	undirected	3 (Section 4.1)	$O(hn^2)$	$O(h\ell)$
		2 (Section 4.2)	$O(hn^2)$	$O(hn\ell)$
	directed-2	3 (Section 5.1)	$O(h^2n^2)$	$O(hn\ell)$
		2 (Section 5.2)	$O(h)$	$O(h\ell)$

* RC denotes round complexity and TC denotes transmission complexity. "TC over 1" is the TC of the PSMT protocol that transmits a single message and "TC over ℓ" is the TC of the protocol that transmits multiple (ℓ) messages, where each message is a field element. "directed-1" are the directed graphs without feedback, and "directed-2" are those with feedback. h is the length of a codeword and n is the number of critical paths (see Section 3).

threshold to model an adversary makes little sense. Indeed, certain platforms are more vulnerable than the others. Also, more hierarchical structures cannot be described by a single adversary. The general adversary model assumes that the adversary is characterized by an adversary structure [11], which consists of a number of subsets of nodes, and the adversary is able to control one of these subsets, instead of any t nodes.

Notable studies on PSMT tolerating adversary structures have been done by Kumar et al. [14] on bi-direction channels, by Desmedt et al. [8] on one-way forward channels, and by both Patra et al. [19] and Yang and Desmedt [24] on mixed forward and feedback channels. However, due to the generality of the adversary structure, the protocols in the previous studies are, in many cases, inefficient in terms of the number of execution rounds[1] (*round complexity*) and the number of *field elements* transmitted (*transmission complexity*). Also some previous results are yet to be further characterized. We shall describe these issues in more detail in Section 3.

Our contributions. In this paper we show how *linear secret sharing schemes* (LSSS) and *linear codes* can be used to design efficient PSMT protocols in the general adversary model. Before we do that, we first show a basic construction of an LSSS and discuss its properties (see Section 2.1). Then we propose a new generalized linear code (see Section 2.2) for the purpose of error-correcting, and also for the purpose of defining *pseudo-basis* and *pseudo-dimension* (see Section 2.3). This follows the idea of Kurosawa and Suzuki [15]. Our study on LSSS and linear codes is shown in Section 2.

Next, in Section 3, we show a further characterization on the problem of PSMT in the general adversary model. We observe that the transmission complexity of most previous PSMT protocols is determined by the number of the *critical paths*.

[1] A round is a transmission from S to R or vice versa.

Thus we shall describe the properties of the critical paths that are effectively used (see Section 3.1). Also in this section, we show how our protocols improve the previous results in terms of *round complexity* (RC) and *transmission complexity* (TC) (see Table 1, which we discuss in detail in Section 3.2). Indeed, not only do we significantly improve the TC of some previous PSMT protocols, but we are also the first to give interactive protocols that have *constant* RC in the studies of the general adversary model. Furthermore, we are also the first to study PSMT over multiple messages in this context.

Section 4 and 5 give our constant round and communication efficient protocols in different network settings. These protocols show comprehensive improvements to the previous results in this area, as shown in Table 1.

2 LSSS and Linear Codes

Secret sharing schemes are key tools in the study of PSMT. Given a set of n participants $P = \{1, \ldots, n\}$, the extensively studied threshold schemes (e.g., Shamir's scheme [20]) allow any subset of $t+1$ participants to learn a secret s, but do not reveal any information of s to any subset of at most t participants. General non-threshold schemes, which realize secret sharing among general *access structures*, are also presented in literature (e.g., Ito et al. [12] and Benaloh and Leichter [3]). A monotone access structure Γ is a family of the subsets of P such that for any set $A \subseteq P$, if $A \in \Gamma$ and $A \subseteq A'$, then $A' \in \Gamma$. Without loss of generality, we assume that $\Gamma \neq \emptyset$. An adversary structure can be defined as $\mathcal{A} = 2^P \setminus \Gamma$. Thus for any set $A \subseteq P$, if $A \in \mathcal{A}$ and $A \supseteq A'$, then $A' \in \mathcal{A}$. It has been shown that LSSS's can be designed for any monotone access structures, so that any set of participants that is in Γ can learn a secret s but any set in $\mathcal{A}$ cannot. Next we show the construction and the properties of such an LSSS.

2.1 Constructing an LSSS

First, it is well-known that monotone span programs are essentially equivalent to LSSS's [13] (see also [5]).

Definition 1. [13] *A* monotone span program *is a triple* $(\mathbb{F}, M, \psi)$*, where* $\mathbb{F}$ *is a finite field,* M *is an* $h \times d$ *matrix* $(h \geq d)$*, and* $\psi : \{1, \ldots, h\} \to \{1, \ldots, n\}$ *is a surjective function that assigns a number of rows in* M *to each participant in* P*.*

For later use, we only allow each row of M to be assigned to a unique participant; i.e., if $\psi(i) = j$, then $\psi(i) \neq j'$ for any $j' \neq j$. This is easy to achieve by making duplicates of the rows that are assigned to multiple participants. Thus h can indicate the total number of shares distributed.

As Shamir's scheme, our construction assumes that $\mathbb{F}$ is sufficiently large. We also assume a message space $\mathbb{M} \subseteq \mathbb{F}$, from which the secret is drawn with respect to a certain probability distribution. Now with $(\mathbb{F}, M, \psi)$, one can share a secret using an LSSS.

Definition 2. *Given a monotone span program* $(\mathbb{F}, M, \psi)$*, a secret* $s \in \mathbb{M}$ *and a random vector* $\mathbf{r} \in \mathbb{F}^{d-1}$*. We regard* $LS : (\mathbb{M}, \mathbb{F}^{d-1}) \rightarrow \mathbb{F}^h$ *as a function such that (T denotes transpose)*

$$LS(s, \mathbf{r}) = M \times (s, \mathbf{r})^T = (s_1, \ldots, s_h)^T,$$

where $s_1, \ldots, s_h$ *are the* h *shares generated by the LSSS, and they are assigned to the* n *participants by* ψ*. For any* $1 \leq t \leq h$ *shares* $s_{i_1}, \ldots, s_{i_t}$ $(1 \leq i_1 < \ldots < i_t \leq h)$*, let* $\psi(i_1, \ldots, i_t)$ *be the set of participants to whom these shares are assigned and* $s_0 \in \mathbb{M}$ *be any possible secret, the LSSS must satisfy the following conditions:*

Secrecy: $\Pr[s = s_0 | s_{i_1}, \ldots, s_{i_t}] = \Pr[s = s_0]$ *if* $\psi(i_1, \ldots, i_t) \in \mathcal{A}$*;*
Reconstruction: $\Pr[s = s_0 | s_{i_1}, \ldots, s_{i_t}] = 0$ *or* 1 *if* $\psi(i_1, \ldots, i_t) \in \Gamma$*.*

Apparently, if $\psi(i_1, \ldots, i_t) \in \Gamma$, then in the linear span of the $i_1, \ldots, i_t$-th rows of M, there must exist the *target vector* $tar = (1, 0, \ldots, 0)$ [13]. This is to satisfy the reconstruction condition.

In the context of the information rate, the size of the secret shares has been studied in literature (e.g., [6,23,4]). However, to the best of our knowledge, there is no results regarding the tight upper bound on the total size of the shares, which is h in our LSSS. In fact, we do not know whether for any access structure, there exists an LSSS with size h polynomial in n. However we can have an exponential size LSSS, which we call *the worst case LSSS*, as follows. The worst case LSSS is defined by a monotone span program $(\mathbb{F}, M_{h \times d}, \psi)$ such that $d = |\mathcal{A}|$ and $h = O(dn)$. h is thus exponential in n because in general $|\mathcal{A}| = O(2^n)$. This construction somehow follows [10] (based on [21]).

The worst case LSSS

Given a set of n participants P and an adversary structure $\mathcal{A}$ on P. Let $\Delta = \{P \setminus A | A \in \mathcal{A}\}$ and $d = |\Delta| = |\mathcal{A}|$. Construct a $d \times d$ matrix M^V, which is an identity matrix except all entries in the first row are changed to 1.
Let $\Delta = \{D_1, ..., D_d\}$, then for each $1 \leq i \leq d$, construct a $|D_i| \times d$ matrix M_i such that each row of M_i is a duplication of the i-th row of M^V. Let $h = \sum_{i=1}^{d} |D_i|$, construct an $h \times d$ matrix M that is filled by $M_1, \ldots, M_d$ from top to bottom.
The function ψ assigns the rows in M to each participant in such a manner that if a participant is in $D_i \in \Delta$ $(1 \leq i \leq d)$, then ψ assigns a row of M_i to this participant. **End.**

See the proof of the secrecy and reconstruction properties of the worst case LSSS in the full version of this paper [1].

2.2 Linear Codes

Given an LSSS defined by $(\mathbb{F}, M_{h \times d}, \psi)$. We denote k as the rank of M, thus $k \leq d$. In the rest of the paper, *we let the first* k *rows of* M *be linearly independent.* Thus $\psi(1, \ldots, k) \in \Gamma$. Indeed, because otherwise $\psi(1, \ldots, k) \in \mathcal{A}$ and

the participants in $\psi(1,\ldots,k)$ can then recover all the other shares using linear combinations. This contradicts the secrecy condition of Definition 2.

Definition 3. *A linear code C is defined by a $k \times h$ generating matrix G in standard form $G = (I_k|A)$ [16], where I_k denotes the $k \times k$ identity matrix and A is a $k \times (h-k)$ matrix.*

The codewords of code C are determined by an encode *function $EC : \mathbb{F}^k \to \mathbb{F}^h$ such that given a k-vector $\mathbf{r} \in \mathbb{F}^k$,*

$$EC(\mathbf{r}) = \mathbf{r} \times G = \mathbf{c},$$

where $\mathbf{c}$ is an h-vector, as a codeword of C, and denoted $\mathbf{c} \in C$.

Evidently code C has $|\mathbb{F}|^k$ codewords.

We link an LSSS with a linear code as follows. In the rest of this section, *we let M_k be a $k \times d$ matrix that consists of the first k rows of M*, so the rank of M_k is k. We construct G in such a manner that the i-th column of G, which we call col_i, has the following property: $(col_i)^T \times M_k = row_i$, where row_i is the i-th row of M. This is possible because the rank of M is k, thus row_i is in the linear span of the first k rows of M (M_k). Therefore, the set $\{LS(s,\mathbf{r})|s \in \mathbb{M}, \mathbf{r} = \mathbb{F}^{d-1}\}$ is a subset of a linear code, because for any $s \in \mathbb{M}, \mathbf{r} \in \mathbb{F}^{d-1}$, we have

$$LS(s,\mathbf{r}) = (s_1,\ldots,s_h) = EC(s_1,\ldots,s_k) \in C.$$

Definition 4. *Let $\mathbf{k}$ be a k-vector such that $\mathbf{k} \times M_k = tar$, where $tar = (1,0,\ldots,0) \in \mathbb{F}^d$ is the target vector*[2]*. Let $\mathbf{r} \in \mathbb{F}^k$. We define a* decode *function $DC : \mathbb{F}^k \to \mathbb{F}$ such that $DC(\mathbf{r}) = \mathbf{r} \times \mathbf{k}^T$. We denote the output of the function, $r = DC(\mathbf{r})$, as* the information of the codeword $\mathbf{c} = EC(\mathbf{r})$.

Theorem 1. *Given any codeword $\mathbf{c} = (c_1,\ldots,c_h) = EC(\mathbf{r}) \in C$. One can decode the information of $\mathbf{c}$ with t entries $c_{i_1},\ldots,c_{i_t}$ $(1 \le i_1 < \ldots < i_t \le h)$ of $\mathbf{c}$ if and only if $\psi(i_1,\ldots,i_t) \in \Gamma$.*

Proof. Let $\mathbf{k}$ be a k vector such that the information of $\mathbf{c}$ is $r = DC(\mathbf{r}) = \mathbf{r} \times \mathbf{k}^T$. Remark that C is defined by G, which is derived from M of the LSSS. Let Λ be a $k \times t$ matrix such that for each $1 \le j \le t$, the j-th column of Λ is the i_j-th column of G, then we have

$$\begin{bmatrix} row_{i_1} \\ \vdots \\ row_{i_t} \end{bmatrix} = \Lambda^T \times M_k, \tag{1}$$

where for each $1 \le j \le t$, row_{i_j} is the i_j-th row of M.

First we show that if $\psi(i_1,\ldots,i_t) \in \mathcal{A}$, then one cannot decode r with $c_{i_1},\ldots,c_{i_t}$. Assume the opposite, i.e., r can be decoded with $c_{i_1},\ldots,c_{i_t}$. Since $r = \mathbf{r} \times \mathbf{k}^T$, the possibility that r can be decoded by $(c_{i_1},\ldots,c_{i_t})$ means that

[2] Because $\psi(1,\ldots,k) \in \Gamma$ as we showed before, $\mathbf{k}$ must exist.

the column vector $\mathbf{k}^T$ is in the linear span of the columns of Λ. That is, there exists a $\mathbf{t}^T$ such that $\mathbf{k}^T = \Lambda \times \mathbf{t}^T$ so that

$$r = \mathbf{r} \times \mathbf{k}^T = \mathbf{r} \times \Lambda \times \mathbf{t}^T = (c_{i_1}, \ldots, c_{i_t}) \times \mathbf{t}^T.$$

Since $\mathbf{k}^T = \Lambda \times \mathbf{t}^T \Rightarrow \mathbf{k} = \mathbf{t} \times \Lambda^T$, by multiplying $\mathbf{t}$ by both sides of Eq. 1, we have

$$\mathbf{t} \times \begin{bmatrix} row_{i_1} \\ \vdots \\ row_{i_t} \end{bmatrix} = \mathbf{t} \times \Lambda^T \times M_k = \mathbf{k} \times M_k = tar.$$

This means that the target vector tar is in the linear span of the rows assigned to the participants $\psi(i_1, \ldots, i_t) \in \mathcal{A}$, which is not allowed in our LSSS due to the secrecy condition.

Next if $\psi(i_1, \ldots, i_t) \in \Gamma$, then by the reverse of the above proof and the reconstruction condition of the LSSS, we can easily prove that one can decode r with $c_{i_1}, \ldots, c_{i_t}$. □

Given that $\mathbf{c} = (c_1, \ldots, c_h)$ is a codeword at the encoding end, and $\mathbf{x} = (x_1, \ldots, x_h)$ is the input at the decoding end, because of the channel noise, it is possible that $\mathbf{x} \neq \mathbf{c}$. We let $\mathbf{e} = (e_1, \ldots, e_h)$ be an *error vector* such that $\mathbf{e} = \mathbf{x} - \mathbf{c}$. Normally we have the following assumption: let $E = \{i | e_i \neq 0\}$ be an *error locator*, we always have $\psi(E) \in \mathcal{A}$. That is, the errors in a codeword are caused by a set in the adversary structure. With this assumption, it is well-known that

- the decoder can detect that $\mathbf{x}$ is not a codeword if and only if $P \notin 2\mathcal{A}$ (i.e., $P \notin \{A_1 \cup A_2 | A_1, A_2 \in \mathcal{A}\}$), where P is the set of all participants, and
- the decoder can decode the information of $\mathbf{c}$ from $\mathbf{x}$ if and only if $P \notin 3\mathcal{A}$ (i.e., $P \notin \{A_1 \cup A_2 \cup A_3 | A_1, A_2, A_3 \in \mathcal{A}\}$).

See a proof of this result in the full version of this paper [1].

2.3 Pseudo-basis and Pseudo-dimension

In Eurocrypt '08, Kurosawa and Suzuki initiated the idea of pseudo-basis and pseudo-dimension in the threshold model with multiple codewords [15]. A generalization of the pseudo-basis and pseudo-dimension is possible if $P \notin 2\mathcal{A}$ (corresponding to $n \geq 2t+1$ in the threshold model), thus *we assume that* $P \notin 2\mathcal{A}$ *in this section*. Next, we let $\psi^{-1} : \{1, \ldots, n\} \to \{1, \ldots, h\}$ be the inverse function of ψ. That is, let $A \subseteq P$, then $\psi^{-1}(A)$ *returns all the locations in a codeword that are assigned to the participants in* A *by* ψ.

Definition 5. *Let* $A \subseteq P$, *we define* $|A|$ *as the* size *of* A *and* $|\psi^{-1}(A)|$ *as the* weight *of* A. *We denote*

$$sz^{\mathcal{A}} = \max\{\text{size of } A | A \in \mathcal{A}\} \text{ and } wt^{\mathcal{A}} = \max\{\text{weight of } A | A \in \mathcal{A}\}.$$

Evidently $sz^{\mathcal{A}} = O(n)$ and $wt^{\mathcal{A}} = O(h)$. The idea of the generalization is as follows. The encoder sends m codewords $\mathbf{c}_1, \ldots, \mathbf{c}_m$, and the decoder receives m h-vectors $\mathbf{x}_1, \ldots, \mathbf{x}_m$ such that for each $1 \leq i \leq m$, $\mathbf{x}_i = \mathbf{c}_i + \mathbf{e}_i$ where $\mathbf{e}_i = (e_{i1}, \ldots, e_{ih})$ is an error vector. For each $\mathbf{e}_i$, let $E_i = \{j | e_{ij} \neq 0\}$ be an error locator, then E_i has the following two properties: (1) $|E_i| \leq wt^{\mathcal{A}}$ and (2) $\psi(E_i) \in \mathcal{A}$ and hence $|\psi(E_i)| \leq sz^{\mathcal{A}}$. We assume that $\bigcup_{i=1}^{m} E_i \in \mathcal{A}$, i.e., the errors in all the codewords are caused by the same set in $\mathcal{A}$. Now we give our pseudo-basis construction scheme as follows.

Pseudo-basis construction scheme

Set $B := \emptyset$. For each $1 \leq i \leq m$, distinguish the following two cases:

1. $B = \emptyset$: if $x_i \in C$, then do nothing, otherwise, then add $\mathbf{x}_i$ in B.
2. Otherwise: let $B = \{\mathbf{x}_{g_1}, \ldots, \mathbf{x}_{g_b}\}$ where $1 \leq g_1 < \ldots < g_b < i$, if there exist $(a_1, \ldots, a_b) \in \mathbb{F}^b$ such that $\mathbf{x}_i + a_1\mathbf{x}_{g_1} + \ldots + a_b\mathbf{x}_{g_b} \in C$, then do nothing, otherwise, add $\mathbf{x}_i$ in B.

Let B be the pseudo-basis. Thus $|B|$ is the pseudo-dimension. **End.**

It is trivial that the pseudo-dimension of our scheme is at most $wt^{\mathcal{A}} = O(h)$, because there are at most $wt^{\mathcal{A}}$ non-zero entries in each error vector. Thus the pseudo-basis has $O(h^2)$ field elements.

Lemma 1. *For any codeword* $\mathbf{c} = (c_1, \ldots, c_h) \in C$, *let* $D = \{i | c_i \neq 0\}$. *If* $P \notin 2\mathcal{A}$ *and* $\psi(D) \in \mathcal{A}$, *then the information of* $\mathbf{c}$ *is* 0.

Proof. Let $O = \{i | c_i = 0\}$. From $P \notin 2\mathcal{A}$ and $\psi(D) \in \mathcal{A}$, we can have $\psi(O) \in \Gamma$. According to Theorem 1, the information of $\mathbf{c}$ can be decoded with all the entries c_i such that $i \in O$. Since all these entries are 0's, the information of $\mathbf{c}$ is 0. □

Given a codeword $\mathbf{c} \in C$ and a vector $\mathbf{x}$, and let $\mathbf{e} = \mathbf{x} - \mathbf{c}$ be an error vector such that $\psi(E) \in \mathcal{A}$. If $\mathbf{e} \in C$, then $\mathbf{x} \in C$. Due to Lemma 1, the information of $\mathbf{e}$ is 0, so the information of $\mathbf{x}$ equals to the information of $\mathbf{c}$. That is, the error vector $\mathbf{e}$ does not actually cause errors, and we call this kind of error vector *invalid*. Evidently, the vector $\mathbf{0} \in \mathbb{F}^h$ is an invalid error vector.

Let $B = \{\mathbf{x}_{g_1}, \ldots, \mathbf{x}_{g_b}\}$ be a pseudo-basis, where $1 \leq g_1 < \ldots < g_b \leq m$, and $E_{g_1}, \ldots, E_{g_b}$ be the respective error locators. we denote $F = \bigcup_{i=1}^{b} E_{g_i}$ as the *final error locator* of B.

Theorem 2. *If the final error locator of a pseudo-basis is known, then the decoder can decode the information of all the codewords.*

Proof. Given the final error locator F of a pseudo-basis $B = \{\mathbf{x}_{g_1}, \ldots, \mathbf{x}_{g_b}\}$, a decoding scheme is as simple as the following:

Decoding scheme from the pseudo-basis

For each $1 \leq i \leq m$, decode the information r_i of $\mathbf{c}_i$ from $\mathbf{x}_i$ such that if $j \in F$, then the j-th entry of $\mathbf{x}_i$ is not used for decoding. **End.**

It is straightforward that if $i \in \{g_1, \ldots, g_b\}$, then the decoded information r_i is correct. Indeed, $P \notin 2\mathcal{A}$ and $\psi(F) \in \mathcal{A}$ imply that $\psi(\{1, \ldots, h\} \setminus F) \in \Gamma$. Thus according to Theorem 1, the entries not indicated by F can be used to decode r_i. Since F contains all the error locations of $\mathbf{x}_i$, all the entries that are used to decode r_i are correct.

Next, if $i \in \{1, \ldots, m\} \setminus \{g_1, \ldots, g_b\}$, then because of the existence of non-zero invalid error vectors, it is possible that $E_i \supsetneq F$. That is, errors may exist in the entries used to decode r_i. Since $\mathbf{x}_i \notin B$, there exist $(a_1, \ldots, a_b) \in \mathbb{F}^b$ such that $\mathbf{x}_i + a_1\mathbf{x}_{g_1} + \ldots + a_b\mathbf{x}_{g_b} \in C$. Thus $\mathbf{e}_i + a_1\mathbf{e}_{g_1} + \ldots + a_b\mathbf{e}_{g_b} \in C$. Let $\mathbf{e}'_i = \mathbf{e}_i + a_1\mathbf{e}_{g_1} + \ldots + a_b\mathbf{e}_{g_b}$, we have that $\mathbf{e}'_i$ is an invalid error vector. Thus one can decode the information r_i of $\mathbf{c}_i$ correctly from the vector $\mathbf{x}'_i = \mathbf{c}_i + \mathbf{e}'_i$. Since $\mathbf{x}_i = \mathbf{c}_i + \mathbf{e}_i$, it is clear that excluding the entries indicated by F, the remaining entries of $\mathbf{x}_i$ are the same as those of $\mathbf{x}'_i$. That is, even though errors may exist in the remaining entries, one can decode the information r_i of $\mathbf{c}_i$ correctly from these entries. □

3 PSMT Preliminaries

We abstract away the concrete network structure and model a network by a graph $G(V, E)$, whose nodes are the parties in the network and edges are point-to-point secure communication channels. We consider two kinds of network graphs in this paper:

1. *Undirected graphs* - in which all the edges are undirected, and allow two-way communication;
2. *Directed graphs* - in which all the edges are one-way directed or bi-directed, and allow mixed communication.

Given an adversary structure $\mathcal{A}$ on the nodes of a graph, we say the sender S and the receiver R are *$d\mathcal{A}$-separated* if there exist d sets $A_1, \ldots, A_d \in \mathcal{A}$ such that all paths between S and R pass through some nodes in $\bigcup_{i=1}^{d} A_i$; otherwise we say they are *$d\mathcal{A}$-connected*.

In the context of PSMT, *perfect security* requires the achievement of *perfect privacy* (i.e., zero probability that the adversary learns the message from the information he gets) and *perfect reliability* (i.e., zero probability that R fails to recover the message correctly). The necessary and sufficient conditions (N&S) for PSMT on different network graphs have been given in previous results:

N&S-undirected: in undirected graphs, S and R are $2\mathcal{A}$-connected [14];
N&S-directed-1: in directed graphs without feedback paths, S and R are $3\mathcal{A}$-connected [8];
N&S-directed-2: in directed graphs with feedback paths, S and R are $2\mathcal{A}$-connected with the forward paths from S to R, and if S and R are $3\mathcal{A}$-separated, then for any three sets $A_1, A_2, A_3 \in \mathcal{A}$ such that $A_1 \cup A_2 \cup A_3$ separates S and R, at most one of these three sets separates S and R on the feedback paths from R to S [19,24].

It can be seen that the paths between S and R play an important role in the study of PSMT. Next we show how a characterization of the *critical paths* determines the PSTM protocols and their transmission complexity (TC).

3.1 Critical Paths

Unlike those in the threshold model, the N&S conditions for PSMT in the general adversary model do not require node-disjoint paths. This rises the question of how to transmit messages in a general network graph. The straightforward solution (though somehow less efficient) is to characterize the graph into all possible paths between S and R. To this end, the idea of *critical paths* was introduced by Kumar et al. [14] in their initial study. We extend their study, by firstly giving a formal definition as follows.

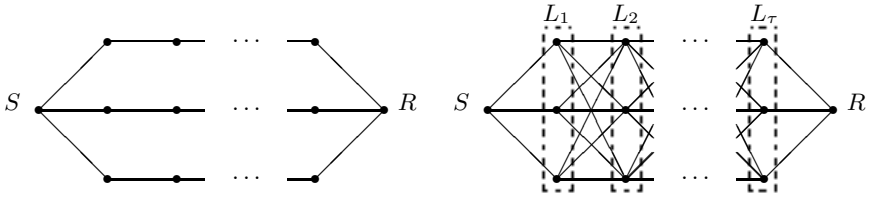
Definition 6. *Given a graph $G(V, E)$, in which S and R are $d\mathcal{A}$-connected. A set of paths W is called* critical*, if S and R are $d\mathcal{A}$-connected with all paths in W, but are $d\mathcal{A}$-separated with all paths in any $W' \subsetneq W$. Let $\mathcal{W}$ be the set of all critical sets of paths, we define a* minimal critical set *W^* such that $W^* \in \mathcal{W}$ and $|W^*| = \min\{|W| : W \in \mathcal{W}\}$.*

Without loss of generality, we assume that there does not exist a trusted path between S and R; i.e., $|W^*| > 1$.

Observation 1. *With any graph in which S and R are $d\mathcal{A}$-connected, $|W^*|$ can be as small as $d+1$ or as large as exponential in the size of the graph.*

We give two examples in Fig. 1. In the examples we assume that S and R are $2\mathcal{A}$-connected. First suppose a graph G_1 is as shown in Fig. 1(a), in which there are only 3 paths between S and R. The adversary structure $\mathcal{A}$ has the following property: all nodes in any set $A \in \mathcal{A}$ are on the same path. Thus it is clear that in G_1, S and R are $2\mathcal{A}$-connected, and all the 3 paths are in W^*.

Next suppose a graph G_2 is as shown in Fig. 1(b). We assume that except S and R, there are 3τ nodes in G_2. We can view S and R as they are connected by τ levels $L_1, \ldots, L_\tau$, where each level L_i $(1 \le i \le \tau)$ is a set of 3 nodes, and there is an edge between each node in L_i and each node in L_{i+1}. The adversary



(a) G_1: $|W^*| = d + 1 = 3$. **(b)** G_2: $|W^*|$ is exponential in n.

Fig. 1. $2\mathcal{A}$-connectivity in different graphs

structure $\mathcal{A}$ has the following property: for each set $A \in 2^P$, if there exist two nodes $v_1, v_2 \in A$ such that $v_1, v_2 \in L_i$ $(1 \leq i \leq \tau)$, then $A \notin \mathcal{A}$; otherwise $A \in \mathcal{A}$. In other words, the adversary E can control at most 1 node of each level.

Obviously S and R are $2\mathcal{A}$-connected in G_2, but if we remove any edge from the graph, then they are $2\mathcal{A}$-separated. Also straightforwardly $|W^*| = 3^\tau$, because the critical paths are all the paths with exactly one node of each level on them. Thus we have that $|W^*|$ is exponential in the size of the network, which is $9\tau - 3$.

Of course our examples can easily be adapted to other connectivity, e.g., $3\mathcal{A}$-connectivity.

Therefore, if a PSMT protocol is executed via the paths in the graphs, then it is impossible to determine its TC in the size of the network, because the number of paths varies remarkably in different graphs with the same connectivity (e.g., G_1 and G_2). Thus we determine TC in the number of critical paths. For this purpose, a re-characterization of the adversary structure is needed.

In general, the participants in an adversary structure are considered to be the nodes in the network graph. We denote this adversary structure as $\mathcal{A}^V$. Given a critical set of paths W, we define a new adversary structure $\mathcal{A}^W$ such that $|\mathcal{A}^W| = |\mathcal{A}^V|$, and for each set $A^V \in \mathcal{A}^V$, there is a corresponding set $A^W \in \mathcal{A}^W$ such that A^W consists of all the paths in W that pass through nodes in A^V.

It is clear that if S and R are $d\mathcal{A}^V$-connected, then they are $d\mathcal{A}^W$-connected with W. In the rest of the paper, we use $\mathcal{A}^W$ as the considered adversary structure. Thus we let $\mathcal{A} = \mathcal{A}^W$ and *the participants of the adversary structure are the critical paths of the network graph.*

3.2 Improvements to the Previous Results

In the rest of the paper, we let $n = |W|$ be the number of critical paths, and $\mathcal{A}$ be an adversary structure over the n paths.

Because the previous protocols use different characterizations for PSMT, it is not straightforward to compare their TC with our result. In fact, we need to compare the three parameters $(n, |\mathcal{A}|, h)$ [3] that determine the TC of the protocols. First we do not know the tight upper bound on h, but our worst case LSSS achieves $h \leq O(|\mathcal{A}|n)$, so h should not be larger. In general $|\mathcal{A}|$ is exponential in n, but due to the way that the critical paths are selected, n can be polynomial in $|\mathcal{A}|$ in some network graph [14]. Either way, our results significantly improve the previous results in terms of round complexity (RC) and transmission complexity (TC) over a single message. We also present some efficient protocols to transmit $\ell > 1$ messages. The problem of multiple message transmission has not been studied before in the general adversary model.

A summary of the results are shown in Table 1 in Section 1. Note that Desmedt et al.'s protocol [8] is executed in directed graphs without feedback, which means that the receiver R cannot send messages to the sender S. Thus

[3] As shown in the previous section, h is the size of the LSSS as well as the length of the codewords.

the protocols in this graph must be non-interactive and can only have 1-round. Their protocol is actually an alternative use of the worst case LSSS that we showed before. Thus the protocol can easily be reformed into a 1-round protocol with TC $O(h)$. The protocol by Yang and Desmedt [24] uses the settings in [19], which require both the RC and TC to be exponential in $|\mathcal{A}|$. As we discussed before, both h and n are at most polynomial in $|\mathcal{A}|$, so our improvements are obvious. We also remark that in the studies of the general adversary model, our results are the first to have constant RC in undirected and directed-2 graphs.

3.3 Other Preliminaries

We assume that each message s is drawn from the message space $\mathbb{M} \subseteq \mathbb{F}$ with respect to a certain probability distribution. Since two different type of graphs are considered, we have the following: in an undirected graph, we denote $W = \{w_1, \ldots, w_n\}$ as a critical set of undirected paths; in a directed graph, we denote $W = \{w_1, \ldots, w_n\}$ as a critical set of the forward paths and $Q = \{q_1, \ldots, q_u\}$ as a critical set of the feedback paths, where $u = O(n)$.

Given that S and R are $2\mathcal{A}$-connected with W, if S sends the same message via all paths in W, then R is able to receive the message perfectly reliably [14]. In our protocols we say "*S broadcasts a message via W*" to indicate this kind of transmission. Thus the TC of the broadcast of 1 field element is $O(n)$.

Note that the linear code is constructed considering the critical paths as the participants. When S sends a codeword $\mathbf{c} = (c_1, \ldots, c_h)$ in such a manner that for each $1 \leq j \leq h$, if $\psi(j) = w_i$ for some $1 \leq i \leq n$, then S sends c_j via w_i, we say "*S sends $\mathbf{c}$ via W with respect to ψ*" to indicate this kind of transmission. Thus the TC of the transmission of 1 codeword is $O(h)$.

In our protocols, we omit some indices for the communication. For example, if S sends a pseudo-basis to R, then generally S should attach a index in the transmission to indicate exactly which codeword each vector in the pseudo-basis corresponds to. Indexing is very cheap in terms of TC. Thus in our protocols, we omit some indices to make the protocols easier to read.

4 PSMT in Undirected Graphs

In this section we show our PSMT protocols in undirected graphs. According to N&S-undirected, S and R must be $2\mathcal{A}$-connected in an undirected graph. We first give 3-round protocols in Section 4.1 for the transmissions of a single message and multiple messages, and then give 2-round protocols in Section 4.2. The protocols given in this section are along the lines of the results in [15].

4.1 3-Round Undirected Protocols

We omit the 3-round protocols in this section due to lack of space, and also because they are relatively simple. However, the TC of our 3-round protocol over a single message is $O(hn^2)$, and the TC of our 3-round protocol over multiple

(ℓ) messages is $O(h\ell)$ where $\ell = wt^{\mathcal{A}}h$. Thus the TC of both protocols are about optimal in the context of PSMT in the general adversary model. For the details of the 3-round undirected protocols, see the full version of this paper [1].

4.2 2-Round Undirected Protocols

First we give a 2-round protocol to transmit a single message.

2-round undirected protocol for a single message s

Round 1 - R **to** S**:**

1. R chooses n random k-vectors $\mathbf{r}_1, \ldots, \mathbf{r}_n \in \mathbb{F}^k$, and for each $1 \leq i \leq n$, R encodes $\mathbf{r}_i$ to get codeword $\mathbf{c}_i = EC(\mathbf{r}_i) = (c_{i1}, \ldots, c_{ih})$.
2. For each $1 \leq i \leq n$, R sends vector $\mathbf{r}_i$ via path w_i, and sends codeword $\mathbf{c}_i$ via W with respect to ψ.

Round 2 - S **to** R**:**

1. S receives n k-vectors $\mathbf{r}'_1, \ldots, \mathbf{r}'_n$ and n h-vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n$ from W. For each $1 \leq i \leq n$, let $\mathbf{x}_i = (x_{i1}, \ldots, x_{ih})$.
2. For each $1 \leq i \leq n$, S encodes $\mathbf{r}'_i$ to get codeword $\mathbf{c}'_i = EC(\mathbf{r}'_i) = (c'_{i1}, \ldots, c'_{ih})$. S then constructs a set D_i such that for each $1 \leq j \leq h$, iff $x_{ij} \neq c'_{ij}$, then $(x_{ij}, j) \in D_i$.
3. S finds a k-vector $\mathbf{r}^S$ such that $s = DC(\mathbf{r}^S)$, and then encodes $\mathbf{c}^S = EC(\mathbf{r}^S) = (c^S_1, \ldots, c^S_h)$. For each $1 \leq j \leq h$, if $\psi(j) = w_i$, then S computes $z_j = c^S_j + c'_{ij}$. Finally S sets $\mathbf{z} = (z_1, \ldots, z_h)$.
4. S broadcasts $\mathbf{z}$ and $D_1, \ldots, D_n$ via W.

Recovery Phase

1. R receives $\mathbf{z}$ and $D_1, \ldots, D_n$ from W.
2. R sets $F := \emptyset$. For each $1 \leq i \leq n$, if there exists a pair $(x_{ij}, j) \in D_i$ such that $x_{ij} = c_{ij}$, then R sets $F := F \cup \{i\}$.
3. For each $1 \leq j \leq h$, if $\psi(j) = w_i$, then R computes $c^R_j = z_j - c_{ij}$. R then decodes s' as the information of $(c^R_1, \ldots, c^R_h)$ such that for any $\psi(j) = w_i$ where $i \in F$, the entry c^R_j is not used for decoding. **End.**

Proof of perfect security. Omitted. See the full version of this paper [1].

TC of the protocol. Let $TC(i)$ be the TC of Round i for $1 \leq i \leq 3$. In this protocol:

$$TC(1) = hn + kn = O(hn)$$
$$TC(2) = O(n(h + 2hn)) = O(hn^2)$$

We have that the total TC is $O(hn^2)$ field elements.

Next, before we show our 2-round PSMT protocol that transmits multiple messages, we employ a well-known technique in this context: the ***randomness extractor*** [22,2,15]. Suppose that the adversary has no knowledge on ℓ out of m random elements $r_1, \ldots, r_m \in \mathbb{F}$. Let $f(x)$ be a polynomial of degree $deg\ f(x) \leq m - 1$ such that $f(i) = r_i$ for each $1 \leq i \leq m$, then the adversary has no knowledge on $z_j = f(m + j)$ for each $1 \leq j \leq \ell$. We denote a function $RE : \mathbb{F}^m \to \mathbb{F}^\ell$ as a randomness extractor such that $RE(r_1, \ldots, r_m) = (z_1, \ldots, z_\ell)$. This function will be used in the following 2-round PSMT protocol.

2-round undirected protocol for $\ell = wt^{\mathcal{A}}(n - sz^{\mathcal{A}} - 1)$ messages $s_1, \ldots, s_\ell$

Round 1 - R to S:

1. R chooses $wt^{\mathcal{A}}n$ random k-vectors $\mathbf{r}_1, \ldots, \mathbf{r}_{wt^{\mathcal{A}}n} \in \mathbb{F}^k$, and for each $1 \leq i \leq wt^{\mathcal{A}}n$, S encodes $\mathbf{r}_i$ to get codeword $\mathbf{c}_i = EC(\mathbf{r}_i) = (c_{i1}, \ldots, c_{ih})$.
2. For each $1 \leq i \leq n$, R sends vectors $\mathbf{r}_{i+0 \cdot wt^{\mathcal{A}}}, \mathbf{r}_{i+1 \cdot wt^{\mathcal{A}}}, \ldots, \mathbf{r}_{i+(wt^{\mathcal{A}}-1)wt^{\mathcal{A}}}$ via path w_i. R also sends codewords $\mathbf{c}_1, \ldots, \mathbf{c}_{wt^{\mathcal{A}}n}$ via W with respect to ψ.

Round 2 - S to R:

1. S receives $wt^{\mathcal{A}}$ k-vectors $\mathbf{r}'_{i+0 \cdot wt^{\mathcal{A}}}, \mathbf{r}'_{i+1 \cdot wt^{\mathcal{A}}}, \ldots, \mathbf{r}'_{i+(wt^{\mathcal{A}}-1)wt^{\mathcal{A}}}$ on each path w_i $(1 \leq i \leq n)$, and also receives $wt^{\mathcal{A}}n$ h-vectors $\mathbf{x}_1, \ldots, \mathbf{x}_{wt^{\mathcal{A}}n}$ from W. For each $1 \leq i \leq wt^{\mathcal{A}}n$, let $\mathbf{x}_i = (x_{i1}, \ldots, x_{ih})$.
2. For each $1 \leq i \leq wt^{\mathcal{A}}n$, S uses the pseudo-basis construction scheme to construct a pseudo-basis B from $\mathbf{x}_1, \ldots, \mathbf{x}_{wt^{\mathcal{A}}n}$. Let b be the pseudo-dimension of B, then $b \leq wt^{\mathcal{A}}$.
3. For each $1 \leq i \leq wt^{\mathcal{A}}n$, S encodes $\mathbf{r}'_i$ to get codeword $\mathbf{c}'_i = EC(\mathbf{r}'_i) = (c'_{i1}, \ldots, c'_{ih})$. S then constructs a set D_i such that for each $1 \leq j \leq h$, iff $x_{ij} \neq c'_{ij}$, then $(c'_{ij}, j) \in D_i$.
4. For each $1 \leq i \leq wt^{\mathcal{A}}n$, S decodes $r'_i = DC(\mathbf{r}'_i)$. S then constructs a set T such that iff $|D_i| \leq wt^{\mathcal{A}}$, then $r'_i \in T$. S uses the randomness extractor to get $(z_1, \ldots, z_\ell) = RE(T)$, and for each $1 \leq i \leq \ell$, S computes $\sigma_i = s_i + z_i$.
5. S broadcasts the pseudo-basis B and $\sigma_1, \ldots, \sigma_\ell$. For each $1 \leq i \leq wt^{\mathcal{A}}n$, if $|D_i| > wt^{\mathcal{A}}$, then S broadcasts "ignore i"; else, then S broadcasts D_i.

Recovery Phase

1. R finds the final error locator F from B.
2. For each D_i that R receives on W, R constructs an h-vector $\mathbf{c}''_i = (c''_{i1}, \ldots, c''_{ih})$ such that for each $1 \leq j \leq h$, if $(c'_{ij}, j) \in D_i$, then $c''_{ij} = c'_{ij}$; else, then $c''_{ij} = c_{ij}$. R then decodes the information r''_i of $\mathbf{c}''_i$ such that for any $j \in F$, c''_{ij} is not used for decoding. R puts r''_i in a set T'.
3. R uses the randomness extractor to get $(z'_1, \ldots, z'_\ell) = RE(T')$, and for each $1 \leq i \leq \ell$, R computes $s'_i = \sigma_i - z'_i$. **End.**

Proof of perfect security. Omitted. See the full version of this paper [1].

TC of the protocol. In this protocol:

$$TC(1) = (k + h)wt^{\mathcal{A}}n = O(h\ell)$$
$$TC(2) = O(n(wt^{\mathcal{A}}h + \ell + wt^{\mathcal{A}}n \cdot 2h)) = O(h^2n^2) = O(hn\ell)$$

We have that the total TC is $O(hn\ell)$ field elements.

5 PSMT in Directed Graphs

In this section we show our PSMT protocols in directed graphs. We let $W = \{w_1, \ldots, w_n\}$ be the critical set of forward paths and $Q = \{q_1, \ldots, q_u\}$ be the critical set of feedback paths.

In a directed graph without feedback ($Q = \emptyset$), S only needs to send a codeword $\mathbf{c}$, of which the information is the message s, to R via W with respect to ψ. Due to N&S-directed-1, S and R are $3\mathcal{A}$-connected, R can decode the information of $\mathbf{c}$ by correcting errors. Thus the protocol is perfectly secure and the TC is $O(h)$. We remark that Desmedt et al.'s protocol [8] is actually an alternative use of the worst case LSSS.

Next we consider a directed graph with feedback ($Q \neq \emptyset$). We give our 3-round protocols under the condition of N&S-directed-2 in Section 5.1. In Section 5.2, we show that N&S-directed-2 is not sufficient for 2-round PSMT protocols, and hence we give a new N&S condition and propose our protocols under this condition. The protocols given in this section are along the lines of the results in [18,17].

5.1 3-Round Directed Protocols

Before we show our 3-round protocols, we notice that the adversary structure $\mathcal{A}$ is over all paths in $W \cup Q$. However, in our 3-round protocols, we do not need to assign shares (or entries) to the paths in Q. Thus we denote an adversary structure $\mathcal{A}'$ over the paths in W only, i.e., for any set $A \in \mathcal{A}$, there is a corresponding set $A' \in \mathcal{A}'$ such that $A' = A \cap W$. Thus S and R are $2\mathcal{A}'$-connected with the paths in W. Note that in this section, the linear codes in our protocols are constructed with respect to $\mathcal{A}'$.

3-round directed protocol for a single message s

Round 1 - S to R:

1. S chooses $wt^{\mathcal{A}}(u+1)+1$ random k-vectors $\mathbf{r}_1, \ldots, \mathbf{r}_{wt^{\mathcal{A}}(u+1)+1} \in \mathbb{F}^k$, and for each $1 \leq i \leq wt^{\mathcal{A}}(u+1)+1$, S encodes $\mathbf{r}_i$ to get codeword $\mathbf{c}_i = EC(\mathbf{r}_i) = (c_{i1}, \ldots, c_{ih})$.
2. For each $1 \leq i \leq wt^{\mathcal{A}}(u+1)+1$, S sends $\mathbf{c}_i$ via W with respect to ψ.

Round 2 - R to S:

1. R receives $wt^{\mathcal{A}}(u+1)+1$ h-vectors $\mathbf{x}_1, \ldots, \mathbf{x}_{wt^{\mathcal{A}}(u+1)+1}$ from W. R uses the pseudo-basis construction scheme (see Section 2.3) to construct a pseudo-basis B from $\mathbf{x}_1, \ldots, \mathbf{x}_{wt^{\mathcal{A}}(u+1)+1}$, and then broadcasts B via all paths $q_1, \ldots, q_u \in Q$.

Round 3 - S to R:

1. For each $1 \leq v \leq u$, let B_v be the pseudo-basis that S receives on path q_v, and let b_v be the pseudo-dimension of B_v.
2. For each $1 \leq v \leq u$, if $b_v > wt^{\mathcal{A}}$, then S broadcasts "ignore v" via W; else then S finds the final error locator F_v from B_v. If $|F_v| > wt^{\mathcal{A}}$, then S broadcasts "ignore v" via W; else then S broadcasts B_v and F_v via W.
3. S sets $U := \emptyset$ and $T := \emptyset$. For each $1 \leq v \leq u$ such that $b_v \leq wt^{\mathcal{A}}$, S adds all the actual codewords ($\mathbf{c}_i$'s) that correspond to the h-vectors in B_v to U. Thus at last, $|U| \leq wt^{\mathcal{A}}u$. For each $\mathbf{r}_i$ such that $EC(\mathbf{r}_i) = \mathbf{c}_i \notin U$, if $i \notin T$ and $|T| < wt^{\mathcal{A}} + 1$, then S sets $T := T \cup \{i\}$. Thus at last, $|T| = wt^{\mathcal{A}} + 1$. For each $i \in T$, S decodes $r_i = DC(\mathbf{r}_i)$. S computes $\sigma = s + \sum_{i \in T} r_i$, and broadcasts σ and T via W.

Recovery Phase

Let $v := 1$, while $v \leq u$:

1. if R receives "ignore v" from W, then R sets $v := v + 1$;
2. else if R receives B_v and F_v from W, then
   (a) if $B_v \neq B$, then R sets $v := v + 1$;
   (b) else, then with F_v, σ and T, R uses the decoding scheme from pseudo-basis (see Section 2.3) to get the information r_i of $\mathbf{c}_i$ for each $i \in T$. R then recovers $s = \sigma - \sum_{i \in T} r_i$, and terminates the protocol.

If $v > u$, then R knows that S did not receive the correct pseudo-basis B, so all paths $q_1, \ldots, q_u \in Q$ are corrupted. For each $i \in T$, R finds a set $A \in \mathcal{A}$ such that $Q \subseteq A$, and if A's entries in $\mathbf{x}_i$ are removed, all the remaining entries are a part of a codeword $\mathbf{c}'_i \in C$, then R decodes r'_i as the information of $\mathbf{c}'_i$. R recovers $s' = \sigma - \sum_{i \in T} r'_i$. **End.**

Proof of perfect security. Omitted. See the full version of this paper [1].

TC of the protocol. In this protocol:

$$
\begin{aligned}
TC(1) &= h(wt^{\mathcal{A}}(u+1)+1) = O(h^2 n) \\
TC(2) &= O(u(wt^{\mathcal{A}} h)) = O(h^2 n) \\
TC(3) &= O(n(wt^{\mathcal{A}} hu + wt^{\mathcal{A}} u + 1 + wt^{\mathcal{A}} + 1)) = O(h^2 n^2)
\end{aligned}
$$

We have that the total TC is $O(h^2 n^2)$ field elements.

Our 3-round protocol that transmits multiple messages is a generalization of the above protocol for a single message transmission. Thus we only show their differences as follows.

3-round directed protocol for $\ell = wt^{\mathcal{A}} u$ message $s_1, \ldots, s_\ell$

Round 1 - S to R: S does the same only for $wt^{\mathcal{A}}(u+1)+\ell$ random k-vectors.
Round 2 - R to S: R does the same.
Round 3 - S to R: S does the same until step 3.

3. S sets $U := \emptyset$. For each $1 \leq v \leq u$ such that $b_v \leq wt^{\mathcal{A}}$, S adds all the actual codewords ($\mathbf{c}_i$'s) that correspond to the h-vectors in B_v to U. Thus at last, $|U| \leq wt^{\mathcal{A}} u$.
4. S sets $T_1, \ldots, T_\ell := \emptyset$. For each $\mathbf{r}_i$ such that $EC(\mathbf{r}_i) = \mathbf{c}_i \notin U$, for each $1 \leq j \leq \ell$, if $i \notin T_j$ and $|T_j| < wt^{\mathcal{A}}$, then S sets $T_j := T_j \cup \{i\}$. Thus at last, all $T_1, \ldots, T_\ell$ are *the same* and $|T_j| = wt^{\mathcal{A}}$. There are at least ℓ vectors $\mathbf{r}_i$ such that $EC(\mathbf{r}_i) = \mathbf{c}_i \notin U$ and $i \notin T_j$ [4]. Let $\mathbf{r}_{i_1}, \ldots, \mathbf{r}_{i_\ell}$ be ℓ such vectors, then for each $1 \leq j \leq \ell$, S sets $T_j := T_j \cup \{i_j\}$. Thus $|T_j| = wt^{\mathcal{A}} + 1$, and all $T_1, \ldots, T_\ell$ are *different*. For each $1 \leq j \leq \ell$ and $i \in T_j$, S decodes $r_i = DC(\mathbf{r}_i)$, computes $\sigma_j = s_j + \sum_{i \in T_j} r_i$, and broadcasts σ_j and T_j via W.

Recovery Phase For each $1 \leq j \leq \ell$, R does the same to recover s_j. **End.**

[4] This is because $|U| \leq wt^{\mathcal{A}} u$, $|T_j| = wt^{\mathcal{A}}$ and the total number of vectors $\mathbf{r}_i$ is $wt^{\mathcal{A}}(u+1)+\ell$.

Proof of perfect security. Omitted. See the full version of this paper [1].

TC of the protocol. In this protocol:

$$TC(1) = h(wt^{\mathcal{A}}(u+1) + wt^{\mathcal{A}}u) = O(h\ell)$$
$$TC(2) = O(u(wt^{\mathcal{A}}h)) = O(h\ell)$$
$$TC(3) = O(n(wt^{\mathcal{A}}hu + wt^{\mathcal{A}}u + wt^{\mathcal{A}}u(1 + wt^{\mathcal{A}} + 1))) = O(hn\ell)$$

We have that the total TC is $O(hn\ell)$ field elements.

5.2 2-Round Directed Protocols

In [18], Patra et al. showed that in the threshold model, the minimal connectivity for PSMT in directed graph is not sufficient for a 2-round protocol. Here we do the similar. That is, we prove that in the general adversary model, N&S-directed-2 is not sufficient for a 2-round protocol. Note that the general assumption is that the feedback channels are not reliable (i.e., not $2\mathcal{A}$-connected).

Theorem 3. *Given a directed graph $G(V, E)$ and an adversary structure $\mathcal{A}$, 2-round PSMT is possible if and only if S and R are $2\mathcal{A}$-connected with the forward paths and $3\mathcal{A}$-connected in G.*

Proof. First we prove the necessity of the condition. $2\mathcal{A}$-connectivity with the forward paths is obviously necessary. Now assume that S and R are $3\mathcal{A}$-separated in G and there is a 2-round PSMT protocol Π. Let $view^S$ and $view^R$ be the views of S and R respectively. In Round 1 of Π, $view^S$ and $view^R$ can be different if the adversary corrupts some feedback paths. Since the feedback paths are not reliable, S cannot detect the differences. Thus after Round 2, because Π is perfectly private, with respect to $\mathcal{A}$, we regard $view^S$ as a codeword whose information is the message. Thus $view^R$ is $view^S$ plus an error vector caused by a set $A \in \mathcal{A}$. Since S and R are $3\mathcal{A}$-separated, R cannot correct the errors and decode the message. Thus Π is not perfectly reliable. We have a contradiction.

Next we show a 2-round PSMT protocol under this condition. We let $\mathcal{A}' = \mathcal{A} \cup \{Q\}$ (if $Q \in \mathcal{A}$, then $\mathcal{A}' = \mathcal{A}$). Since S and R are $2\mathcal{A}$-connected with the forward paths, they are $3\mathcal{A}'$-connected in G. The linear code in this protocol is constructed with respect to $\mathcal{A}'$.

2-round directed protocol for a single message s

Round 1 - R to S: R chooses a random k-vector $\mathbf{r}$, and encodes it to get the codeword $\mathbf{c} = EC(\mathbf{r}) = (c_1, \ldots, c_h)$. Suppose that $c_1, \ldots, c_t$ are the entries in $\mathbf{c}$ such that $\psi(c_1, \ldots, c_t) = Q$, the linear code allows all these entries to be independent[5]. R then sends the entries $c_1, \ldots, c_t$ via Q with respect to ψ.

Round 2 - S to R: Upon the entries $c'_1, \ldots, c'_t$ that S receives on Q, S constructs a k-vector $\mathbf{r}'$ such that $c'_1, \ldots, c'_t$ are in the codeword $\mathbf{c}' = EC(\mathbf{r}') = (c'_1, \ldots, c'_h)$. S decodes $r' = DC(\mathbf{r}')$. S then sends $c'_{t+1}, \ldots, c'_h$ via W with respect to ψ and broadcasts $\sigma = s + r'$.

[5] This is possible. See the full version of this paper [1] for more details.

Recovery Phase R receives $c''_{t+1}, \ldots, c''_h$ and σ on W. R constructs an h-vector $\mathbf{x} = (c_1, \ldots, c_t, c''_{t+1}, \ldots, c''_h)$. Thus $\mathbf{x} = \mathbf{c}' + \mathbf{e}$ where $\mathbf{e}$ is an error vector caused by a set $A \in \mathcal{A}'$. Due to the $3\mathcal{A}'$-connectivity, R can decode the information r' of $\mathbf{c}'$ from $\mathbf{x}$ and recover $s = \sigma - r'$. **End.**

Proof of perfect security is omitted. See the full version of this paper [1].

Clearly the TC of this protocol is $O(h)$, and the protocol can transmit ℓ messages with a TC of $O(h\ell)$. □

6 Conclusion and Open Problems

In this paper, we regarded general access structures as a special linear code and exploited its properties to design PSMT protocols in the general adversary model. The construction of our protocols is based on the idea of defining adversary structure over critical paths. We are the first to study interactive PSMT with a constant round complexity. Moreover, the transmission complexity of our protocols is similar to the best protocols that use non-constant rounds, which is quite unexpected. Also our study on PSMT over multiple messages is new in this context.

Evidently, there are still many unknown properties of the linear codes we proposed. The most obvious one is the tight upper bound on h, which is open for decades. Another interesting problem is whether in the presence of non zero invalid error-vectors, it is possible to have a pseudo-dimension that is smaller than $O(h)$.

The TC of our 2-round undirected and 3-round directed protocols for multiple message transmission is $O(hn\ell)$. In [22,2,15], the authors used a technique called *generalized broadcast* to reduce the TC by $O(n)$. We wonder if generalized broadcast can further reduce the TC of our protocols to $O(h\ell)$.

Acknowledgment. We would like to thank the anonymous referees for their helpful comments on the earlier version of the paper.

References

1. The full version of this paper will be available on the authors' web pages
2. Agarwal, S., Cramer, R., de Hann, R.: Asymptotically optimal two-round perfectly secure message transmission. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 394–408. Springer, Heidelberg (2006)
3. Benaloh, J., Leichter, J.: Generalized secret sharing and monotone functions. In: Goldwasser, S. (ed.) CRYPTO 1988. LNCS, vol. 403, pp. 27–35. Springer, Heidelberg (1990)
4. Blundo, C., De Santis, A., De Simone, R., Vaccaro, U.: Tight bounds on the information rate of secret sharing schemes. Des. Codes Cryptography 11(2), 107–122 (1997)
5. Cramer, R., Damgård, I., Maurer, U.M.: General secure multi-party computation from any linear secret-sharing scheme. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 316–334. Springer, Heidelberg (2000)

6. Csirmaz, L.: The size of a share must be large. J. Cryptography 10(4), 223–231 (1997); A Preliminary version published in 1995.
7. Desmedt, Y., Wang, Y.: Perfectly secure message transmission revisited. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 502–517. Springer, Heidelberg (2002)
8. Desmedt, Y., Wang, Y., Burmester, M.: A complete characterization of tolerable adversary structures for secure point-to-point transmissions without feedback. In: Deng, X., Du, D.-Z. (eds.) ISAAC 2005. LNCS, vol. 3827, pp. 277–287. Springer, Heidelberg (2005)
9. Dolev, D., Dwork, C., Waarts, O., Yung, M.: Perfectly secure message transmission. J. ACM 40(1), 17–47 (1993)
10. Frankel, Y., Desmedt, Y.: Classification of ideal homomorphic threshold schemes over finite Abelian groups. In: Rueppel, R.A. (ed.) EUROCRYPT 1992. LNCS, vol. 658, pp. 25–34. Springer, Heidelberg (1993)
11. Hirt, M., Maurer, U.M.: Player simulation and general adversary structures in perfect multiparty computation. J. Cryptology 13(1), 31–60 (2000)
12. Ito, M., Saito, A., Nishizeki, T.: Secret sharing schemes realizing general access structure. In: Proc. IEEE Globecom 1987, pp. 99–102 (1987)
13. Karchmer, M., Wigderson, A.: On span programs. In: Proc. IEEE Structure in Complexity Theory, pp. 102–111 (1993)
14. Kumar, M., Goundan, P., Srinathan, K., Rangan, C.P.: On perfectly secure communication over arbitrary networks. In: Proc. ACM PODC 2002, pp. 293–202 (2002)
15. Kurosawa, K., Suzuki, K.: Truly efficient 2-round perfectly secure message transmission scheme. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 324–340. Springer, Heidelberg (2008); Also available in IEEE Transaction on Information Theory, 55(11)5223–5232 (2009)
16. MacWilliams, F.J., Sloane, N.J.A.: The theory of error-correcting codes. North-Holland Publishing Company, Amsterdam (1978)
17. Patra, A., Choudhary, A., Rangan, C.P.: On communication complexity of secure message transmission in directed networks. In: Proc. ICDCN 2010. LNCS, vol. 5935, pp. 42–53 (2010)
18. Patra, A., Cloudhary, A., Rangan, C.P.: Brief announcement: perfectly secure message transmission in directed networks re-visited. In: Proc. ACM PODC 2009, pp. 278–279 (2009)
19. Patra, A., Shankar, B., Choudhary, A., Srinathan, K., Rangan, C.P.: Perfectly secure message transmission in directed networks tolerating threshold and non threshold adversary. In: Bao, F., Ling, S., Okamoto, T., Wang, H., Xing, C. (eds.) CANS 2007. LNCS, vol. 4856, pp. 80–101. Springer, Heidelberg (2007)
20. Shamir, A.: How to share a secret. ACM Commun. 22(11), 612–613 (1979)
21. Simmons, G.J., Jackson, W., Martin, K.: The geometry of shared secret schemes. Bulletin of the Institute of Combinatorics and its Applications 1(1), 71–88 (1991)
22. Srinathan, K., Narayanan, A., Rangan, C.P.: Optimal perfectly secure message transmission. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 545–561. Springer, Heidelberg (2004)
23. van Dijk, M.: On the information rate of perfect secret sharing schemes. Des. Codes Cryptography 6(2), 143–169 (1995)
24. Yang, Q., Desmedt, Y.: Cryptanalysis of secure message transmission protocols with feedback. In: Kurosawa, K. (ed.) Information Theoretic Security. LNCS, vol. 5973, pp. 159–176. Springer, Heidelberg (2010)

On Invertible Sampling and Adaptive Security

Yuval Ishai[1,⋆], Abishek Kumarasubramanian[2], Claudio Orlandi[3,⋆⋆], and Amit Sahai[2,⋆⋆⋆]

[1] Technion and UCLA
yuvali@cs.technion.ac.il
[2] UCLA
{abishekk,sahai}@cs.ucla.edu
[3] Aarhus University
claudio@cs.au.dk

Abstract. Secure multiparty computation (MPC) is one of the most general and well studied problems in cryptography. We focus on MPC protocols that are required to be secure even when the adversary can *adaptively* corrupt parties during the protocol, and under the assumption that honest parties cannot reliably erase their secrets prior to corruption.

Previous feasibility results for adaptively secure MPC in this setting applied either to deterministic functionalities or to randomized functionalities which satisfy a certain technical requirement. The question whether adaptive security is possible for *all* functionalities was left open.

We provide the first convincing evidence that the answer to this question is negative, namely that some (randomized) functionalities cannot be realized with adaptive security.

We obtain this result by studying the following related *invertible sampling* problem: given an efficient sampling algorithm A, obtain another sampling algorithm B such that the output of B is computationally indistinguishable from the output of A, but B can be efficiently inverted (even if A cannot). This invertible sampling problem is independently motivated by other cryptographic applications. We show, under strong but well studied assumptions, that there exist efficient sampling algorithms A for which invertible sampling as above is impossible. At the same time, we show that a general feasibility result for adaptively secure MPC implies that invertible sampling is possible for every A, thereby reaching a contradiction and establishing our main negative result.

1 Introduction

Secure multiparty computation (MPC) is one of the most fundamental problems in cryptography. The goal of MPC is to allow two or more parties to compute

⋆ Supported in part by ISF grant 1310/06, BSF grant 2008411, and NSF grants 0830803, 0716835, 0627781.

⋆⋆ This work was done while the author was visiting UCLA.

⋆⋆⋆ Supported in part by NSF grants 0916574, 0830803, 0716389, and 0627781, an equipment grant from Intel, and an Okawa Foundation Research Grant.

M. Abe (Ed.): ASIACRYPT 2010, LNCS 6477, pp. 466–482, 2010.

some functionality (a deterministic or randomized mapping from inputs to outputs) while emulating an ideal evaluation of the functionality in which a trusted trusted party receives all inputs and delivers all outputs. This is formally captured by simulation-based security definitions, which (roughly speaking) require that whatever an adversary can achieve by attacking the real execution of the protocol can also be achieved by a *simulator* which attacks the above ideal evaluation process.

Since the introduction of MPC in the 1980s [42,28,5,15], many security definitions have been proposed and feasibility results shown. In particular, significant research efforts have been invested in realizing *adaptively secure* MPC protocols, whose security is required to hold in the presence of an adversary that can corrupt parties *adaptively* at any point during the protocol. When considering adaptive security, it is typically assumed that honest parties cannot reliably erase their secrets. This is an assumption we make throughout this work. The main challenge in proving the security of cryptographic protocols in this setting is that when a new party is corrupted, the simulator needs to provide an explanation of the internal randomness for this party that has to be consistent with the simulated view so far and with the party's input.

Adaptively secure protocols in this setting were first constructed by Canetti, Feige, Goldreich and Naor [13] in a standalone model and then by Canetti, Lindell, Ostrovsky and Sahai [14] in the universal composability (UC) model [9]. These protocols applied to all deterministic functionalities, but in the case of randomized functionalities they were restricted to so called *adaptively well-formed* functionalities [14]. Intuitively, randomized functionalities can present the following problem: when the adversary corrupts *all* the parties in the real execution[1], he learns the private randomness of all parties. However in the ideal world, if the ideal functionality tosses some coins that are kept private and used during the computation, the ideal adversary (the simulator) will never learn these private coins, even after corrupting every party. The presence of private randomness in the ideal world makes it problematic to realize randomized functionalities in which the randomness cannot be efficiently computed from the inputs and outputs. The "adaptively well formed" functionalities satisfy the syntactic requirement that they reveal all their internal randomness when all parties are corrupted. (In other words, securely realizing such functionalities does not pose the challenge of hiding the internal randomness of the functionality from the adversary.) The question for general functionalities was left open.

In this paper we show that, under strong but well studied computational assumptions, there exist functionalities which *cannot* be realized with adaptive

[1] At first glance, it may seem strange to require any security when all parties involved in a protocol are eventually corrupted. However, this is important when protocols are meant to be composed (even sequentially). For instance, a sub-protocol of a larger protocol may involve only a small subset S of the participants of the larger protocol. In such a situation, guaranteeing security of the larger protocol when (only) the players in S are corrupted would require analyzing the security of the sub-protocol when *all* the participants of the sub-protocol are corrupted.

security. Concretely, our main negative result relies on the following two assumptions: (1) the existence of so-called *extractable one-way functions* [10,11,18] (this is a common generalization of several "knowledge-of-exponent" style assumptions from the literature [19,31,4,39]), and (2) the existence of non-interactive zero-knowledge (NIZK) proofs for NP [6,7].

Our negative result applies to almost every model of adaptively secure computation without erasures from the literature. This includes stand-alone security in the semi-honest and malicious models (under the definition of [8]), UC-security in the CRS model (under the definition of [14]) or even to security in the OT-hybrid model, where every functionality can be *unconditionally* realized with non-adaptive UC-security [35,34]. Our negative result does *not* apply to the case where only a strict subset of the parties can be corrupted (in particular, to MPC with an honest majority). The existence of uncorrupted parties allows the simulator to avoid the need for "explaining" the output of the functionality by providing its internal randomness. Our negative result also does not apply to adaptive security in the standalone model without post-execution corruption [8]; this (nonstandard) notion of adaptive security does not support even sequential composition. See Section 1.2 below.

Invertible sampling. A key concept which we use to obtain our negative result and is of independent interest is that of *invertible sampling* (Definition 1 of [20]). Suppose we are given an efficient sampling algorithm A. Can we always obtain an *alternative* efficient sampling algorithm B such that the output of B is indistinguishable from the output of A, but B can be efficiently inverted in the sense that its randomness can be efficiently computed based on its output? Here we refer to a distributional notion of inversion, namely an inversion algorithm B^{-1} is successful if the pair $(r', \mathsf{B}(r'))$ is computationally indistinguishable from the pair $(\mathsf{B}^{-1}(\mathsf{B}(r')), \mathsf{B}(r'))$ where r' is a uniform random input for B. We refer to the hypothesis that every efficient A admits an efficient B as above as the *invertible sampling hypothesis (ISH)*. While our study of ISH is primarily motivated by its relevance to adaptive security, this question is independently motivated by other cryptographic applications (such as settling the relation between public-key encryption and oblivious transfer); see Section 6 for details.

The ISH may seem easy to refute under standard assumptions. Indeed, if we require the outputs of A and B to be *identically* distributed, then ISH could be refuted based on the existence of any pseudorandom generator G: Let A output $G(r)$. The existence of B as above would allow one to distinguish between $G(r)$ (for which B^{-1} will find an inverse under B with overwhelming probability) and a uniformly random string of the same length (which with overwhelming probability has no inverse under B). However, the case where the outputs of B and A should only be *computationally* indistinguishable appears to be much more challenging. In particular, note that a pseudorandom distribution *does* admit an invertible alternative sampler: the sampler B just outputs a uniformly random string. Since this output is computationally indistinguishable from the actual distribution, it is consistent with the above formulation of ISH.

We show, under the assumptions described above, that there exist efficient sampling algorithms A for which the ISH fails. At the same time, we show that a general feasibility result for adaptively secure MPC implies that invertible sampling is possible for every A, thereby reaching a contradiction and establishing our main negative result.

More precisely, we show that general adaptively secure computation implies (and in fact, is equivalent to) a stronger version of ISH in which the sampling algorithms A, B are given an input x in addition to their random input, and where the inversion algorithm B^{-1} should be successful on *every* input x. This stronger flavor of ISH is ruled out by the assumptions mentioned above, namely the existence of extractable one-way functions and NIZK proof systems for NP. To rule out the weaker variant of ISH (with no input x) we need to use somewhat stronger assumptions: a non-standard (but still plausible) variant of an extractable one-way function, and the existence of non-interactive witness-indistinguishable (NIWI) protocols for NP without a common reference string [22,1,29,30].

1.1 Our Techniques

We now give some intuition on our construction of an efficient sampling algorithm A for which ISH does not hold. For this purpose, it is convenient to first describe a *relativized* world (defined via a randomized oracle) in which such A provably exists. As a first attempt, suppose that we have an oracle computing a random function $f : \{0,1\}^n \to \{0,1\}^{2n}$. Now, consider the efficient sampling algorithm A which outputs a random image of f, namely $\mathsf{A}(r) = f(r)$. (Note that A is efficient given oracle access to f.) Similarly to the previous PRG example, such an algorithm is not enough to refute the computational version of ISH: indeed, the alternative sampler B can simply output a uniformly random string of length $2n$. The high level idea for ruling out such an alternative sampler is to make the outputs of f efficiently verifiable. Formally, we add to f an additional oracle g which decides whether a given string $y \in \{0,1\}^{2n}$ is in the image of f. (A similar oracle was used by Wee [41] in the seemingly unrelated context of separating two notions of computational entropy.)

We now informally argue that ISH is false relative to the randomized oracle (f, g). Let $\mathsf{A}(r) = f(r)$ as before. Assume towards a contradiction that an alternative sampling algorithm $\mathsf{B}(r')$ as required by ISH exists. We argue that B can be used to efficiently invert f on a random output $y = f(x)$, which remains hard even when given the decision oracle g. By the computational indistinguishability requirement, it suffices (in order to reach a contradiction) to successfully invert f on a random output y' sampled by B. Moreover, since indistinguishability holds relative to the verification oracle g we are guaranteed that (with overwhelming probability) y' as above will be in the image of f.

The inversion algorithm for f, when given y' sampled by B, uses the inversion algorithm B^{-1} guaranteed by ISH to obtain a preimage r' of y' under B. Since f is a random function, it is impossible to efficiently find an image y' of f without querying f on the corresponding pre-image. (Jumping ahead, this is

the step where our explicit non-relativized construction will rely on a knowledge assumption.) Thus, the inversion algorithm can use r' to extract a preimage x of y' under f by running B on r' until it queries f on a point x such that $f(x) = y$.

To obtain an explicit version of the above A we use extractable one-way functions to implement f and a NIZK proof for proving range membership to emulate the role of g (the latter is similar to the use of NIZK proofs in [37,32]; see Section 1.2). For technical reasons that have to do with the common reference string required by the NIZK proof system, we cannot use this approach to refute the basic version of ISH described above. For this, we need to employ a somewhat more complicated proof strategy and apply NIWI proofs instead of NIZK proofs. See Section 4 for details.

1.2 Related Work

Adaptively secure MPC (without erasures) was first realized in [13] for the stand-alone case. In [8], a variant of the notion of adaptive security that guarantees sequential composition was introduced: we refer to the variant from [8] which requires security against *post execution corruption (PEC)*. Namely, after the simulation is complete, the environment can ask the adversary to corrupt additional parties and simulate their views. This variant is used in [8] to prove sequential composition. In fact, a separation between adaptive security with PEC and without it has been shown in [12]. We stress that the negative results from [12] apply to specific *protocols* rather than functionalities. That is, [12] builds protocols which are shown to be adaptively secure in one setting but not adaptively secure in another setting, but does not show any *functionality* which cannot be realized with adaptive security, as opposed to our impossibility result.

In the UC security framework [9] the main feasibility result for securely realizing adaptively well-formed functionalities against an adaptive adversary was obtained in [14] (see also [17,25]). This work also suggested the following plausible candidate for a randomized functionality which *cannot* be realized with adaptive security: on input a security parameter k, output the product of two random k-bit primes. However, we do not know how to relate the possibility of realizing this functionality with adaptive security to any well-studied assumption.

If one is willing to assume that honest parties can reliably erase their data, security against adaptive adversaries becomes a much easier task. Our negative results do not apply to this alternative model, and general feasibility results in this model were obtained in [3,36].

The Invertible Sampling Hypothesis is related to questions of oblivious sampling that have been studied in other cryptographic contexts. For instance, the question of generating a public key for an encryption scheme without learning how to decrypt is related to the goal of constructing an oblivious transfer protocol from a public-key encryption scheme [23,26]; virtually any non-committing encryption scheme [2,20,25,16] requires some form of oblivious sampling of public keys; in a recent result [21] the question of whether ISH holds has been informally asked, in the context of turning UC-secure protocols in the common reference string model into semi-honest secure stand-alone protocols. If the common

reference string is a random string, the problem trivially reduces to having one party publish a random string. If the CRS instead is sampled using some generic distribution, is not clear whether a semi-honest party can sample the common reference string without learning the trapdoor. See Section 6 for a further discussion of these additional connections of ISH with cryptography.

The *knowledge of exponent assumption* was introduced in [19], and since then other specific knowledge of exponent assumptions have been proposed [4,39], until in some recent work [11,10,18] the abstract notion of *extractable functions* has been introduced. Our impossibility results rely on assumptions of this type. The use of knowledge assumptions in security proofs has received criticism in the cryptographic community, especially because such assumptions seem hard to disprove [38] (even though in [4] a "wrong" knowledge assumption from [31] has been disproved). As far as we know, our work is the first to apply such assumptions towards *negative* results in cryptography.

Finally, our use of NIZK and NIWI proofs for NP was inspired by the use of NIZK in [37] to construct a class of distributions where efficient learning with an evaluator is possible but coming up with a generator that approximates the given distribution is infeasible, and by [41,32] in the context of separating conditional HILL and Yao entropies. Note, however, that none of these works made use of knowledge assumptions; such assumptions appear to be crucial to our techniques.

2 Preliminaries

Notation. We use n as a length parameter; all probability distributions we consider in this work will be over strings of length polynomial in n. We let U_n denote the uniform distribution over $\{0,1\}^n$. We use $x \leftarrow X$ to denote the process of sampling x from the distribution X. If X is a set, $x \leftarrow X$ denotes a uniform choice of x from X. For any distribution X and algorithm A, we denote by $\mathsf{A}(X)$ the probability distribution on the outputs of A taken over the coin tosses (if any) of A and an independent random choice of the input x from X.

We use the standard notation $\{C_1; C_2; \ldots; C_m : D\}$ to denote the distribution of D obtained as a result of the sampling process defined by the sequence of instructions $C_1, \ldots, C_m$. For example, $\{a \leftarrow X; b \leftarrow \mathsf{A}(a) : (a,b)\}$ denotes the distribution of pairs (a,b) obtained by first picking a from X and then obtaining b by running A on a. Similarly, we use $\Pr[C_1; C_2; \ldots; C_m : E]$ to denote the probability of event E in the probability space defined by the sequence of instructions $C_1, \ldots, C_m$. For instance, $\Pr[a \leftarrow X; b \leftarrow Y : a \neq b]$ is the probability that when a is chosen according to X and b is independently chosen according to Y, a and b are not equal.

We assume that the reader is familiar with the concepts of *negligible function*, *one-way function*, *pseudorandom generator*, and *non-interactive zero-knowledge proof system*. Suitable definitions can be found in the full version or in [27].

By default we assume efficient algorithms to be uniform and efficient distinguishers to be nonuniform. We will use $\epsilon(\cdot)$ to denote an unspecified negligible function.

Let $I \subseteq \{0,1\}^*$ be an arbitrary infinite index set. We say that two distribution ensembles $\{X_w\}_{w\in I}$ and $\{Y_w\}_{w\in I}$ are *computationally indistinguishable* if for every polynomial-size circuit family C_n there exists a negligible function ϵ such that for every $w \in I$,

$$|\Pr[C_{|w|}(X_w) = 1] - \Pr[C_{|w|}(Y_w) = 1]| \leq \epsilon(|w|).$$

Sampling algorithms. We will view any probabilistic polynomial time (PPT) algorithm A as defining an *efficient sampling algorithm* (or *sampler* for short). We let $\mathsf{A}(w)$ denote the output distribution of A on input w and $\mathsf{A}(w; r_\mathsf{A})$ denote the output when the random input (i.e., sequence of coin-tosses) is given by r_A. Without loss of generality, we can associate with every efficient A a polynomial $\ell(\cdot)$ such that r_A is a random input of length $\ell(|w|)$. Under this convention, $\mathsf{A}(w)$ is distributed identically to $\mathsf{A}(w; U_{\ell(|w|)})$. We will use this convention throughout the paper. Finally, we will sometimes be interested in the special case of samplers over a unary input alphabet; in this case A defines a sequence of distributions $\{\mathsf{A}(1^n)\}_{n\in\mathbb{N}}$.

We say that a sampling algorithm A is *inverse-samplable* if there exists a PPT inversion algorithm which, given an input w and a sample y from the output $\mathsf{A}(w)$, outputs a random input r for A which is consistent with w, y. Moreover, the choice of r should be "correctly distributed" in the sense that (w, y, r) should be computationally indistinguishable from $(w, \mathsf{A}(r_\mathsf{A}), r_\mathsf{A})$ where $r_\mathsf{A} \leftarrow U_{\ell(|w|)}$. (Such a distributional inversion requirement is similar in spirit to the definition of a distributionally one-way function [33].)

Definition 1 (Inverse-Samplable Algorithm). *We say that an efficient sampling algorithm* A *is* inverse-samplable *if there exists a PPT* inverter *algorithm* A^{-1} *such that the distribution ensembles* $\{r_\mathsf{A} \leftarrow U_{\ell(|w|)} : (r_\mathsf{A}, \mathsf{A}(w; r_\mathsf{A}))\}_{w\in\{0,1\}^*}$ *and* $\{r_\mathsf{A} \leftarrow U_{\ell(|w|)} : (\mathsf{A}^{-1}(w, \mathsf{A}(w; r_\mathsf{A})), \mathsf{A}(w; r_\mathsf{A}))\}_{w\in\{0,1\}^*}$ *are computationally indistinguishable.*

3 Invertible Sampling Hypothesis

The Invertible Sampling Hypothesis (ISH) is concerned with the possibility of inverse-sampling arbitrary efficiently samplable distributions. It is easy to see that if one-way functions exist, then there are efficient sampling algorithms which are not inverse-samplable. Thus, we settle for the hope that for every efficient sampling algorithm A there exists an efficient and inverse-samplable algorithm B whose output is computationally indistinguishable from that of A. The ISH captures the above hope. We will also consider a weaker variant of ISH, referred to as *weak ISH*, which restricts the sampler A to have a unary input alphabet. This is formalized below.

Hypothesis 1 (Invertible Sampling Hypothesis: ISH). *For every efficient sampling algorithm* A *there exists an efficient sampling algorithm* B *satisfying the following two requirements.*

1. Closeness: *The distribution ensembles* $\{\mathsf{A}(w)\}_{w\in\{0,1\}^*}$ *and* $\{\mathsf{B}(w)\}_{w\in\{0,1\}^*}$ *are computationally indistinguishable.*
2. Invertibility: B *is inverse-samplable (see Definition 1).*

Hypothesis 2 (Weak ISH). *The* weak ISH *is defined exactly as ISH above, except that the inputs* w *for* A *and* B *are restricted to be unary (i.e., are of the form* 1^n*).*

Clearly, ISH implies Weak ISH. The weaker flavor of ISH is somewhat more natural in that it refers to the traditional notion of a sampling algorithm (defining a single probability distribution for each length parameter n) as opposed to the more general notion of a probabilistic algorithm. Moreover, the weak ISH suffices for the motivating applications discussed in Section 6. However, it turns out that the stronger flavor can be refuted under more standard assumptions and that ruling out this flavor suffices for obtaining our main negative result on adaptively secure MPC. Thus, in the following we will consider both variants of ISH.

We will start (in Section 4) by refuting the weak ISH assuming the existence of a strong variant of extractable one-way functions as well as NIWI proof systems for NP. We will then (Section 5) refute the original and stronger variant of ISH under the weaker assumptions that standard extractable one-way functions (generalizing various "knowledge-of-exponent assumptions" from the literature) exist, as well as NIZK protocols for NP in the CRS model. At a high level, refuting the stronger flavor of ISH is easier because the additional "external" input allows us to introduce randomness over which the alternative sampler B has no control. This randomness can be used for choosing the CRS for a NIZK proof or random parameters for a family of extractable one-way functions.

4 Conditional Refutation of Weak ISH

As already discussed in the introduction, any pseudorandom generator $G : \{0,1\}^n \to \{0,1\}^{2n}$ provides a nontrivial example of a sampling algorithm for which weak ISH holds. Indeed, if $\mathsf{A}(1^n)$ outputs $G(r_\mathsf{A})$ where $r_\mathsf{A} \leftarrow U_n$, then $\mathsf{B}(1^n)$ can simply output r_B where $r_\mathsf{B} \leftarrow U_{2n}$.

This example suggests that in order to provide a counterexample for the (weak) ISH, it does not suffice for the computation performed by the sampler to be *one-way* and for its output support to be *sparse*, but its output should also be *verifiable* (a feature missing in the aforementioned example). Jumping ahead, verifiability will be achieved via variants of non-interactive zero-knowledge. It turns out that even the "sparseness" requirement needs to be significantly strengthened in order to rule out the possibility of directly sampling an output without knowing a corresponding input. Classes of sparse one-way functions with a similar property were studied in [19,4,39] under the umbrella of "knowledge assumptions." Crudely speaking, a knowledge assumption for a function f states that if any efficient algorithm A outputs a point in $\mathsf{image}(f)$, then the only way A could have computed this image is by choosing an x and computing $f(x)$ (here it is necessary that $\mathsf{image}(f)$ be sparse). Thus the algorithm "knows" x. This is

formally captured by requiring the existence of an efficient algorithm that can extract x from A's input and randomness.

A brief outline of our refutation of weak ISH is as follows. Suppose a function f is both "extractable" and one-way. Given an algorithm which produces valid points in $\mathsf{image}(f)$, if we can obtain the randomness that it used, then we can use f's "knowledge extractor" to find pre-images and thus break the one-wayness of f. However to obtain this randomness, we need the algorithm to be inverse-samplable. Since weak ISH hypothesizes the existence of such an algorithm we can invert f and contradict its one-wayness.

Next, we formally prove that weak ISH is false assuming the existence of a strong notion of an Extractable One-Way Function (EOWF) and the assumption that Non-Interactive Witness Indistinguishable Proofs (NIWI) exist for all of NP.

We start by defining the two primitives we rely on. An extractable one-way function is a one-way function f with the following extraction property: for any efficient A which, on random input r_A, attempts to output an element y in the image of f, there is an efficient extractor K_A which given the random input r_A of A succeeds in finding a preimage $x \in f^{-1}(y)$ with roughly the same success probability. Formally:

Definition 2 (Extractable One-Way Function (EOWF)). *Let f be a one-way function. We say that f is an* extractable one-way function *if for every PPT algorithm A with running time $\ell(n)$ there is a PPT extractor algorithm K_A such that for every n:*

$$Pr[r_A \leftarrow U_{\ell(n)}; y = A(1^n; r_A); x \leftarrow K_A(1^n, r_A) : \\ (f(x) = y) \vee (\forall x', f(x') \neq y)] \geq 1 - \epsilon(n)$$

for some negligible function ϵ.

We note that the above definition appears stronger than similar definitions from the literature in that it requires f to be a single, explicit one-way function, as opposed to a keyed collection of functions. In particular, EOWF as above *can not* be instantiated using concrete knowledge assumptions from the literature such as the ones in [19,4,39]. However, it still seems plausible that (length-flexible versions of) practical cryptographic functions satisfy the above definition. In Section 5 we will rely on a more standard notion of EOWF (which allows f to depend on a random key and captures previous assumptions from the literature) in order to refute the strong variant of ISH.

Next we need the notion of *non-interactive witness indistinguishable (NIWI)* proof systems [1,29,30]. A NIWI proof is used to efficiently prove that an input x is in some NP-language L without allowing the verifier to distinguish between any two possible witnesses. While the latter witness indistinguishability property is weaker than the zero-knowledge property of NIZKs, it turns out that it is sufficient for our purposes. The important advantage of NIWI proofs is that they can be implemented (under stronger assumptions) without a trusted common reference string, which is inherently required for NIZK proofs.

Definition 3 (Non-Interactive Witness Indistinguishable Proof System [22,1,29]). *Let L be any NP language, and R_L a fixed witness relation for L. Then $\mathcal{P} = (P, V)$ is called a* non-interactive witness indistinguishable (NIWI) proof system *for R_L if P and V are PPT algorithms and the following conditions hold for some negligible function ϵ:*

1. Completeness. *For all $(x, w) \in R_L$*

$$\Pr[\pi \leftarrow P(x, w); b \leftarrow V(x, \pi) : b = 1] \geq 1 - \epsilon(|x|).$$

2. Soundness. *For all $x \notin L$, for all proof strings π^**

$$\Pr[V(x, \pi^*) = 1] \leq \epsilon(|x|).$$

3. Witness Indistinguishability (WI). *For every polynomial-size circuit family C_n, and every x, w_0, w_1 such that $(x, w_0) \in R_L$ and $(x, w_1) \in R_L$,*

$$|\Pr[C_{|x|}(P(x, w_0)) = 1] - \Pr[C_{|x|}(P(x, w_1)) = 1]| \leq \epsilon(|x|).$$

NIWI proofs exist for all of NP under well-studied assumptions [22,1,29].

We now use the above two primitives to establish the main result of this section.

Theorem 1. *If EOWF exists and NIWI proofs exist for NP, then Weak ISH is false.*

Proof (sketch): Let f be an EOWF. We first define an efficient sampling algorithm A, which outputs two random points in $\mathsf{image}(f)$ and also a NIWI proof that at least one of the points was correctly computed. That is, the sampling algorithm picks random $x_0, x_1 \leftarrow \{0,1\}^n$ and outputs $(f(x_0), f(x_1), \pi)$, where π is a NIWI proof that *either* $f(x_0)$ or $f(x_1)$ is in the image of f. More concretely, π is obtained by running a NIWI prover for the NP relation defined by

$$R_L((y_0, y_1), w) = 1 \text{ iff } f(w) = y_0 \vee f(w) = y_1$$

on input $(f(x_0), f(x_1))$ and witness x_0. From Weak ISH, we obtain A's invertible alternate sampling algorithm B and its inverter B^{-1}. By the soundness property of the NIWI proof, we are (essentially) ensured that the alternate sampler B outputs at least one valid point in the image of f. But then we can construct a new algorithm X that runs B and outputs at random one of the two images y_b.

Now X is an algorithm that outputs (with significant probability) valid points in the image of f. Given that f is an EOWF, there must exist also an extractor K_{X} that given the random input of X outputs x_b such that $y_b = f(x_b)$. Using B^{-1} to inverse-sample the random input of X and feeding it to K_{X} we can efficiently invert f, contradicting its one-wayness. See the full version for more details. □

5 Conditional Refutation of ISH

In this section we refute the main (strong) variant of ISH under weaker and more standard assumptions than those used to refute Weak ISH.

We start by defining a relaxed notion of extractable one-way function which is similar to the notion of (non-interactively) extractable function family ensemble put forward by Canetti and Dakdouk [10,11,18]. In contrast to the previous notion from Definition 2, the relaxed notion follows from previous concrete knowledge assumptions in the literature such as Damgård's *knowledge of exponent* assumption [19].

Definition 4 (Function Family Ensemble). *A* function family*, indexed by a key space K, is a set of functions $F = \{f_k\}_{k\in K}$ in which each function has the same domain and range. A* function family ensemble*, $\mathcal{F} = \{F^n\}_{n\in\mathbb{N}}$, is defined as an ensemble of function families F_n with key spaces $\{K_n\}_{n\in\mathbb{N}}$.*

Definition 5 (One-Way Function Family Ensemble). *A function family ensemble is* one-way *if:*

- *f_k can be evaluated (given 1^n, $k \in K_n$, and $x \in domain(f_k)$) in time polynomial in n, and*
- *for every polynomial-size circuit family C_n there is a negligible function ϵ such that for every n,*

$$Pr[k \leftarrow K_n; x \leftarrow domain(f_k); x' = C_n(1^n, k, f_k(x)) : f_k(x') = f_k(x)] \leq \epsilon(n).$$

Definition 6 (Non-Interactively Extractable One-Way Function Family Ensembles [18]). *We say that an one-way function family ensemble is* non-interactively extractable *(without auxiliary information) if for any efficient sampling algorithm A running in time $\ell(n)$ (with random input $r_A \in U_{\ell(n)}$), there exists a PPT algorithm K_A and a negligible function ϵ such that for all n:*

$$Pr[k \leftarrow K_n; r_A \leftarrow U_{\ell(n)}; y = A(1^n, k; r_A); x \leftarrow K_A(1^n, k, r_A) : (f_k(x) = y) \vee (\forall x', f_k(x') \neq y)] \geq 1 - \epsilon(n).$$

The difference between the above notion of extractable one-way function family ensembles and the notion of EOWF from Definition 2 is that extraction is not guaranteed for all functions in the function family but only for a randomly chosen function (concretely, the first step $k \leftarrow K_n$ chooses a random function). Furthermore, the process of picking the random function may use private randomness that is not available to the algorithm A.

The above difference makes it possible to derive extractable one-way function family ensembles from existing knowledge assumptions in literature [19,31,4,39]. As an example, the Knowledge of Exponent (KEA) Assumption [19] informally states that there exists an ensemble of groups $\{G_n\}_{n\in\mathbb{N}}$ where the discrete logarithm problem is hard to solve and any PPT adversary A that on input g, w can compute a pair of the form (g^r, w^r) must *know* r, in the sense that there

exists an efficient extractor K_A which given the random input of A can compute r. Mapping this example to Definition 6, the key space is $K_n = G_n \times G_n$ and the function f_k with $k = (w, g)$ is defined by $f_k(r) = (w^r, g^r)$.

Next, we replace the previous NIWI primitive with non-interactive zero knowledge (NIZK) proofs in the common reference string (CRS) model [6,40]. We omit the (standard) definition of NIZK, but note that the assumptions on which NIZK proof systems for NP can be based are significantly more general than the corresponding assumptions for NIWI, and include the existence of trapdoor permutations [24].

We are now ready to state the main theorem of this section.

Theorem 2. *If non-interactively extractable one-way function family ensembles exist and NIZK proof systems exist for NP, then ISH is false.*

Proof (sketch): The proof follows the same outline as the one from Theorem 1, but the use of NIZK instead of NIWI allows it to take a somewhat simpler form. Let $\mathcal{F}$ be a non-interactively extractable one-way function family ensemble. We first define an efficient sampling algorithm A whose inputs are pairs of strings (k, σ): k is a key from the key space of $\mathcal{F}$ and σ is a uniformly random string to be used as a CRS for a NIZK proof system. A outputs a random image of f_k and a NIZK proof (under σ) that the output is valid. Let B be the alternate invertible sampler hypothesized by ISH. Due to the soundness of the NIZK proof system, B outputs valid images of f_k when σ is chosen uniformly at random. Since $\mathcal{F}$ is extractable, we can use B, its extractor K_{B} and its inverter B^{-1} to construct an efficient inversion algorithm for the family ensemble $\mathcal{F}$, contradicting its one-wayness property. See the full version for details. □

6 Applications of ISH

While our main motivation for studying ISH is its relevance to adaptively secure MPC (discussed later in Section 7) we start by presenting two other consequences of (weak) ISH. In order to avoid any confusion, we remind the reader that in the previous sections we disproved ISH under some specific computational assumptions. However, as we couldn't disprove ISH *unconditionally* (or even under standard cryptographic assumptions), it is still interesting to investigate the consequences of ISH in order to put ISH in the proper cryptographic context and to further motivate our study.

PKE and OT: As a first consequence, we note that if ISH holds, this would settle the question of the relationship between public key encryption (PKE) and oblivious transfer (OT), as studied in [26].

Theorem 3. *If ISH holds, then the existence of semantic secure PKE implies the existence of an oblivious transfer protocol.*

Proof (sketch): The proof follows by considering a protocol for $\frac{1}{2}$-OT similar in spirit to the EGL protocol [23], where the receiver samples one public key with the key generation algorithm (thus learning the secret key), and the other using the alternate inverse-samplable algorithm, as described in ISH. Receiver's security loosely follows from the closeness property of ISH, while sender security can be deduced by the semantic security of the PKE scheme. See the full version for details. □

Assumptions for UC-secure computation: A systematic study of the minimal setup and computational assumptions for UC-secure computation has been recently undertaken in [21]. A question that the authors left open is whether the existence of stand-alone oblivious transfer (SA-OT) is a necessary assumption for UC-secure oblivious transfer (UC-OT) in the common reference string (CRS) model, where the string is sampled from an arbitrary distribution. If ISH holds, one could answer this question affirmatively. To show that SA-OT is necessary for UC-OT we will show how to construct a protocol for SA-OT assuming that UC-OT in the CRS model exists. Intuitively we need to generate a CRS to make the protocol work, but we don't want any party to learn the corresponding trapdoor. Unfortunately, we cannot let the parties use MPC in order to generate this CRS, since unconditional MPC is impossible, and we cannot assume that OT exists (or any equivalent computational assumption). But if ISH holds, there is a way of sampling any CRS without learning the trapdoor by using the invertible sampler, after which parties can run the UC-OT with respect to this CRS. Also note that we don't need this fake CRS to be distributed exactly as the real CRS, but just computationally close: if the UC-OT protocol works with the real CRS but not with the fake CRS, it could be used as a distinguisher, thus violating ISH. Standard compilation techniques can be used to turn this protocol into a protocol secure against a malicious adversary.

7 Adaptive Security and ISH

In this section we show that our strong variant of ISH (Hypothesis 1) is closely related to secure multi-party computation with security against adaptive adversaries (*adaptive MPC* or *AMPC* for short). We first show that if *all* randomized functionalities admit AMPC protocols, then ISH is true. Combined with Theorem 2, this gives the first strong evidence that *general* AMPC is impossible. Then, we proceed to show that if ISH is true and all the parties are mutually connected with OT-channels,[2] then general AMPC is possible – thus showing that ISH is essentially equivalent to general AMPC.

As discussed in the introduction, our results apply to a wide range of AMPC models from the literature. For convenience, we will refer to the two-party semi-honest model, under the definition of [8] which requires security against *post execution corruption (PEC)*. The latter means that after the execution is complete, the environment can ask the adversary to corrupt additional parties. The

[2] Our use of ideal OT can be replaced by any adaptively secure OT protocol, which can be based on standard cryptographic assumptions.

PEC requirement is needed to prove sequential composition of adaptively secure protocols, and is implied by most other definitions of adaptive security from the literature (such as adaptive UC-security). Our negative result does not hold for adaptively secure protocols without PEC (since in the semi-honest two-party case, security in this model is equivalent to non-adaptive security [12]).

Brief Preliminaries. This section is an informal introduction to (adaptively secure) MPC protocols. In an MPC protocol, *adaptive security* implies that an adversarial entity can adaptively choose the parties he wants to corrupt at any point in the protocol. An adversary is *semi-honest* if the parties that he corrupts always follow the prescribed protocol. His goal is to try and obtain as much information as possible under this constraint. Security against such adversaries is a basic requirement for any cryptographic protocol. An ideal model of security for MPC protocols is one in which there exists a trusted third party who (via secure private channels) receives all the inputs from the participants of the protocol and sends back their respective outputs. Semi-honest adversaries in this model can only learn the input and output of the parties that he corrupts. Considering this as a basis for security, in the ideal-real model of [8], a real world protocol for MPC is secure if for every adversary A in the real execution, there exists an ideal world adversary S (also known as the simulator), such that the outputs of A and S are computationally indistinguishable. We refer the reader to [8,36] for a more precise definition of this notion.

7.1 Adaptively Secure MPC Implies ISH

First we show that if AMPC protocols exist for every functionality $\mathcal{F}$, then ISH (Hypothesis 1) is true.

Theorem 4. *If for every PPT functionality $\mathcal{F}$ there exists a protocol Π that securely realizes $\mathcal{F}$ against an adaptive semi-honest adversary (with PEC), then ISH is true.*

Proof (sketch): Consider a two-party randomized functionality $\mathcal{F}$ that takes input from both parties and uses some internal coins and compute some function A. Now if there exist a protocol π between P_1, P_2 that securely implements $\mathcal{F}$, in particular the following two conditions will be satisfied: 1) The output of the protocol π and the functionality $\mathcal{F}$ are computationally close (because the protocol is *correct*); 2) There exist a simulator S that can explain the randomness used by P_1, P_2 in π to produce the output z, without access to the functionality random tape $r_{\mathcal{F}}$. Therefore we can use the protocol and the simulator (π, S) as a foundation to build the inverse-samplable algorithm $\mathsf{B}, \mathsf{B}^{-1}$ that satisfy the requirement of ISH. The inverse-samplable algorithm B can be constructed by simulating a run of the protocol π between P_1 and P_2 "in the head", while the inverter B^{-1} will run the simulator S as a subroutine. See the full version for more details. □

7.2 ISH Implies Adaptively Secure MPC

In the previous section we showed that AMPC implies ISH. Now we show that the converse is true too.

To make the result stronger, we will show that ISH implies the strongest variant of MPC i.e., multiparty computation secure against an active, adaptive adversary in the universally composable security framework (UC-AMPC). Given that UC computation is impossible in plain model, we look at the OT-hybrid model where it is possible to evaluate adaptively well-formed functionalities [14], and we show how ISH would allow us to extend this result to all functionalities. We refer the reader to [9] for the definition of UC-AMPC. We look at adaptive security, where the adversary $\mathcal{A}$ can corrupt any of the two parties P_1, P_2 at any point during the protocol π.

Theorem 5. *If ISH holds, then active secure UC-AMPC is possible for any functionality in the UC-OT hybrid model.*

Proof (sketch): It is known that any deterministic functionality can be securely implemented in the OT-hybrid model [35,34]. Using the UC composition theorem and ISH we extend the result for randomized functionalities.

Consider a general randomized functionality $(z_1, z_2) \leftarrow \mathcal{F}(x, y; \rho)$, where ρ is the private randomness of $\mathcal{F}$, (x, z_1) the input/output of P_1, and (y, z_2) the input/output of P_2. Let $z_i = f_i(x, y; \rho)$. Then from Strong ISH we know that there exist f_i', f_i^{-1}, the alternative sampler and the inverter.

Now define a new, deterministic functionality $\mathcal{G}$ as $(z_1, z_2) = \mathcal{G}((x, \rho_1), (y, \rho_2))$, where $z_i = f_i'(x, y; \rho_1^i \oplus \rho_2^i)$, and where f_i' is the alternative sampler for f_i. Being a deterministic functionality, $\mathcal{G}$ can be securely realized with adaptive security in the OT-hybrid model.

Now the protocol to implement $\mathcal{F}$ in the $\mathcal{G}$-hybrid model proceeds as follows. Party P_i picks ρ_i at random, feeds it into $\mathcal{G}$ together with its input, and waits to receive the output. Note that the protocol does not exactly compute the required functionality f, but f'. The indistinguishability requirements of ISH imply that the output of f and of f' are indistinguishable too, and that suffices for UC-computation. This protocol can be shown to be UC-secure, see the full version for more details. □

References

1. Barak, B., Ong, S.J., Vadhan, S.P.: Derandomization in cryptography. SIAM J. Comput. 37(2), 380–400 (2007)
2. Beaver, D.: Plug and play encryption. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 75–89. Springer, Heidelberg (1997)
3. Beaver, D., Haber, S.: Cryptographic protocols provably secure against dynamic adversaries. In: Rueppel, R.A. (ed.) EUROCRYPT 1992. LNCS, vol. 658, pp. 307–323. Springer, Heidelberg (1993)
4. 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)

5. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In: STOC, pp. 1–10 (1988)
6. Blum, M., Feldman, P., Micali, S.: Non-interactive zero-knowledge and its applications (extended abstract). In: STOC, pp. 103–112 (1988)
7. Blum, M., De Santis, A., Micali, S., Persiano, G.: Noninteractive zero-knowledge. SIAM J. Comput. 20(6), 1084–1118 (1991)
8. Canetti, R.: Security and composition of multiparty cryptographic protocols. J. Cryptology 13(1), 143–202 (2000)
9. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: FOCS, pp. 136–145 (2001)
10. Canetti, R., Dakdouk, R.R.: Extractable perfectly one-way functions. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, LNCS, vol. 5126, pp. 449–460. Springer, Heidelberg (2008)
11. Canetti, R., Dakdouk, R.R.: Towards a theory of extractable functions. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 595–613. Springer, Heidelberg (2009)
12. Canetti, R., Damgård, I., Dziembowski, S., Ishai, Y., Malkin, T.: Adaptive versus non-adaptive security of multi-party protocols. J. Cryptology 17(3), 153–207 (2004)
13. Canetti, R., Feige, U., Goldreich, O., Naor, M.: Adaptively secure multi-party computation. In: STOC, pp. 639–648 (1996)
14. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: STOC, pp. 494–503 (2002)
15. Chaum, D., Crépeau, C., Damgård, I.: Multiparty unconditionally secure protocols (extended abstract). In: STOC, pp. 11–19 (1988)
16. Choi, S.G., Dachman-Soled, D., Malkin, T., Wee, H.: Improved non-committing encryption with applications to adaptively secure protocols. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 287–302. Springer, Heidelberg (2009)
17. Choi, S.G., Soled, D.D., Malkin, T., Wee, H.: Simple, black-box constructions of adaptively secure protocols. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 387–402. Springer, Heidelberg (2009)
18. Dakdouk, R.R.: Theory and application of extractable functions (2009), http://cs-www.cs.yale.edu/homes/jf/Ronny-thesis.pdf
19. Damgård, I.: 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)
20. Damgård, I., Nielsen, J.B.: Improved non-committing encryption schemes based on a general complexity assumption. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 432–450. Springer, Heidelberg (2000)
21. Damgård, I., Nielsen, J.B., Orlandi, C.: On the necessary and sufficient assumptions for UC computation. In: Micciancio, D. (ed.) Theory of Cryptography. LNCS, vol. 5978, pp. 109–127. Springer, Heidelberg (2010)
22. Dwork, C., Naor, M.: Zaps and their applications. SIAM J. Comput. 36(6), 1513–1543 (2007)
23. Even, S., Goldreich, O., Lempel, A.: A randomized protocol for signing contracts. ACM Commun. 28(6), 637–647 (1985)
24. Feige, U., Lapidot, D., Shamir, A.: Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In: FOCS, pp. 308–317 (1990)
25. Garay, J.A., Wichs, D., Zhou, H.-S.: Somewhat non-committing encryption and efficient adaptively secure oblivious transfer. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 505–523. Springer, Heidelberg (2009)

26. Gertner, Y., Kannan, S., Malkin, T., Reingold, O., Viswanathan, M.: The relationship between public key encryption and oblivious transfer. In: FOCS, pp. 325–335 (2000)
27. Goldreich, O.: Foundations of cryptography: Basic applications. Cambridge Univ. Pr., Cambridge (2004)
28. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or a completeness theorem for protocols with honest majority, pp. 218–229 (1987)
29. Groth, J., Ostrovsky, R., Sahai, A.: Non-interactive Zaps and new techniques for NIZK. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 97–111. Springer, Heidelberg (2006)
30. 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)
31. Hada, S., Tanaka, T.: A relationship between one-wayness and correlation intractability. In: Imai, H., Zheng, Y. (eds.) PKC 1999. LNCS, vol. 1560, pp. 82–96. Springer, Heidelberg (1999)
32. Hsiao, C.-Y., Lu, C.-J., Reyzin, L.: Conditional computational entropy, or toward separating pseudoentropy from compressibility. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 169–186. Springer, Heidelberg (2007)
33. Impagliazzo, R., Luby, M.: One-way functions are essential for complexity based cryptography (extended abstract). In: FOCS, pp. 230–235 (1989)
34. 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)
35. Kilian, J.: Founding cryptography on oblivious transfer. In: STOC, pp. 20–31 (1988)
36. Lindell, A.Y.: Adaptively secure two-party computation with erasures. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 117–132. Springer, Heidelberg (2009)
37. Naor, M.: Evaluation may be easier than generation (extended abstract). In: STOC, pp. 74–83 (1996)
38. Naor, M.: On cryptographic assumptions and challenges. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 96–109. Springer, Heidelberg (2003)
39. Prabhakaran, M., Xue, R.: Statistically hiding sets. In: Fischlin, M. (ed.) CT-RSA 2009. M. Prabhakaran R. Xue, vol. 5473, pp. 100–116. Springer, Heidelberg (2009)
40. De Santis, A., Micali, S., Persiano, G.: Non-interactive zero-knowledge proof systems. In: Pomerance, C. (ed.) CRYPTO 1987. LNCS, vol. 293, pp. 52–72. Springer, Heidelberg (1988)
41. Wee, H.: On pseudoentropy versus compressibility. In: IEEE Conference on Computational Complexity, pp. 29–41 (2004)
42. Yao, A.C.-C.: How to generate and exchange secrets, pp. 162–167 (1986)

Multiparty Computation for Modulo Reduction without Bit-Decomposition and a Generalization to Bit-Decomposition*

Chao Ning and Qiuliang Xu**

School of Computer Science and Technology, Shandong University,
Jinan, 250101, China
ncnfl@mail.sdu.edu.cn, xql@sdu.edu.cn

Abstract. Bit-decomposition, which is proposed by Damgård *et al.*, is a powerful tool for multi-party computation (MPC). Given a sharing of secret a, it allows the parties to compute the sharings of the bits of a in constant rounds. With the help of bit-decomposition, constant-rounds protocols for various MPC problems can be constructed. However, bit-decomposition is relatively expensive, so constructing protocols for MPC problems without relying on bit-decomposition is a meaningful work. In multi-party computation, it remains an open problem whether the *modulo reduction problem* can be solved in constant rounds without bit-decomposition.

In this paper, we propose a protocol for (public) modulo reduction without relying on bit-decomposition. This protocol achieves constant round complexity and linear communication complexity. Moreover, we show a generalized bit-decomposition protocol which can, in constant rounds, convert the sharing of secret a into the sharings of the digits of a, along with the sharings of the bits of every digit. The digits can be base-m for any $m \geq 2$.

Keywords: Multiparty Computation, Constant-Rounds, Modulo Reduction, Generalization to Bit-Decomposition.

1 Introduction

Secure multi-party computation (MPC) allows the computation of a function f when the inputs to f are secret values held by distinct parties. After running the MPC protocol, the parties obtains only the predefined outputs but nothing else, and the privacy of their inputs are guaranteed. Although generic solutions for MPC already exist [2,10], the efficiency of these protocols tends to be low. So we focus on constructing efficient protocols for specific functions. More exactly, we are interested in integer arithmetic in the information theory setting [12].

* Supported by the National Natural Science Foundation of China under Grant No. 60873232.

** Corresponding author.

M. Abe (Ed.): ASIACRYPT 2010, LNCS 6477, pp. 483–500, 2010.

A proper choice of representation of the inputs can have great influence on the efficiency of the computation [7,18]. For example, when we want to compute the *sum* or the *product* of some private integer values, we'd better represent these integers as elements of a prime field $\mathbb{Z}_p$ and perform the computations using an arithmetic circuit as additions and multiplications are trivial operations in the field. If we use the binary representation of the integers and a Boolean circuit to compute the expected result, then we will get a highly inefficient protocol as the bitwise addition and multiplication are very expensive [4,5]. On the other hand, if we want to *compare* some (private) integer values, the binary representation will be of great advantage as comparison is a *bit-oriented* operation. In this case, the arithmetic circuit over $\mathbb{Z}_p$ will be a bad choice.

To bridge the gap between the arithmetic circuits and the Boolean circuits, Damgård *et al.* [7] proposed a novel protocol, called bit-decomposition, to convert a sharing of secret a into the sharings of the bits of a. This protocol is very useful both in theory and application. However, the bit-decomposition protocol is relatively expensive in terms of round and communication complexities. So the work on constructing (constant-rounds) protocols for MPC problems without relying on bit-decomposition is not only interesting but also meaningful. Recently, in [12], Nishide *et al.* constructed more efficient protocols for comparison, interval test and equality test of shared secrets without relying on the bit-decomposition protocol. However, it remains an open problem whether the *modulo reduction problem* can be solved in constant rounds without bit-decomposition [17]. In this paper, we show a linear protocol for the (public) modulo reduction problem without relying on bit-decomposition. What's more, the bit-decomposition protocol of [7] can only de-composite the sharing of secret a into the sharings of the *bits* of a. However, especially in practice, we may often need the sharings of the *digits* of a. Here the digits can be base-m for any $m \geq 2$. For example, in real life, integers are (almost always) represented as base-10 digits. Then, MPC protocols for practical use may often require the base-10 digits of the secret shared integers. Another example is as follows. If the integers are about *time* and *date*, then base-24, base-30, base-60, or base-365 digits may be required. So, to meet these requirements, we propose a generalization to bit-decomposition in this paper.

1.1 Our Contributions

First we introduce some necessary notations. We focus mainly on the multi-party computation based on linear secret sharing schemes. Assume that the underlying secret sharing scheme is built in field $\mathbb{Z}_p$ where p is a prime with bit-length l (i.e. $l = \lceil \log p \rceil$). For secret $a \in \mathbb{Z}_p$, we use $[a]_p$ to denote the secret sharing of a, and $[a]_B$ to denote the sharings of the bits of a, i.e. $[a]_B = \left([a_{l-1}]_p, ..., [a_1]_p, [a_0]_p\right)$.

The *public modulo reduction problem* can be formalized as follows:

$$[x \mod m]_p \leftarrow Modulo - Reduction([x]_p, m)$$

where $x \in \mathbb{Z}_p$ and $m \in \{2, 3, ..., p-1\}$.

In existing public modulo reduction protocols [7,17], the bit-decomposition is involved, incurring $O(l \log l)$ communication complexity. What's more, in the worst case, the communication complexity of this protocol may go up to $\Theta(l^2)$. Specifically, the existing modulo reduction protocol uses the bit-decomposition protocol to reduce the "size" of the problem, and then uses up to l *comparisons*, which is non-trivial, to determine the final result. This is essentially an "exhaustive search". If the bit-length of the inputs to the comparison protocol is relatively long, e.g. $\Theta(l)$ which is often the case, the overall complexity will go up to $\Theta(l^2)$. So, the efficiency of the protocol may be very poor. To solve this problem, we propose a protocol, which achieves constant round complexity and linear communication complexity, for public modulo reduction without relying on bit-decomposition. Besides this, we also propose an enhanced protocol that can output the sharings of the bits of $x \mod m$, i.e. $[x \mod m]_B$.

Moreover, we also construct a generalized bit-decomposition protocol which can, in constant rounds, convert the sharing of secret a into the sharings of the digits of a, along with the sharings of the bits of every digit. The digits can be base-m for any $m \geq 2$. We name this protocol the *Base-m Digit-Bit-Decomposition Protocol*. The asymptotic communication complexity of this protocol is $O(l \log l)$. Obviously, when m is a power of 2, this protocol degenerates to the bit-decomposition protocol.

For illustration, we will show an example here. Pick a binary number

$$a = (11111001)_2 = 249.$$

If $[a]_p$ is given to the bit-decomposition protocol as input, it outputs

$$[a]_B = ([1]_p, [1]_p, [1]_p, [1]_p, [1]_p, [0]_p, [0]_p, [1]_p);$$

if $[a]_p$ and $m = 2$ (or $m = 4, 8, 16, 32, ...$) are given to our *Base-m Digit-Bit-Decomposition* protocol as inputs, it will output the same result with the bit-decomposition protocol above; however, when $[a]_p$ and $m = 10$ are given to our *Base-m Digit-Bit-Decomposition* protocol, it will output

$$([2]_B, [4]_B, [9]_B) = \Big(([0]_p, [0]_p, [1]_p, [0]_p), ([0]_p, [1]_p, [0]_p, [0]_p), ([1]_p, [0]_p, [0]_p, [1]_p)\Big)$$

which is significantly different from the output of bit-decomposition.

We also propose a simplified version of the protocol, called *Base-m Digit-Decomposition Protocol*, which outputs $\Big([2]_p, [4]_p, [9]_p\Big)$ when given $[a]_p$ and $m = 10$ as inputs.

Finally, we strongly recommend the interested readers to read [13] which is the full version of this paper. Many of the details are omitted in the present paper due to space constraints.

1.2 Related Work

The problem of bit-decomposition is a basic problem in MPC and was partially solved by Algesheimer *et al.* in [1]. However, their solution is not constant-rounds and can only handle values that are noticeably smaller than p. Damgård *et al.* proposed the first constant-rounds (full) solution to the problem of bit-decomposition in [7]. This ice-break work is based on linear secret sharing schemes [2,11]. Independently, Shoenmakers and Tuyls [16] solved the problem

of bit-decomposition for multiparty computation based on (Paillier) threshold homomorphic cryptosystems [3,8]. Somewhat surprisingly, Nishide and Ohta proposed solutions for comparison, interval test and equality test of shared secrets without relying on bit-decomposition [12]. Their techniques are novel, and have enlightened us a lot. Recently, Toft showed a novel technique that can reduce the communication complexity of bit-decomposition to *almost linear* [18]. Although we do not focus on the "almost linear" property of protocols, some techniques proposed in their paper are so inspiring and enlightening to us. In a followup work, Reistad and Toft proposed a linear bit-decomposition protocol [14]. However, the security of their protocol is non-perfect.

As for the problem of modulo reduction (without bit-decomposition), Guajardo *et al.* proposed a partial solution to this problem in the threshold homomorphic setting [9]. In [6], Catrina *et al.* dealt with the non-constant-rounds private modulo reduction protocol with the incomplete accuracy and statistical privacy in the setting where shared secrets are represented as fixed-point numbers.

2 Preliminaries

In this section we introduce some important notations and some known primitives which will be frequently mentioned in the rest of the paper.

2.1 Notations and Conventions

The multiparty computation considered in this paper is based on linear secret sharing schemes, such as Shamir's [15]. As mentioned above, we denote the underlying field as $\mathbb{Z}_p$ where p is a prime with binary length l.

As in previous works, such as [7] and [12], we assume that the underlying secret sharing scheme allows to compute $[a+b \bmod p]_p$ from $[a]_p$ and $[b]_p$ without communication, and that it allows to compute $[ab \bmod p]_p$ from (public) $a \in \mathbb{Z}_p$ and $[b]_p$ without communication. We also assume that the secret sharing scheme allows to compute $[ab \bmod p]_p$ from $[a]_p$ and $[b]_p$ through communication among the parties. We call this procedure the *multiplication protocol*. Obviously, for multiparty computation, the multiplication protocol is a dominant factor of complexity as it involves communication. So, as in previous works, the round complexity of the protocols is measured by the number of rounds of parallel invocations of the multiplication protocol, and the communication complexity is measured by the number of invocations of the multiplication protocol. For example, if a protocol involves a multiplications in parallel and then another b multiplications in parallel, then we can say that the round complexity is 2 and the communication complexity is $a+b$ multiplications. We have to say that the complexity analysis made in this paper is somewhat rough for we focus mainly on the ideas of the solution, but not on the details of the implementation.

As in [12], when we write $[C]_p$, where C is a Boolean test, it means that $C \in \{0,1\}$ and $C=1$ iff C is true. For example, we use $[x \overset{?}{<} y]_p$ to denote the output of the comparison protocol, i.e. $(x \overset{?}{<} y) = 1$ iff $x < y$ holds.

For the base $m \in \{2, 3, ..., p-1\}$, define $L(m) = \lceil \log m \rceil$. It is easy to see that we should use $L(m)$ bits to represent a base-m digit. For example, when $m = 10$, we have $L(m) = \lceil \log 10 \rceil = 4$, this means we must use 4 bits to represent a base-10 digit. Notice that we have $2^{L(m)-1} < m \leq 2^{L(m)}$ and $m = 2^{L(m)}$ holds iff m is a power of 2. Moreover, we have $L(m) \leq l$ as $m \leq p - 1$.

Define $l^{(m)} = \lceil \log_m p \rceil$. Obviously, $l^{(m)}$ is the length of p when p is coded base-m. Note that $l^{(m)} = \lceil \log_m p \rceil = \left\lceil \frac{\log p}{\log m} \right\rceil = \left\lceil \frac{l}{\log m} \right\rceil \leq l$ as $m \geq 2$.

For any $a \in \mathbb{Z}_p$, the secret sharing of a is denoted by $[a]_p$. We use $[a]_B$ to denote the bitwise sharing of a.

We use $[a]_D^m = \left([a_{l^{(m)}-1}]_p^m, ..., [a_1]_p^m, [a_0]_p^m\right)$ to denote the *digit-wise sharing* of a. For $i \in \{0, 1, ..., l^{(m)} - 1\}$, $[a_i]_p^m$ denotes the sharing of the $i'th$ base-m digit of a with $0 \leq a_i \leq (m-1)$.

The *digit-bit-wise sharing* of a, which is denoted by $[a]_{D,B}^m$, is defined as below:

$$[a]_{D,B}^m = \left([a_{l^{(m)}-1}]_B^m, ..., [a_1]_B^m, [a_0]_B^m\right),$$

in which $[a_i]_B^m = \left([a_i^{L(m)-1}]_p, ..., [a_i^1]_p, [a_i^0]_p\right)$ $(i \in \{0, 1, ..., l^{(m)} - 1\})$ denotes the bitwise sharing of the $i'th$ base-m digit of a. Note that $[a_i]_B^m$ has $L(m)$ bits.

Sometimes, if m can be inferred from the context, we may write $[a_i]_p^m$ (or $[a_i]_B^m$) as $[a_i]_p$ (or $[a_i]_B$) for simplicity.

In this paper, we often need to get the *digit-wise representation* or the *digit-bit-wise representation* of some public value c, i.e. $[c]_D^m$ or $[c]_{D,B}^m$. This can be done freely as c is public.

It's easy to see that if we have obtained $[x]_B$, then $[x]_p$ can be freely obtained by a linear combination. We can think of this as $[x]_B$ contains "more information" than $[x]_p$. For example, if we get $[a]_{D,B}^m = \left([a_{l^{(m)}-1}]_B^m, ..., [a_1]_B^m, [a_0]_B^m\right)$, then $[a]_D^m = \left([a_{l^{(m)}-1}]_p^m, ..., [a_1]_p^m, [a_0]_p^m\right)$ is implicitly obtained. In protocols that can output both $[x]_B$ and $[x]_p$, which is often the case in this paper, we always omit $[x]_p$ for simplicity.

Given $[c]_p$, we need a protocol to reveal c, which is denoted by $c \leftarrow reveal([c]_p)$.

When we write command $C \leftarrow b?A : B$, where $A, B, C \in \mathbb{Z}_p$ and $b \in \{0, 1\}$, it means the following:

$$if\ b = 1,\ then\ C\ is\ set\ to\ A;\ otherwise,\ C\ is\ set\ to\ B.$$

We call this command the *conditional selection command*. When all the variables in this command are public, this "selection" can of course be done. When the variables are shared or even bitwise shared, this can also be done. Specifically, the command

$$[C]_p \leftarrow [b]_p?[A]_p : [B]_p$$

can be realized by setting

$$[C]_p \leftarrow [b]_p([A]_p - [B]_p) + [B]_p$$

which costs 1 round and 1 multiplication; the command

$$[C]_B \leftarrow [b]_p?[A]_B : [B]_B$$

can be realized by the following procedure:

For $i = 0, 1, ..., l-1$ in parallel: $[C_i]_p \leftarrow [b]_p([A_i]_p - [B_i]_p) + [B_i]_p$
$[C]_B \leftarrow \left([C_{l-1}]_p, ..., [C_1]_p, [C_0]_p\right)$ $\triangleright |A| = |B| = |C| = l$

Note that the above procedure costs 1 round, l invocations of multiplication.

Other cases, such as $[C]_D^m \leftarrow [b]_p?[A]_D^m : [B]_D^m$ and $[C]_{D,B}^m \leftarrow [b]_p?[A]_{D,B}^m : [B]_{D,B}^m$ can be realized similarly. We will frequently use this *conditional selection command* in our protocols.

2.2 Known Primitives

We will now simply introduce some existing primitives which are important building blocks of this paper. All these primitives are proposed in [7].

▲***Random-Bit.*** The *Random-Bit* protocol is the most basic primitive which can generate a shared uniformly random bit unknown to all parties. In the linear secret sharing setting, which is the case in this paper, it takes only 2 rounds and 2 multiplications.

▲***Bitwise-LessThan.*** Given two bitwise shared inputs $[x]_B$ and $[y]_B$, the *Bitwise-LessThan* protocol can compute a shared bit $[x \overset{?}{<} y]_p$. We note that using the method of [18], this protocol can be realized in 6 rounds and $13l + 6\sqrt{l}$ multiplications. Notice that $13l + 6\sqrt{l} \leq 14l$ holds for $l \geq 36$ which is often the case in practice. So, for simplicity, we refer to the complexity of this protocol as 6 rounds and $14l$ multiplications.

▲***Bitwise-Addition.*** Given two bitwise shared inputs, $[x]_B$ and $[y]_B$, the *Bitwise-Addition* protocol outputs $[x + y]_B$. An important point of this protocol is that $d = x + y$ holds over the integers, not (only) mod p. This protocol, which costs 15 rounds and $47l \log l$ multiplications, is the most expensive primitive of the bit-decomposition protocol of [7]. We will not use this primitive in this paper, but use *Bitwise-Subtraction* instead. However, the asymptotic complexity of our *Bitwise-Subtraction* protocol is the same with that of the *Bitwise-Addition* since they both involve a *generic prefix protocol* which costs $O(l \log l)$ multiplications. We will introduce our *Bitwise-Subtraction* protocol later.

3 A Simple Introduction to Our New Primitives

In this section, we will simply introduce the new primitives proposed in this paper. We will only describe the inputs and the outputs of the protocols, along with some simple comments. All these new primitives will be described in detail in Section 6.

•***Bitwise-Subtraction.*** The *Bitwise-Subtraction* protocol accepts two bitwise shared values $[x]_B$ and $[y]_B$ and outputs $[x - y]_B$. This protocol is in fact first proposed in [18] and is re-described (in a widely different form) in this paper. In our protocols, we only need a restricted version (of *Bitwise-Subtraction*) which requires $x \geq y$. A run of this *restricted* protocol is denoted by

$$[x-y]_B \leftarrow Bitwise-Subtraction^*([x]_B,[y]_B).$$

It costs 15 rounds and $47l\log l$ multiplications.

• ***BORROWS.*** This protocol is used as a sub-protocol in the *Bitwise-Subtraction* protocol above to compute the borrow bits (as well as in the *Bitwise-Subtraction** protocol). Given two bitwise sharings $[x]_B$ and $[y]_B$, this protocol outputs

$$([b_{l-1}]_p,...,[b_1]_p,[b_0]_p) \leftarrow BORROWS([x]_B,[y]_B)$$

where $[b_i]_p$ is the sharing of the borrow bit at bit-position $i \in \{0,1,...,l-1\}$.

• ***Random-Digit-Bit.*** Given $m \in \{2,3,...,p-1\}$ as input, the *Random-Digit-Bit* protocol outputs

$$[d]_B^m = \left([d^{L(m)-1}]_p,...,[d^1]_p,[d^0]_p\right) \leftarrow Random-Digit-Bit(m)$$

where $d \in \{0,1,...,m-1\}$ represents a base-m digit. Notice that $[d]_p^m$ is implicitly obtained. The complexity of this protocol is 8 rounds and $16L(m)$ multiplications.

• ***Digit-Bit-wise-LessThan.*** The *Digit-Bit-wise-LessThan* protocol accepts two digit-bit-wise shared values $[x]_{D,B}^m$ and $[y]_{D,B}^m$ and outputs

$$[x \overset{?}{<} y]_p \leftarrow Digit-Bit-wise-LessThan([x]_{D,B}^m,[y]_{D,B}^m).$$

The complexity of this protocol is 6 rounds and $14l$ multiplications.

• ***Random-Solved-Digits-Bits.*** Using the above two primitives as sub-protocols, we can construct the *Random-Solved-Digits-Bits* protocol which, when given $m \in \{2,3,...,p-1\}$ as input, outputs a digit-bit-wise shared random value $[r]_{D,B}^m$ satisfying $r < p$. We denote a run of this protocol by

$$[r]_{D,B}^m \leftarrow Random-Solved-Digits-Bits(m).$$

This protocol takes 14 rounds and $312l$ multiplications.

• ***Digit-Bit-wise-Subtraction.*** This protocol is a novel generalization to the bitwise subtraction protocol and is very important to this paper. It accepts two digit-bit-wise shared values $[x]_{D,B}^m$ and $[y]_{D,B}^m$ and outputs $[x-y]_{D,B}^m$. Again, in this paper, we need only a restricted version which requires $x \geq y$. A run of this restricted protocol is denoted by

$$[x-y]_{D,B}^m \leftarrow Digit-Bit-wise-Subtraction^*([x]_{D,B}^m,[y]_{D,B}^m).$$

This restricted protocol costs 30 rounds and $47l\log l + 47l\log(L(m))$ multiplications. What's more, if we don't need $[x-y]_{D,B}^m$ but (only) need $[x-y]_D^m$ instead, then this restricted protocol can be (further) simplified. We denote a run of this (further) simplified protocol by

$$[x-y]_D^m \leftarrow Digit-Bit-wise-Subtraction^{*-}([x]_{D,B}^m,[y]_{D,B}^m).$$

The complexity of this protocol goes down to 21 rounds and $16l+47l^{(m)}\log\left(l^{(m)}\right)$ multiplications.

With the above primitives, we can construct our *Modulo-Reduction* protocol and *Base-m Digit-Bit-Decomposition* protocol, which will be described in detail separately in Section 4 and Section 5.

4 Multiparty Computation for Modulo Reduction without Bit-Decomposition

In this section, we give out our (public) *Modulo-Reduction* protocol which is realized without relying on bit-decomposition. This protocol is constant-rounds and involves only $O(l)$ multiplications. Informally speaking, our *Modulo-Reduction* protocol is essentially the *Least Significant Digit Protocol* and is a natural generalization to the *Least Significant Bit Protocol* (i.e. the *LSB* protocol) in [12]. Recall that for an integer a, the sharing of the least significant base-m digit of a is denoted by $[a_0]_p^m$, and the bitwise sharing of the least significant base-m digit of a is denoted by $[a_0]_B^m$. The protocol is described in detail in **Protocol 1**.

Protocol 1. The modulo reduction protocol, $Modulo - Reduction(\cdot)$, for computing the residue of a shared integer modulo a public integer.

Input: $[x]_p$ with $x \in \mathbb{Z}_p$ and $m \in \{2, 3, ..., p-1\}$.
Output: $[x \mod m]_p$.
Process:
$[r]_{D,B}^m \leftarrow Random - Solved - Digits - Bits(m)$
$c \leftarrow reveal([x]_p + [r]_{D,B}^m)$ ▷Note that $[r]_{D,B}^m$ implies $[r]_p^m$. (1.a)
$[X_1]_p^m \leftarrow [c_0]_p^m - [r_0]_B^m$ ▷ $[r_0]_B^m$ implies $[r_0]_p^m$.
$[X_2]_p^m \leftarrow [c_0]_p^m - [r_0]_B^m + m$
$[s]_p \leftarrow Bitwise - LessThan([c_0]_B^m, [r_0]_B^m)$ (1.b)
$[X]_p^m \leftarrow [s]_p?[X_2]_p^m : [X_1]_p^m$ ▷A *conditional selection command*.

$c' \leftarrow c + p$ ▷Addition over the integers.
$[X'_1]_p^m \leftarrow [c'_0]_p^m - [r_0]_B^m$
$[X'_2]_p^m \leftarrow [c'_0]_p^m - [r_0]_B^m + m$
$[s']_p \leftarrow Bitwise - LessThan([c'_0]_B^m, [r_0]_B^m)$ (1.c)
$[X']_p^m \leftarrow [s']_p?[X'_2]_p^m : [X'_1]_p^m$

$[t]_p \leftarrow Digit - Bit - wise - LessThan([c]_{D,B}^m, [r]_{D,B}^m)$ (1.d)
$[x \mod m]_p = [x_0]_p^m \leftarrow [t]_p?[X']_p^m : [X]_p^m$
Return $[x \mod m]_p$

Correctness: By simulating a *base-m addition* process, the protocol extracts $[x_0]_p^m$ which is just $[x \mod m]_p$. See [13] for the details.

Privacy: The only possible information leakage takes place in line (1.a), where a *reveal* command is involved. However, the revealed value, i.e. c, is uniformly random, so it leaks no information about the secret x. So the privacy is guaranteed.

Complexity: Complexity comes mainly from the invocations of sub-protocols. Note that the two invocations of *Bitwise-LessThan* and the invocation of *Digit-Bit-wise-LessThan* can be scheduled in parallel. In all it will cost 22 rounds and

$$312l + 14L(m) + 1 + 14L(m) + 1 + 14l + 1 = 326l + 28L(m) + 3$$

multiplications. Recall that $L(m) \leq l$, so the communication complexity is upper bounded by $354l + 3$ multiplications.

The original modulo reduction problem does not require the sharings of the bits of the residue, i.e. $[x \bmod m]_B$. So in the above protocol, $[x \bmod m]_B$ is not computed. However, if we want, we can get $[x \bmod m]_B$ using an enhanced version of the above *Modulo-Reduction* protocol. This enhanced protocol will be denoted by $Modulo-Reduction^{+}(\cdot)$. The construction is seen as **Protocol 2**.

Protocol 2. The *enhanced* modulo reduction protocol, $Modulo-Reduction^{+}(\cdot)$, for computing the *bitwise shared* residue of a shared integer modulo a public integer.

Input: $[x]_p$ with $x \in \mathbb{Z}_p$ and $m \in \{2, 3, ..., p-1\}$.
Output: $[x \mod m]_B$.
Process:
$[r]^m_{D,B} \leftarrow Random-Solved-Digits-Bits(m)$
$c \leftarrow reveal([x]_p + [r]^m_{D,B})$
$[\bar{M}_1]^m_B \leftarrow [c_0]^m_B \qquad [\bar{S}_1]^m_B \leftarrow [r_0]^m_B$
$[\bar{M}_2]^m_B \leftarrow [c_0 + m]^m_B \quad [\bar{S}_2]^m_B \leftarrow [r_0]^m_B$ ▷Addition over the integers.
$[s]_p \leftarrow Bitwise-LessThan([c_0]^m_B, [r_0]^m_B)$
$[\bar{M}]^m_B \leftarrow [s]_p?[\bar{M}_2]^m_B : [\bar{M}_1]^m_B$ ▷Involving $L(m)$ multiplications.
$[\bar{S}]^m_B \leftarrow [s]_p?[\bar{S}_2]^m_B : [\bar{S}_1]^m_B$

$c' \leftarrow c + p$
$[\bar{M}'_1]^m_B \leftarrow [c'_0]^m_B \qquad [\bar{S}'_1]^m_B \leftarrow [r_0]^m_B$
$[\bar{M}'_2]^m_B \leftarrow [c'_0 + m]^m_B \quad [\bar{S}'_2]^m_B \leftarrow [r_0]^m_B$
$[s']_p \leftarrow Bitwise-LessThan([c'_0]^m_B, [r_0]^m_B)$
$[\bar{M}']^m_B \leftarrow [s']_p?[\bar{M}'_2]^m_B : [\bar{M}'_1]^m_B$
$[\bar{S}']^m_B \leftarrow [s']_p?[\bar{S}'_2]^m_B : [\bar{S}'_1]^m_B$

$[t]_p \leftarrow Digit-Bit-wise-LessThan(c, [r]^m_{D,B})$
$[M]^m_B \leftarrow [t]_p?[\bar{M}']^m_B : [\bar{M}]^m_B$ ▷M is the minuend.
$[S]^m_B \leftarrow [t]_p?[\bar{S}']^m_B : [\bar{S}]^m_B$ ▷S is the subtrahend.

$[x \bmod m]_B = [x_0]^m_B \leftarrow Bitwise-Subtraction^{*}([M]^m_B, [S]^m_B)$
Return $[x \bmod m]_B$

The correctness and privacy of this protocol can be proved similarly to the *Modulo-Reduction* protocol above. By carefully selecting the *Minuend* and the *Subtrahend*, we can get the expected result by using *only one invocation* of the *Bitwise-Subtraction** protocol. The overall complexity of this protocol is 37 rounds and

$$326l+28L(m)+47L(m)\log(L(m))+6\ L(m) = 326l+34L(m)+47L(m)\log(L(m))$$

multiplications.

5 A Generalization to Bit-Decomposition

In this section, we will propose our generalization to bit-decomposition, i.e. the *Base-m Digit-Bit-Decomposition* protocol. The details of this protocol are

presented in **Protocol 3**. The main framework of this protocol is similar to the bit-decomposition protocol of [18].

Protocol 3. The *Base-m Digit-Bit-Decomposition* protocol, $Digit - Bit - Decomposition(\cdot, m)$, for converting the sharing of secret x into the digit-bit-wise sharing of x.

Input: $[x]_p$ with $x \in \mathbb{Z}_p$ and the base $m \in \{2, 3, ..., p-1\}$.
Output: $[x]^m_{D,B}$
Process:
$[r]^m_{D,B} \leftarrow Random - Solved - Digits - Bits(m)$
$c \leftarrow reveal([x]_p + [r]^m_{D,B})$ (3.a)
$c' \leftarrow c + p$
$[t]_p \leftarrow Digit - Bit - wise - LessThan([c]^m_{D,B}, [r]^m_{D,B})$ (3.b)
$[\tilde{c}]^m_{D,B} = [t]_p ? [c']^m_{D,B} : [c]^m_{D,B}$ $\triangleright Note\ that\ \tilde{c} = x + r$
$[x]^m_{D,B} \leftarrow Digit - Bit - wise - Subtraction^*([\tilde{c}]^m_{D,B}, [r]^m_{D,B})$ (3.c)
Return $[x]^m_{D,B}$

Correctness is described in detail in [13]. Privacy is straightforward. The overall complexity of this protocol is $14 + 6 + 30 = 50$ rounds and

$$312l + 14l + (47l \log l + 47l \log (L(m))) = 326l + 47l \log l + 47l \log (L(m))$$

multiplications. The communication complexity is upper bounded by $326l + 94l \log l$ multiplications as $L(m) \leq l$.

If we do not need $[x]^m_{D,B}$ but (only) need $[x]^m_D$ instead, then the above protocol can be simplified. The method is to replace the *Digit-Bit-wise-Subtraction** protocol with the *Digit-Bit-wise-Subtraction*$^{*-}$ protocol. We call this simplified protocol the *Base-m Digit-Decomposition Protocol*, a run of which is denoted by $Digit - Decomposition(\cdot, m)$. The correctness and privacy of this protocol can be similarly proved. The complexity goes down to $14 + 6 + 21 = 41$ rounds and

$$312l + 14l + \left(16l + 47l^{(m)} \log \left(l^{(m)}\right)\right) = 342l + 47l^{(m)} \log \left(l^{(m)}\right)$$

multiplications. Recall that $l^{(m)} = \lceil \log_m p \rceil \leq l$, so the communication complexity is upper bounded by $342l + 47l \log l$ multiplications.

6 Realizing the Primitives

In this section, we will describe in detail the (new) primitives which are essential for the protocols of our paper. Informally, most of the protocols in this section are generalized version of the protocols of [7] from base-2 to base-m for any $m \geq 2$. It will be seen that, when m is a power of 2, some of our primitives degenerate to the existing primitives in [7]. So, in the complexity analysis, we focus on the case where m is not a power of 2, i.e. $m < 2^{L(m)}$.

6.1 Bitwise-Subtraction

We describe the *Bitwise-Subtraction* protocol here. In fact, this protocol is already proposed in [18]. They reduced the problem of bitwise-subtraction to

the *Post-fix Comparison* problem. Here, we re-consider the problem of bitwise-subtraction and solve it in a (highly) similar manner to the *Bitwise-Addition* protocol of [7].

As is mentioned in Section 3, we will first propose a restricted (*bitwise-subtraction*) protocol, *Bitwise-Subtraction**, which requires that *the minuend is not less than the subtrahend*. We will only use this restricted version in this paper. The general version without the above restriction can be realized with the help of the *Bitwise-LessThan* protocol. See [13] for the details. Given a *BORROWS* protocol that can compute the sharings of the borrow bits, the *Bitwise-Subtraction** protocol can be realized as in **Protocol 4**.

Protocol 4. The *restricted* bitwise-subtraction protocol, $Bitwise-Subtraction^*(\cdot)$, for computing the bitwise sharing of the difference between two bitwise shared values. This protocol requires that the minuend is not less than the subtrahend.

Input: $[x]_B = ([x_{l-1}]_p, ..., [x_1]_p, [x_0]_p)$ and $[y]_B = ([y_{l-1}]_p, ..., [y_1]_p, [y_0]_p)$ satisfying $x \geq y$.
Output: $[x-y]_B = [d]_B = ([d_{l-1}]_p, ..., [d_1]_p, [d_0]_p)$.
Process:
$([b_{l-1}]_p, ..., [b_1]_p, [b_0]_p) \leftarrow BORROWS([x]_B, [y]_B)$
$[d_0]_p \leftarrow [x_0]_p - [y_0]_p + 2[b_0]_p$
For $i = 1, 2, ..., l-1$ in parallel: $[d_i]_p \leftarrow [x_i]_p - [y_i]_p + 2[b_i]_p - [b_{i-1}]_p$
$[x-y]_B = [d]_B \leftarrow ([d_{l-1}]_p, ..., [d_1]_p, [d_0]_p)$
Return $[x-y]_B$

Note that the output of this protocol, i.e. $[x-y]_B$, is of bit length l, not $l+1$. This is because $x \geq y$ holds and thus we do not need a sign bit. Correctness and privacy is straightforward. The complexity of this protocol is 15 rounds and $47l \log l$ multiplications.

6.2 Computing the Borrow Bits

We now describe the *BORROWS* protocol which can compute the sharings of the borrow bits. In fact our *BORROWS* protocol is highly similar to the *CARRIES* protocol in [7]. So only the difference is sketched here. As in [7], we use an operator $\circ : \sum \times \sum \rightarrow \sum$, where $\sum = \{S, P, K\}$, which is defined by $S \circ x = S$ for all $x \in \sum$, $K \circ x = K$ for all $x \in \sum$, $P \circ x = x$ for all $x \in \sum$. Here, $\circ$ represents the *borrow-propagation* operator, whereas in [7] it represents the *carry-propagation* operator. When computing $[x-y]_B$ (where $x \geq y$ holds) with two bitwise shared inputs

$$[x]_B = ([x_{l-1}]_p, ..., [x_1]_p, [x_0]_p) \quad and \quad [y]_B = ([y_{l-1}]_p, ..., [y_1]_p, [y_0]_p),$$

for bit-position $i \in \{0, 1, \ldots, l-1\}$, let $e_i = S$ iff a borrow is set at position i (i.e. $x_i < y_i$); $e_i = P$ iff a borrow would be propagated at position i (i.e. $x_i = y_i$); $e_i = K$ iff a borrow would be killed at position i (i.e. $x_i > y_i$). It can be easily verified that $b_i = 1$ (i.e. the $i'th$ borrow bit is set, which means the $i'th$ bit needs

to borrow a "1" from the $(i+1)'th$ bit) iff $e_i \circ e_{i-1} \circ \cdots \circ e_0 = S$. It can be seen that in the case where $\circ$ represents the borrow-propagation operator and in the case where $\circ$ represents the carry-propagation operator, the rules for $\circ$ (i.e. $S \circ x = S$, $K \circ x = K$ and $P \circ x = x$ for all $x \in \sum$) are *completely the same*. This means that when computing the borrow bits, once all the $e_i's$ are obtained, the residue procedure of our *BORROWS* protocol will be (completely) the same with that of the CARRIES protocol (in [7]). So, the only difference lies in the procedure of computing the $e_i's$, which will be sketched below.

As in [7], we represent S, P and K with bit vectors

$$(1,0,0),(0,1,0),(0,0,1) \in \{0,1\}^3.$$

Then, for every bit-position $i \in \{0,1,...,l-1\}$, $[e_i]_B = ([s_i]_p, [p_i]_p, [k_i]_p)$ can be obtained as follows: $[s_i]_p = [y_i]_p - [x_i]_p[y_i]_p$; $[p_i]_p = 1 - [x_i]_p - [y_i]_p + 2[x_i]_p[y_i]_p$; $[k_i]_p = [x_i]_p - [x_i]_p[y_i]_p$, which in fact need only one multiplication (i.e. $[x_i]_p[y_i]_p$). Correctness follows readily from the above arguments. Privacy is straightforward. The complexity of the protocol is 15 rounds and $47l \log l$ multiplications.

6.3 Random-Digit-Bit

We will now introduce the *Random-Digit-Bit* protocol for generating a random bitwise shared base-m digit, which is denoted by d here. In fact, d is a random integer satisfying $0 \leq d \leq m-1$. The details are presented in **Protocol 5**.

Protocol 5. The *Random-Digit-Bit* protocol, $Random - Digit - Bit(\cdot)$, for generating the bitwise sharing of a random digit. The digit is base-m for any $m \geq 2$.

Input: The base m satisfying $2 \leq m \leq p-1$.
Output: $[d]_B^m = ([d^{L(m)-1}]_p, ..., [d^1]_p, [d^0]_p)$ with $0 \leq d \leq m-1$.
Process:
For $i = 0, 1, \ldots, L(m) - 1$ in parallel: $[d^i]_p \leftarrow Random - Bit()$.
$[d]_B^m \leftarrow ([d^{L(m)-1}]_p, ..., [d^1]_p, [d^0]_p)$
If $m = 2^{L(m)}$, then **Return** $[d]_B^m$. Otherwise proceed as below.
$[r]_p \leftarrow Bitwise - LessThan([d]_B^m, m)$
$r \leftarrow reveal([r]_p)$
If $r = 0$, then abort. Otherwise **Return** $[d]_B^m$.

See [13] for the correctness. As for the privacy, when this protocol does not abort, the only information leaked is that $d < m$, which is an *a priori* fact. As for the complexity, when m is not a power of 2, the total complexity of one run of this protocol is 8 rounds and $16L(m)$ multiplications. As in [7], using a Chernoff bound, it can be seen that if this protocol has to be repeated in parallel to get a lower abort probability, then the round complexity is still 8, and the amortized communication complexity goes up to $4 \times 16L(m) = 64L(m)$ multiplications.

6.4 Digit-Bit-Wise-LessThan

The *Digit-Bit-wise-LessThan* protocol proposed here is a natural generalization to the *Bitwise-LessThan* protocol. Recall that when we write $[C]_p$, where C is a Boolean test, it means that $C \in \{0,1\}$ and $C = 1$ iff C is true. The details of the protocol are presented in **Protocol 6**.

Protocol 6. The *Digit-Bit-wise-LessThan* protocol,
$Digit-Bit-wise-LessThan(\cdot)$, for comparing two digit-bit-wise shared values.

Input: Two digit-bit-wise shared values $[x]^m_{D,B} = ([x_{l^{(m)}-1}]^m_B, ..., [x_1]^m_B, [x_0]^m_B)$ and $[y]^m_{D,B} = ([y_{l^{(m)}-1}]^m_B, ..., [y_1]^m_B, [y_0]^m_B)$.
Output: $[(x \overset{?}{<} y)]_p$, where $(x \overset{?}{<} y) = 1$ iff $x < y$ holds.
Process:
$[X]_B \leftarrow ([x^{L(m)-1}_{l^{(m)}-1}]_p, ..., [x^1_{l^{(m)}-1}]_p, [x^0_{l^{(m)}-1}]_p,$
...
...
...
$[x^{L(m)-1}_1]_p, ..., [x^1_1]_p, [x^0_1]_p,$
$[x^{L(m)-1}_0]_p, ..., [x^1_0]_p, [x^0_0]_p)$

$[Y]_B \leftarrow ([y^{L(m)-1}_{l^{(m)}-1}]_p, ..., [y^1_{l^{(m)}-1}]_p, [y^0_{l^{(m)}-1}]_p,$
...
...
...
$[y^{L(m)-1}_1]_p, ..., [y^1_1]_p, [y^0_1]_p,$
$[y^{L(m)-1}_0]_p, ..., [y^1_0]_p, [y^0_0]_p)$

$[(x \overset{?}{<} y)]_p = [(X \overset{?}{<} Y)]_p \leftarrow Bitwise - LessThan([X]_B, [Y]_B)$
Return $[(x \overset{?}{<} y)]_p$

Correctness is presented in [13]. Privacy follows readily from only using private sub-protocols. The complexity of the protocol is 6 rounds and (about) $14l$ multiplications.

6.5 Random-Solved-Digits-Bits

The *Random-Solved-Digits-Bits* protocol is an important primitive which can generate a digit-bit-wise shared random value unknown to all parties. It is a natural generalization to the *Random-Solved-Bits* protocol in [7]. The details are presented in **Protocol 7**.

Recall that the bitwise representation of the most significant base-m digit of p is $[p_{l^{(m)}-1}]^m_B = \left(p^{L(m)-1}_{l^{(m)}-1}, ..., p^1_{l^{(m)}-1}, p^0_{l^{(m)}-1}\right)$. Suppose $p^j_{l^{(m)}-1}$ $(j \in \{0, 1, ...,$

$L(m)-1\}$) is the left-most "1" in $[p_{l^{(m)}-1}]_B^m$. Then, in order to get an acceptable abort probability, the bit-length of the most significant base-m digit of r should be $j+1$ because an acceptable r must be less than p. In this protocol, for simplicity, we assume that $p_{l^{(m)}-1}^{L(m)-1}=1$. Under this assumption, we can generate $[r_{l^{(m)}-1}]_B^m$ using the *Random-Digit-Bit* protocol. If $p_{l^{(m)}-1}^{L(m)-1}=0$, then $[r_{l^{(m)}-1}]_B^m$ can be generated by using the *Random-Bit* protocol directly.

Protocol 7. The *Random-Solved-Digits-Bits* protocol, $Random-Solved-Digits-Bits(\cdot)$, for jointly generating a digit-bit-wise shared value which is uniformly random from $\mathbb{Z}_p$.

Input: m, i.e. the expected base of the digits.
Output: $[r]_{D,B}^m$, in which r is a uniformly random value satisfying $r<p$.
Process:
For $i=0,1,...,l^{(m)}-1$ in parallel: $[r_i]_B^m \leftarrow Random-Digit-Bit(m)$.
$[r]_{D,B}^m \leftarrow ([r_{l^{(m)}-1}]_B^m,...,[r_1]_B^m,[r_0]_B^m)$
$[c]_p \leftarrow Digit-Bit-wise-LessThan([r]_{D,B}^m,[p]_{D,B}^m)$
$c \leftarrow reveal([c]_p)$
If $c=0$, then abort. Otherwise **Return** $[r]_{D,B}^m$.

The correctness and the privacy is straightforward. The amortized complexity of this protocol is 8+6=14 rounds and $\left(l^{(m)}\cdot 64L(m)+14l\right)*4=312l$ multiplications.

6.6 Digit-Bit-Wise-Subtraction

In this section, we will describe in detail the restricted version, *Digit-Bit-wise-Subtraction**, which requires that the minuend is not less than the subtrahend. The general version, which can be realized using the techniques in the *Bitwise-Subtraction* protocol and which is not used in the paper, is omitted for simplicity.

•**The Restricted Digit-Bit-Wise-Subtraction.** We will now describe in detail the *Digit-Bit-wise-Subtraction** protocol. This protocol is novel and is the most important primitive in our *Base-m Digit-Bit-Decomposition* protocol. The details are presented in **Protocol 8**.

Protocol 8. The *restricted Digit-Bit-wise-Subtraction* protocol, $Digit-Bit-wise-Subtraction^*(\cdot)$, for computing the digit-bit-wise sharing of the difference between two digit-bit-wise shared values with the minuend not less than the subtrahend.

Input: $[x]_{D,B}^m=([x_{l^{(m)}-1}]_B^m,...,[x_1]_B^m,[x_0]_B^m)$ and
$[y]_{D,B}^m=([y_{l^{(m)}-1}]_B^m,...,[y_1]_B^m,[y_0]_B^m)$ satisfying $x\geq y$.
Output: $[x-y]_{D,B}^m=[d]_{D,B}^m=([d_{l^{(m)}-1}]_B^m,...,[d_1]_B^m,[d_0]_B^m)$.

Process:

$$[X]_B \leftarrow ([x_{l^{(m)}-1}^{L(m)-1}]_p, ..., [x_{l^{(m)}-1}^{1}]_p, [x_{l^{(m)}-1}^{0}]_p,$$
$$\cdots$$
$$\cdots$$
$$\cdots$$
$$[x_1^{L(m)-1}]_p, ..., [x_1^1]_p, [x_1^0]_p,$$
$$[x_0^{L(m)-1}]_p, ..., [x_0^1]_p, [x_0^0]_p)$$

$$[Y]_B \leftarrow ([y_{l^{(m)}-1}^{L(m)-1}]_p, ..., [y_{l^{(m)}-1}^{1}]_p, [y_{l^{(m)}-1}^{0}]_p,$$
$$\cdots$$
$$\cdots$$
$$\cdots$$
$$[y_1^{L(m)-1}]_p, ..., [y_1^1]_p, [y_1^0]_p,$$
$$[y_0^{L(m)-1}]_p, ..., [y_0^1]_p, [y_0^0]_p)$$

$$([b_{l^{(m)}-1}^{L(m)-1}]_p, ..., [b_{l^{(m)}-1}^{1}]_p, [b_{l^{(m)}-1}^{0}]_p,$$
$$\cdots$$
$$\cdots$$
$$\cdots \quad (8.a)$$
$$[b_1^{L(m)-1}]_p, ..., [b_1^1]_p, [b_1^0]_p,$$
$$[b_0^{L(m)-1}]_p, ..., [b_0^1]_p, [b_0^0]_p) \leftarrow BORROWS([X]_B, [Y]_B)$$

$$[t_0^0]_p = [x_0^0]_p - [y_0^0]_p + 2[b_0^0]_p \quad (8.b)$$

For $j = 1, ..., L(m) - 1$, in parallel: $[t_0^j]_p = [x_0^j]_p - [y_0^j]_p + 2[b_0^j]_p - [b_0^{j-1}]_p$.

For $i = 1, ..., l^{(m)} - 1$ do

$[t_i^0]_p = [x_i^0]_p - [y_i^0]_p + 2[b_i^0]_p - [b_{i-1}^{L(m)-1}]_p$

For $j = 1, ..., L(m) - 1$, in parallel: $[t_i^j]_p = [x_i^j]_p - [y_i^j]_p + 2[b_i^j]_p - [b_i^{j-1}]_p$.

End for (8.c)

$C \leftarrow 2^{L(m)} - m$ ▷ Note that C is public. (8.d)

For $i = 0, 1, ..., l^{(m)} - 1$ do

$[t_i]_B^m \leftarrow ([t_i^{L(m)-1}]_p, ..., [t_i^1]_p, [t_i^0]_p)$

If $m < 2^{L(m)}$ then ▷Recall that $m < 2^{L(m)}$ means m is not a power of 2.

$$[d_i]_B^m \leftarrow Bitwise - Subtraction^* \left([t_i]_B^m, \left([b_i^{L(m)-1}]_p ? C : 0\right)\right) \quad (8.e)$$

Else

$[d_i]_B^m \leftarrow [t_i]_B^m$

End if

End for (8.f)

$[x - y]_{D,B}^m = [d]_{D,B}^m \leftarrow ([d_{l^{(m)}-1}]_B^m, ..., [d_1]_B^m, [d_0]_B^m)$

Return $[x - y]_{D,B}^m$

Correctness is described in detail in [13]. Privacy follows readily from the fact that we only call private sub-protocols. The complexity of this protocol

is 30 rounds and $47l\log l + 47l\log(L(m))$ multiplications. The communication complexity is upper bounded by $94l\log l$ multiplications since $L(m) \leq l$.

•**A Simplified Version.** If we do not need $[x-y]^m_{D,B}$ but (only) need $[x-y]^m_D$ instead, a simplified version of the above protocol, *Digit-Bit-wise-Subtraction*$^{*-}$, can be obtained by simply replacing all the statements after *statement* (8.a) with the following.

$[d_0]^m_p = [x_0]^m_p - [y_0]^m_p + m[b_0^{L(m)-1}]_p$
For $i = 1, ..., l^{(m)} - 1$ in parallel: $[d_i]^m_p = [x_i]^m_p - [y_i]^m_p + m[b_i^{L(m)-1}]_p - [b_{i-1}^{L(m)-1}]_p$
$[x-y]^m_D = [d]^m_D \leftarrow ([d_{l^{(m)}-1}]^m_p, ..., [d_1]^m_p, [d_0]^m_p)$
Return $[x-y]^m_D$

Note that the above process is free. Correctness and privacy is straightforward. The complexity of this protocol goes down to 15 rounds and $47l\log l$ multiplications as the expensive *Bitwise-Subtraction** protocol is omitted.

If this (simplified) protocol is constructed from scratch, then, for relatively large m, the borrow bits for every digit-position, i.e. $[b_i^{L(m)-1}]_p$ for $i \in \{0, 1, ..., l^{(m)} - 1\}$, can be obtained with a lower cost. For every digit-position $i \in \{0, 1, ..., l^{(m)} - 1\}$, $e_i \in \{S, P, K\}$ can be obtained by calling the linear primitive *Bitwise-LessThan*. Specifically, we have

$$e_i = S \Leftrightarrow [x_i]^m_B < [y_i]^m_B;\ \ e_i = P \Leftrightarrow [x_i]^m_B = [y_i]^m_B;\ \ e_i = K \Leftrightarrow [x_i]^m_B > [y_i]^m_B.$$

So, using the *Bitwise-LessThan* protocol *in both ways*, which costs $l + \sqrt{l}$ more multiplications and no more rounds than one single invocation [18], we can get all the $e_i's$. Then as in the *BORROWS* protocol (or the *CARRIES* protocol), the target borrow bits (for every digit-position) can be obtained by using a generic prefix protocol which costs 15 rounds and $47l^{(m)}\log l^{(m)}$ multiplications. So the *Digit-Bit-wise-Subtraction*$^{*-}$ protocol can be realized in 6+15=21 rounds and (less than) $16l + 47l^{(m)}\log\left(l^{(m)}\right)$ multiplications. Recall that $l^{(m)} = \lceil \log_m p \rceil$. Then for relatively large m, e.g. $m \approx p^{\frac{1}{10}}$ where $l^{(m)} = 10$, the communication complexity may be very low.

7 Comments

As in [12], although we describe all our protocols in the secret sharing setting, our techniques are also applicable to the threshold homomorphic setting. All the protocols in our paper can be similarly realized in this setting. However, some of the protocols in this setting may be less efficient than their counterpart in the secret sharing setting because the *Random-Bit* protocol, which is a basic building block, is more expensive in the threshold homomorphic setting.

It is easy to see that using our *Base-m Digit-Decomposition* protocol which extracts all the base-m digits of the shared input, we can also solve the modulo reduction problem (which requires only the least significant base-m digit). However, our *Modulo-Reduction* protocol is meaningful because it achieves linear communication complexity and thus is much more efficient.

Obviously, we can say that the bit-decomposition protocol (of [7]) is a special case of our *Base-m Digit-Bit-Decomposition* protocol when m is a power of 2. In fact, we can also view the bit-decomposition protocol as a special case of our *enhanced Modulo-Reduction protocol* when the modulus m is just p, i.e. we have

$$[x]_B = Bit - Decomposition([x]_p) = Modulo - Reduction^{+}([x]_p, p)$$

for any $x \in \mathbb{Z}_p$. Our *enhanced Modulo-Reduction* protocol can handle not only the special case where $m = p$ but also the general case where $m \in \{2, 3, ..., p-1\}$, so it can also be viewed as a generalization to bit-decomposition.

We note that, in [18], a novel technique is proposed which can reduce the communication complexity of the bit-decomposition protocol to *almost linear*. We argue that their technique can also be used in our *Base-m Digit-Bit-Decomposition protocol* (as well as our *Base-m Digit-Decomposition protocol*) to reduce the (communication) complexity to almost linear, because their technique is in fact applicable to any $PreFix-\circ$ (or $PostFix-\circ$) protocol (which is a dominant factor of the communication complexity) assuming a linear protocol for computing the $UnboundedFanIn-\circ$ exists, which is just the case in our protocols.

8 Applications and Future Work

In [13], we will show some applications of our new protocols, such as efficient *Integer Division* protocol, *Divisibility Test* protocol, *Conversion of Integer Representation between Number Systems*, etc.

Although we are successful in providing an (efficient) solution to the *public* modulo reduction problem, we fail in solving the *private* modulo reduction problem where the modulus is (also) secret shared. The absence of the knowledge of the exact value of m makes our techniques useless. We leave it an open problem to construct efficient protocols for private modulo reduction without relying on bit-decomposition.

Acknowledgments. We would like to thank the anonymous reviewers for their careful work and helpful comments.

References

1. Algesheimer, J., Camenisch, J.L., Shoup, V.: Efficient Computation Modulo A Shared Secret with Application to the Generation of Shared Safe-Prime Products. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 417–432. Springer, Heidelberg (2002)
2. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness Theorems for Noncryptographic Fault-Tolerant Distributed Computations. In: 20th Annual ACM Symposium on Theory of Computing, pp. 1–10. ACM Press, New York (1988)
3. Cramer, R., Damgård, I.B., Nielsen, J.B.: Multiparty Computation from Threshold Homomorphic Encryption. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 280–300. Springer, Heidelberg (2001)
4. Chandra, A.K., Fortune, S., Lipton, R.J.: Lower Bounds for Constant Depth Circuits for Prefix Problems. In: Díaz, J. (ed.) ICALP 1983. LNCS, vol. 154, pp. 109–117. Springer, Heidelberg (1983)

5. Chandra, A.K., Fortune, S., Lipton, R.J.: Unbounded Fan-In Circuits and Associative Functions. In: 15th Annual ACM Symposium on Theory of Computing, pp. 52–60. ACM Press, New York (1983)
6. Catrina, O., Saxena, A.: Secure Computation with Fixed-Point Numbers. In: Sion, R. (ed.) FC 2010. LNCS, vol. 6052, pp. 35–50. Springer, Heidelberg (2010)
7. Damgård, I.B., Fitzi, M., Kiltz, E., Nielsen, J.B., Toft, T.: Unconditionally Secure Constant-Rounds Multi-Party Computation for Equality, Comparison, Bits and Exponentiation. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 285–304. Springer, Heidelberg (2006)
8. Damgård, I.B., Nielsen, J.B.: Universally Composable Efficient Multiparty Computation from Threshold Homomorphic Encryption. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 247–264. Springer, Heidelberg (2003)
9. Guajardo, J., Mennink, B., Schoenmakers, B.: Modulo Reduction for Paillier Encryptions and Application to Secure Statistical Analysis. In: Sion, R. (ed.) FC 2010. LNCS, vol. 6052, pp. 375–382. Springer, Heidelberg (2010)
10. Goldreich, O., Micali, S., Wigderson, A.: How to Play Any Mental Game or A Complete Theorem for Protocols with Honest Majority. In: 19th Annual ACM Symposium on Theory of Computing, pp. 218–229. ACM Press, New York (1987)
11. Gennaro, R., Rabin, M.O., Rabin, T.: Simplified Vss and Fast-Track Multiparty Computations with Applications to Threshold Cryptography. In: 17th ACM Symposium on Principles of Distributed Computing, pp. 101–110. ACM Press, New York (1998)
12. Nishide, T., Ohta, K.: Multiparty Computation for Interval, Equality, and Comparison without Bit-Decomposition Protocol. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 343–360. Springer, Heidelberg (2007)
13. Ning, C., Xu, Q.: Multiparty Computation for Modulo Reduction without Bit-Decomposition and A Generalization to Bit-Decomposition. Cryptology ePrint Archive, Report 2010/266, http://eprint.iacr.org/2010/266
14. Reistad, T., Toft, T.: Linear, Constant-Rounds Bit-Decomposition. In: Lee, D., Hong, S. (eds.) ICISC 2009. LNCS, vol. 5984, pp. 245–257. Springer, Heidelberg (2010)
15. Shamir, A.: How to Share A Secret. Communications of the ACM 22(11), 612–613 (1979)
16. Schoenmakers, B., Tuyls, P.: Efficient Binary Conversion for Paillier Encrypted Values. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 522–537. Springer, Heidelberg (2006)
17. Toft, T.: Primitives and Applications for Multi-party Computation. PhD thesis, University of Aarhus (2007), http://www.daimi.au.dk/~ttoft/publications/dissertation.pdf
18. Toft, T.: Constant-Rounds, Almost-Linear Bit-Decomposition of Secret Shared Values. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 357–371. Springer, Heidelberg (2009)

A Closer Look at Anonymity and Robustness in Encryption Schemes

Payman Mohassel

Computer Science Department, University of Calgary
pmohasse@cpsc.ucalgary.ca

Abstract. In this work, we take a closer look at *anonymity* and *robustness* in encryption schemes. Roughly speaking, an anonymous encryption scheme hides the identity of the secret-key holder, while a robust encryption scheme guarantees that every ciphertext can only be decrypted to a valid plaintext under the intended recipient's secret key.

In case of anonymous encryption, we show that if an anonymous PKE or IBE scheme (in presence of CCA attacks) is used in a hybrid encryption, all bets regarding the anonymity of the resulting encryption are off. We show that this is the case even if the symmetric-key component is anonymous. On the positive side, however, we prove that if the key-encapsulation method is, additionally *weakly robust* the resulting hybrid encryption remains anonymous. Some of the existing anonymous encryption schemes are known to be weakly robust which makes them more desirable in practice.

In case of robust encryption, we design several *efficient* constructions for transforming any PKE/IBE scheme into weakly and strongly robust ones. Our constructions only add a minor computational overhead to the original schemes, while achieving better ciphertext sizes compared to the previous constructions. An important property of our transformations is that they are non-keyed and do not require any modifications to the public parameters of the original schemes.

We also introduce a relaxation of the notion of robustness we call *collision-freeness*. We primarily use collision-freeness as an *intermediate notion* by showing a more efficient construction for transforming any collision-free encryption scheme into a strongly robust one. We believe that this simple notion can be a plausible replacement for robustness in some scenarios in practice. The advantage is that most existing schemes seem to satisfy collision-freeness without any modifications.

1 Introduction

The classical definitions of security for encryption schemes are mainly concerned with the secrecy of encrypted data. Particularly, the widely accepted notions of indistinguishability and non-malleability under chosen plaintext and ciphertext attacks [15,19,12], are all directed at capturing various aspects of data-secrecy in encryption schemes. However, since encryption schemes are employed in a wide range of applications, one often requires them to satisfy additional properties.

M. Abe (Ed.): ASIACRYPT 2010, LNCS 6477, pp. 501–518, 2010.

Two such properties, which have been the subject of formal studies in the cryptographic literature, are *anonymity* [5,2] and *robustness* [3]. Anonymity helps keep the identity of the key-holders in an encryption scheme private, while robustness provides a layer of protection against misuse or error by ensuring that a single ciphertext can only be decrypted by the intended user. In this paper we study several aspects of anonymity and robustness in public-key and identity-based encryption schemes.

1.1 Anonymity of Hybrid Encryption Schemes

The concept of anonymity for encryption schemes has been around for sometime but was first formalized in the context of symmetric-key encryption [1,11,14] and was later extended to the case of public-key encryption (PKE) and identity-based encryption (IBE) [5,2]. Several PKE and IBE schemes in the literature such as the Cramer-Shoup [10], and the Boyen-Waters [9] in the standard model, and DHIES [4] and Boneh-Franklin [8] in the random oracle model are shown to be anonymous.

However, in most cases, PKE and IBE schemes are used as key encapsulation methods (KEM) to encrypt a random key which is then used by a symmetric-key data encapsulation method (DEM) to encrypt the message itself. It is well known that if the KEM component is IND-CCA secure and the DEM component is (one-time) IND-CCA, the resulting hybrid encryption is also IND-CCA secure (e.g. see [10]).[1] From a practical point of view, it is important to determine whether similar statements can be made when considering anonymity.

A NEGATIVE RESULT. At first glance, it seems that the symmetric-key component is harmless as far as anonymity is concerned since it only encrypts a message using a random secret key, which is unlikely to reveal additional information about the public key or the identity (this is in fact the case for CPA attacks). However, somewhat surprisingly, we show that this intuition is wrong in presence of *chosen ciphertext attacks*. Particularly, we show a counterexample by building an anonymous-CCA (ANON-CCA) secure PKE/IBE scheme and a symmetric-key IND-CCA encryption, where it is easy to break the anonymity of the resulting hybrid construction. The negative result extends to the case when the symmetric-key component is also anonymous. An important implication is that:

Designing ANON-CCA PKE or IBE schemes is not sufficient for providing anonymity in practice where, more often than not, encryption schemes are used in hybrid constructions.

A POSITIVE RESULT. On the positive side, we show that if one further assumes that the KEM component is *weakly-robust* (see Section 2 for the definition), the resulting hybrid encryption is in fact ANON-CCA. This implies that despite our negative result, for most ANON-CCA schemes we know of such as the Boneh-Franklin IBE, the Cramer-Shoup PKE, and the DHIES PKE all of which are

[1] Note that the KEM/DEM framework is more general than hybrid encryption but here we are focus on the KEM/DEM framework in the context of hybrid encryption schemes.

known to be weakly-robust [3] (in the appropriate model), using them as part of a hybrid construction preserves their anonymity. The same is however not true for the Boyen-Waters anonymous IBE scheme which is shown not to be weakly robust.

This result reemphasizes the close connection between anonymity and robustness and provides additional motivation to study the robustness property when designing anonymous encryption schemes.

1.2 Robustness

Informally speaking, *weak* robustness requires that a ciphertext does not decrypt to a valid plaintext under distinct secret keys for two different identities. A *stronger* version of robustness requires this to be the case even for adversarially chosen ciphertexts. The concept of robustness was studied in one way or another in [18] and [17], but was only recently formalized by Abdalla *et al.* [3].

It is not hard to see that robustness can be trivially achieved by appending the encryption key to the ciphertext and checking for it upon decryption. The main drawback is that the resulting scheme is no longer anonymous. In fact, as discussed in [3] and further motivated by our results on anonymity of hybrid encryptions, it is exactly for anonymous schemes that robustness is important. In [3], the authors study the robustness properties for several existing anonymous encryption schemes, and design general constructions for transforming any IBE/PKE scheme into robust ones.

A transformation is *keyed* if an additional string needs to be added to the set of public parameters for the original scheme, and is called *non-keyed*, otherwise. An important advantage of non-keyed constructions over keyed ones is that the robustness property can be added to the encryption scheme without having to notify a third party such as a PKI in advance. Consequently, users of a system can add robustness to the scheme after it is deployed.

Non-keyed transformations for robustness. In the standard model, we design a non-keyed construction for transforming any anonymous IBE/PKE scheme into a weakly robust one in presence of CPA attacks. In the random oracle model, we design a non-keyed transformation that provides strong robustness in presence of CCA attacks. In both cases, the computational overhead is very small (it involves one to three invocations of a hash function), and despite being non-keyed the ciphertext sizes we achieve are better than those of the previous work. A curious open question is whether we can achieve the latter transformation in the standard model.

Collision-freeness. We also study the notion of *collision-freeness*, a natural relaxation of robustness. Roughly speaking, an encryption scheme is collision-free if a ciphertext does not decrypt to the same message under two different decryption keys. Collision-freeness can be a sufficient property in some scenarios in practice. For example, if the receiver expects to see a specific message as part of the protocol but after decrypting using his secret key recovers a different one, he can detect an error and stop the communication. Interestingly, we show that schemes such as the El Gamal PKE scheme [13] and the Boyen-Waters

IBE scheme [9] are strongly collision-free even though they are known not to be weakly robust. Hence, collision-freeness seems to be a less restrictive assumption on an encryption scheme and one that most encryption schemes seem to satisfy without any modifications. More importantly, we design a more efficient construction for transforming any collision-free encryption scheme to a strongly robust one.

2 Preliminaries

ONE-WAY FUNCTIONS. Roughly speaking, a function is one-way if it is hard to invert on a random input. More formally, we say that a function f over $\{0,1\}^k$ is *one-way* if

$$\mathbf{Adv}^{\text{owf}}_{f,\mathsf{A}}(k) \;=\; \Pr\left[\, x \xleftarrow{\$} \{0,1\}^k \,;\, y \leftarrow f(x) \,;\, x' \xleftarrow{\$} \mathsf{A}(f,y) \;:\; x = x' \,\right]$$

is negligible for every PPT inverter A.

GENERAL ENCRYPTION SCHEMES. Abdalla *et al.* [3] introduced and used the notion of general encryption schemes which encompass both PKE and IBE schemes. Similar to their work we will use this notion, since all our transformations are applicable to both PKE and IBE schemes.

A general encryption (GE) scheme consists of a tuple $GE = (\mathsf{Pg}, \mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$ of algorithms. The parameters generation algorithm Pg takes no input and returns common parameters $pars$ and a master secret key msk. On input $pars, msk, id$, the key generation algorithm Kg produces an encryption key ek and the decryption key dk. On inputs $pars, ek, M$ the encryption algorithm Enc produces a ciphertext C encrypting plaintext M. On input $pars, ek, dk, C$, the deterministic decryption algorithm Dec returns either a plaintext M or $\perp$ to indicate that it rejects. GE is a PKE scheme if $msk = \epsilon$ and Kg ignores its id input. GE is an IBE scheme if $ek = id$, meaning the encryption key generated by Kg on inputs $pars, msk, id$ is always id. Finally, we point out that the notion of general encryption contains PKE schemes, IBE schemes and more. In other words, there are general encryption schemes that are neither PKE nor IBE schemes.

AI-{CPA,CCA} SECURITY. Traditionally, the definitions of privacy [15,19,12] and anonymity [5,2] for encryption schemes are introduced separately. However, when considering robustness, it makes sense to consider both notions simultaneously. Hence we follow the definition of [3] who combine the two into a single game. We define the AI-{CPA,CCA} security (AI = ANON + IND) of a general encryption scheme $GE = (\mathsf{Pg}, \mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$ via a security game between the adversary and the challenger.

- **Setup:** Challenger runs $(pars, msk) \leftarrow \mathsf{Pg}(1^k)$; $b \xleftarrow{\$} \{0,1\}$; $S, T, U, V \leftarrow \emptyset$.
- **Queries:**
  - Public key query id. Challenger lets $U \leftarrow U \cup \{id\}$; $(Ek[id], Dk[id]) \xleftarrow{\$} \mathsf{Kg}(pars, msk, id)$ and returns $Ek[id]$.

- Decryption-key query id. If $id \notin U$ or $id \in S$ return $\perp$. Else $V \leftarrow V \cup \{id\}$ and return $Dk[id]$.
- Decryption query (C, id). If $id \notin U$ or $(id, C) \in T$ return $\perp$. Else let $M \leftarrow \mathsf{Dec}(pars, Ek[id], Dk[id], C)$, and return M.
- Challenge query $(id_0^*, id_1^*, M_0^*, M_1^*)$. If $id_0^* \notin U$ or $id_1^* \notin U$ or $id_0^* \in V$, or $id_1^* \in V$ return $\perp$. Else let $C^* \xleftarrow{\$} \mathsf{Enc}(pars, Ek[id_b], M_b^*)$; $S \leftarrow S \cup \{id_0^*, id_1^*\}$; $T \leftarrow T \cup \{(id_0^*, C^*), (id_1^*, C^*)\}$ and return C^*.

– **Adversary's guess.** Adversary returns a bit b'.

Note that there is only one challenge query. In case of CPA attacks, no decryption queries are allowed. Adversary A's advantage in the AI-{CPA,CCA} game is:

$$\mathbf{Adv}_{GE}^{\text{ai-\{cpa,cca\}}}(\mathsf{A}) = \Pr[b' = b] - 1/2$$

In some cases however, we consider the security notions for anonymity (ANON-{CPA,CCA}) and indistinguishability (IND-{CPA,CCA}), individually. The challenge query in the above security game can be modified in the obvious way to capture each of these definitions separately. We point out that similar definitions can also be adapted for the case of symmetric-key encryption.

Robustness. Following [3], we consider two definitions of robustness for a general encryption scheme, namely weak robustness (WROB) and strong robustness (SROB). The following game defines both notions. As noted below the only difference is in the final message sent by the adversary to the challenger:

– **Setup:** Challenger runs $(pars, msk) \leftarrow \mathsf{Pg}(1^k)$; $b \xleftarrow{\$} \{0,1\}$; $U, V \leftarrow \emptyset$.
– **Queries:**
  - Public key query id. Challenger lets $U \leftarrow U \cup \{id\}$; $(Ek[id], Dk[id]) \xleftarrow{\$} \mathsf{Kg}(pars, msk, id)$ and returns $Ek[id]$.
  - Decryption-key query id. If $id \notin U$ or $id \in S$ return $\perp$. Else $V \leftarrow V \cup \{id\}$ and return $Dk[id]$.
  - Decryption query (C, id). If $id \notin U$ return $\perp$. Else let $M \leftarrow \mathsf{Dec}(pars, Ek[id], Dk[id], C)$, and return M.
  - Final message (id_0^*, id_1^*, M) (for WROB). If $id_0 = id_1$ or $id_0^* \notin U$ or $id_1^* \notin U$ or $id_0^* \in V$, or $id_1^* \in V$ return 0. Else let $C^* \xleftarrow{\$} \mathsf{Enc}(pars, Ek[id_0], M)$; $M' \leftarrow \mathsf{Dec}(pars, Ek[id_1], Dk[id_1], C^*)$; if $M' \neq \perp$ return 1, else return 0.
  - Final message (id_0^*, id_1^*, C) (for SROB). If $id_0 = id_1$ or $id_0^* \notin U$ or $id_1^* \notin U$ or $id_0^* \in V$, or $id_1^* \in V$ return 0. Else let $M_0 \leftarrow \mathsf{Dec}(pars, Ek[id_0], Dk[id_0], C)$; $M_1 \leftarrow \mathsf{Dec}(pars, Ek[id_1], Dk[id_1], C)$; if $M_0 \neq \perp$ and $M_1 \neq \perp$ return 1, else return 0.

Similar to above, in case of CPA attacks, no decryption queries are allowed. Adversary A's advantage in the {WROB,SROB}-{CPA,CCA} game is:

$$\mathbf{Adv}_{GE}^{\text{\{wrob,srob\}-\{cpa,cca\}}}(\mathsf{A}) = \Pr[G^{\mathsf{A}} \to 1]$$

In the WROB game the adversary produces a message M, and C is its encryption under the encryption key of one of the given identities, while in the SROB

game adversary produces C directly, and may not obtain it as an honest encryption. Note that in case of PKE schemes, the adversary does not get to choose the encryption keys of the identities it is targeting. Those are honestly and independently chosen by the identities themselves in real life and and by the games in the above formalizations.

3 Anonymous-CCA Hybrid Encryption

In this section we take a closer look at anonymous encryption schemes in presence of *chosen ciphertext attacks* (ANON-CCA) as defined in Section 2. Previous works on anonymous public-key and identity-based encryption [5,2] have studied this security notion and provided constructions satisfying it.

However, in most scenarios in practice, PKE and IBE schemes are used in the KEM/DEM paradigm. It is known that if the KEM component is IND-CCA secure and the DEM component is (one-time) IND-CCA, the resulting hybrid encryption is also IND-CCA secure. For practical reasons, it is crucial to determine whether we can make similar statements when considering the anonymity of the resulting hybrid construction. More specifically, we try to answer the following question:

Given an ANON-CCA PKE or IBE scheme and an (ANON-CCA + IND-CCA) symmetric-key encryption scheme, is the resulting hybrid encryption scheme ANON-CCA?

3.1 A Negative Result

Somewhat surprisingly, we answer the above question in the negative. First we show a counterexample by building an ANON-CCA secure PKE/IBE scheme and a symmetric-key IND-CCA encryption, where it is easy to break the anonymity of the resulting hybrid construction. The negative result easily extends to the case when the symmetric-key component is also ANON-CCA. An important implication is that *designing ANON-CCA PKE or IBE schemes is not sufficient for providing anonymity in practice where, more often than not, encryption schemes are used in hybrid constructions.*

Claim 3.1. *There exist an ANON-CCA PKE/IBE scheme and a symmetric-key authenticated encryption scheme (assuming there are secure schemes at all) such that the resulting hybrid encryption is not ANON-CCA.*

The intuition behind the counterexample is that since the adversary has access to a decryption oracle, he can take advantage of the fact that decrypting one ciphertext under two different secret keys can result in different answers. Particularly, these different answers can be used by the adversary to compromise the anonymity of the scheme.

Proof. We describe the proof for the case of a PKE scheme, but an identical proof works for IBE schemes as well. Let $\mathsf{PKE}_1 = (\mathsf{Kg}_1, \mathsf{Enc}_1, \mathsf{Dec}_1)$ be an (ANON-CCA + WROB-CCA) PKE encryption scheme. The Cramer-Shoup encryption

scheme or any of the constructions in this paper will do. We build the encryption scheme $\mathsf{PKE}_2 = (\mathsf{Kg}_2, \mathsf{Enc}_2, \mathsf{Dec}_2)$ by letting the key-generation and encryption algorithms be identical to those of PKE_1, and modifying the decryption algorithm such that whenever the Dec_1 algorithm returns the symbol $\perp$, the decryption algorithm Dec_2 returns 0^n instead, and otherwise works similar to Dec_1. It is easy to verify that after this simple modification, PKE_2 remains ANON-CCA. PKE_2 will be the key encapsulation method in our counterexample.

For the DEM component we use an IND-CCA encryption scheme that is also *key-binding*, a notion introduced in [14].

Definition 1. *A symmetric-key encryption scheme $\mathcal{E} = (\mathcal{SK}, \mathcal{SE}, \mathcal{SD})$ is called key-binding if for any key k generated by $\mathcal{SK}$, any message m, and randomness r, there does not exist a key k' such that $k' \neq k$ and $\mathcal{SD}_{k'}(\mathcal{SE}_k(m, r)) \neq \perp$.*

The key-binding property guarantees that a ciphertext created using one secret key, does not decrypt correctly under any other secret key. Fischlin [14] showed simple constructions of such encryption schemes from any PRF. For the purpose of our counterexample it suffices to know that an IND-CCA encryption scheme $\mathcal{E}$ with such a property exists.

Now, we show that combining PKE_2 and $\mathcal{E}$ into a hybrid encryption is not ANON-CCA. Particularly, an attacker with the following simple strategy can break the anonymity of the scheme.

Recall the ANON-CCA security game. Attacker A initially sends a message m as his challenge in the ANON-CCA game and receives the ciphertext $C = (c_1, c_2) = (\mathsf{Enc}(pk_{id_b}, k), \mathcal{SE}_k(m))$ for a random bit $b \in \{0, 1\}$ and a random key $k \in \{0, 1\}^n$. Then, A makes a decryption query for the ciphertext $(c_1, \mathcal{SE}_{0^n}(m'))$ under public key pk_{id_0}, for an arbitrary message m'. If the answer is $\perp$, A outputs 0 and else outputs 1.

To see why A breaks the ANON-CCA security of the encryption scheme note that if $b = 1$ then $k' = \mathsf{Dec}_2(sk_{id_1}, \mathsf{Enc}_2(pk_{id_0}, k)) = 0^n$ given the way we have defined PKE_2. Hence, we have that $\mathcal{SD}_{0^n}(\mathcal{SE}_{0^n}(m')) = m' \neq \perp$. On the other hand if $b = 0$ then $k' = \mathsf{Dec}_2(sk_{id_0}, \mathsf{Enc}_2(pk_{id_0}, k)) = k$. Hence we have $\mathcal{SD}_k(\mathcal{SE}_{0^n}(m')) = \perp$ due to the key-binding property of $\mathcal{E}$ and the fact that $k \neq 0^n$ with all but negligible probability. Therefore, A guesses the bit b correctly with high probability.

A closer look at the above attack strategy reveals that a much weaker property than that of definition 1 for the symmetric-key scheme suffices for our argument to go through. In particular, we only need the *key binding* property to hold for a fixed message and a fixed secret key (m' and 0^n, respectively).

Strengthening the DEM component? One potential solution is to use a symmetric-key encryption scheme that possesses some additional properties. Particularly, one natural question is whether using an anonymous-CCA symmetric-key encryption as the DEM component would yield an anonymous hybrid construction. Unfortunately, the answer to this question is also negative. It is easy to verify that the above negative result extends to work for any security notion considered for symmetric-key encryption, as long as that security notion can be achieved in

conjunction with the *key-binding* property. In all such cases, the proof given above works without any significant changes.

Anonymity of symmetric-key encryption schemes has been studied under the name *key-hiding* in [14] where the authors also design IND-CCA secure symmetric-key encryption schemes that are simultaneously key-hiding and key-binding. This leads to the following claim:

Claim 3.2. *There exist an ANON-CCA PKE/IBE scheme and an (ANON-CCA + IND-CCA) symmetric-key encryption scheme such that the resulting hybrid encryption is not ANON-CCA.*

3.2 A Positive Result

In light of the above negative results, it is natural to ask what additional property the KEM component should have in order to preserve its ANON-CCA security in a hybrid construction. We show that if one further assumes that the KEM component is *weakly-robust*, the resulting hybrid encryption is in fact ANON-CCA. This implies that despite the negative results we gave above, for most ANON-CCA schemes we know such as the Boneh-Franklin IBE, the Cramer-Shoup PKE, and the DHIES PKE all of which are known to be weakly-robust [3], using them as part of a hybrid construction is safe. The intuition behind the proof is that weak robustness ensures that the decryption algorithm behaves in a *predictable* way, when decrypting a ciphertext under two different secret keys, and this predictable behavior combines quite nicely with the security properties of an authenticated symmetric encryption scheme, namely, IND-CCA security and the ciphertext integrity (CTXT-INT).

In the following claim we prove a stronger result than what we need here by considering the notion of AI-CCA security which combines ANON-CCA security and IND-CCA security into one definition. The main reason is that we need this stronger claim in a following section. The proof for the case when one is only interested in ANON-CCA secure hybrid schemes is identical.

Claim 3.3. *If the KEM component* PKE *of a hybrid construction is an (AI-CCA + WROB-CCA) general encryption, and* $\mathcal{E}$ *is a one-time authenticated symmetric encryption, then the resulting hybrid encryption* PKE' *is also an AI-CCA general encryption scheme.*

Proof. We prove the above claim via a sequence of games.

Game 0. Game 0 is simply the AI-CCA game. Denote by b the random bit generated by the challenger, by C^* the challenge ciphertext $C^* = (c_1^*, c_2^*)$ where c_1^* is the KEM component and c_2^* is the DEM component, and by k^* the secret key used for the DEM component.

Game 1. Game 1 is similar to game 0, except that for any decryption queries of the form (c_1, c_2) for pk_{id_b} where $c_1 = c_1^*$ and $c_2 \neq c_2^*$, challenger uses k^* to decrypt c_2 and recover the message (as opposed to decrypting c_1).

It is easy to see that the difference between the advantage of any adversary in these two games is bounded by the decryption error. For simplicity we assume that there is no decryption error and therefore

$$\mathbf{Adv}_{G_0}(\mathsf{A}) = \mathbf{Adv}_{G_1}(\mathsf{A})$$

Game 2. Similar to game 1 except that for any decryption queries of the form (c_1, c_2) for $pk_{id_{1-b}}$ where $c_1 = c_1^*$ and $c_2 \neq c_2^*$), challenger returns $\perp$.

Note that games 1 and 2 are different only when c_1^* which is the encryption of message m_b under pk_{id_b}, also decrypts correctly under the $pk_{id_{1-b}}$. This probability is bounded by the advantage of an adversary B in winning the WROB-CCA game and hence:

$$\mathbf{Adv}_{G_2}(\mathsf{A}) - \mathbf{Adv}_{G_1}(\mathsf{A}) \leq \mathbf{Adv}^{\mathrm{wrob-cca}}_{\mathsf{PKE}}(\mathsf{B})$$

Game 3. Similar to game 2 except that the challenger generates and uses a random key k' (instead of k^*) when encrypting the private-key component of the ciphertext for the challenge query.

The difference between the advantages of an adversary in games 2 and 3 is bounded by the AI-CCA security of the PKE scheme:

$$\mathbf{Adv}_{G_3}(\mathsf{A}) - \mathbf{Adv}_{G_2}(\mathsf{A}) \leq \mathbf{Adv}^{\mathrm{ai-cca}}_{\mathsf{PKE}}(\mathsf{B}')$$

Game 4. We modify game 3 in two ways. First, for the challenge query, instead of encrypting the message m_b, the challenger encrypts the constant message 0^k. Second, for decryption queries (c_1, c_2) under pk_{id_b} where $c_1 = c_1*$ the challenger returns $\perp$.

The probability of distinguishing the first change is bounded by the IND-CCA advantage of an adversary against the $\mathcal{E}$ scheme, while for second change, the probability is bounded by the advantage of an adversary playing the ciphertext integrity (CTXT-INT) game with $\mathcal{E}$. Both the IND-CCA security and the CTXT-INT security are properties that are possessed by any authenticated encryption scheme.

$$\mathbf{Adv}_{G_4}(\mathsf{A}) - \mathbf{Adv}_{G_3}(\mathsf{A}) \leq \mathbf{Adv}^{\mathrm{ind-cca}}_{\mathcal{E}}(\mathsf{B}'') + \mathbf{Adv}^{\mathrm{ctxt-int}}_{\mathcal{E}}(\mathsf{B}''')$$

Finally, it is easy to see that the adversary's view in game 4 is independent of the bit b and hence adversary's advantage in guessing b is exactly $1/2$. Putting things together we have:

$$\mathbf{Adv}^{\mathrm{ai-cca}}_{\mathsf{PKE}'}(\mathsf{A}) \leq \mathbf{Adv}^{\mathrm{wrob-cca}}_{\mathsf{PKE}}(\mathsf{B}) + \mathbf{Adv}^{\mathrm{ai-cca}}_{\mathsf{PKE}}(\mathsf{B}') + \mathbf{Adv}^{\mathrm{ind-cca}}_{\mathcal{E}}(\mathsf{B}'') + \mathbf{Adv}^{\mathrm{ctxt-int}}_{\mathcal{E}}(\mathsf{B}''')$$

4 Non-keyed Transformations for Robustness

Having further motivated the study of robust encryption schemes, we next focus on efficient ways of transforming general encryption schemes into robust ones. As mentioned earlier, such a transformation is called a keyed transformation if

an additional string is added to the original set of public parameters, and is called non-keyed otherwise.

In Section 4.1, we design an efficient and non-keyed transformation for weak-robustness, in presence of CPA attacks (in the standard model). In Section 4.2, we design a non-keyed transformation for strong-robustness in presence of CCA attacks (in the random oracle model). Despite being non-keyed, our transformations have better ciphertext sizes compared to previous work. In other words, not adding an extra string to the public parameters does translate to larger ciphertexts (see the efficiency comparison sections).

4.1 A Transformation for AI-CPA Schemes

The following non-keyed construction takes any AI-CPA encryption scheme, and transforms it to a (AI-CPA + WROB-CPA) scheme.

Construction 4.1. Let $\mathsf{PKE} = (\mathsf{Pg}, \mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$ be a AI-CPA general encryption scheme, and let f be a one-way function over $\{0,1\}^k$. We construct the general encryption scheme $\mathsf{PKE}' = (\mathsf{Pg}', \mathsf{Kg}', \mathsf{Enc}', \mathsf{Dec}')$:

- **Parameter Generation**(Pg'): On input 1^k return $(pars, msk) \xleftarrow{\$} \mathsf{Pg}(1^k)$.
- **Key Generation**(Kg'): On input $pars, msk, id$, return $(pk_{id}, sk_{id}) \xleftarrow{\$} \mathsf{Kg}(pars, msk, id)$.
- **Encryption**(Enc'): On input $pars, pk_{id}, m$, generate a random $r \in \{0,1\}^k$ and return $(\mathsf{Enc}(pars, pk_{id}, m||r), f(r))$.
- **Decryption**(Dec'): On inputs $pars, pk_{id}, sk_{id}, (c_1, c_2)$, compute $m'||r' \xleftarrow{\$} \mathsf{Dec}(pars, pk_{id}, sk_{id}, c_1)$. If $r' \neq \perp$ and $f(r') = c_2$ return m'; else return $\perp$.

Note that in construction 4.1, instead of a one-way function, we can also use a target collision-resistant (TCR) hash function (a universal one-way hash function). Particularly, it is easy to show that any TCR function that is sufficiently compressing is a good one-way function.

We will shortly prove the security of the above scheme, but first lets briefly study its efficiency.

Efficiency comparison. To implement our scheme one can use a fixed-length cryptographic hash function h with output length of 128 bits (e.g. constructed by suitably modifying the output length of a hash function from the SHA family). The reason that we only need 128 bits of output is that we only require the hash function to be one-way as opposed to collision-resistant. Furthermore, it is sufficient for us to let $k = 256$ where r is chosen from $\{0,1\}^k$.[2] This means that the PKE scheme has to encrypt a message that is only 256 bits longer than the original message and the ciphertext is at most expanded by an additive factor of 384 bits as opposed to 768 bits in construction of Abdalla *et al.* [3].

[2] When computing hash of r, we can pad r with enough 0's in order to match the input block-size requirement for the hash function. Note that this does not effect the efficiency of the encryption or the size of ciphertext in any way.

Theorem 1. *Let* PKE *be a AI-CPA secure general encryption scheme and* f *be a one-way function. Then, the* PKE′ *scheme of construction 4.1 is both AI-CPA secure and WROB-CPA secure.*

Proof. We prove the theorem in two separate claims. Claim 4.2 ensures that the above transformation preserves the AI-CPA security of the original scheme. Claim 4.3 states that the resulting scheme PKE′ is also weakly robust.

Claim 4.2. *For any PPT adversary* A *against* PKE′*, there exist a PPT adversary* B *against* PKE *such that:*

$$\mathbf{Adv}^{\text{ai-cpa}}_{\mathsf{PKE}'}(A) = \mathbf{Adv}^{\text{ai-cpa}}_{\mathsf{PKE}}(B)$$

Proof. B runs A. When A sends its challenge request (id_0, id_1, M_0, M_1), B generates a random value $r \in \{0,1\}^k$ and sends $(id_0, id_1, M_0||r, M_1||r)$ to its own challenger in the AI-CPA game for PKE. B receives back $c^* = \mathsf{Enc}(pars, id_b, pk_{id_b}, M_b||r)$ and sends $(c^*, f(r))$ to A. The decryption-key queries made by A are forwarded to the corresponding oracle in B's game. Since we only consider CPA attacks, no decryption queries on id_0 or id_1 are allowed. Eventually, A outputs a bit b'. B also outputs b' and halts. It is straightforward to see that the advantage of B against the PKE is the same as A's advantage against the PKE′ scheme.

Claim 4.3. *For any PPT adversary* A *against the* PKE′ *in the WROB-CPA game, there exist PPT adversaries* B_1 *against the* PKE *in the AI-CPA game and* B_2 *against* f *in the one-wayness game such that:*

$$\mathbf{Adv}^{\text{wrob-cpa}}_{\mathsf{PKE}'}(A) = 2\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{PKE}}(\mathsf{B}_1) + \mathbf{Adv}^{\text{owf}}_{f}(\mathsf{B}_2)$$

Proof. We prove this claim in a sequence of two games.

Game 0. Game 0 is the WROB-CPA game against the PKE′ scheme as defined earlier. More specifically, adversary sends the tuple (id_0, id_1, M) to the challenger. Challenger computes $C_0 = \mathsf{Enc}'(pars, id_0, pk_{id_0}, M) = (\mathsf{Enc}(pars, pk_{id_0}, M||r), f(r))$ for random $r \in \{0,1\}^k$. He then computes $M_1 = \mathsf{Dec}'(pars, pk_{id_1}, sk_{id_1}, C_0)$. If $M_1 \neq \perp$, the challenger outputs 1. Else it outputs 0.

Game 1. Game 1 is similar to game 0, except that C_0 is computed in the following way:

$$C_0 = (\mathsf{Enc}(pars, pk_{id_0}, M||0^k), f(r))$$

The rest of the game stays the same.

First we show that there exist an adversary B_1 such that $\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{PKE}}(\mathsf{B}_1) = 1/2(\Pr[G_1^A \to 1] - \Pr[G_0^A \to 1])$. B_1 runs A and receives the tuple (id_0, id_1, M) from her. B_1 queries the key oracle for (pk_{id_1}, sk_{id_1}). He then generates a random $r \in \{0,1\}^k$ and sends $(id_0, m'_0 = M||r, m'_1 = M||0^k)$ to the challenger in the IND-CPA game against PKE and receives $C_0 = \mathsf{Enc}(pars, id_0, pk_{id_0}, m'_b)$ for a random bit b. B_1 then decrypts $(C_0, f(r))$ using the Dec' algorithm and the secret key

sk_{id_1}. If the result of decryption is not $\perp$, B_1 lets $b' = 1$ and else $b' = 0$. Then we have

$$\begin{aligned}
&\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{PKE}}(\mathsf{B}_1) = \Pr[b' = b] - 1/2 = \\
&\Pr[b = 1] \cdot \Pr[\mathsf{B}_1^{\text{ind-cpa}} \to 1 | b = 1] + \Pr[b = 0] \cdot \Pr[\mathsf{B}_1^{\text{ind-cpa}} \to 0 | b = 0] = \\
&1/2 \Pr[\mathsf{B}_1^{\text{ind-cpa}} \to 1 | b = 1] + 1/2 \Pr[\mathsf{B}_1^{\text{ind-cpa}} \to 0 | b = 0] - 1/2 = \\
&1/2 \Pr[G_1^{\mathsf{A}} \to 1] + 1/2(1 - \Pr[G_0^{\mathsf{A}} \to 1]) - 1/2 = \\
&1/2(\Pr[G_1^{\mathsf{A}} \to 1] - \Pr[G_0^{\mathsf{A}} \to 1])
\end{aligned} \tag{1}$$

We now show that there exist an adversary B_2 such that $\Pr[G_1^{\mathsf{A}} \to 1] = \mathbf{Adv}^{\text{owf}}_f(\mathsf{B}_2)$. B_2 generates the $(pars, msk)$ for the general encryption, and runs A. When B_2 receives the tuple (id_0, id_1, M) he computes (pk_{id_0}, sk_{id_0}), (pk_{id_1}, sk_{id_1}) and $C_0 = \mathsf{Enc}(pars, pk_{id_0}, M_0||0^k)$. He then requests his challenge for the one-wayness game and receives $f(r)$ for a random r. B_2 then decrypts using $(C_0, f(r))$ using the Dec' algorithm and the secret key sk_{id_1}. If the result is $\perp$ it outputs fail and halts. Else, it parses the decrypted plaintext into $M'||r'$ and returns r' to his own challenger.

B_2 wins the one-wayness game if $f(r') = f(r)$. Note that according to the definition of Dec', whenever the decryption algorithm does not output $\perp$ we have $f(r') = f(r)$. Hence

$$\mathbf{Adv}^{\text{owf}}_f(\mathsf{B}_2) = 1 - \Pr[\mathsf{B}_{2f}^{\text{owf}} \to \mathsf{fail}] = 1 - \Pr[{G_1}^{\mathsf{A}} \to 0] = \Pr[{G_1}^{\mathsf{A}} \to 1]$$

Putting things together we have:

$$\begin{aligned}
&\mathbf{Adv}^{\text{wrob-cpa}}_{\mathsf{PKE}'}(\mathsf{A}) = \Pr[G_0^{\mathsf{A}} \to 1] = \\
&\Pr[G_0^{\mathsf{A}} \to 1] - \Pr[G_1^{\mathsf{A}} \to 1] + \Pr[G_1^{\mathsf{A}} \to 1] = \\
&2\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{PKE}}(\mathsf{B}_1) + \mathbf{Adv}^{\text{owf}}_f(\mathsf{B}_2)
\end{aligned}$$

4.2 A Transformation for AI-CCA Schemes

Unfortunately, the transformation we gave above does not work in case of AI-CCA encryption schemes. Nevertheless, we are able to design an efficient and non-keyed transformation for any AI-CCA encryption scheme, in the random oracle model. The construction follows:

Construction 4.4. Let $\mathsf{PKE} = (\mathsf{Pg}, \mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$ be an AI-CCA general encryption scheme, and let $G, H, H' : \{0,1\}^* \to \{0,1\}^k$ be three hash functions. We construct the general encryption scheme $\mathsf{PKE}' = (\mathsf{Pg}', \mathsf{Kg}', \mathsf{Enc}', \mathsf{Dec}')$:

- **Parameter Generation**(Pg'): On input 1^k return $(pars, msk) \xleftarrow{\$} \mathsf{Pg}(1^k)$.
- **Key Generation**(Kg'): On input $pars, msk, id$, return $(pk_{id}, sk_{id}) \xleftarrow{\$} \mathsf{Kg}(pars, msk, id)$.

- **Encryption**(Enc'):
  On input $pars, pk_{id}, m$, generate a random $r \in \{0,1\}^k$ and return $(\mathsf{Enc}(pars, pk_{id}, r; H(r)), G(r) \oplus m, H'(pk, r, m))$.
- **Decryption**(Dec'): On inputs $pars, pk_{id}, sk_{id}, (c_1, c_2, c_3)$, compute $r' \xleftarrow{\$} \mathsf{Dec}(pars, pk_{id}, sk_{id}, c_1)$. If $r' = \perp$ or $\mathsf{Enc}(pars, pk_{id}, r'; H(r')) \neq c_1$, return $\perp$, else compute $m \leftarrow c_2 \oplus G(r)$; if $H'(pk, r, m) = c_3$ return m, else return $\perp$.

The above construction is an adaptation of an earlier version of the OAEP scheme (see [6]) based on any one-way trapdoor function (TDF). The two main differences are that (i) we are transforming a randomized encryption scheme instead of a one-way TDF which is why we use $H(r)$ to generate the randomness for the encryption algorithm, and (ii) since our goal is to also achieve robustness, the third component of the ciphertext hashes the public key along with the message and randomness.

It is also interesting to note that unlike the optimized OAEP scheme [7] which encrypts $c_2||c_3$ as part of the message (in order to obtain shorter ciphertexts), due to the impossibility result of [3] who rule out non-keyed redundancy codes, there is no hope of doing the same in our case.

EFFICIENCY COMPARISON. The overhead for the ciphertext size is two hash values each of which leads to 512 bits of overhead. The alternative existing solution would be to combine a weakly robust encryption scheme with the weak-to-strong transformation of [3]. This leads to $768 + x$ bits where the x is the ciphertext overhead of the weak-to-strong transformation which can be quite large itself (depending on the commitment scheme used).

Theorem 2. *Let* PKE *be an AI-CCA secure general encryption scheme and* H, G, *and* H' *be random oracles. Then, the* PKE' *scheme of construction 4.4 is both AI-CCA secure and SROB-CCA secure.*

We prove the above theorem via two separate claims. Claim 4.5 ensures that the above transformation preserves the AI-CCA security of the original scheme. Claim 4.6 states that the resulting scheme PKE' is also weakly robust.

Claim 4.5. *For any PPT adversary* A *against* PKE', *there exist a PPT adversary* B *against* PKE *such that:*

$$\mathbf{Adv}^{\text{ai-cca}}_{\mathsf{PKE}'}(\mathsf{A}) < q_D/2^k + q_H \mathbf{Adv}^{\text{ai-cca}}_{\mathsf{PKE}}(\mathsf{B})$$

Proof. We prove this claim in a sequence of games.

Game 0. In this game the adversary plays the AI-CCA game with the challenger using the construction above. The challenger initializes three empty lists H_{list}, G_{list}, and H'_{list}. For any oracle query q made to H (G, or H'), if a tuple of the form (q, a) for any a is present in H_{list} (G_{list} or H'_{list}) returns a as the answer. Else, challenger generates a random $a \in \{0,1\}^k$, adds (q, a) to the H_{list} (G_{list} or H'_{list}) and returns a to the adversary. Denote the adversary's challenge query by (m_0, m_1, id_0, id_1), and the response ciphertext by

$c^* = (c_1^*, c_2^*, c_3^*) = (\mathsf{Enc}(pars, pk_{id_b}, r; H(r)), G(r) \oplus m_b, H'(pk_{id_b}, r, m_b))$ for a random bit $b \in \{0,1\}$ and $r \in \{0,1\}^k$. Decryption queries are answered by the challenger using the decryption algorithm described above. Adversary eventually outputs the bit b' and wins if $b' = b$. For any PPT adversary A we have

$$\mathbf{Adv}^{\text{ai-cca}}_{\mathsf{PKE}'}(\mathsf{A}) = \mathbf{Adv}_{G_0}(\mathsf{A}) - 1/2$$

Game 1. Similar to game 0, except that on decryption queries of the form $c = (c_1, c_2, c_3)$ where $c_1 = c_1^*$, if there exist a tuple of the form $(q, c_3) \in H'_{list}$, challenger parses $(pk, r, m) \leftarrow q$, and recomputes the first two components of the ciphertext using these values. If they match c_1 and c_2 sent by the adversary, it returns m. If the values do not match or the tuple of the form (q, c_3) does not exist, challenger returns $\perp$.

A's view in the two games is different only in the case that he has not queried q to the list but is able to guess $c_3 = H'(q)$. This only happens with probability $1/2^k$ for every decryption query. Hence

$$\mathbf{Adv}_{G_0}(\mathsf{A}) - \mathbf{Adv}_{G_1}(\mathsf{A}) \leq q_D/2^k$$

Game 2. This game is identical to game 1 except that if A makes an oracle query for H or G on input r where r is the random message encrypted in the challenge ciphertext, the challenger outputs $fail$ and ends the game.

Based on the fundamental lemma game playing we have

$$\mathbf{Adv}_{G_1}(\mathsf{A}) - \mathbf{Adv}_{G_2}(\mathsf{A}) \leq \Pr[G_1^{\mathsf{A}} \rightarrow fail]$$

Next we will bound the probability of outputting fail, by the advantage of an adversary B who the one-way-CCA game against the PKE scheme. We show that for any adversary A winning the game G_2, there exist a PPT adversary B winning the one-way-CCA game against the original scheme PKE.

B generates a random index $i \in [1..q_H]$. B then runs A. When A makes his challenge query (m_0, m_1, id_0, id_1), B generates a random bit b, and asks for his challenge ciphertext under id_b to receive $c_1^* = \mathsf{Enc}(pk_{id_b}, r)$ for a random message r. B computes c_2^* and c_3^* on his own and replies to A with (c_1^*, c_2^*, c_3^*).

On an oracle query a (for any of the three oracles), if this is the ith oracle query, B outputs a to his own challenger and halts. Else, if a was queried before, he returns the same answer, and if not, he generates a random answer and adds the tuple to the corresponding list.

On a decryption query (c_1, c_2, c_3) where $c_1 \neq c_1^*$, B uses his own decryption oracle for Dec and performs the Dec' decryption algorithm. Here, it is critical for the randomness used in the encryption algorithm to be derivable from the decrypted message, and this is why $H(\cdot)$ is used as the randomness (or else B would not be able to perform the verification component of Dec'). For any decryption query (c_1, c_2, c_3) where $c_1 = c_1^*$, B performs exactly what the challenger in game 1 does. It is easy to see that

$$\Pr[G_2^{\mathsf{A}} \rightarrow fail] \leq q_H \mathbf{Adv}^{\text{ow}-\text{cca}}_{\mathsf{PKE}}(\mathsf{B}) \leq q_H \mathbf{Adv}^{\text{ai-cca}}_{\mathsf{PKE}}(\mathsf{B}).$$

For any adversary A who makes an oracle query for the challenge random message, there is an adversary B' who does not make such a query and has a better advantage (since such a query does not help the adversary win)

$$\mathbf{Adv}_{G_2}(\mathsf{A}) \leq \mathbf{Adv}_{G_2}(\mathsf{B}')$$

Finally, given that B' does not query r to the oracle, the challenge ciphertext is completely independent of the challenge bit b and hence

$$\mathbf{Adv}_{G_2}(\mathsf{B}') = 1/2$$

Putting everything together we have:

$$\mathbf{Adv}^{\text{ai-cca}}_{\mathsf{PKE}'}(\mathsf{A}) < q_D/2^k + q_H \mathbf{Adv}^{\text{ai-cca}}_{\mathsf{PKE}}(\mathsf{B})$$

Claim 4.6. *For any adversary* A *against* PKE' *we have* $\mathbf{Adv}^{\text{srob-cca}}_{\mathsf{PKE}'}(\mathsf{A}) \leq 1/2^k$.

Proof. The proof of the above claim is simple. The main observation is that a ciphertext c_1, c_2, c_3 is valid under two different public keys only if $H'(pk, \cdot, \cdot) = H'(pk', \cdot, \cdot)$ where $pk \neq pk'$. But this only happens with probability $1/2^k$ due to the fact that H' is a random oracle.

5 Collision-Free Encryption and Robustness

In this section we introduce the notion of *collision-freeness*, a natural relaxation of the notion of robustness for general encryption schemes. Intuitively, collision-freeness requires that a ciphertext decrypts to two *different* plaintexts when decrypted using distinct secret keys. Our main motivation is to use collision-freeness as a stepping stone for designing robust encryption schemes. Particularly, we design a more efficient construction for transforming collision-free encryption schemes to strongly robust ones. However, we also believe that collision-freeness is a sufficient property in some scenarios in practice.

Similar to the notion of robustness, we consider weak and strong collision-freeness (WCFR and SCFR). Interestingly, we show that schemes such as the El Gamal PKE scheme [13] and the Boyen-Waters IBE scheme [9] are strongly collision-free even though they are known not to be even weakly robust. Hence, collision-freeness seems to be a less restrictive assumption on an encryption scheme and one that most encryption schemes seem to satisfy without any modifications. The following security game defines the two variants:

- **Setup:** Challenger runs $(pars, msk) \leftarrow \mathsf{Pg}(1^k)$; $b \xleftarrow{\$} \{0,1\}$; $U, V \leftarrow \emptyset$.
- **Queries:**
  - Public key query id. Challenger lets $U \leftarrow U \cup \{id\}$; $(Ek[id], Dk[id]) \xleftarrow{\$} \mathsf{Kg}(pars, msk, id)$ and returns $Ek[id]$.
  - Decryption-key query id. If $id \notin U$ or $id \in S$ return $\perp$. Else $V \leftarrow V \cup \{id\}$ and return $Dk[id]$.

- Decryption query (C, id). If $id \notin U$ return $\perp$. Else let $M \leftarrow \mathsf{Dec}(pars, Ek[id], Dk[id], C)$, and return M.
- Final message (id_0^*, id_1^*, M) (for WCFR). If $id_0 = id_1$ or $id_0^* \notin U$ or $id_1^* \notin U$ or $id_0^* \in V$, or $id_1^* \in V$ return 0. Else let $C^* \xleftarrow{\$} \mathsf{Enc}(pars, Ek[id_0], M)$; $M' \leftarrow \mathsf{Dec}(pars, Ek[id_1], Dk[id_1], C^*)$; if $M' = M$ return 1, else return 0.
- Final message (id_0^*, id_1^*, C) (for SCFR). If $id_0 = id_1$ or $id_0^* \notin U$ or $id_1^* \notin U$ or $id_0^* \in V$, or $id_1^* \in V$ return 0. Else let $M_0 \leftarrow \mathsf{Dec}(pars, Ek[id_0], Dk[id_0], C)$; $M_1 \leftarrow \mathsf{Dec}(pars, Ek[id_1], Dk[id_1], C)$; if $M_0 = M_1$ return 1, else return 0.

In case of CPA attacks, no decryption queries are allowed. Adversary A's advantage in the {WCFR,SCFR}-{CPA,CCA} game is:

$$\mathbf{Adv}_{GE}^{\{\text{wcfr,scfr}\}-\{cpa,cca\}}(\mathsf{A}) = \Pr[G^{\mathsf{A}} \rightarrow 1]$$

Collision-freeness of an encryption scheme can be a sufficient requirement in some scenarios in practice. For example, if the receiver expects to see a specific message as part of the protocol but after decrypting using his secret key recovers a different one, he can detect an error and stop the communication. This makes collision-freeness a particularly attractive definition, since most of the existing anonymous encryption schemes, already satisfy this property without any additional modifications. The following claim mentions two well-known encryption schemes both of which are known not to be weakly-robust but which are collision-free.

Claim 5.1. *The El Gamal PKE scheme and the Boyen-Waters anonymous IBE scheme are SCFR-CPA scheme.*

The proof of the above claim quite simple but is omitted due to lack of space. Next we give a construction for transforming any strongly collision-free AI-CPA scheme into a strongly robust one. First we use the collision-free encryption scheme PKE to encrypt a random message r. Then, we hash the random message using a compressing collision resistant hash function h. We then use a strong extractor (e.g. a universal hash function) to extract the remaining randomness in r and use it as the key to a one-time symmetric-key encryption scheme.

The intuition is that (1) the collision-freeness of the PKE and the collision-resistance of the hash function h combined imply the strong robustness of the resulting scheme. More specifically, it is not hard to show that given any adversary that breaks the strong robustness of PKE', there exist an adversary that finds a collision for h: The collision-finding adversary decrypts the same ciphertext using the secret keys for two different public keys (identities) and outputs the two plaintexts as his collision for the hash function. The collision-freeness of the PKE ensures that the two plaintexts are different with high probability. (2) Given that r is chosen uniformly at random, PKE is IND-CPA secure, and $h(r)$ only leaks a fraction of bits of r, we can use the leftover hash lemma [16] to extract most of the remaining randomness and use it as the secret key to the symmetric-key encryption scheme.

Construction 5.2. Let $\mathsf{PKE} = (\mathsf{Pg}, \mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$ be a (SCFR-CPA + AI-CPA) general encryption scheme; $h : \{0,1\}^{\ell_1} \to \{0,1\}^{\ell_2}$ be a collision-resistant hash function; $Ext : \{0,1\}^k \times \{0,1\}^{\ell_1} \to \{0,1\}^{\ell_3}$ be a family of pairwise independent hash functions, where $\ell_3 \approx \ell_1 - \ell_2$; and $\mathcal{E} = (\mathcal{SK}, \mathcal{SE}, \mathcal{SD})$ be a one-time IND-CPA symmetric-key encryption scheme. We construct the general encryption scheme $\mathsf{PKE}' = (\mathsf{Pg}', \mathsf{Kg}', \mathsf{Enc}', \mathsf{Dec}')$:

- **Parameter Generation**: On input 1^k return $(pars, msk) \xleftarrow{\$} \mathsf{Pg}(1^k)$.
- **Key Generation**: On input $pars, msk, id$, return $(pk_{id}, sk_{id}) \xleftarrow{\$} \mathsf{Kg}(pars, msk, id)$.
- **Encryption**: On input $pars, pk_{id}, m$, generate a random $r \in \{0,1\}^{\ell_1}$ and $K \in \{0,1\}^k$ and return $(\mathsf{Enc}(pars, pk_{id}, r), h(r), K, \mathcal{SE}(Ext(K, r), m))$.
- **Decryption**: On inputs $pars, pk_{id}, sk_{id}, (c_1, c_2, c_3)$, compute $r' \xleftarrow{\$} \mathsf{Dec}(pars, pk_{id}, sk_{id}, c_1)$. If $h(r') = c_2$ return $m' \leftarrow \mathcal{SD}(Ext(K, r'), c_3)$, else return $\perp$.

The following theorem summarizes the result. Due to lack of space, we defer the proof to the full version of the paper.

Theorem 3. *Let* PKE *be a (AI-CPA + SCFR-CPA) secure general encryption scheme, h be a CRHF, Ext be a pairwise independent hash function and $\mathcal{E}$ be a one-time IND-CPA symmetric-key encryption scheme. Then, the* PKE' *scheme of construction 5.2 is both AI-CPA secure and SROB-CPA secure.*

Efficiency and Comparison. The computational overhead for the transformation is negligible as it includes one invocation of a collision-resistant hash function and a pairwise-independent hash function. As an alternative to the above construction, one could also combine the construction 4.1, which leads to a weakly robust encryption, with the weak-to-strong-robustness transformations of [3] to achieve the same goal. However, the resulting transformations are less efficient than the above transformation since we also took advantage of the collision-freeness of the encryption scheme. Furthermore, since all the encryption schemes we know of seem to possess the collision-freeness property, the improved efficiency comes for "free".

References

1. Abadi, M., Rogaway, P.: Reconciling two views of cryptography (the computational soundness of formal encryption) 20(3) 395 (2007)
2. Abdalla, M., Bellare, M., Catalano, D., Kiltz, E., Kohno, T., Lange, T., Malone-Lee, J., Neven, G., Paillier, P., Shi, H.: Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions, vol. 21(3), pp. 350–391 (2008)
3. Abdalla, M., Bellare, M., Neven, G.: Robust encryption. In: Micciancio, D. (ed.) Theory of Cryptography. LNCS, vol. 5978, pp. 480–497. Springer, Heidelberg (2010)

4. Abdalla, M., Bellare, M., Rogaway, P.: The oracle Diffie-Hellman assumptions and an analysis of DHIES, pp. 143–158 (2001)
5. Bellare, M., Boldyreva, A., Desai, A., Pointcheval, D.: Key-privacy in public-key encryption, pp. 566–582 (2001)
6. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols, pp. 62–73 (1993)
7. Bellare, M., Rogaway, P.: Optimal asymmetric encryption, pp. 92–111 (1994)
8. Boneh, D., Franklin, M.K.: Identity-based encryption from the Weil pairing, pp. 213–229 (2001)
9. Boyen, X., Waters, B.: Anonymous hierarchical identity-based encryption (without random oracles), pp. 290–307 (2006)
10. Cramer, R., Shoup, V.: Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack, vol. 33(1), pp. 167–226 (2003)
11. Desai, A.: The security of all-or-nothing encryption: Protecting against exhaustive key search, pp. 359–375 (2000)
12. Dolev, D., Dwork, C., Naor, M.: Non-malleable cryptography (1998) (manuscript)
13. El Gamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms, vol. 31, pp. 469–472 (1985)
14. Fischlin, M.: Pseudorandom function tribe ensembles based on one-way permutations: Improvements and applications, pp. 432–445 (1999)
15. Goldwasser, S., Micali, S.: Probabilistic encryption. Journal of Computer and System Sciences 28(2), 270–299 (1984)
16. Håstad, J., Impagliazzo, R., Levin, L.A., Luby, M.: A pseudorandom generator from any one-way function, vol. 28(4), pp. 1364–1396 (1999)
17. Hofheinz, D., Weinreb, E.: Searchable encryption with decryption in the standard model. Cryptology ePrint Archive, Report 2008/423 (2008), http://eprint.iacr.org/
18. Katz, J., Sahai, A., Waters, B.: Predicate encryption supporting disjunctions, polynomial equations, and inner products, pp. 146–162 (2008)
19. Rackoff, C., Simon, D.R.: Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack, pp. 433–444 (1992)

Limitations on Transformations from Composite-Order to Prime-Order Groups: The Case of Round-Optimal Blind Signatures

Sarah Meiklejohn[1], Hovav Shacham[1], and David Mandell Freeman[2]

[1] University of California, San Diego
La Jolla, CA 92037, USA
{smeiklej,hovav}@cs.ucsd.edu
[2] Stanford University
Stanford, CA 94305, USA
dfreeman@cs.stanford.edu

Abstract. Beginning with the work of Groth and Sahai, there has been much interest in transforming pairing-based schemes in composite-order groups to equivalent ones in prime-order groups. A method for achieving such transformations has recently been proposed by Freeman, who identified two properties of pairings using composite-order groups — "cancelling" and "projecting" — on which many schemes rely, and showed how either of these properties can be obtained using prime-order groups.

In this paper, we give evidence for the existence of limits to such transformations. Specifically, we show that a pairing generated in a natural way from the Decision Linear assumption in prime-order groups can be simultaneously cancelling and projecting only with negligible probability.

As evidence that these properties can be helpful together as well as individually, we present a cryptosystem whose proof of security makes use of a pairing that is both cancelling and projecting. Our example cryptosystem is a simple round-optimal blind signature scheme that is secure in the common reference string model, without random oracles, and based on mild assumptions; it is of independent interest.

1 Introduction

Composite-order groups were introduced for pairing-based cryptography in 2005 by Boneh, Goh, and Nissim [12] and have since been used to realize a large number of cryptographic systems (see, e.g., the schemes surveyed by Freeman [24]). At the same time, the limited number of elliptic curve families on which composite-order groups can be instantiated and the larger parameter sizes associated with composite-order groups (cf. [23,13]) has motivated research on translating these schemes to or obtaining similar ones in the prime-order setting.

In one of the first papers to unify the composite- and prime-order settings, Groth and Sahai [31] developed non-interactive zero-knowledge schemes that not only can be instantiated either in composite- or prime-order groups, but are

M. Abe (Ed.): ASIACRYPT 2010, LNCS 6477, pp. 519–538, 2010.

in fact described identically in either instantiation. What facilitates this flexibility is a new abstraction for pairing-based cryptography in terms of modules over finite commutative rings with an associated bilinear map; this abstraction allows for the simultaneous treatment of three different cryptographic assumptions: the Subgroup Hiding (SGH) assumption of Boneh, Goh, and Nissim [12] in composite-order groups; the Decision Linear (DLIN) assumption of Boneh, Boyen, and Shacham [11], and its k-Linear family of generalizations [45,33],[1] in prime-order groups; and the so-called Symmetric External Diffie-Hellman assumption [7], also in prime-order groups.

More recently, Freeman [24] and Garg, Sahai, and Waters [27] have proposed methods for transforming schemes secure in the composite-order setting into ones secure (under different but analogous assumptions) in the prime-order setting. Freeman, in particular, identifies two properties of pairings on composite-order groups, *projecting* and *cancelling*, and shows how either can be obtained in prime-order groups. He then demonstrates how to transform several known cryptosystems that rely on one of these properties from composite- to prime-order groups.

Our contribution: limits on transformations from composite to prime order. In this paper, we show limits to the feasibility of composite-to-prime transformations such as those mentioned above, suggesting that some schemes cannot be transformed mechanically from one setting to the other. In our main theorem, Theorem 6.5, we show that no pairing over prime-order groups can — except in a negligible fraction of cases — be both projecting and cancelling when subgroup indistinguishability relies in a natural way on k-Linear, where "natural" simply means that we follow the definition of the assumption as closely as possible.

If no cryptosystem required a pairing that is both projecting and cancelling, however, our Theorem 6.5 would not be particularly interesting. As such, we present a new cryptosystem — a natural pairing-based blind signature scheme that is of independent interest, and discussed below — whose proof of security calls for a pairing that is both projecting and cancelling.[2]

Blind signatures were introduced by Chaum in 1982 [17]. In a blind signature scheme, a user interacts in a protocol with a signer to obtain a signature on a message of the user's choice. When the protocol execution ends, the user obtains the signature but the signer learns nothing about the message that was signed. Blind signatures have been used as a building block in a variety of applications, including electronic cash [20] and electronic voting [19].

One useful feature of a blind signature scheme is *concurrency*. For example, if a blind signature used to build an electronic cash system does not retain its security even when the signer engages in multiple protocol executions concurrently, it leaves the issuing bank susceptible to denial-of-service attacks. Concurrency turns out to be as difficult to achieve for blind signatures as it is for other cryptographic

[1] A family of progressively strictly weaker decisional assumptions, where 1-Linear is DDH and 2-Linear is DLIN.

[2] We emphasize that it is the security proof, not the statement of the scheme, that uses the two pairing properties. We thus do not rule out the possibility that a *different* proof strategy will show our scheme secure in prime-order groups.

protocols. While many blind signature schemes have proofs of security only for sequential executions of the protocol, the problem is not merely with proofs. In one example, Martinet, Poupard, and Sola [38] show that signatures in a partially blind signature scheme of Cao, Lin and Xue [16] are forgeable if the signer interacts with two users concurrently.

Our contribution: a round-optimal blind signature scheme. As mentioned above, we present a new pairing-based blind signature scheme. Our blind signing protocol is round-optimal: it consists of only two moves (a request and a response), which implies that it is secure even in the presence of concurrent signing protocol executions. Our scheme is practical, has a proof of security (without random oracles) in the common reference string model, and relies for its security on falsifiable and non-interactive assumptions: computational Diffie-Hellman and Subgroup Hiding. These assumptions are milder than those used in any previous practical concurrently secure blind signature, including those in the random oracle model. ("Practical" in this sense means not relying on general NIZKs for NP as a building block.) Our scheme extends in a natural way to give a partially blind signature scheme [3] with the same properties.

Our blind signatures combine the Waters signature scheme [46] with non-interactive witness-indistinguishable proofs developed in a line of papers by Groth, Ostrovsky, and Sahai [30,29,31]. In this structure our scheme is related to the group signature scheme of Boyen and Waters [15]. The primary disadvantage of our scheme, as with the Boyen-Waters group signature, is its bit-at-a-time nature, which makes the user's blind signing request large: some 40 kilobytes at the 1024-bit security level. The signer's response and the resulting signatures, however, are short.

Related work. The blind signature literature is extensive and varied. Below, we briefly survey the most closely related schemes with concurrent security; see [5,4] for more complete recent treatments.

In the random oracle model, there exist elegant round-optimal blind signatures, due to Chaum [18] and Boldyreva [10], that feature short public keys, short signatures, and an efficient blind signing protocol. Unfortunately the security proofs for these schemes rely on strong interactive assumptions: the RSA known-target inversion assumption [9] and the chosen-target CDH assumption (by contrast, the underlying ordinary signatures can be shown secure using RSA and CDH, respectively).

In the common reference string model, several practical concurrently secure blind signature schemes have been proposed. Unlike our scheme, these schemes rely on assumptions that are interactive or whose statement size grows with the number of queries in the reduction (i.e., "q-type"). Kiayias and Zhou [35] give four-move blind and partially-blind signature schemes secure under the (interactive) LRSW assumption [37], the Paillier assumption [42], and DLIN. Okamoto [40] gives four-move blind and partially blind signature schemes based on the (q-type) Two-Variable Strong Diffie-Hellman assumption and Paillier. Fuchsbauer [25] gives two-move blind signature schemes based on the (q-type)

Asymmetric Double Hidden Strong Diffie-Hellman assumption, the Asymmetric Weak Flexible CDH assumption, and DLIN. Finally, Abe, Haralambiev, and Ohkubo [4] give two-move blind signature schemes based on the (q-type) Simultaneous Flexible Pairing assumption and DLIN. (The last two papers appeared together as [2].)

Also in the common reference string model, blind signatures that use general NIZKs for NP (and are therefore not practical) were given by Juels, Luby, and Ostrovsky [34], Fischlin [22], and Abe and Ohkubo [5]. The Fischlin and Abe-Ohkubo schemes are round-optimal.

Okamoto [40] first observed that the Waters signature can be combined with witness-indistinguishable proofs for a simple NP language to yield blind and partially blind signatures. Our scheme could be viewed as an instantiation of Okamoto's framework (though we blind the message differently) where we take advantage of Groth-Ostrovsky-Sahai proofs to eliminate a round of interaction.

Until recently, no concurrently secure blind signature schemes were known in the plain public-key model. The first such scheme was given by Hazay et al. [32]; it relies on general NIZKs, and its round complexity is poly-logarithmic in the number of concurrent executions for which security must be guaranteed.

Applications and extensions. Finally, as an application of our techniques, we show (in the full version of our paper [39]) how our blind signatures may be used within the Waters IBE system [46] to yield a blind IBE scheme, as introduced by Green and Hohenberger [28]. Compared to Green and Hohenberger's blind extraction protocol, our protocol achieves concurrent security but adds a common reference string and a reliance on the SGH assumption.[3] Furthermore, the Waters signature naturally extends into a hierarchical identity-based signature (cf. [43]); applying our construction at level 2 of the resulting signature gives an identity-based blind signature [47] concurrently secure in the common reference string model.[4] Alternatively, using the Boyen-Waters group signature [15] at level 1 of the hierarchy and our blind signature at level 2 gives a group blind signature [36] concurrently secure in the common reference string model.

2 Mathematical Background

In this paper, we work with *bilinear groups*, which are cyclic groups G of some finite order that admit a nondegenerate bilinear map $e\colon G \times G \to G_T$. Because we generalize the concept of a group and work with modules, we are able to describe

[3] The efficient range proofs due to Boudot [14] rely on the Strong RSA assumption (due to Baric and Pfitzmann [8]) and require a common reference string. If the scheme of Green and Hohenberger is instantiated with these range proofs then its assumptions and setup model are comparable to those of our scheme, but without providing concurrent security.

[4] One could also obtain an identity-based blind signature through generic composition of our blind signature and an ordinary signature [26].

our scheme without relying on any particular properties of the underlying group (with the caveat, as mentioned above, that the scheme is provably secure only for composite-order groups).

2.1 Modules

We first recall the definition of a module; this serves as the foundation for our blind signature scheme, and more specifically for the Groth-Sahai commitments used in our scheme. (See [21, Ch. 10] for further background on modules.)

Definition 2.1. Let $(\mathcal{R}, +, \cdot, 0, 1)$ be a finite commutative ring. An $\mathcal{R}$-*module* A is an abelian group $(A, +, 0)$ such that there exists an operator (namely, scalar multiplication) $\mathcal{R} \times A \to A$, denoted by $(r, x) \mapsto rx$, satisfying the following four properties for all $r, s \in \mathcal{R}$ and $x, y \in A$:

- $(r + s)x = rx + sx$.
- $r(x + y) = rx + ry$.
- $r(sx) = (rs)x$.
- $1x = x$.

When A is written multiplicatively our operator becomes exponentiation and the requirements are written as $x^{r+s} = x^r \cdot x^s$, $(x \cdot y)^r = x^r \cdot y^r$, $(x^r)^s = x^{rs}$, and $x^1 = x$ for all $r, s \in \mathcal{R}$ and $x, y \in A$.

The concept of a module generalizes that of a vector space: when $\mathcal{R}$ is a field, the definitions of an $\mathcal{R}$-module and an $\mathcal{R}$-vector space coincide. The concept of a module also generalizes the concept of an abelian group, as any abelian group can be viewed as a $\mathbb{Z}$-module. If A is isomorphic to $\mathcal{R}^r$ as abelian groups, then r is the *rank* of A. When $\mathcal{R}$ is a field, module rank is the same as vector space dimension. In cryptography, we are most used to working with $\mathbb{Z}/n\mathbb{Z}$- and $\mathbb{F}_p$-modules; for example, any finite group of exponent p can be viewed as a $\mathbb{F}_p$-module.

2.2 Groth-Sahai Commitments

Groth and Sahai [31] devise two types of commitments: commitments to elements in an $\mathcal{R}$-module A, and commitments to elements in the ring $\mathcal{R}$. For our purposes, we will need only commitments to bits; we can simplify things even further by always setting $A = G$ for our bilinear group G.

To form commitments to module elements, Groth and Sahai define an $\mathcal{R}$-module B and two homomorphisms $\tau : A \to B$ and $\rho : B \to A$.[5] These maps are defined such that, for some elements $h_1, \ldots, h_m$ in B, $\rho(h_i) = 1$ for all i and ρ is non-trivial for all x that are not contained in $B_1 := \langle h_1, \ldots, h_m \rangle$. A commitment to $x \in A$ is then defined as $c(x) = \tau(x) \prod_{i=1}^{m} h_i^{r_i}$ for random values $r_1, \ldots, r_m \leftarrow \mathcal{R}$. This means that the h_i elements act as keys for the commitment scheme, and that the common reference string is $(\mathcal{R}, A, B, \tau, \rho, h_1, \ldots, h_m)$. There are two cases:

[5] Our notation differs from that of Groth and Sahai, but the ideas are the same.

- Hiding keys: in this case, the h_i elements generate the whole module B; in other words, $B_1 = \langle h_1, \ldots, h_m \rangle = B$. This implies that $\tau(A) \subseteq B_1$, which means that $c(x)$ will be perfectly hiding (as each commitment will be a random element of B).
- Binding keys: in this case, $B_1 \neq B$ and $\rho(c) = \rho(\tau(x)h^r) = \rho \circ \tau(x)$ for some restricted space of inputs x. To determine what this restricted space is, we see that c will generally reveal the coset of B_1 where $\tau(x)$ lives. Thus in order for the commitment to be perfectly binding we must restrict the space of inputs x to be the inverse image of $B_2 \simeq B/B_1$; because we know that $B_1 \neq B$, both B_2 and $\tau^{-1}(B_2)$ are non-trivial and so this domain restriction is actually meaningful. (Since B is an abelian group, the quotient module is always well-defined.)

It is clear from these definitions that a set of keys cannot be both hiding and binding, as the settings require very different properties of the commitment keys $h_1, \ldots, h_m$. To get any meaningful blindness properties, however, we need these two settings to be indistinguishable. We therefore require an assumption that implies this indistinguishability; the choice of assumption depends on the instantiation being used.

3 Security Notions for Blind Signatures

We define a blind or partially blind signature scheme in the common reference string (CRS) model to be a collection of four protocols: a $\mathsf{Setup}(1^k)$ algorithm that outputs the CRS σ_{CRS}, a $\mathsf{KeyGen}(\sigma_{CRS})$ algorithm that outputs the signing key pair (pk, sk), a BlindSign protocol, which consists of an interaction of the form $\mathsf{User}(\sigma_{CRS}, pk, M) \leftrightarrow \mathsf{Signer}(\sigma_{CRS}, sk)$ (in which the signer outputs success if the protocol is successful, and the user outputs success and the unblinded signature σ), and finally a $\mathsf{Verify}(\sigma_{CRS}, pk, M, \sigma)$ algorithm that outputs accept if the signature is valid and fail if not.

In general, there are two properties that blind and partially blind signatures must satisfy: blindness and one-more unforgeability. Informally, the blindess requirement says that in the process of signing a user's message, the signer does not learn anything abut the message he is signing. The one-more unforgeability requirement says that if the user interacts with the signer ℓ times, then he should be able to produce ℓ signatures and no more (so in particular, he cannot produce $\ell + 1$). We now describe these properties more formally.

3.1 Blind Signatures

Formal definitions of blind signatures were introduced by Juels, Luby, and Ostrovsky [34], although both properties were considered informally in Chaum's original paper on the subject [17], and one-more unforgeability was considered formally in Pointcheval and Stern's work on security arguments for signatures [44].

In the Juels-Luby-Ostrovsky formalization, the blindness property is defined as follows: the adversary is given a public-private key pair and outputs two messages M_0 and M_1. He then engages in two signing protocols with honest users: the first user requests a signature on message M_b and the second on message M_{1-b}, where b is a random bit unknown to the adversary. The adversary is then given the resulting signatures σ_0 and σ_1, but only if both interactions are successful, and his goal is to guess the bit b (given the messages, the corresponding signatures, and the signing protocol transcripts).

In this paper, we use a stronger version of the blindness property which allows the adversary to generate the signing key pair himself, possibly in a malicious manner. This strengthening was proposed independently in several recent papers [1,41,35].

The one-more unforgeability property can be defined as follows: the adversary is given a public key and engages in multiple executions of the blind signing protocol with a signer; the adversary is able to choose how to interleave the executions. At the end, the adversary is considered successful if he is able to output $\ell + 1$ distinct message-signature pairs $(M_1, \sigma_1), \ldots, (M_{\ell+1}, \sigma_{\ell+1})$, where ℓ is the number of executions in which the signer outputs success.

In this definition, two message-signature pairs (M_i, σ_i) and (M_j, σ_j) are considered distinct even if $M_i = M_j$ (so if σ_i and σ_j are just two different signatures on the same message) for $i \neq j$. Unfortunately, this means that any signature scheme in which signatures can be re-randomized (like our signature scheme, as we will see in Section 4) will automatically be unable to satisfy one-more unforgeability. We therefore weaken the property by requiring that the adversary be unable to output $\ell + 1$ message-signature pairs in which the *messages* are all distinct.[6] This modified definition was also considered recently by Okamoto [41].

We put all this information together and give a formal definition of security for blind signature schemes in the full version of this paper [39].

3.2 Partially Blind Signatures

The security properties of blind signatures can also be extended to partially blind signatures; these formalizations are due to Abe and Okamoto [6]. For partially blind signatures, the adversary outputs two info strings $info^{(0)}$ and $info^{(1)}$ in addition to its messages M_0 and M_1. It then interacts with two separate users in the same manner as before, except this time the first user requests a signature on M_b using $info^{(0)}$ and the second requests a signature on M_{1-b} with info $info^{(1)}$. The adversary is given the resulting signatures σ_0 and σ_1 if both interactions were successful and if $info^{(0)} = info^{(1)}$. As before, his goal is to guess the bit b.

The one-more unforgeability property is also quite similar to the property for blind signatures; here, the adversary is allowed to choose the info string for each interaction with the signer. The goal is then for the adversary to output an info

[6] We observe that blind signatures satisfying this weakened unforgeability property are still sufficient for e-cash and other standard applications of blind signatures.

string $info^*$ as well as $\ell + 1$ message-signature pairs $(M_1, \sigma_1), \ldots, (M_{\ell+1}, \sigma_{\ell+1})$, where ℓ represents the number of interactions in which the signer output success while using the info string $info^*$.

4 Underlying Signature Scheme

As our underlying signature scheme we use a slightly modified version of the Waters signature scheme [46]. Essentially, we just need to generalize the Waters signature scheme by bringing it into the language of modules so that we can use it in combination with Groth-Sahai commitments to create our blind signature scheme.

4.1 CRS Setup

For the Waters signature, the required elements for the common reference string are a bilinear group G, the target group G_T and the bilinear map $e\colon G \times G \to G_T$, as well as generators $g, u', u_1, \ldots, u_k$ for G, where k denotes the length of the messages to be signed. We now add in the elements discussed in Section 2.2: we start with a ring $\mathcal{R}$ such that G can be interpreted as an $\mathcal{R}$-module. We then add in an $\mathcal{R}$-module B, a map $\tau\colon G \to B$, a map $\rho\colon B \to G$, and a bilinear map $E\colon B \times B \to B_T$, which also requires us to specify a target module B_T and the resulting τ_T and ρ_T maps. This means that the CRS will be $\sigma_{sig} = (\mathcal{R}, G, G_T, B, B_T, e, E, \tau, \tau_T, \rho, \rho_T, g, u', u_1, \ldots, u_k)$. The relations between all these maps are summarized in the following figure:

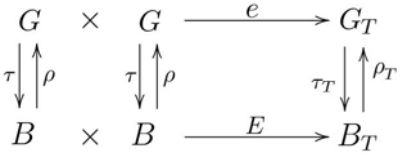


Fig. 1. Commutative diagram for our modules

4.2 Signing Protocol

In our generalized Waters signature, the size of the message space will be $\{0,1\}^k$ for some value k (for example, to use hash-and-sign with SHA-1 as the hash function we would set $k = 160$). As noted above, the CRS, which is shared between the user and the signer, will contain $k + 1$ random generators of G.

- $\mathsf{Setup}(1^k)$: Output a tuple σ_{sig} that has been computed as described in the previous section.
- $\mathsf{KeyGen}(\sigma_{sig})$: Pick a random value $\alpha \leftarrow \mathcal{R}$ and set $A = E(\tau(g), \tau(g))^\alpha$. The public key will be $pk = A$ and the secret key will be $sk = \alpha$ (actually, $\tau(g)^\alpha$ will suffice).

– $\mathsf{Sign}(\sigma_{sig}, sk, M)$: Write the message out bitwise as $M = b_1 \dots b_k$, and write $sk = \tau(g)^\alpha$. Pick a random $r \leftarrow \mathcal{R}$ and compute

$$S_1 = \tau(g)^\alpha \Big(\tau(u') \prod_{i=1}^{k} \tau(u_i)^{b_i}\Big)^r \quad \text{and} \quad S_2 = \tau(g)^{-r}.$$

Output the signature $\sigma = (S_1, S_2)$.

– $\mathsf{Verify}(\sigma_{sig}, pk, M, \sigma)$: Again, write the message out bitwise as $M = b_1 \dots b_k$; also write the signature as $\sigma = (S_1, S_2)$ and the public key as $pk = A$. Check that these values satisfy the following equation:

$$E(S_1, \tau(g)) \cdot E\Big(S_2, \tau(u') \prod_{i=1}^{k} \tau(u_i)^{b_i}\Big) = A. \tag{1}$$

If they do, output accept; else, output fail.

One nice property of the Waters signature (and our extended Waters signature) is that anyone can re-randomize a signature by choosing $s \leftarrow \mathcal{R}$ and computing $S_1' = S_1 \cdot \big(\tau(u') \prod_i \tau(u_i)^{b_i}\big)^s$ and $S_2' = S_2 \cdot \tau(g)^{-s}$. Since this results in $S_1' = \tau(g)^\alpha \big(\tau(u') \prod_i \tau(u_i)^{b_i}\big)^{r+s}$ and $S_2' = \tau(g)^{-(r+s)}$, the re-randomization process really does give us a valid signature. In particular, the randomness in the resulting signature (S_1', S_2') will be information-theoretically independent from the randomness r chosen by the signer in the signature (S_1, S_2).

We recall the computational Diffie-Hellman (CDH) assumption used for the Waters signature:

Assumption 4.1. *Let $\mathcal{G}$ be an algorithm that outputs a tuple (q, G, g), where G is a group of order q (not necessarily prime) and g is a generator of g. We say that G satisfies the computational Diffie-Hellman assumption if it is computationally infeasible to compute the value g^{ab} given the tuple (g, g^a, g^b). More formally, for all PPT adversaries $\mathcal{A}$ there exists a negligible function $\nu(\cdot)$ and a security parameter k_0 such that the following holds for all $k > k_0$:*

$$\Pr\big[(q, G, g) \leftarrow \mathcal{G}(1^k);\ a, b \leftarrow \mathbb{Z}_q\ :\ \mathcal{A}(g, g^a, g^b) = g^{ab}\big] = \nu(k).$$

The Waters signature scheme is existentially unforgeable if G satisfies the CDH assumption. In our extended version, the signature scheme will be existentially unforgeable if B satisfies the CDH assumption. As the proof is a trivial extension of Waters' proof, we will not include it here.

5 Our Blind Signature Scheme

In this section we describe our blind signature scheme. Although we describe only the partially blind setting, our description also encapsulates the fully blind setting, which corresponds to the case $k_0 = 0$.

5.1 CRS Setup

In our CRS we must include all the necessary elements for Groth-Sahai commitments as well as values in the tuple σ_{sig} of Section 4.1. This means our CRS will be $\sigma_{CRS} = (\sigma_{sig}, h_1, \ldots, h_m)$, where the h_i elements are binding keys for Groth-Sahai commitments. Specifically, the elements h_i generate a proper submodule B_1 of the module B used in the Waters signature scheme.

5.2 The Partially Blind Protocol

In the following protocol, the user and signer both have access to an info string *info*. At the end of the protocol, the user obtains a signature on $info||M$ for a message M, while the signer learns nothing beyond the fact that the message M followed the guidelines laid out in *info*. In addition, the user and the signer will have agreed upon the length of the *info* string; call it k_0 for $0 \leq k_0 \leq k$. Setting $k_0 = 0$ corresponds to a fully blind signature, while setting $k_0 = k$ corresponds to an ordinary run of the (generalized) Waters signature scheme.

- $\mathsf{Setup}(1^k)$: Output σ_{CRS} as described in the previous section (Section 5.1).
- $\mathsf{KeyGen}(\sigma_{CRS})$: Same as KeyGen from Section 4.2.
- $\mathsf{User}(\sigma_{CRS}, pk, info, M)$: First write the info string out bitwise, as $info = b_1 \ldots b_{k_0}$, and similarly write the message as $M = b_{k_0+1} \ldots b_k$. Now, for each i such that $k_0 < i \leq k$, pick random values $t_{i1}, \ldots, t_{im} \leftarrow \mathcal{R}$ and compute

$$c_i = \tau(u_i)^{b_i} \cdot \prod_{j=1}^{m} h_j^{t_{ij}} \quad \text{and} \quad \pi_{ij} = \left(\tau(u_i)^{2b_i - 1} \cdot \prod_{j=1}^{m} h_j^{t_{ij}}\right)^{t_{ij}},$$

  where c_i acts as a GS commitment to the bit b_i and $\vec{\pi}_i = \{\pi_{ij}\}_{j=1}^m$ acts as a proof that the value contained in c_i is in fact a 0 or a 1. Send the tuple $req = (c_{k_0+1}, \vec{\pi}_{k_0+1}, \ldots, c_k, \vec{\pi}_k)$ as a request to the issuer (and save some state information *state*).
- $\mathsf{Signer}(\sigma_{CRS}, sk, info, req)$: First, write $info = b_1 \ldots b_{k_0}$ and $sk = \tau(g)^\alpha$. Upon receiving the request, check that each c_i is indeed a commitment to a 0 or 1 by checking that

$$E\big(c_i, \tau(u_i)^{-1} c_i\big) = \textstyle\prod_{j=1}^{m} E(h_j, \pi_{ij}) \tag{2}$$

  for each $k_0 < i \leq k$. If this equation fails to hold for any value of i, abort the protocol and output $\perp$. Otherwise, compute the value

$$c = \tau(u')\Big(\prod_{i=1}^{k_0} \tau(u_i)^{b_i}\Big)\Big(\prod_{i=k_0+1}^{k} c_i\Big).$$

  Finally, pick a random value $r \leftarrow \mathcal{R}$ and compute

$$K_1 = \tau(g)^\alpha \cdot c^r, \quad K_2 = \tau(g)^{-r}, \quad \text{and} \quad K_{3j} = h_j^{-r} \quad \text{for } 1 \leq j \leq m.$$

  Set $\vec{K}_3 = \{K_{3j}\}_{j=1}^m$, send the tuple $(K_1, K_2, \vec{K}_3)$ back to the user, and output success and *info*.

– $\mathsf{User}(state, (K_1, K_2, \vec{K}_3))$: First, check that K_2 and $\vec{K}_3$ were formed properly by checking satisfiability of

$$E\big(K_{3j}, \tau(g)\big) = E(K_2, h_j) \tag{3}$$

for each $1 \le j \le m$. If this equation does not verify for some j, abort and output $\perp$. Otherwise, unblind the signature by computing

$$S_1 = K_1 \prod_{i=k_0+1}^{k} \prod_{j=1}^{m} K_{3j}^{\,t_{ij}} \quad \text{and} \quad S_2 = K_2. \tag{4}$$

Next verify that this is a valid signature on $info||M$ by running $\mathsf{Verify}(\sigma_{CRS}, pk, info||M, (S_1, S_2))$. If this step outputs fail, abort the protocol and output $\perp$. If it outputs accept, however, re-randomize the signature by choosing a random value $s \leftarrow \mathcal{R}$ and computing

$$S_1' = S_1\Big(\tau(u') \prod_{i=1}^{k} \tau(u_i)^{b_i}\Big)^{s} \quad \text{and} \quad S_2' = S_2 \cdot \tau(g)^{-s}.$$

The final signature is $\sigma = (S_1', S_2')$; output σ as well as $info$ and $\mathsf{success}$.

– $\mathsf{Verify}(\sigma_{CRS}, pk, M, \sigma)$: Same as Verify from Section 4.2.

Theorem 5.1. *The blind signature scheme outlined above is correct and partially blind, under the assumption that the h_i values in the hiding and binding settings are indistinguishable.*

The proof of Theorem 5.1 appears in the full version of this paper [39]. The theorem demonstrates correctness and (partial) blindness, but it does not show one-more unforgeability. In order to prove this last property, we need to define two properties of pairings, adapted from Freeman [24, §3] for our purposes:

Definition 5.2. A pairing $E\colon B \times B \to B_T$ is *cancelling* if there exists a decomposition $B = B_1 \times B_2$ such that $E(b_1, b_2) = 1$ for all $b_1 \in B_1$, $b_2 \in B_2$.

Definition 5.3. A pairing $E\colon B \times B \to B_T$ is *projecting* if there exists a decomposition $B = B_1 \times B_2$, a submodule $B_T' \subset B_T$, and homomorphisms $\pi\colon B \to B$ and $\pi_T\colon B_T \to B_T$, such that $B_1 \subseteq \ker(\pi)$, $\pi(x) = x$ for $x \in B_2$, $B_T' \subseteq \ker(\pi_T)$, and $\pi_T(E(x, y)) = E(\pi(x), \pi(y))$ for all $x, y \in B$.

As we will see below, the pairing E has both of these properties (with respect to the same decomposition $B = B_1 \times B_2$) when instantiated using composite-order groups under the Subgroup Hiding (SGH) assumption. Because SGH also provides the necessary indistinguishability properties, we obtain the following theorem, a proof of which can be found in the full version of this paper [39]:

Theorem 5.4. *The blind signature scheme outlined above is one-more unforgeable under the SGH assumption and the assumption that the modified Waters signature scheme in Section 4 is existentially unforgeable on the submodule $B_2 \subseteq B$.*

5.3 Instantiation under the SGH Assumption

We first recall the Subgroup Hiding (SGH) assumption:

Assumption 5.5 ([12]). *Let $\mathcal{G}$ be an algorithm that outputs a tuple (p, q, G, G_T, e) such that G and G_T are both groups of order $n = pq$ and $e\colon G \times G \to G_T$ is a nondegenerate bilinear map. We say that G satisfies the Subgroup Hiding assumption if it is computationally infeasible to distinguish between an element of G and an element of G_q. More formally, for all PPT adversaries $\mathcal{A}$ there exists a negligible function $\nu(\cdot)$ and a security parameter k_0 such that the following holds for all $k > k_0$:*

$$\Big|\Pr\big[(p,q,G,G_T,e) \leftarrow \mathcal{G}(1^k); n = pq; x \leftarrow G : \mathcal{A}(n,G,G_T,e,x) = 0\big] - \Pr\big[(p,q,G,G_T,e) \leftarrow \mathcal{G}(1^k); n = pq; x \leftarrow G : \mathcal{A}(n,G,G_T,e,x^p) = 0\big]\Big| < \nu(k).$$

To instantiate our blind signature scheme under this assumption, we use a group G of order $n = pq$ with p, q prime. We define $B = G$ and τ to be the identity map; this means that we can use $E = e$. We need only one h_i element, namely an h_1 such that h_1 generates G_q in the binding setting and h_1 generates the whole group G in the hiding setting. The SGH assumption tells us that these choices of h_1 are indistinguishable. We can also describe our ρ map as $\rho(c_i) = c_i^q = (u_i^q)^{b_i}$ since h_1 has order q. Because the u_i are all generators for G and therefore $u_i^q \neq 1$, we see that the ρ map will indeed reveal the bit b_i.

Because h_1 will generate either G or G_q, we have $B = G_p \times G_q$, which means (looking back at the statement of Theorem 5.4) that we assume for the security of our blind signature that CDH is hard in G_p. To see that the pairing e is cancelling, note that every element of G_p can be written as $a = g^{\alpha q}$ for some $\alpha \in \mathbb{F}_p$ and every element of G_q can be written as as $b = g^{\beta p}$ for some $\beta \in \mathbb{F}_q$. Then $e(a,b) = e(g^{\alpha q}, g^{\beta p}) = e(g^{\alpha\beta pq}, g) = e\big((g^n)^{\alpha\beta}, g\big) = 1$ because G has order n. To see that e is projecting, note that there exists a $\lambda \in \mathbb{Z}_n$ such that $\lambda \equiv 1 \bmod p$ and $\lambda \equiv 0 \bmod q$, and that furthermore this value is efficiently computable (given the factorization of n) using the Chinese Remainder Theorem. Thus exponentiating by λ cancels out the G_q component of a group element while leaving the G_p component unchanged. This allows us to define $\pi(z) = z^\lambda$ for $z \in G$ and $\pi_T(z_T) = (z_T)^\lambda$ for $z_T \in G_T$. These maps are easily seen to satisfy the projecting properties.

Finally, to compute the value h_1 we can set $h_1 = g$ in the hiding setting and $h_1 = g^p$ in the binding setting. This means that, as with the map ρ, the factorization of n will be required as a trapdoor to compute h_1.

The obvious downside of using our scheme under the SGH assumption is the use of a composite-order group, which necessitates a common reference string generated by a trusted third party.[7] The upside, on the other hand, is that the

[7] It is an open problem to replace the trusted third party with an efficient secure multiparty computation protocol for computing the CRS.

scheme is as efficient as possible under this assumption, as each part of the signature involves only one group element.[8]

6 Converting to a Prime-Order Setting

In this section we argue that our scheme cannot be instantiated securely under a natural usage of the k-Linear family of assumptions in prime-order groups. The k-Linear family generalizes the Decision Diffie-Hellman and Decision Linear [11] assumptions (which can be recovered by setting $k = 1$ or 2, respectively) and is defined as follows:

Assumption 6.1 ([45,33]). *Let $\mathcal{G}$ be a generation algorithm that outputs a tuple (p, G, g) such that p is prime, G is a group of order p, and g is a generator of G. We say that G satisfies the k-Linear assumption if it is computationally infeasible to distinguish between tuples of the form $(g^{\alpha_1}, \ldots, g^{\alpha_{k+1}}, g^{\alpha_1 r_1}, \ldots, g^{\alpha_k r_k}, g^{\alpha_{k+1}\sum_{i=1}^{k} r_i})$ and tuples of the form $(g^{\alpha_1}, \ldots, g^{\alpha_{k+1}}, g^{\alpha_1 r_1}, \ldots, g^{\alpha_{k+1} r_{k+1}})$ for random $\alpha_i, r_i \leftarrow \mathbb{F}_p$. More formally, for all PPT adversaries $\mathcal{A}$ there exists a negligible function $\nu(\cdot)$ and a security parameter k_0 such that the following holds for all $k > k_0$:*

$$\left| \Pr\left[\begin{array}{l} (p,G,g)\leftarrow\mathcal{G}(1^k) \\ \alpha_1,\ldots,\alpha_{k+1}\xleftarrow{R}\mathbb{F}_p \\ r_1,\ldots,r_k\xleftarrow{R}\mathbb{F}_p \end{array} : \mathcal{A}(g^{\alpha_1}, \ldots, g^{\alpha_{k+1}}, g^{\alpha_1 r_1}, \ldots, g^{\alpha_{k+1}\sum_{i=1}^{k} r_i}) = 0 \right] \right.$$
$$\left. - \Pr\left[\begin{array}{l} (p,G,g)\leftarrow\mathcal{G}(1^k) \\ \alpha_1,\ldots,\alpha_{k+1}\xleftarrow{R}\mathbb{F}_p \\ r_1,\ldots,r_{k+1}\xleftarrow{R}\mathbb{F}_p \end{array} : \mathcal{A}(g^{\alpha_1}, \ldots, g^{\alpha_{k+1}}, g^{\alpha_1 r_1}, \ldots, g^{\alpha_{k+1} r_{k+1}}) = 0 \right] \right| < \nu(k),$$

Let G be a bilinear group of prime order p with a pairing $e\colon G \times G \to G_T$. When we refer to a "natural" use of the k-Linear assumption, we mean that we define the module B to be G^{k+1} and the module B_1 to be generated by k elements of B that span a rank-k submodule. Indeed, one way to interpret the k-Linear assumption is that a random element in the submodule B_1 of G^{k+1} generated by elements of the form $(1, \ldots, 1, g^{\alpha_i}, 1, \ldots, 1, g)$ for $i = 1, \ldots, k$ is indistinguishable from a random element of G^{k+1}. Our use of the assumption generalizes this interpretation only slightly, by randomizing the generators of B_1. Note that in our setup the quotient module $B_2 = B/B_1$ has $\mathbb{F}_p$-rank 1.

Following Freeman [24, §2], we define a (symmetric) pairing on B by setting $B_T = (G_T)^m$ for some integer m and choosing $(k+1) \times (k+1)$ (symmetric) matrices $E^{(\ell)}$ over $\mathbb{F}_p$ for $\ell = 1, \ldots, m$. We then set the ℓth component of the pairing to be

$$E\Big((g_1, \ldots, g_{k+1}), (h_1, \ldots, h_{k+1})\Big)^{(\ell)} := \prod_{i,j=1}^{k+1} e(g_i, h_j)^{e_{ij}^{(\ell)}}, \tag{5}$$

[8] Of course, the number of bits taken to represent the composite-order group element, approximately 1024, is much larger than it would be for a prime-order group element, which can be as small as 160 bits (at the 80-bit security level).

where $e_{ij}^{(\ell)}$ denotes the (i,j)-th entry of $E^{(\ell)}$. The connection between this setup and the k-Linear assumption is given by the following theorem:

Theorem 6.2 ([24, Theorem 2.5]). *Let G, B, B_1, B_T be as described above, with B_1 a uniformly random rank-k submodule of B. If G satisfies the k-Linear assumption, then a random element of B_1 is computationally indistinguishable from a random element in B.*

While any scheme based on Groth-Sahai proofs requires the projecting property of Definition 5.3 and the indistinguishability of elements in B_1 and B (i.e., the indistinguishability of hiding and binding commitment keys), our scheme also requires the cancelling property of Definition 5.2. In the remainder of this section, we show that for any k, the three properties (projecting, cancelling, and key indistinguishability) cannot simultaneously be obtained in prime-order groups using the k-Linear assumption as described above, except with negligible probability (over the choice of the module B_1).

We start by showing that the image of a symmetric pairing on a group G of prime order p must also have order p. In what follows, we denote by $E(B,B)$ the submodule of B_T generated by all elements of the form $E(x,y)$ for $x, y \in B$.

Lemma 6.3. *If G is a group of prime order p and $e\colon G \times G \to G_T$ is a nondegenerate symmetric bilinear map, then the order of $e(G,G)$ is p.*

Proof. We first observe that $e(G,G)$ has exponent p; to see this, note that since G has order p, we have $e(x,y)^p = e(x^p,y) = e(1,y) = 1$ for any $x, y \in G$. Since $e(G,G)$ has exponent p, any element is of the form $z = \prod_i e(x_i,y_i)^{c_i}$ for $x_i, y_i \in G$ and $c_i \in \mathbb{F}_p$. Since G is cyclic, we can write $x_i = g^{a_i}$ and $y_i = g^{b_i}$ for a generator g and unique $a_i, b_i \in \mathbb{F}_p$. By bilinearity, we can write $z = e(g,g)^{\sum_i a_i b_i c_i}$, and therefore $e(G,G)$ is a cyclic group generated by $e(g,g)$; the nondegeneracy of e implies that $e(g,g)$ has order p. □

Lemma 6.3 shows that by replacing G_T with $e(G,G)$, we may assume without loss of generality that G_T has order p. We make this assumption in the remainder of the section. We will also assume that the values used to define the pairing E on the module B are independent of the submodules B_1 and B_2; if they are not independent, then the fact that they are related to the (publicly known) generators of B_1 gives an adversary information about B_1 that could be used to break the indistinguishability assumption. Similarly, if the pairing depends on B_2, then the adversary may be able to use this information to compute an element $y \in B_2$; then given an element x in either B_1 or B, he could compute $E(x,y)$ and conclude that $x \in B_1$ if and only if the resulting value is 1.

We can now show that in the prime-order setting our indistinguishability restrictions on B and its submodules will, with high probability, yield a pairing E that can be either projecting or cancelling, but not both at the same time. Our approach is to construct a cancelling pairing and then show that it implies that the image of the pairing E is of order p. We will then show that this implies that the pairing cannot satisfy the projecting property.

In general, there are two methods in the literature for cancelling paired group elements. As seen in Section 5.3, the cancelling in the composite setting is fairly straightforward: it follows from the fact that the orders of the G_p and G_q subgroups are relatively prime. In a prime-order setting this is not an option, as every component (i.e., G, G_T, B, B_1, B_2, B_T) has exponent p. We therefore need to use linear combinations of exponents in order to successfully cancel elements. Our next result can be interpreted as showing that forming these linear combinations requires us to combine elements in the pairing and thus shrink the size of the pairing's image. To simplify notation, we state the proposition relative to the $(k-1)$-Linear assumption.

Proposition 6.4. *Let G be a bilinear group of prime order p with pairing $e\colon G \times G \to G_T$. Let B be the rank-k G-module G^k, let $B_T = (G_T)^m$ for some positive integer m, and let $E\colon B \times B \to B_T$ be a nondegenerate pairing defined as in (5). If B_1 is a uniformly random rank-$(k-1)$ submodule of B and E is a cancelling pairing that is independent of the decomposition $B = B_1 \times B_2$, then $e(B, B)$ has order p with overwhelming probability.*

Proof. To start, we write elements in B as either $\mathbf{a} = (a_1, \ldots, a_k)$ or $\mathbf{b} = (b_1, \ldots, b_k)$ with $a_i, b_i \in G$. Equivalently, we can fix a generator g of G and write $\mathbf{a} = (g^{\alpha_1}, \ldots, g^{\alpha_k})$ and $\mathbf{b} = (g^{\beta_1}, \ldots, g^{\beta_k})$ for exponents $\alpha_i, \beta_i \in \mathbb{F}_p$. As we saw in (5) above, the element $E(\mathbf{a}, \mathbf{b}) \in B_T$ is a tuple of elements of G_T, in which each entry is of the form $T = \prod_{i,j} e(a_i, b_j)^{e_{ij}}$. By assumption, the coefficients $e_{ij} \in \mathbb{F}_p$ are independent of the α_i and β_i values.

Suppose that $\mathbf{a} \in B_1$ and $\mathbf{b} \in B_2$; let us see what we require in order to have $T = 1$. Let $\mathbf{a}_1, \ldots, \mathbf{a}_{k-1}$ be a set of generators of B_1, and write $\mathbf{a}_u = (g^{\alpha_{u1}}, \ldots, g^{\alpha_{u,k-1}})$ for $u = 1, \ldots, k-1$. Then a general element of B_1 is given by $\mathbf{a} = \mathbf{a}^{r_1} \cdots \mathbf{a}^{r_{k-1}}$ for arbitrary $r_1, \ldots, r_{k-1} \in \mathbb{F}_p$. Since B_1 has rank $k-1$, the submodule B_2 has rank 1 and a general element of B_2 is given by $\mathbf{b} = (g^{\beta_1 t}, \ldots, g^{\beta_k t})$ for some fixed $\beta_1, \ldots, \beta_k \in \mathbb{F}_p$ and arbitrary $t \in \mathbb{F}_p$. Looking back at how our element T is computed in (5), we can see that the condition $T = 1$ is equivalent to

$$\sum_u r_u \left(\sum_i \alpha_{ui} \left(\sum_j e_{ij} \beta_j t \right) \right) = 0$$

In matrix notation, this is $\vec{r} \cdot A \cdot E \cdot \vec{b} \cdot t = 0$, where $\vec{r}$ is the row vector $(r_1, \ldots, r_{k-1})$, $E = [e_{ij}]$ is the $k \times k$ matrix specifying the pairing coefficients (denoted $E^{(\ell)}$ in (5)), $A = [\alpha_{ui}]$ is the $(k-1) \times k$ matrix whose rows are the vectors corresponding to the generators of B_1, and $\vec{b}$ is the column vector $(\beta_1, \ldots, \beta_k)$. Because this requirement must hold for all values of $\vec{r}$ and t, we can further reduce the equation to $A \cdot E \cdot \vec{b} = 0$. We now consider two different cases: when E is invertible and when E is singular.

We first consider the case where E is singular. The cancelling property requires that $A \cdot E \cdot \vec{b} = 0$. If $E \cdot \vec{b} = 0$, then the pairing is degenerate in this component, as *any* element paired with $\vec{b}$ will be 1. Therefore, this cannot be the only type

of element in the pairing tuple, or else the entire pairing would be degenerate. On the other hand, if $E \cdot \vec{b} \neq 0$, then since $A \cdot E \cdot \vec{b} = 0$, we see that $E \cdot \vec{b}$ is a nonzero vector in both the image of E and the kernel of A.

Next we consider the case where E is invertible, and consider not only the element T but also another element T' in the target tuple. The element T' will have its own associated coefficient matrix E', with the requirement that $A \cdot E' \cdot \vec{b} = 0$. Then we have $A \cdot E \cdot \vec{b} = A \cdot E' \cdot \vec{b} = 0$, which implies that $\vec{b}$ is contained in both the kernel of $A \cdot E$ and the kernel of $A \cdot E'$. Since A has rank $k-1$ and we are assuming E to be invertible, we know that the dimension of $\ker(A \cdot E)$ is 1. Furthermore, since E is invertible we can write

$$A \cdot E' \cdot \vec{b} = A \cdot (E \cdot E^{-1}) \cdot E' \cdot \vec{b} = A \cdot E \cdot (E^{-1} \cdot E' \cdot \vec{b}) = 0,$$

which implies that $E^{-1}E' \cdot \vec{b}$ is also contained in the kernel of $A \cdot E$. Since this kernel is one-dimensional, $E^{-1}E' \cdot \vec{b}$ must be a constant multiple of $\vec{b}$; i.e., $E^{-1}E' \cdot \vec{b} = \lambda \cdot \vec{b}$ for some $\lambda \in \mathbb{F}_p$ and $\vec{b}$ is an eigenvector of $E^{-1}E'$.

We now observe that because A has rank $k-1$, its kernel has rank one; furthermore, choosing a rank-$(k-1)$ submodule B_1 is equivalent to choosing the one-dimensional subspace $\ker(A)$. Since E is invertible and independent of B_1, this is equivalent to choosing the one-dimensional subspace $\ker(A \cdot E)$. Let $\vec{u}$ be any vector in $\ker(A \cdot E)$. Then $\vec{u} = \gamma \cdot \vec{b}$ for some $\gamma \in \mathbb{F}_p$, and our analysis above shows that $\vec{u}$ is an eigenvector of $E^{-1}E$. Since $\ker(A \cdot E)$ can contain *any* nonzero vector $\vec{u}$, this implies that *every* vector is an eigenvector of $E^{-1}E$. Therefore $E^{-1}E'$ must be a diagonal matrix with the same value in each diagonal entry; in other words, we have $E^{-1}E' = cI$ for some constant $c \in \mathbb{F}_p$. Thus we have $E' = cI \cdot E = c \cdot E$, and so $T' = T^c$.

It remains only to put everything together. Let $E^{(\ell)}$ be the coefficient matrix from (5) used to compute the ℓth component of the pairing. Our argument above shows that if *one* of the matrices $E^{(\ell)}$ is invertible, then *all* matrices $E^{(\ell')}$ are constant multiples of $E^{(\ell)}$, and therefore the order of $e(B,B)$ is the same as the order of $e(G,G) = G_T$, which is p. Thus if the pairing E is cancelling and the order of $e(B,B)$ is greater than p, then *none* of the matrices $E^{(\ell)}$ can be invertible.

Now suppose all of the $E^{(\ell)}$ are singular. Our consideration of this case above shows that if the pairing E is cancelling, then there must be *some* matrix $E^{(\ell)}$ with $\ker(A) \cap \mathrm{im}(E^{(\ell)}) \neq \{0\}$. As noted above, choosing the module B_1 is equivalent to choosing the one-dimensional subspace $\ker A$. Since $E^{(\ell)}$ is not invertible, we have $\dim(\mathrm{im}(E^{(\ell)})) \leq k-1$. Thus the number of one-dimensional subspaces in $\mathrm{im}(E^{(\ell)})$ is at most $(p^{k-1}-1)/(p-1)$, while the number of one-dimensional subspaces in $\mathbb{F}_p^k$ is $(p^k-1)/(p-1)$. We conclude that the probability (taken over a uniformly random choice of $\ker(A)$ and thus also of A) that $\ker(A)$ has nontrivial intersection with the image of $E^{(\ell)}$ is at most $(p^{k-1}-1)/(p^k-1) < 2/p$. Taking a union bound, we conclude that the probability that $\ker(A) \cap \mathrm{im}(E^{(\ell)}) \neq 0$ for *some* ℓ is at most $2m/p$, which is negligible. □

Putting all this together, we can prove our main theorem:

Theorem 6.5. *Let G be a bilinear group of prime order p with pairing $e\colon G \times G \to G_T$. Let B be the rank-k G-module G^k, let $B_T = (G_T)^m$ for some positive integer m, and let $E\colon B \times B \to B_T$ be a nondegenerate pairing defined as in (5). If B_1 is a uniformly random rank-$(k-1)$ submodule of B and E is a cancelling pairing that is independent of the decomposition $B = B_1 \times B_2$, then with overwhelming probability the pairing E cannot be projecting (with respect to the same decomposition $B = B_1 \times B_2$).*

Proof. By Proposition 6.4, we know that if E is cancelling, then $E(B, B)$ has order p with overwhelming probability. This means that $E(B, B)$ is cyclic and any nonzero element is a generator.

Suppose E is projecting and choose some $x \in B_1$. Since E is nondegenerate, there is some $y \in B$ such that $E(x, y) \neq 1$. Now the projecting property implies that $\pi_T(E(x, y)) = E(\pi(x), \pi(y)) = E(1, \pi(y)) = 1$. Since $E(x, y)$ generates $E(B, B)$, we conclude that $\pi_T(E(B, B)) = \{1\}$.

On the other hand, now choose some $x' \in B_2$. Then there is some $y' \in B$ such that $E(x', y') \neq 1$. Furthermore, the cancelling property implies that without loss of generality we can assume $y' \in B_2$. The projecting property now implies that $\pi_T(E(x', y')) = E(\pi(x'), \pi(y')) = E(x', y') \neq 1$, so we conclude that $\pi_T(E(B, B)) = E(B, B)$, contradicting our conclusion above. □

7 Conclusions and Open Problems

In this paper we have shown that there are limitations on transformations of pairing-based cryptosystems from composite- to prime-order groups. In particular, we have given evidence that two properties of composite-order pairings identified by Freeman — cancelling and projecting — cannot be simultaneously obtained in prime-order groups.

Specifically, we have shown that a pairing defined in a natural way with subgroup hiding provided by the Decision Linear assumption can be both cancelling and projecting with only negligible probability. As evidence that both properties are sometimes called for simultaneously, we have presented a natural cryptographic scheme whose proof of security calls for a pairing that is both cancelling and projecting. This scheme is a practical round-optimal blind (and partially blind) signature secure in the common reference string model, under mild assumptions and without random oracles.

Many open questions remain. First, we would like either to generalize our result so it applies to a wider class of pairings constructed from prime order groups (possibly including asymmetric pairings), or instead to show that no such generalization is possible by exhibiting a pairing in prime-order groups that is simultaneously projecting and cancelling. Second, we have given evidence that our specific proof strategy for our blind signature scheme is unlikely to generalize to prime-order groups, but have not settled the question of whether our scheme when instantiated in prime-order groups is in fact provably secure (by means of a different, ad-hoc proof) or insecure (i.e., actually susceptible to attack). Finally, it is interesting to consider whether a more general procedure (not relying on

Freeman's properties) can be used to transform every composite-order scheme to a prime-order one, or whether some schemes provably cannot be so transformed.

Acknowledgements. We are grateful to Melissa Chase for helpful discussions about this work and to our anonymous reviewers for their helpful comments. The first author is supported in part by a MURI grant administered by the Air Force Office of Scientific Research and in part by a graduate fellowship from the Charles Lee Powell Foundation. The third author is supported by an NSF Mathematical Sciences Postdoctoral Research Fellowship.

References

1. Abdalla, M., Namprempre, C., Neven, G.: On the (im)possibility of blind message authentication codes. In: Pointcheval, D. (ed.) CT-RSA 2006. LNCS, vol. 3860, pp. 262–279. Springer, Heidelberg (2006)
2. Abe, M., Fuchsbauer, G., Groth, J., Haralambiev, K., Ohkubo, M.: Structure-preserving signatures and commitments to group elements. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 209–236. Springer, Heidelberg (2010)
3. Abe, M., Fujisaki, E.: How to date blind signatures. In: Kim, K., Matsumoto, T. (eds.) ASIACRYPT 1996. LNCS, vol. 1163, pp. 244–251. Springer, Heidelberg (1996)
4. Abe, M., Haralambiev, K., Ohkubo, M.: Signing on elements in bilinear groups for modular protocol design. Cryptology ePrint Archive, Report 2010/133 (2010), http://eprint.iacr.org/
5. Abe, M., Ohkubo, M.: A framework for universally composable non-committing blind signatures. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 435–450. Springer, Heidelberg (2009)
6. Abe, M., Okamoto, T.: Provably secure partially blind signatures. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 271–286. Springer, Heidelberg (2000)
7. Ballard, L., Green, M., de Medeiros, B., Monrose, F.: Correlation-resistant storage via keyword-searchable encryption. Cryptology ePrint Archive, Report 2005/417, (2005), http://eprint.iacr.org/
8. Baric, 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)
9. Bellare, M., Namprempre, C., Pointcheval, D., Semanko, M.: The power of RSA inversion oracles and the security of Chaum's RSA-based blind signature scheme. In: Syverson, P.F. (ed.) FC 2001. LNCS, vol. 2339, pp. 319–338. Springer, Heidelberg (2002)
10. Boldyreva, A.: Threshold signature, multisignature and blind signature schemes based on the gap-Diffie-Hellman-group signature scheme. In: Desmedt, Y. (ed.) PKC 2003. LNCS, vol. 2567, pp. 31–46. Springer, Heidelberg (2002)
11. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)
12. 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)
13. Boneh, D., Rubin, K., Silverberg, A.: Finding composite order ordinary elliptic curves using the Cocks-Pinch method. Cryptology ePrint Archive, Report 2009/533, (2009), http://eprint.iacr.org/2009/533

14. Boudot, F.: Efficient proofs that a committed number lies in an interval. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 431–444. Springer, Heidelberg (2000)
15. Boyen, X., Waters, B.: Compact group signatures without random oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 427–444. Springer, Heidelberg (2006)
16. Cao, T., Lin, D., Xue, R.: A randomized RSA-based partially blind signature scheme for electronic cash. Computers and Security 24(1), 44–49 (2005)
17. Chaum, D.: Blind signatures for untraceable payments. In: Chaum, D., Rivest, R., Sherman, A. (eds.) Proceedings of Crypto 1982, pp. 199–204. Plenum Press, New York (1983)
18. Chaum, D.: Blind signature system (abstract). In: Chaum, D. (ed.) Proceedings of Crypto 1983, p. 153. Plenum Press, New York (1984)
19. Chaum, D.: Elections with unconditionally-secret ballots and disruption equivalent to breaking RSA. In: Günther, C. (ed.) EUROCRYPT 1988. LNCS, vol. 330, pp. 177–182. Springer, Heidelberg (1988)
20. Chaum, D., Fiat, A., Naor, M.: Untraceable electronic cash. In: Goldwasser, S. (ed.) CRYPTO 1988. LNCS, vol. 403, pp. 319–327. Springer, Heidelberg (1990)
21. Dummit, D., Foote, R.: Abstract Algebra, 2nd edn. Prentice-Hall, Upper Saddle River (1999)
22. Fischlin, M.: Round-optimal composable blind signatures in the common reference string model. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 60–77. Springer, Heidelberg (2006)
23. Freeman, D., Scott, M., Teske, E.: A taxonomy of pairing-friendly elliptic curves. J. Cryptology 23(2), 224–280 (2010)
24. Freeman, D.M.: Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 44–61. Springer, Heidelberg (2010)
25. Fuchsbauer, G.: Automorphic signatures in bilinear groups and an application to round-optimal blind signatures. Cryptology ePrint Archive, Report 2009/320 (2009), http://eprint.iacr.org/
26. Galindo, D., Herranz, J., Kiltz, E.: On the generic construction of identity-based signatures with additional properties. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 178–193. Springer, Heidelberg (2006)
27. Garg, S., Sahai, A., Waters, B.: Efficient fully collusion-resilient traitor tracing scheme. Cryptology ePrint Archive, Report 2009/532 (2009), http://eprint.iacr.org/2009/532
28. Green, M., Hohenberger, S.: Blind identity-based encryption and simulatable oblivious transfer. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 265–282. Springer, Heidelberg (2007)
29. Groth, J., Ostrovsky, R., Sahai, A.: Non-interactive zaps and new techniques for NIZK. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 97–111. Springer, Heidelberg (2006)
30. Groth, J., Ostrovsky, R., Sahai, A.: Perfect non-interactive zero knowledge for NP. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 339–358. Springer, Heidelberg (2006)
31. 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)

32. Hazay, C., Katz, J., Koo, C.-Y., Lindell, Y.: Concurrently-secure blind signatures without random oracles or setup assumptions. In: Vadhan, S. (ed.) TCC 2007. LNCS, vol. 4392, pp. 323–341. Springer, Heidelberg (2007)
33. 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)
34. Juels, A., Luby, M., Ostrovsky, R.: Security of blind digital signatures. In: Kaliski Jr., B. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 150–164. Springer, Heidelberg (1997)
35. Kiayias, A., Zhou, H.-S.: Concurrent blind signatures without random oracles. In: Yung, M. (ed.) SCN 2006. LNCS, vol. 4116, pp. 49–62. Springer, Heidelberg (2006)
36. Lysyanskaya, A., Ramzan, Z.: Group blind digital signatures: A scalable solution to electronic cash. In: Hirschfeld, R. (ed.) FC 1998. LNCS, vol. 1465, pp. 184–197. Springer, Heidelberg (1998)
37. Lysyanskaya, A., Rivest, R., Sahai, A., Wolf, S.: Pseudonym systems. In: Heys, H., Adams, C. (eds.) SAC 1999. LNCS, vol. 1758, pp. 184–199. Springer, Heidelberg (2000)
38. Martinet, G., Poupard, G., Sola, P.: Cryptanalysis of a partially blind signature scheme, or How to make $100 bills with $1 and $2 ones. In: Crescenzo, G.D., Rubin, A. (eds.) FC 2006. LNCS, vol. 4107, pp. 171–176. Springer, Heidelberg (2006)
39. Meiklejohn, S., Shacham, H., Freeman, D.M.: Limitations on transformations from composite-order to prime-order groups: the case of round-optimal blind signatures. Cryptology ePrint Archive, Report 2010/474 (2010), http://eprint.iacr.org/2010/474
40. Okamoto, T.: Efficient blind and partially blind signatures without random oracles. Cryptology ePrint Archive, Report 2006/102 (2006), http://eprint.iacr.org/
41. Okamoto, T.: Efficient blind and partially blind signatures without random oracles. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 80–99. Springer, Heidelberg (2006)
42. 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)
43. Paterson, K., Schuldt, J.: Efficient identity-based signatures secure in the standard model. In: Batten, L.M., Safavi-Naini, R. (eds.) ACISP 2006. LNCS, vol. 4058, pp. 207–222. Springer, Heidelberg (2006)
44. Pointcheval, D., Stern, J.: Security arguments for digital signatures and blind signatures. J. Cryptology 13(3), 361–396 (2000)
45. Shacham, H.: A Cramer-Shoup encryption scheme from the linear assumption and from progressively weaker linear variants. Cryptology ePrint Archive, Report 2007/074 (2007), http://eprint.iacr.org/
46. Waters, B.: Efficient identity-based encryption without random oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)
47. Zhang, F., Kim, K.: ID-based blind signature and ring signature from pairings. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 533–547. Springer, Heidelberg (2002)

The Semi-Generic Group Model and Applications to Pairing-Based Cryptography*

Tibor Jager[1] and Andy Rupp[2]

[1] Horst Görtz Institute for IT Security, Ruhr-University Bochum, Germany
tibor.jager@rub.de
[2] University of Trier, Germany
andy.rupp@rub.de

Abstract. In pairing-based cryptography the Generic Group Model (GGM) is used frequently to provide evidence towards newly introduced hardness assumptions. Unfortunately, the GGM does not reflect many known properties of bilinear group settings and thus hardness results in this model are of limited significance. This paper proposes a novel computational model for pairing-based cryptography, called the Semi-Generic Group Model (SGGM), that is closer to the standard model and allows to make more meaningful security guarantees. In fact, *the best algorithms currently known for solving pairing-based problems are semi-generic* in nature. We demonstrate the usefulness of our new model by applying it to study several important assumptions (BDDH, Co-DH). Furthermore, we develop master theorems facilitating an easy analysis of other (future) assumptions. These master theorems imply that (unless there are better algorithms than the semi-generic ones) great parts of the zoo of novel assumptions over bilinear groups are reducible to just *two* (more or less) standard assumptions over finite fields. Finally, we examine the appropriateness of the SGGM as a tool for analyzing the security of *practical* cryptosystems without random oracles by applying it to the BLS signature scheme.

Keywords: Restricted models of computation, generic groups, semi-generic group model, cryptographic assumptions, master theorems, provable security, pairing-based cryptography.

1 Introduction

Assuming that certain computational problems, mostly from algebra, number theory, and coding theory, are intractable builds the foundation of public-key cryptography. However, proving the validity of these assumptions in the standard model of computation seems to be impossible with currently available techniques.

Why do we believe in such hardness assumptions, though they are not provable in general? For classic number-theoretic problems, such as integer factorization (IF) or the discrete logarithm (DL) problem, this is certainly due to the absence of efficient

* This is an extended abstract, see [20] for the full version. This research has been supported by the European Community (FP7/2007-2013) under grant agreement number ICT-2007-216646 - European Network of Excellence in Cryptology II (ECRYPT II).

M. Abe (Ed.): ASIACRYPT 2010, LNCS 6477, pp. 539–556, 2010.

algorithms in spite of intensive long-term research by many brilliant people. However, besides such well-known assumptions, there frequently appear new assumptions building the basis for novel cryptosystems with original properties. What can be done to provide evidence for these assumptions apart from trying to find efficient algorithms over decades? Clearly, we should try to underpin the belief in novel assumptions by searching for reductions to a more mature assumption; but unfortunately finding such a reduction often fails.

An important approach to (nevertheless) gain immediate evidence towards hardness assumptions is to prove them with respect to a restricted but still meaningful class of algorithms. This is the motivation behind the invention of black-box models for algebraic structures like groups, fields, and rings, where algorithms are limited to perform only operations "commonly" available over these structures. Probably, the most famous of these models is the *generic group model (GGM)* introduced by Shoup in his seminal paper [34] from 1997, and refined by Maurer in [28]. In this model one considers algorithms – so-called *generic group algorithms* – that, given a group $\mathbb{G}$ as black-box, may only perform a set of basic operations on the elements of $\mathbb{G}$ such as applying the group law, inversion of group elements and equality testing. Since the group is treated as a black-box, the algorithms cannot exploit any special properties of a concrete group representation. As a consequence, such algorithms are generic in the sense that they can be applied to any concrete instantiation of a group (e.g., $\mathbb{Z}_p^*$ or $E(\mathbb{F}_p)$) in order so solve a problem. Natural examples of this class of algorithms are the Pohlig-Hellman [30] and Pollard's Rho [31] algorithm for computing discrete logarithms.

It should be noted that one has to take care when interpreting results in the GGM like intractability results as evidence in practice, since this model abstracts away from potentially many properties an algorithm might be able to exploit in the real world [15]. On the one hand, there exist cryptographic groups (such as certain elliptic curve groups) for which not many properties beyond the axioms of an algebraic group are known. Hence, modeling such groups as generic can be seen as a reasonable abstraction. On the other hand, there are groups, also used in cryptography, featuring many further properties, which clearly makes the generic model an inappropriate reflection for them. A prime example are multiplicative groups of finite fields or rings. These structures offer many well-understood properties beyond the group axioms, such as additional efficient algebraic operations (e.g., addition in the field or ring), and other properties of the group representation (e.g., the notion of prime integers and irreducible polynomials), that are simply ignored by the generic group model, but give rise to more efficient algorithms for certain problems (e.g., index calculus algorithms for computing discrete logarithms).

But should a minimal requirement on such an idealized model of computation not be that at least all currently known algorithms are captured? There exist some first approaches in the cryptographic literature to tackle this issue: The pseudo-free group model proposed by Hohenberger [19] and Rivest [32] does not treat a group as a black-box. Unfortunately, the definition of pseudo-freeness is very restrictive in the sense that a number of important groups (like all known-order groups) are immediately excluded and important problems, such as Diffie-Hellman-type problems, seem not to be covered. Other approaches due to Leander and Rupp [27] and Aggarwal and Maurer [1] take into account that the RSA group $\mathbb{Z}_n^*$ is embedded in the ring $\mathbb{Z}_n$. They use

a generic *ring* model, where algorithms may perform both multiplication and addition operations on $\mathbb{Z}_n$ to show that breaking RSA is equivalent to factoring. Unfortunately, recent work [21] shows that even computing the *Jacobi symbol* is equivalent to factoring in this model. So this approach has not led to a satisfying abstraction of $\mathbb{Z}_n^*$ yet.

Over the last decade a considerable number of innovative cryptosystems, such as identity-based encryption [7] or short digital signatures with strong security [9,10], have been proposed over bilinear groups. A bilinear group setting consists of groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_3$, with a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_3$, called a pairing. Along with these cryptosystems also many new assumptions have been introduced, e.g., Bilinear Diffie-Hellman (BDH) [23,24], q-Strong Diffie-Hellman [4,14,22], Decision Linear Diffie-Hellman (DLIN) [6], Co-Diffie-Hellman (Co-DH) [9,8], and many more. Unfortunately, for virtually all of them no reduction to a well-analyzed assumption like DL is known. In fact, finding such reductions seems to be a difficult task, since the algebraic settings underlying classic problems (e.g., a single cyclic group for DL) significantly differ from bilinear settings. Hence, given an instance of a classic problem, it appears to be hard to transform this instance to one of the bilinear problem in order to leverage an algorithm for the latter.

Consequently, the only way to provide some immediate evidence for such novel assumptions consists in proofs in restricted models of computation. So far, the only such model for bilinear settings is a straightforward extension of the generic group model, where all three groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_3$ are modeled as generic groups [33,11,25]. In all known instances of bilinear settings the groups $\mathbb{G}_1$ and $\mathbb{G}_2$ are elliptic curve groups, thus modeling these groups as generic may be considered as a reasonable abstraction. However, in contrast to that, the group $\mathbb{G}_3$ is usually a subgroup of the multiplicative group of a finite field. So there definitely exist non-generic algorithms for cryptographic problems like BDH, Co-DH, etc. featuring a running time which is *at most* sub-exponential: these sub-exponential algorithms map the inputs over $\mathbb{G}_1$ and $\mathbb{G}_2$ (given as part of a problem instance) to $\mathbb{G}_3$ using the bilinear mapping (MOV reduction [29]) and determine the discrete logarithms of these elements over $\mathbb{G}_3$ using index calculus. Knowledge of these discrete logarithms allows to compute the solution to the problem instance using a few exponentiations. Note that there might be even more efficient algorithms especially for potentially easier problems like decisional or gap problems. Hence, modeling bilinear settings in this way is clearly inappropriate.

OUR CONTRIBUTION. We propose the *Semi-Generic Group Model (SGGM)* which leverages this observation as follows: The elliptic curve groups $\mathbb{G}_1$ and $\mathbb{G}_2$ are modeled as generic groups, while $\mathbb{G}_3$ is given in the standard model, i.e., algorithms may perform *any* computation over $\mathbb{G}_3$ that is possible in the subgroup of a finite field. The SGGM is thus closer to the standard model than the GGM and can provide stronger evidence towards hardness assumptions in pairing-based cryptography. In fact, to the best of our knowledge all algorithms currently known for solving pairing-based problems are semi-generic in nature. In particular, the sub-exponential algorithms applying a MOV reduction described above are covered by the SGGM.

We analyzed some of the most important computational and decisional assumptions of pairing-based cryptography in our new model. In this extended abstract we restrict to consider Co-DH and decisional BDH. The full version of the paper [20] covers

additional problems, including q-strong DH and DLIN. We are able to reduce the considered assumptions (with respect to semi-generic algorithms) to fairly standard assumptions over finite fields like Square DH and a slight variation of DL. That means, the bilinear assumptions are at least as hard as certain more standard assumption over $\mathbb{G}_3$ provided that there are no non-semi-generic algorithms. Furthermore, we developed master theorems ensuring the hardness of broad classes of computational and decisional problems in the SGGM. Studying such generalizations is not only important in order to structure and facilitate the analysis of the rapidly growing set of cryptographic assumptions as motivated in [3], but improves our understanding of the properties which need to be satisfied by a problem to be intractable. Results like [12,33,11] are in this vein. Boyen [11] (see also [5]) developed master theorems for the hardness of some general classes of decisional problems in the generic group model for bilinear settings. Rupp et al. [33] provide hardness conditions for even broader classes of computational problems and algebraic settings, but still in the GGM. Bresson et al. [12] study a general class of decisional assumptions over a single group in the standard model and show that this class can be reduced to DDH (under certain restrictions). In the scope of the proof of our master theorem for decisional problems we enhance Bresson et al.'s results for the standard model and apply them to the SGGM.

The security of public-key cryptosystems, especially of *practical* cryptosystems, can often only be proven in an idealized model, such as the random oracle model (ROM) [2]. An issue with the ROM is that it idealizes a hash function in a way such that it has *all* properties of a "perfect" hash function (collision resistance, (second) preimage resistance, random output, ...) at the same time. When the cryptosystem (and thus the random oracle) is implemented in practice, one has to choose an adequate hash function instantiating the random oracle. An important question is whether providing *all* properties of the random oracle at the same time is really necessary to provide security.

We examine the useability of the SGGM as a tool complementing the ROM. We are able to prove the security of the Boneh-Lynn-Shacham (BLS) short signature scheme [9,10] against semi-generic adversaries without random oracles, however, requiring non-standard properties for the employed hash function. It is left as an interesting open problem to study whether these requirements can actually be satisfied by a reasonably efficient practical hash function.

2 The Semi-Generic Group Model

Let $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_3$ be groups of prime order p and $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$ be corresponding generators. For the sake of simplicity of the subsequent formalizations we use multiplicative notation for all groups.

Definition 1. *A* pairing *is a map* $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$ *with the following properties:*

1. Bilinearity*:* $\forall (a, b) \in \mathbb{G}_1 \times \mathbb{G}_2$ *and* $x_1, x_2 \in \mathbb{Z}_p$ *holds that* $e(a^{x_1}, b^{x_2}) = e(a, b)^{x_1 x_2}$.
2. Non-degeneracy*:* $g_3 := e(g_1, g_2)$ *is a generator of* $\mathbb{G}_3$*, i.e.,* $g_3 \neq 1$.
3. e *is efficiently computable.*

Following [17], we distinguish three different types of bilinear group settings:

- *Type 1:* $\mathbb{G}_1 = \mathbb{G}_2$. *We will call this the setting with* symmetric *bilinear map.*
- *Type 2:* $\mathbb{G}_1 \neq \mathbb{G}_2$, *there is an* efficiently computable *isomorphism* $\psi : \mathbb{G}_1 \to \mathbb{G}_2$.
- *Type 3:* $\mathbb{G}_1 \neq \mathbb{G}_2$, *there is no efficiently computable isomorphism* $\psi : \mathbb{G}_1 \to \mathbb{G}_2$.

FORMAL DEFINITION OF THE SGGM. We base our formal description of the SGGM for bilinear settings on the generic group model introduced by Maurer [28], though our proofs can be adapted to Shoup's GGM [34] as well. The main difference between Maurer's and Shoup's formalization is that in the first model group elements are encoded deterministically whereas in the second model encodings are random.

An algorithm $\mathcal{A}$ in the SGGM interacts with a *semi-generic group oracle* $\mathcal{O}$, which computes the group operations and evaluates the pairing and isomorphism on behalf of $\mathcal{A}$. $\mathcal{O}$ receives as input two vectors of group elements (the problem instance)

$$I_1 = (a_{1,1}, \ldots, a_{1,k_1}) \in \mathbb{G}_1^{k_1} \quad \text{and} \quad I_2 = (a_{2,1}, \ldots, a_{2,k_2}) \in \mathbb{G}_2^{k_2} .$$

It maintains two lists $\mathcal{E}_1 \subseteq \mathbb{G}_1$ and $\mathcal{E}_2 \subseteq \mathbb{G}_2$, with $\mathcal{E}_{i,j}$ denoting the j-th entry of list $\mathcal{E}_i$, which are initialized such that $\mathcal{E}_{i,j} := a_{i,j}$ for all possible (i, j). We denote with $[a]_i$ the smallest index j (also called encoding) such that $\mathcal{E}_{i,j} = a$. Index $[a]_i$ is undefined, if $a \notin \mathcal{E}_i$. We may always assume that semi-generic algorithms only provide defined indices as input to the oracle. During initialization of the lists $\mathcal{E}_1$ and $\mathcal{E}_2$, the corresponding indices pointing to the contained elements are sent to the algorithm.

The oracle implements the following *public procedures*, which may be called by $\mathcal{A}$:

- `GroupOp`$([a]_i, [b]_i, i)$: This procedure takes as input two indices $[a]_i, [b]_i$ and a list index i. It determines the group elements $a, b \in \mathbb{G}_i$ by list lookup, computes $c = a \cdot b \in \mathbb{G}_i$, appends c to $\mathcal{E}_i$, and returns $[c]_i$.
- `BilinearMap`$([a]_1, [b]_2)$: This procedure takes as input two indices $[a]_1, [b]_2$. It determines the corresponding group elements $a \in \mathbb{G}_1, b \in \mathbb{G}_2$ by list lookup and returns $e(a, b)$ in the *standard* representation of $\mathbb{G}_3$ (i.e., as finite field element).

When considering Type 2 settings the algorithm may also query to apply the isomorphism ψ to an element of $\mathbb{G}_1$:

- `Isomorphism`$([a]_1)$: This procedure takes as input an index $[a]_1$, determines the element $a \in \mathbb{G}_1$, computes $b = \psi(a)$, appends b to $\mathcal{E}_2$ and returns $[b]_2$.

Note that a random group element can be efficiently sampled by a semi-generic algorithm by using `GroupOp`$(\cdot)$ to raise the generator (which is always part of a problem instance) to some $r \xleftarrow{\$} \mathbb{Z}_p$.

2.1 Essential Ingredients for Proofs in the SGGM

This section describes a few general observations that will turn out to be the essential ingredients for proofs in the semi-generic model.

Observation 1: Components inside oracle are exchangeable. Semi-generic algorithms due to its nature are "blind" with respect to the internal details of the groups

$\mathbb{G}_1$ and $\mathbb{G}_2$ as well as the pairing e and the isomorphism ψ. These components are hidden within a black-box. Hence, we can plug-in "something else" for these components as long as these replacements behave like cyclic groups with a bilinear map and an isomorphism. We will utilize this observation in a novel way to map inputs given over $\mathbb{G}_3$ back to $\mathbb{G}_1$ and $\mathbb{G}_2$ by setting $\mathbb{G}_1 := \mathbb{G}_2 := \mathbb{G}_3$ internally and simulating a virtual bilinear map $e : \mathbb{G}_3 \times \mathbb{G}_3 \rightarrow \mathbb{G}_3$ and isomorphism $\psi : \mathbb{G}_3 \rightarrow \mathbb{G}_3$.

Observation 2: Computed elements over $\mathbb{G}_1$ and $\mathbb{G}_2$ are linear polynomials in initial inputs. Let $I_1 \in \mathbb{G}_1^m$ and $I_2 \in \mathbb{G}_2^n$ be inputs given to the semi-generic oracle (as part of a problem instance). We have $I_2 = I_1$ in the case of a Type 1 setting. In the following, we always assume that at least the generators g_1 and g_2 are given (as the first components of these input tuples). So we can write $I_1 = (g_1, g_1^{x_2}, \ldots, g_1^{x_m})$ and $I_2 = (g_2, g_2^{y_2}, \ldots, g_2^{y_n})$ for some *unknown* $x_j, y_k \in \mathbb{Z}_p$ (no assumptions about their distribution are made here) and $\psi(g_1) = g_2$ in the case of a Type 2 setting. Then we define the tuple $I_1' := I_1$ and the tuple $I_2' := I_2$ in the case of a Type 1 and Type 3 setting or $I_2' := (g_2, g_2^{x_2}, \ldots, g_2^{x_m}, g_2^{y_2}, \ldots, g_2^{y_n})$ for a Type 2 setting. These tuples are called the *initial inputs* to semi-generic algorithms. Using this notation, we can describe the following observation: Over $\mathbb{G}_1$ and $\mathbb{G}_2$ a semi-generic algorithm can only perform the group law on the initial inputs. Thus, any element $a \in \mathbb{G}_i$ ($i \in \{1, 2\}$) computed by a semi-generic algorithm is a product of the elements in I_i'. Hence we can represent such an element as $a = g_i^{P(x_2,\ldots,x_m,y_2,\ldots,y_n)}$ for some linear multivariate polynomial $P = \alpha_1 + \sum_{j=2}^{m} \alpha_j X_j + \sum_{j=2}^{n} \beta_j Y_j$, where the β_j are zero in the case $i = 1$ or if we consider a Type 1 setting. It is important to observe that all coefficients of this polynomial are *known* to the oracle.

Observation 3: Pairing is simulatable knowing images of initial inputs. Let $a \in \mathbb{G}_1$ and $b \in \mathbb{G}_2$ be two elements computed by a semi-generic algorithm. Then by using the above observation and setting $x_1 := 1$ it is easy to see that

$$\begin{aligned} e(a,b) &= e(g_1^{\sum_{i=1}^{m} \alpha_i x_i}, g_2^{\sum_{j=1}^{m} \alpha_j' x_j + \sum_{k=2}^{n} \beta_k' y_k}) \\ &= \prod_{i=1}^{m} \prod_{j=1}^{m} e(g_1^{x_i}, g_2^{x_j})^{\alpha_i \alpha_j'} \cdot \prod_{i=1}^{m} \prod_{k=2}^{n} e(g_1^{x_i}, g_2^{y_k})^{\alpha_i \beta_k'} \end{aligned}$$

From this equation it follows that by *knowing the images of the initial inputs* under the pairing, one can *compute the output of* e *on arbitrary inputs* provided by a semi-generic algorithm without actually evaluating the pairing explicitly. In other words, an oracle equipped with a table containing $e(a, b)$ for all combinations of a in I_1' and b in I_2' would be able to handle all `BilinearMap` queries.

3 Analysis of Selected Problems in the Semi-Generic Model

In this section we exemplarily analyze the hardness of the computational Co-DH and the decisional BDH problem. Certainly, the list of problems we are considering here is by no means complete. Our main purpose is to give concrete analyses of some important problems of bilinear cryptography, thereby illustrating the basic ideas and techniques underlying proofs in this model, before dealing with the more intricate case of general classes of problems in Section 4.

3.1 Reducing 2-DL to Co-DH

The Co-DH problem has been used in [9,8] for the construction of short and aggregate signatures over bilinear groups. Over a Type 2 setting it can be defined as follows: Given $(g_1, g_1^{x_0}, g_2, g_2^{x_1}, g_3)$, where $(x_0, x_1) \xleftarrow{\$} \mathbb{Z}_p^2$ are secret random choices, output $g_2^{x_0 x_1}$.

It is easy to see that in order to prove something about the hardness of Co-DH, we definitely need to make the assumption that the discrete logarithm problem over $\mathbb{G}_3$ is intractable. But is this enough? Our answer is "not quite": We are going to relate the hardness of Co-DH to the 2-DL problem over $\mathbb{G}_3$, a slightly easier variant of DL. The q-DL problem can be defined as follows: Given $(g_3, g_3^x, \ldots, g_3^{x^q})$, where $x \xleftarrow{\$} \mathbb{Z}_p$ is a secret random value, output x. The additional input $g_3^{x^2}$ (in comparison to standard DL) is needed in order to be able to simulate the pairing when running the Co-DH algorithm.

Theorem 1. *Suppose there exists a semi-generic group algorithm $\mathcal{A}$ solving* Co-DH *over a Type 2 bilinear group setting in time t with success probability ϵ. Then there exists an algorithm $\mathcal{B}$ solving the* 2-DL *problem over $\mathbb{G}_3$ in time $t' \approx t$ with success probability $\epsilon' \geq \frac{1}{2}\epsilon$.*

Proof. Given an instance of the 2-DL problem, $\mathcal{B}$ sets up an instance of the Co-DH problem in the semi-generic model in a way that it can leverage a solution to Co-DH computed by $\mathcal{A}$ to solve the 2-DL instance. In particular, $\mathcal{B}$ will play the role of the semi-generic oracle. We exploit Observation 1 from Section 2.1 to setup such an useful instance: Since $\mathcal{A}$ is "blind" with respect to the internal details of $\mathbb{G}_1$, $\mathbb{G}_2$, e, and ψ, we set $\mathbb{G}_1 := \mathbb{G}_2 := \mathbb{G}_3$ and try to simulate a virtual bilinear map $e : \mathbb{G}_3 \times \mathbb{G}_3 \to \mathbb{G}_3$.

We are now ready to describe our reduction algorithm $\mathcal{B}$. $\mathcal{B}$ takes as input an instance $a_0 := g_3, a_1 := g_3^x, a_3 := g_3^{x^2}$ of the 2-DL problem over $\mathbb{G}_3$. Then it chooses $i^* \xleftarrow{\$} \{0,1\}$, $x_{1-i^*} \xleftarrow{\$} \mathbb{Z}_p$ and sets $a_2 := g_3^{x_{1-i^*}}$. The wanted discrete logarithm x is now embedded as the implicit secret choice x_{i^*} in an instance of the Co-DH problem. More precisely, $\mathcal{B}$ sets up a problem instance and simulates the oracle $\mathcal{O}$ as follows:

- The lists $\mathcal{E}_1$ and $\mathcal{E}_2$ are initialized with $g_3, g_3^{x_0}$ and $g_3, g_3^{x_1}$, respectively, where $g_3^{x_{i^*}}$ is set to be a_1. The indices $[g_3]_1, [g_3^{x_0}]_1, [g_3]_2, [g_3^{x_1}]_2$, and g_3 are sent out to $\mathcal{A}$.
- `GroupOp` can be simulated since $\mathcal{B}$ knows how to perform the group law over $\mathbb{G}_3$.
- `Isomorphism`$([a]_1)$ can be simulated by looking up a in $\mathcal{E}_1$, appending it to $\mathcal{E}_2$, and then determining the index $[a]_2$.
- Using Observation 3 from Section 2.1, we can easily see that `BilinearMap` can be simulated: Let $[b]_1, [c]_2$ be the two indices given as input to the procedure by $\mathcal{A}$. Then we can write

$$e(b,c) = e(g_3^{\sum_{j=-1}^{0} z_j x_j}, g_3^{\sum_{k=-1}^{1} z'_k x_k}) = \prod_{j=-1}^{0} \prod_{k=-1}^{1} (g_3^{x_j x_k})^{z_j z'_k}$$

  where $x_{-1} := 1$ and z_j and z'_k are known to $\mathcal{B}$. Since $\mathcal{B}$ is given $a_0, \ldots, a_3$ and knows i^*, x_{1-i^*}, it can compute the required elements $g_3, g_3^{x_0}, g_3^{x_1}, g_3^{x_0^2}, g_3^{x_0 x_1}$ to simulate the pairing: $g_3 = a_0$, $g_3^{x_{i^*}} = a_1$, $g_3^{x_{1-i^*}} = a_2$, $g_3^{x_0^2} = a_3$ if $i^* = 0$ and $g_3^{x_0^2} = a_2^{x_0}$ else, $g_3^{x_0 x_1} = a_1^{x_{1-i^*}}$.

Given some instance of Co-DH, algorithm $\mathcal{A}$ eventually outputs some valid index $[c]_2$. The corresponding element $c \in \mathbb{G}_2$ can be written as $c = g_2^{P(x_0,x_1)}$ for some known polynomial $P = z_0 + z_1X_0 + z_2X_1 \in \mathbb{Z}_p[X_0, X_1]$ (Observation 2, Section 2.1). So alternatively we can say that $\mathcal{A}$ wins if $(P - X_0X_1)(x_0, x_1) \equiv 0 \bmod p$. This success event can be split up into the following disjoint events:

- Event $\mathcal{S}_1$: The univariate polynomial $(P - X_0X_1)(x_0)$, i.e., the polynomial $P - X_0X_1$ where we only evaluate the variable X_0 with x_0, is zero modulo p. Let the probability of this event be denoted by α_1.
- Event $\mathcal{S}_2$: The univariate polynomial $(P - X_0X_1)(x_0)$ is not zero modulo p but $(P - X_0X_1)(x_0, x_1)$ is. Let the probability of this event be denoted by α_2.

Clearly, we have $\epsilon = \alpha_1 + \alpha_2$.

Let us consider the events $\mathcal{S}_1$ and $\mathcal{S}_2$ when $\mathcal{B}$ runs $\mathcal{A}$ for certain choices of i^*. Note that $\mathcal{B}$ knows the coefficients of P since it responded to $\mathcal{A}$'s queries. With probability $\frac{1}{2}\alpha_1$, we have $i^* = 0$ and $\mathcal{S}_1$. This means $z_0 + z_1x + z_2X_1 - xX_1 \equiv 0$. But in this case z_2 needs to be equal to x. So $\mathcal{B}$ wins by simply returning the known coefficient z_2. Furthermore, with probability $\frac{1}{2}\alpha_2$, we have $i^* = 1$ and $\mathcal{S}_2$. Hence, the wanted DL is the root of the uni-variate non-zero polynomial $z_0 + z_1x_0 + z_2X_1 - x_0X_1$ known to $\mathcal{B}$. It can thus be determined as $x \equiv (z_0 + z_1x_0)(x_0 - z_2)^{-1} \bmod p$. It is easy to verify that the inverse $(x_0 - z_2)^{-1}$ always exists.

To summarize, if i^* happens to be zero, $\mathcal{B}$ outputs z_2, otherwise it outputs $(z_0 + z_1x_0)(x_0 - z_2)^{-1}$. In this way, its success probability is at least $\frac{1}{2}\alpha_1 + \frac{1}{2}\alpha_2 = \frac{1}{2}\epsilon$. □

3.2 Reducing SqDDH to BDDH

The bilinear decisional Diffie-Hellman problem (BDDH) is certainly among the most well-known problems over bilinear groups. It has originally been introduced in a seminal paper by Joux [23] and, e.g., further been used by Boneh and Franklin [7] to construct an identity based encryption scheme. Let us consider BDDH over a Type 1 setting where it can be defined as follows: Given $(g_1, g_1^{x_1}, g_1^{x_2}, g_1^{x_3}, g_3^{r_b})$, where $(x_1, x_2, x_3) \xleftarrow{\$} \mathbb{Z}_p^3$, $b \xleftarrow{\$} \{0,1\}$, $r_1 = x_1x_2x_3$, and $r_0 \xleftarrow{\$} \mathbb{Z}_p$ are secret choices, output b.

We relate the hardness of BDDH with respect to semi-generic algorithms to the hardness of the well-known decisional Diffie-Hellman (DDH) problem and the square decisional Diffie-Hellman (SqDDH) problem over $\mathbb{G}_3$. SqDDH is a potentially easier variant of DDH: Given $(g_3, g_3^x, g_3^{r_b})$, where $x \xleftarrow{\$} \mathbb{Z}_p$, $b \xleftarrow{\$} \{0,1\}$, $r_1 = x^2$, and $r_0 \xleftarrow{\$} \mathbb{Z}_p$ are secret choices, output b. Our result is formalized in Theorem 2. It is worth mentioning that in contrast to computational problems (like Co-DH) for decisional problems usually multiple reduction steps are required. In the proof we apply the idea of DDH-steps [12] to the bilinear setting and introduce the new concept of SqDDH-steps. Since the DDH assumption reduces to the SqDDH assumption [38] the hardness of BDDH can be formulated with respect to SqDDH only (Corollary 1).

Theorem 2. *Suppose there exists a semi-generic group algorithm $\mathcal{A}$ solving* BDDH *over a Type 1 setting in time t with advantage ϵ. Then there exists an algorithm $\mathcal{B}_{\mathsf{SqDDH}}$ solving* SqDDH *over $\mathbb{G}_3$ in time $t_{\mathsf{SqDDH}} \approx t$ with advantage $\epsilon_{\mathsf{SqDDH}}$ and an algorithm*

Table 1. Transforming a semi-generic oracle for real BDDH into one for random BDDH using SqDDH and DDH steps

	Game									
	G_1	G_2	G_3	G_4	G_5	G_6	G_7	G_8	G_9	G_{10}
$g_3^{r_b}$	$g_3^{x_1x_2x_3}$	$g_3^{x_1x_2x_3}$	$g_3^{x_1x_2x_3}$	$g_3^{x_1x_2x_3}$	$g_3^{\mathbf{x_7}x_3}$	$g_3^{\mathbf{x_8}}$	$g_3^{x_8}$	$g_3^{x_8}$	$g_3^{x_8}$	$g_3^{x_8}$
$e(g_1, g_1)$	g_3	g_3	g_3	g_3	g_3	g_3	g_3	g_3	g_3	g_3
$e(g_1, g_1^{x_1})$	$g_3^{x_1}$	$g_3^{x_1}$	$g_3^{x_1}$	$g_3^{x_1}$	$g_3^{x_1}$	$g_3^{x_1}$	$g_3^{x_1}$	$g_3^{x_1}$	$g_3^{x_1}$	$g_3^{x_1}$
$e(g_1, g_1^{x_2})$	$g_3^{x_2}$	$g_3^{x_2}$	$g_3^{x_2}$	$g_3^{x_2}$	$g_3^{x_2}$	$g_3^{x_2}$	$g_3^{x_2}$	$g_3^{x_2}$	$g_3^{x_2}$	$g_3^{x_2}$
$e(g_1, g_1^{x_3})$	$g_3^{x_3}$	$g_3^{x_3}$	$g_3^{x_3}$	$g_3^{x_3}$	$g_3^{x_3}$	$g_3^{x_3}$	$g_3^{x_3}$	$g_3^{x_3}$	$g_3^{x_3}$	$g_3^{x_3}$
$e(g_1^{x_1}, g_1^{x_1})$	$g_3^{x_1^2}$	$g_3^{\mathbf{x_4}}$	$g_3^{x_4}$	$g_3^{x_4}$	$g_3^{x_4}$	$g_3^{x_4}$	$g_3^{x_4}$	$g_3^{x_4}$	$g_3^{x_4}$	$g_3^{\mathbf{x_1^2}}$
$e(g_1^{x_2}, g_1^{x_2})$	$g_3^{x_2^2}$	$g_3^{x_2^2}$	$g_3^{\mathbf{x_5}}$	$g_3^{x_5}$	$g_3^{x_5}$	$g_3^{x_5}$	$g_3^{x_5}$	$g_3^{x_5}$	$g_3^{\mathbf{x_2^2}}$	$g_3^{x_2^2}$
$e(g_1^{x_3}, g_1^{x_3})$	$g_3^{x_3^2}$	$g_3^{x_3^2}$	$g_3^{x_3^2}$	$g_3^{\mathbf{x_6}}$	$g_3^{x_6}$	$g_3^{x_6}$	$g_3^{x_6}$	$g_3^{\mathbf{x_3^2}}$	$g_3^{x_3^2}$	$g_3^{x_3^2}$
$e(g_1^{x_1}, g_1^{x_2})$	$g_3^{x_1x_2}$	$g_3^{x_1x_2}$	$g_3^{x_1x_2}$	$g_3^{x_1x_2}$	$g_3^{\mathbf{x_7}}$	$g_3^{x_7}$	$g_3^{\mathbf{x_1x_2}}$	$g_3^{x_1x_2}$	$g_3^{x_1x_2}$	$g_3^{x_1x_2}$
$e(g_1^{x_1}, g_1^{x_3})$	$g_3^{x_1x_3}$	$g_3^{x_1x_3}$	$g_3^{x_1x_3}$	$g_3^{x_1x_3}$	$g_3^{x_1x_3}$	$g_3^{x_1x_3}$	$g_3^{x_1x_3}$	$g_3^{x_1x_3}$	$g_3^{x_1x_3}$	$g_3^{x_1x_3}$
$e(g_1^{x_2}, g_1^{x_3})$	$g_3^{x_2x_3}$	$g_3^{x_2x_3}$	$g_3^{x_2x_3}$	$g_3^{x_2x_3}$	$g_3^{x_2x_3}$	$g_3^{x_2x_3}$	$g_3^{x_2x_3}$	$g_3^{x_2x_3}$	$g_3^{x_2x_3}$	$g_3^{x_2x_3}$
		SqDDH	SqDDH	SqDDH	DDH	DDH	DDH	SqDDH	SqDDH	SqDDH
	Justification									

$\mathcal{B}_{\mathsf{DDH}}$ *solving* DDH *over* $\mathbb{G}_3$ *in time* $t_{\mathsf{DDH}} \approx t$ *with advantage* ϵ_{DDH} *such that* $\epsilon \leq 3\epsilon_{\mathsf{DDH}} + 6\epsilon_{\mathsf{SqDDH}}$.

Corollary 1. *If* SqDDH *is* (ϵ, t)*-hard over* $\mathbb{G}_3$*, then* BDDH *is* $(9\epsilon, t)$*-hard for semi-generic algorithms.*

Proof (Theorem 2). In the following we show that a for a semi-generic algorithm a "real" BDDH tuple $(g_1, g_1^{x_1}, g_1^{x_2}, g_1^{x_3}, g_3^{r_1} = g_3^{x_1x_2x_3})$ is computationally indistinguishable from a "random" tuple $(g_1, g_1^{x_1}, g_1^{x_2}, g_1^{x_3}, g_3^{r_0})$, unless SqDDH or DDH are easy over $\mathbb{G}_3$. We do this by considering a series of games played between a semi-generic algorithm $\mathcal{A}$ and an oracle $\mathcal{O}$. We start with $\mathcal{A}$ given oracle access to a real BDDH tuple. We then gradually transform this tuple as well as the output of the oracle until we end up with a random tuple. One can show that if $\mathcal{A}$ can distinguish two consecutive games G_{i-1} and G_i then it can be used to build an algorithm solving SqDDH or DDH.

The games are described by Table 1. Each of the columns labeled with G_i specifies the (direct) input over $\mathbb{G}_3$ (see Row 1) or the output of `BilinearMap` in game G_i for all possible inputs over $\mathbb{G}_1$. Bold-printed parts of a value highlight the actual changes in comparison to the previous game. The entry in the last row of a column G_i indicates which assumption (SqDDH or DDH) justifies the indistinguishability of the Games G_{i-1} and G_i. If a new x_j $(j > 3)$ appears in a column, this means that this value has been added to the corresponding game and the oracle chooses x_j uniformly from $\mathbb{Z}_p$.

As one can see from the table, by means of the Games G_2 to G_4 we remove all squares x_i^2 $(1 \leq i \leq 3)$ from the output of the pairing oracle. We do this simply by replacing each square with a new value x_j $(4 \leq j \leq 6)$. These transformations are called (bilinear) SqDDH steps and are prerequisites for the subsequent DDH steps performed in Games G_5 to G_6. During these DDH steps we selectively remove all

products $x_i x_j$ that involve variables being part of the challenge. Again, this is done by replacing the products by fresh uniformly chosen values x_j ($j \in \{7, 8\}$). In Game G_6 the challenge $g_3^{r_b} = g_3^{x_8}$ is finally independent of the input since x_8 does not appear anywhere else. After that, in Games G_7 to G_{10} we reverse the changes we did to the input and `BilinearMap` during G_2 to G_6 in reverse order. More, precisely in G_{6+j} we reverse the changes we did in G_{6-j} for $1 \leq j \leq 4$. Finally, in G_{10} we have reversed all changes (except for the one in G_6). This last game corresponds to the situation where $\mathcal{A}$ is given oracle access to a random BDDH tuple. If all intermediate games have been computationally indistinguishable (under the SqDDH and DDH assumption) then certainly also a real BDDH tuple is computationally indistinguishable from a random tuple, with respect to semi-generic algorithms.

For the sake of clarity, let us consider the transition from G_1 to G_2 (SqDDH Step) and G_4 to G_5 (DDH Step) in some more detail and quantify the involved reductions. The oracle $\mathcal{O}_{G_1}$ in Game G_1 corresponds to the original semi-generic oracle for BDDH providing access to a real BDDH tuple. The oracle in $\mathcal{O}_{G_2}$ in G_2 is equal to $\mathcal{O}_{G_1}$ except for the following changes: $\mathcal{O}_{G_2}$ additionally chooses $x_4 \xleftarrow{\$} \mathbb{Z}_p$ and uses a slightly modified table for computing pairing outputs as specified in Table 1. Let us assume $\mathcal{A}$ distinguishes the two games in time t with advantage

$$\epsilon_1 = \mathsf{Adv}_{\mathcal{A}}^{G_1,G_2} = \left|\Pr[1 \leftarrow \mathcal{A}^{\mathcal{O}_{G_1}}] - \Pr[1 \leftarrow \mathcal{A}^{\mathcal{O}_{G_2}}]\right| .$$

Then from $\mathcal{A}$ we can build an algorithm $\mathcal{B}$ for SqDDH. Again, we make use of the observation that semi-generic algorithms are blind with respect to $\mathbb{G}_1$ and e and set $\mathbb{G}_1 := \mathbb{G}_3$ and $e : \mathbb{G}_3 \times \mathbb{G}_3 \to \mathbb{G}_3$. Now let an instance

$$g_3, g_3^{x_1}, g_3^{r'_{b'}} = \begin{cases} g_3^{x_1^2}, & b' = 1 \\ g_3^{x_4}, & b' = 0 \end{cases}$$

of the SqDDH problem over $\mathbb{G}_3$ be given. $\mathcal{B}$ chooses $x_2, x_3 \xleftarrow{\$} \mathbb{Z}_p$. Then it simulates $\mathcal{O}_{G_1}$ and $\mathcal{O}_{G_2}$ as follows (we indicate below how group elements are computed though x_1, x_1^2, x_4, and b' are unknown to $\mathcal{B}$):

- The list $\mathcal{E}_1$ is initialized with $g_3, g_3^{x_1}, g_3^{x_2}, g_3^{x_3}$. Over $\mathbb{G}_3$ $\mathcal{A}$ is given $g_3, (g_3^{x_1})^{x_2 x_3}$.
- For simulating `BilinearMap`, we use the fact that we only need to know the pairing output for all possible initial inputs. These elements can be computed as described by the following table:

a	g_3	g_3	g_3	g_3	$g_3^{x_1}$	$g_3^{x_2}$	$g_3^{x_3}$	$g_3^{x_1}$	$g_3^{x_1}$	$g_3^{x_2}$
b	g_3	$g_3^{x_1}$	$g_3^{x_2}$	$g_3^{x_3}$	$g_3^{x_1}$	$g_3^{x_2}$	$g_3^{x_3}$	$g_3^{x_2}$	$g_3^{x_3}$	$g_3^{x_3}$
$e(a,b)$	g_3	$g_3^{x_1}$	$g_3^{x_2}$	$g_3^{x_3}$	$g_3^{r'_{b'}}$	$g_3^{x_2^2}$	$g_3^{x_3^2}$	$(g_3^{x_1})^{x_2}$	$(g_3^{x_1})^{x_3}$	$g_3^{x_2 x_3}$

It is easy to see that if $b' = 1$, algorithm $\mathcal{B}$ exactly simulates $\mathcal{O}_{G_1}$ and $\mathcal{O}_{G_2}$ otherwise. Thus, by simply forwarding the output of $\mathcal{A}$, $\mathcal{B}$ solves the SqDDH problem instance with the same advantage ϵ_1.

Let us now consider the transition from G_4 to G_5. The oracle $\mathcal{O}_{G_5}$ in G_5 coincides with $\mathcal{O}_{G_4}$ except for the following changes: $\mathcal{O}_{G_5}$ additionally chooses $x_7 \xleftarrow{\$} \mathbb{Z}_p$ and

uses a modified table for computing pairing outputs as specified in Table 1. Assume $\mathcal{A}$ distinguishes the two games in time t with advantage $\epsilon_4 = \mathsf{Adv}_{\mathcal{A}}^{G_4,G_5}$. Then we can use $\mathcal{A}$ to build an algorithm $\mathcal{B}$ for DDH. Given an instance

$$g_3, g_3^{x_1}, g_3^{x_2}, g_3^{r'_{b'}} = \begin{cases} g_3^{x_1 x_2}, & b' = 1 \\ g_3^{x_7}, & b' = 0 \end{cases}$$

of DDH over $\mathbb{G}_3$, $\mathcal{B}$ chooses $x_3, x_4, x_5, x_6 \xleftarrow{\$} \mathbb{Z}_p$ and simulates $\mathcal{O}_{G_5}$ and $\mathcal{O}_{G_6}$:

- The list $\mathcal{E}_1$ is initialized with $g_3, g_3^{x_1}, g_3^{x_2}, g_3^{x_3}$. Over $\mathbb{G}_3$ $\mathcal{A}$ is given $g_3, (g_3^{r'_{b'}})^{x_3}$.
- For simulating `BilinearMap` we use the following table of pairing outputs:

a	g_3	g_3	g_3	g_3	$g_3^{x_1}$	$g_3^{x_2}$	$g_3^{x_3}$	$g_3^{x_1}$	$g_3^{x_1}$	$g_3^{x_2}$
b	g_3	$g_3^{x_1}$	$g_3^{x_2}$	$g_3^{x_3}$	$g_3^{x_1}$	$g_3^{x_2}$	$g_3^{x_3}$	$g_3^{x_2}$	$g_3^{x_3}$	$g_3^{x_3}$
$e(a,b)$	g_3	$g_3^{x_1}$	$g_3^{x_2}$	$g_3^{x_3}$	$g_3^{x_4}$	$g_3^{x_5}$	$g_3^{x_6}$	$g_3^{r'_{b'}}$	$(g_3^{x_1})^{x_3}$	$(g_3^{x_2})^{x_3}$

If $b' = 1$, $\mathcal{B}$ behaves like $\mathcal{O}_{G_4}$ whereas it behaves like $\mathcal{O}_{G_5}$ if $b' = 0$. By simply forwarding the output of $\mathcal{A}$, $\mathcal{B}$ solves the DDH problem instance with advantage ϵ_4.

The bound on ϵ follows now from $\epsilon \leq \sum_{i=1}^{9} \epsilon_i$, where $\epsilon_i = \mathsf{Adv}_{\mathcal{A}}^{G_i,G_{i+1}}$, and setting $\epsilon_{\mathsf{SqDDH}} = \max_{i\in\{1,2,3,7,8,9\}}(\epsilon_i)$, $\epsilon_{\mathsf{DDH}} = \max_{i\in\{4,5,6\}}(\epsilon_i)$. □

4 Analysis of General Problem Classes

Analyzing general problem classes instead of individual problems is important for at least two reasons: First, it improves our understanding of the properties that need to be satisfied by a problem to be intractable with respect to semi-generic algorithms. Second, master theorems for these classes alleviate the burden of analyzing future problems.

Generalized Pairing-Based Problems. Let a Type 1, 2, or 3 setting according to Definition 1 be given. Furthermore, let $\ell \in \mathbb{N}, d \in \{1,2,3\}$ be positive integers, $\mathbf{I_1}, \mathbf{I_2}, \mathbf{I_3} \subset \mathbb{Z}_p[X_1, \ldots, X_\ell]$ be finite sets of (publicly known) polynomials (called *input polynomials*) and $Q \in \mathbb{Z}_p[X_1, \ldots, X_\ell]$ be a single (publicly known) polynomial (called *challenge polynomial*). Then we define a $(\mathbf{I_1}, \mathbf{I_2}, \mathbf{I_3}, Q)$-$\mathsf{BDH}_{\mathbb{G}_d}$ problem as: Given

$$((g_1^{R(\mathbf{x})})_{R\in\mathbf{I_1}}, (g_2^{R(\mathbf{x})})_{R\in\mathbf{I_2}}, (g_3^{R(\mathbf{x})})_{R\in\mathbf{I_3}}),$$

where $\mathbf{x} \xleftarrow{\$} \mathbb{Z}_p^\ell$ are secret random values, output $g_d^{Q(\mathbf{x})}$. A decisional variant of such problems can be defined analogously. In the following we always assume that the polynomial 1 is contained in each $\mathbf{I_i}$ which corresponds to the natural assumption that for each group a generator is given.

Informally speaking, a $(\mathbf{I_1}, \mathbf{I_2}, \mathbf{I_3}, Q)$-$\mathsf{BDH}_{\mathbb{G}_d}$ problem is *non-trivial* if there is no way to compute Q using only the input polynomials and the operations on them which are implicitly given by the underlying bilinear setting. Let us restrict here to consider the case $d \in \{1,2\}$. Let $\mathbf{I_1} = \{R_1, \ldots, R_t\}$ and $\mathbf{I_2} = \{S_1, \ldots, S_{t'}\}$. Then using

Observation 2 (Section 2.1), one can see that the output $[c]_d$ of a semi-generic algorithm for the considered problem can be written as $g_d^{P(\mathbf{x})}$ for some P of the form

$$P = \begin{cases} \sum_{j=1}^{t} z_j R_j, & d = 1 \\ \sum_{j=1}^{t'} z'_j S_j, & d = 2 \text{ and Type 3 setting} \\ \sum_{j=1}^{t} z_j R_j + \sum_{j=1}^{t'} z'_j S_j, & d = 2 \text{ and Type 2 setting} \end{cases} \tag{1}$$

We call a $(\mathbf{I_1}, \mathbf{I_2}, \mathbf{I_3}, Q)$-$\mathsf{BDH}_{\mathbb{G}_d}$ non-trivial if there is no P of the above form such that $g_d^{P(\mathbf{x})} = g_d^{Q(\mathbf{x})}$ *for all* $\mathbf{x} \in \mathbb{Z}_p^\ell$, i.e., if $P \neq Q \in \mathbb{Z}_p[X_1, \ldots, X_\ell]$. More formal and general definitions can be found in the full version of this paper [20].

Reductions for Generalized Problems. Theorem 3 extends the reduction we have seen for Co-DH in Section 3.1 to the more general class of $(\mathbf{I_1}, \mathbf{I_2}, \mathbf{I_3}, Q)$-$\mathsf{BDH}_{\mathbb{G}_d}$ problems. The crucial difference and novelty lies in the technique for extracting the wanted discrete logarithm given the output of the semi-generic algorithm.

Theorem 3. *Let* $d \in \{1, 2\}$ *and* $(\mathbf{I_1}, \mathbf{I_2}, \mathbf{I_3}, Q)$-$\mathsf{BDH}_{\mathbb{G}_d}$ *be a non-trivial problem with challenge and input polynomials in* $\mathbb{Z}_p[X_1, \ldots, X_\ell]$. *Let* $k = \max_i(\deg_{X_i}(\mathbf{I_1} \cup \mathbf{I_2} \cup \mathbf{I_3}))$. *Suppose there is a semi-generic algorithm* $\mathcal{A}$ *solving* $(\mathbf{I_1}, \mathbf{I_2}, \mathbf{I_3}, Q)$-$\mathsf{BDH}_{\mathbb{G}_d}$ *in time* t *with success probability* ϵ. *Then there is an algorithm* $\mathcal{B}$ *solving* $2k$-DL *in* $\mathbb{G}_3$ *in time* $t' \approx t + \tilde{O}(k' \log p)$, *where* $k' = \max(k, \deg(Q))$, *with probability* $\epsilon' \geq \frac{\epsilon}{\ell}$.

Proof. Let $k_1 = 2k$. $\mathcal{B}$ takes as input a k_1-DL challenge $a_0 = g_3, a_1 = g_3^{x^1}, \ldots, a_{k_1} = g_3^{x^{k_1}}$. It then chooses $i^* \xleftarrow{\$} \{1, \ldots, \ell\}$ and $x_1, \ldots, x_{i^*-1}, x_{i^*+1}, \ldots, x_\ell \xleftarrow{\$} \mathbb{Z}_p$. The unknown x is treated as the secret choice x_{i^*} in the context of a $(\mathbf{I_1}, \mathbf{I_2}, \mathbf{I_3}, Q)$-$\mathsf{BDH}_{\mathbb{G}_d}$ instance. We only sketch important points in the simulation of the semi-generic oracle: Each internal list $\mathcal{E}_j$ is initialized with the elements $(g_3^{P(\mathbf{x})})_{P \in \mathbf{I_j}}$ where for a polynomial $P = \sum_{e=(e_1,\ldots,e_\ell) \in E} b_e X_1^{e_1} \cdots X_\ell^{e_\ell}$, $E \subset \mathbb{Z}_p^\ell$, the element $g_3^{P(\mathbf{x})}$ can be computed as $g_3^{P(\mathbf{x})} = \prod_e a_{e_{i^*}}^{b_e \prod_{s \neq i^*} x_s^{e_s}}$ using the given instance of the k_1-DL problem. This is possible because the degree in X_{i^*} of the polynomials in each set $\mathbf{I_j}$ is upper bounded by k_1. Similarly, the table for simulating `BilinearMap` can be created since for each entry $g_3^{P(\mathbf{x})}$ in this table, P is again of degree at most k_1 in X_{i^*}.

Given an $(\mathbf{I_1}, \mathbf{I_2}, \mathbf{I_3}, Q)$-$\mathsf{BDH}_{\mathbb{G}_d}$ instance, $\mathcal{A}$ eventually outputs an index $[c]_d$. Then c can be written as $g_3^{P(\mathbf{x})}$ for some known polynomial P as described in Equation 1. Thus, $\mathcal{A}$ wins if $Q(\mathbf{x}) \equiv P(\mathbf{x}) \bmod p$. Since $Z := Q - P$ is not zero modulo p (the problem is non-trivial) this success event can be split into disjoint events $\mathcal{S}_1, \ldots, \mathcal{S}_\ell$, where $\mathcal{S}_j$ is defined as:

$$Z(X_1 = x_1, \ldots, X_{j-1} = x_{j-1}) \not\equiv 0 \text{ and } Z(X_1 = x_1, \ldots, X_j = x_j) \equiv 0 \tag{2}$$

Denoting the probability of event $\mathcal{S}_j$ by α_j we obtain $\epsilon = \alpha_1 + \cdots + \alpha_\ell$.

Now assume that event S_{i^*} occurs, which happens with probability ϵ/ℓ. Consider the polynomial $Z_{i^*} = Z(X_1 = x_1, \ldots, X_{i^*-1} = x_{i^*-1}) \bmod p \in \mathbb{Z}_p[X_{i^*}, \ldots, X_\ell]$. This polynomial is of the form $Z_{i^*} = \sum_{e=(e_{i^*},\ldots,e_\ell) \in E} b_e X_{i^*}^{e_{i^*}} \cdots X_\ell^{e_\ell}$, for some $E \subset \mathbb{Z}_p^{\ell - i^* + 1}$, where in at least one monomial the variable X_{i^*} appears with a non-zero

exponent e_{i^*}. Let $M = b'_e X_{i^*}^{e'_{i^*}} \cdots X_\ell^{e'_\ell}$ be one of these monomials. Then consider the polynomial Z'_{i^*} we obtain by summing up all monomials of Z_{i^*} containing the submonomial $X_{i^*+1}^{e'_{i^*+1}} \cdots X_\ell^{e'_\ell}$:

$$Z'_{i^*} = \sum_{\substack{e=(e_{i^*},\ldots,e_\ell)\in E \\ e_{i^*+1}=e'_{i^*+1},\ldots,e_\ell=e'_\ell}} b_e X_{i^*}^{e_{i^*}} X_{i^*}^{e'_{i^*}} \cdots X_\ell^{e'_\ell}$$

Clearly, we have $Z'_{i^*} \not\equiv 0 \bmod p$ and since $Z_{i^*}(X_{i^*} = x_{i^*}) \equiv 0 \bmod p$ it also holds that $Z'_{i^*}(X_{i^*} = x_{i^*}) \equiv 0 \bmod p$. Hence, $x_{i^*} = x$ is a root of the non-zero uni-variate polynomial

$$Z''_{i^*} = \sum_{\substack{e=(e_{i^*},\ldots,e_\ell)\in E \\ e_{i^*+1}=e'_{i^*+1},\ldots,e_\ell=e'_\ell}} b_e X_{i^*}^{e_{i^*}}$$

Note that Algorithm $\mathcal{B}$ can easily construct the polynomial Z''_{i^*} by picking an *arbitrary* monomial from Z_{i^*} for which X_{i^*} appears with non-zero exponent. The coefficients b_e can also be easily computed since the coefficients of Z are known and $x_1, \ldots, x_{i^*-1}$ have been chosen by $\mathcal{B}$. So by applying an efficient standard algorithm for computing roots of polynomials over $\mathbb{Z}_p$, such as [36, Algorithm 14.15], $\mathcal{B}$ can find the wanted DL $x_{i^*} = x$ by computing all roots of the polynomial Z''_{i^*}. These at most $k' = \max(k, \deg(Q))$ different roots can be computed in time $\tilde{O}(k' \log p)$ [36, Corollary 14.16]. Whether a root x' equals x can be tested by verifying $g^{x'} \stackrel{?}{=} a_1$. □

We have also been able to find a reduction for a general class of decisional problems which is efficient for virtually all problems of this class considered in practice. Essentially, our reduction from the SqDDH problem over $\mathbb{G}_3$ works for all $(\mathbf{I_1}, \mathbf{I_2}, \mathbf{I_3}, Q)$-BDDH$_{\mathbb{G}_3}$ problems where variables in $\mathbf{I_1} \cup \mathbf{I_2}$ and $\mathbf{I_3} \cup \{Q\}$ appear with at most linear and quadratic exponents, respectively. Our approach for this general reduction differs from the one for BDDH we have seen in Section 3.2 in the following way: The BDDH reduction is direct in the sense that *all reduction steps take place directly in the semi-generic model*. As an alternative, one could also first "project" BDDH to the group $\mathbb{G}_3$ by finding an "appropriate" problem which reduces *in a single step* to BDDH (with respect to semi-generic algorithms) and then *apply all* DDH *and* SqDDH *reduction steps to this problem in the standard model*. We follow this latter approach in our proof for general bilinear decisional problems since it has the advantage that we can resort to Bresson et al.'s results for generalized DDH problems [12] in the standard model. However, this is not straightforward. Since their results are quite restricted we need to enhance them to more general problem classes. For more details on our result for bilinear decisional problems we refer to the full version [20].

5 Analyzing Cryptosystems in the Semi-Generic Model

Besides for studying cryptographic hardness assumptions, it would also be interesting to use the SGGM as a tool to analyze the security of practical pairing-based cryptosystems. Similar analyzes have been made in the classical GGM [35, 13]. In this section

we consider the Boneh-Lynn-Shacham (BLS) signature scheme [9,10] in the SGGM. It turns out that it is possible to prove security of this scheme under the semi-generic groups heuristic, by requiring concrete (but non-standard) properties of the hash function.

The BLS signature scheme over a Type 1 bilinear setting is defined as follows. Let H_1 be a hash function $H_1 : \{0,1\}^\ell \rightarrow \mathbb{G}_1$.

- Gen samples a random generator g of $\mathbb{G}_1$, $s \xleftarrow{\$} \mathbb{Z}_p$, and sets $pk = (g, g^s)$, $sk = s$.
- $\mathsf{Sign}(sk, m)$ computes $H_1(m)$ and returns $\sigma = H_1(m)^s$.
- $\mathsf{Verify}(pk, m, \sigma)$ returns 1, if $e(H_1(m), pk) = e(\sigma, g)$, and 0 otherwise.

Let us now describe the EUF-CMA security experiment for the BLS signature scheme in the SGGM. Here we are facing a technical problem: the BLS scheme utilizes a hash function $H_1 : \{0,1\}^\ell \rightarrow \mathbb{G}_1$, that is, the output of this map is a group element in some given representation. However, in the SGGM we want to consider algorithms which are *independent* of a particular representation of elements of $\mathbb{G}_1$. Since in our model elements of $\mathbb{G}_1$ are given as list indices, we have no representation of group elements that we could use as the range of the hash function.

One possible solution would be to fall back on the formalization of a generic group by Shoup [34]. In this model, group elements are represented by unique random bit strings. Thus, we could use a hash function that maps to bit strings of appropriate size. However, the fact that group elements are encoded as *random* strings has been subject to much criticism [16,26,15]. For instance, the Shoup model can be misused to implement a random oracle, which is of no avail since we want to avoid random oracles in our security proof. Therefore we follow a different approach. We implement H_1 as a generic *group hash function*.

Definition 2. *A group hash function is a pair of algorithms* $\mathbf{H} = (\mathsf{GHGen}, \mathsf{GHEval})$.

- GHGen *takes as input a generator* g *of* $\mathbb{G}_1$*, and returns* $A = (a_1, \ldots, a_\delta) \in \mathbb{G}_1^\delta$*. Vector* A *specifies a function* $H_1 : \{0,1\}^\ell \rightarrow \mathbb{G}_1$*.*
- *Algorithm* GHEval *takes as input a vector* $A \in \mathbb{G}_1^\delta$ *and a string* $m \in \{0,1\}^\ell$*, and returns* $H_1(m) \in \mathbb{G}_1$*.*

We say that a group hash function is generic*, if* GHGen *and* GHEval *perform only group operations on elements of* A*.*

Examples of generic group hash functions are the hash function used in Water's IBE scheme [37] and the programmable hash functions of Hofheinz and Kiltz [18].

Generic group hash functions have the useful property that there exist "trapdoor" set-up and evaluation algorithms $(\mathsf{TrapGen}, \mathsf{TrapEval})$ with the following properties.

- $\mathsf{TrapGen}$ takes as input a generator $g \in \mathbb{G}_1$. It returns a vector $A \in \mathbb{G}_1^\delta$, distributed identically to the output of GHGen for all g, and some trapdoor information td.
- Algorithm $\mathsf{TrapEval}$ takes as input a vector $A \in \mathbb{G}_1^\delta$ and a string $m \in \{0,1\}^\ell$, and returns h such that $g^h = H_1(m)$.

For the security proof we need to demand a strong form of collision resistance.

Definition 3. *A group hash function is* (ϵ, t, q)-algebraic collision resistant*, if*

$$\Pr\left[\mathcal{A}(A) = (m_0, ..., m_q, i_0, ..., i_q) : H_1(m_0) = g^{i_0} \prod_{j=1}^{q} (H_1(m_j))^{i_j}\right] \leq \epsilon$$

for all algorithms $\mathcal{A}$ *running in time* t*.*

By employing techniques from [18] it is possible to construct hash functions satisfying this property under weak assumptions, like the hardness of computing discrete logarithms in $\mathbb{G}_1$, for any *constant* q. A major drawback is, however, that for these constructions the size δ of vector A grows at least linearly with q. We leave it as an open problem to study whether there exists a (possibly probabilistic) trapdoor group hash function such that δ is constant and $q = q(\kappa)$ is a polynomial.

We formalize the EUF-CMA experiment in the SGGM as follows. At the beginning of the game, the challenger samples a random generator g and a secret key x. Then it runs $(a_1, \ldots, a_\delta) \xleftarrow{\$} \mathsf{GHGen}(g)$, sets $I_1 := (g, g^x, a_1, \ldots, a_\delta)$, and implements a semi-generic oracle with input I_1 as described in Section 2. This provides the adversary with the public key, and the ability to perform group operations on elements of $\mathbb{G}_1$.

When the adversary queries a signature for some chosen message m_i, the challenger computes $H(m_i)^x$ and appends it to the list $\mathcal{E}_1$.

We say that the adversary *wins* the game, if it outputs a message m and index $[s]_1$ such that $s = H(m)^x$, that is, the adversary has computed a valid signature for m. We say that a semi-generic adversary $\mathcal{A}$ (ϵ, t)*-breaks* the EUF-CMA security of a signature scheme if $\mathcal{A}$ runs in time t and $\Pr[\mathcal{A} \text{ wins}] \geq \epsilon$.

Theorem 4. *Suppose there exists an adversary* $\mathcal{A}$ (ϵ, t)*-breaking the EUF-CMA security of the BLS signature scheme in the semi-generic model by making* q *chosen-message signature queries. Then there exists an algorithm* $\mathcal{B}_{\mathsf{coll}}$ $(\epsilon_{\mathsf{dl}}, t_{\mathsf{dl}}, q)$*-breaking the algebraic collision resistance of* H_1 *and an algorithm* $\mathcal{B}_{\mathsf{dl}}$ $(\epsilon_{\mathsf{dl}}, t_{\mathsf{dl}})$*-solving the discrete logarithm problem in* $\mathbb{G}_1$*, such that* $t \approx t_{\mathsf{coll}} \approx t_{\mathsf{dl}}$ *and* $\epsilon \leq \epsilon_{\mathsf{coll}} + \epsilon_{\mathsf{dl}}$*.*

Proof. Suppose there exists an adversary $\mathcal{A}$ that outputs a message m and an index $[s]_1$ such that $s = H(m)^x$. In the SGGM, an adversary has to compute a group element of $\mathbb{G}_1$ by applying a sequence of group operations to the initial values $(g, g^x, a_1, \ldots, a_\delta)$ stored in $\mathcal{E}_1$ and to group elements added to the list by the challenger oracle in response to chosen-message signature queries. Thus, when $\mathcal{A}$ outputs $(m, [s]_1)$ such that $s = H(m)^x$, then the oracle obtains an equation

$$H(m)^x = g^{\alpha_1} \cdot (g^x)^{\alpha_2} \cdot \prod_{i=1}^{\delta} a_i^{\beta_i} \cdot \prod_{i=1}^{q} (H(m_i)^x)^{\gamma_i}\,, \tag{3}$$

or equivalently $x \cdot \left(\log_g H(m) - \sum_{i=1}^{q} \gamma_i \log_g H(m_i) - \alpha_2\right) = \alpha_1 + \sum_{i=1}^{\delta} \beta_i \log_g a_i$, for integers $\alpha_i, \beta_i, \gamma_i$ known to the oracle. We consider two types of forgers:

1. A Type-A forger performs a sequence of operations such that

$$\log_g H(m) - \sum_{i=1}^{q} \gamma_i \log_g H(m_i) - \alpha_2 \equiv 0 \bmod p. \tag{4}$$

2. A Type-B forger performs a sequence of operations such that

$$\log_g H(m) - \sum_{i=1}^{q} \gamma_i \log_g H(m_i) - \alpha_2 \not\equiv 0 \bmod p. \tag{5}$$

Lemma 1. *Suppose there exists a Type-A forger $\mathcal{A}$ (ϵ, t)-breaking the EUF-CMA security of the BLS signature scheme by making at most q chosen-message queries. Then there exists an algorithm $\mathcal{B}_{\mathsf{coll}}$ $(\epsilon_{\mathsf{dl}}, t_{\mathsf{dl}}, q)$-breaking the algebraic collision resistance of* $(\mathsf{GHGen}, \mathsf{GHEval})$ *in time $t' \approx t$ with success probability $\epsilon_{\mathsf{coll}} \geq \epsilon$.*

PROOF. Algorithm $\mathcal{B}_{\mathsf{coll}}$ receives as input a vector $A' = (g', a'_1, \ldots, a'_\delta)$. It proceeds exactly like the semi-generic EUF-CMA challenger, except that it sets $g := g'$ and $a_i := a'_i$ instead of sampling g at random and generating A by running $\mathsf{GHGen}(g)$. Thus, in particular $\mathcal{B}_{\mathsf{coll}}$ chooses the secret key $x \xleftarrow{\$} \mathbb{Z}_p$ and thus is able to simulate the original challenger perfectly.

When $\mathcal{A}$ outputs $(m, [s]_1)$ such that $s = H(m)^x$, then $\mathcal{B}_{\mathsf{coll}}$ computes and returns integers $(\alpha_2, \gamma_1, \ldots, \gamma_q)$ as in Equation 4. Observe that if Equation 4 is satisfied, then we have $H(m) = g^{\alpha_2} \cdot \prod_{i=1}^{q} H(m_i)^{\gamma_i}$. ▲

Lemma 2. *Suppose there exists a Type-B forger $\mathcal{A}$ (ϵ, t)-breaking the EUF-CMA security of the BLS signature scheme. Then there exists an algorithm $\mathcal{B}_{\mathsf{dl}}$ solving the discrete logarithm problem in $\mathbb{G}_1$ in time $t_{\mathsf{dl}} \approx t$ with success probability $\epsilon_{\mathsf{dl}} \geq \epsilon$.*

PROOF. Algorithm $\mathcal{B}_{\mathsf{dl}}$ receives as input a tuple (g', y). It sets $g := g'$, $g^x := y$, and runs $(A, td) \xleftarrow{\$} \mathsf{TrapGen}(g)$ to generate the public parameters of the hash function. Recall that A is distributed identically to some A' generated by GHGen. It sets $I_1 := (g, g^x, a_1, \ldots, a_\delta)$, and implements a semi-generic oracle with initial list state I_1.

Since $\mathcal{B}_{\mathsf{dl}}$ does not know the secret-key exponent x, it answers chosen-message signature queries of $\mathcal{A}$ differently. $\mathcal{B}_{\mathsf{dl}}$ makes use of the trapdoor information td generated along with A. Whenever $\mathcal{A}$ submits a chosen-message m_i, $\mathcal{B}_{\mathsf{dl}}$ computes $h_i = \mathsf{TrapEval}(m_i)$ and appends y^{h_i} to $\mathcal{E}_1$. Note that $y^{h_i} = g^{x \log_g H(m_i)} = H(m_i)^x$, thus this is a valid signature.

When $\mathcal{A}$ outputs $(m, [s]_1)$ such that $s = H(m)^x$, then $\mathcal{B}_{\mathsf{dl}}$ computes integers $(\alpha_i, \beta_i, \gamma_i)$ as in Equation 3, and returns

$$x = \log_{g'} y = \frac{\alpha_1 + \sum_{i=1}^{\delta} \beta_i \log_g a_i}{\log_g H(m) - \sum_{i=1}^{q} \gamma_i \log_g H(m_i) - \alpha_2} \bmod p,$$

which is possible since $\log_g H(m) - \sum_{i=1}^{q} \gamma_i \log_g H(m_i) - \alpha_2 \not\equiv 0 \bmod p$. ▲

□

Acknowledgements. We would like to thank Dennis Hofheinz, Jesper Buus Nielsen, and Dominique Unruh for valuable discussions and the anonymous reviewers of Asiacrypt 2010 for their detailed and helpful comments.

References

1. Aggarwal, D., Maurer, U.: Breaking RSA generically is equivalent to factoring. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 36–53. Springer, Heidelberg (2010)
2. 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)
3. Boneh, D.: Number-theoretic assumptions. Invited Talk at TCC's Special Session on Assumptions for Cryptography (2007)
4. Boneh, D., Boyen, X.: Efficient selective-id secure identity-based encryption without random oracles. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)
5. Boneh, D., Boyen, X., Goh, E.: Hierarchical identity based encryption with constant size ciphertext (full paper). Cryptology ePrint Archive, Report 2005/015 (2005), http://eprint.iacr.org/
6. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin, M.K. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)
7. Boneh, D., Franklin, M.K.: Identity-based encryption from the Weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
8. Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and verifiably encrypted signatures from bilinear maps. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 416–432. Springer, Heidelberg (2003)
9. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the Weil pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 514–532. Springer, Heidelberg (2001)
10. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the Weil pairing. J. Cryptology 17(4), 297–319 (2004)
11. Boyen, X.: The Uber-Assumption family. In: Galbraith, S.D., Paterson, K.G. (eds.) Pairing 2008. LNCS, vol. 5209, pp. 39–56. Springer, Heidelberg (2008)
12. Bresson, E., Lakhnech, Y., Mazaré, L., Warinschi, B.: A generalization of DDH with applications to protocol analysis and computational soundness. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 482–499. Springer, Heidelberg (2007)
13. Brown, D.R.L.: Generic groups, collision resistance, and ECDSA. Des. Codes Cryptography 35(1), 119–152 (2005)
14. Cheon, J.: Security analysis of the Strong Diffie-Hellman problem. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 1–11. Springer, Heidelberg (2006)
15. Dent, A.W.: Adapting the weaknesses of the random oracle model to the generic group model. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 100–109. Springer, Heidelberg (2002)
16. Fischlin, M.: A note on security proofs in the generic model. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 458–469. Springer, Heidelberg (2000)
17. Galbraith, S.D., Paterson, K.G., Smart, N.P.: Pairings for cryptographers. Discrete Applied Mathematics 156(16), 3113–3121 (2008)
18. Hofheinz, D., Kiltz, E.: Programmable hash functions and their applications. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 21–38. Springer, Heidelberg (2008)
19. Hohenberger, S.: The cryptographic impact of groups with infeasible inversion. Master's thesis, Massachusetts Institute of Technology (2003)
20. Jager, T., Rupp, A.: The semi-generic group model and applications to pairing-based cryptography (full paper) (2010), http://www.nds.rub.de/chair/publications/
21. Jager, T., Schwenk, J.: On the analysis of cryptographic assumptions in the generic ring model. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 399–416. Springer, Heidelberg (2009)

22. Jao, D., Yoshida, K.: Boneh-Boyen signatures and the Strong Diffie-Hellman problem. Cryptology ePrint Archive, Report 2009/221 (2009), http://eprint.iacr.org/
23. 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)
24. Joux, A.: A one round protocol for tripartite Diffie-Hellman. J. Cryptology 17(4), 263–276 (2004)
25. Katz, J., Sahai, A., Waters, B.: Predicate encryption supporting disjunctions, polynomial equations, and inner products. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 146–162. Springer, Heidelberg (2008)
26. Koblitz, N., Menezes, A.: Another look at generic groups. Advances in Mathematics of Communications 1, 13–28 (2007)
27. Leander, G., Rupp, A.: On the equivalence of RSA and factoring regarding generic ring algorithms. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 241–251. Springer, Heidelberg (2006)
28. Maurer, U.: Abstract models of computation in cryptography. In: Smart, N.P. (ed.) Cryptography and Coding 2005. LNCS, vol. 3796, pp. 1–12. Springer, Heidelberg (2005)
29. Menezes, A., Okamoto, T., Vanstone, S.: Reducing elliptic curve logarithms to logarithms in a finite field. IEEE Transactions on Information Theory 39(5), 1639–1646 (1993)
30. Pohlig, S.C., Hellman, M.E.: An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance. IEEE Transactions on Information Theory 24, 106–110 (1978)
31. Pollard, J.M.: Monte Carlo methods for index computation mod p. Mathematics of Computation 32, 918–924 (1978)
32. Rivest, R.L.: On the notion of pseudo-free groups. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 505–521. Springer, Heidelberg (2004)
33. Rupp, A., Leander, G., Bangerter, E., Dent, A.W., Sadeghi, A.: Sufficient conditions for intractability over black-box groups: Generic lower bounds for generalized DL and DH problems. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 489–505. Springer, Heidelberg (2008)
34. Shoup, V.: Lower bounds for discrete logarithms and related problems. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 256–266. Springer, Heidelberg (1997)
35. Smart, N.P.: The exact security of ECIES in the generic group model. In: Honary, B. (ed.) Cryptography and Coding 2001. LNCS, vol. 2260, pp. 73–84. Springer, Heidelberg (2001)
36. von zur Gathen, J., Gerhard, J.: Modern Computer Algebra, 2nd edn. Cambridge University Press, Cambridge (2003)
37. Waters, B.: Efficient identity-based encryption without random oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)
38. Wolf, S.: Information-theoretically and computationally secure key agreement in cryptography. PhD thesis, ETH Zurich, ETH dissertation No. 13138 (1999)

The Degree of Regularity of HFE Systems*

Vivien Dubois[1] and Nicolas Gama[2]

[1] DGA-MI, France
vivien.dubois@m4x.org
[2] EPFL, Switzerland
nicolas.gama@ens.fr

Abstract. HFE is a public key scheme introduced by Patarin in 1996. An HFE public key is a large system of polynomials in many variables over a small finite field. This system results from some secret composition, based on which the owner can solve it to any arbitrary vector. While the security of the cryptosystem relies on the difficulty of solving the public system without the trapdoor information, in 2002 Faugère found experimentally that Gröbner basis computations perform much better on certain HFE instances than on random systems. More specifically, Faugère observed that the regular behaviour of the Gröbner basis computation collapses at a much lower degree than expected for random systems, letting the computation finish much earlier. Accounting for this distinctive property, Faugère and Joux showed in 2003 that mapping HFE systems to some other multivariate ring exhibits the particular algebraic structure of these systems. Nevertheless, they did not offer the actual computation of the degree of regularity of HFE systems. Later, in 2006, Granboulan, Joux and Stern showed an asymptotic upper bound on the degree of regularity of HFE systems over $GF(2)$ using independent results on overdetermined systems of equations. The case of larger ground fields has remained however completely unsolved. In this paper, we exhibit an additional property of HFE systems that is increasingly significant as the size of the ground field grows. Using this property with a standard combinatorial calculation yields an arguably tight numerical bound on the degree of regularity of HFE systems for any parameters.

Keywords: multivariate polynomials, HFE, algebraic cryptanalysis.

1 Introduction

Solving large systems of multivariate equations over a finite field is one of the most recurrent problems in computer science. Although achieving this task seems very hard in general and can only be tackled for small sizes by current best algorithms, sparse classes of systems exist that can be solved efficiently. In the last fifteen years, attempts have been made at exploiting this gap to build asymmetric cryptographic primitives. In a nutshell, the issue has been to find secure ways of masking structured systems of polynomials.

* This paper is an extended abstract. The full version is available from the authors.

M. Abe (Ed.): ASIACRYPT 2010, LNCS 6477, pp. 557–576, 2010.

The HFE Cryptosystem. One of the most prominent proposals in this area has been the *Hidden Field Equation* cryptosystem, introduced by Patarin in 1996. HFE is based on an elegant idea introduced by Matsumoto and Imai in 1988 of deriving a set of multivariate equations from a single variable equation over a large extension field; this makes use of the vector space structure of this extension field. When the single variable equation can be solved efficiently the same holds for the multivariate system, and access to the large field equation is restricted by applying secret linear bijections on the variables and equations.

More formally, let $\mathbb{F}_q$ denote the finite field with q elements and let ϕ be some linear bijection from $\mathbb{F}_{q^n}$, the degree n extension of $\mathbb{F}_q$, to $(\mathbb{F}_q)^n$. Such a linear bijection is defined by a choice of a linear basis of $\mathbb{F}_{q^n}$. To any polynomial function $P(X)$ on $\mathbb{F}_{q^n}$, one associates the function $\phi \circ P \circ \phi^{-1}$ on $(\mathbb{F}_q)^n$. In HFE, polynomials P have a small degree to ensure efficient root finding. Also, they have a special shape which ensures that $\phi \circ P \circ \phi^{-1}$ is quadratic. This function is then composed with secret linear bijections $S, T : (\mathbb{F}_q)^n \to (\mathbb{F}_q)^n$, $T \circ (\phi \circ P \circ \phi^{-1}) \circ S$ and the result is released as the public function. HFE can be used as a signature scheme and also, with some minor arrangements, as an encryption scheme [16]. Many variations exist and offer potential enhancements.

The Security of HFE. The fundamental issue is whether the public function is a one-way function. Finding a preimage by the public function is the same as finding a solution to the corresponding system of quadratic equations. Denote by $\mathrm{MQ}(q, n)$ the set of systems of n quadratic equations in n variables over $\mathbb{F}_q$, and by $\mathrm{HFE}(q, n, D)$ the subset of HFE systems where D is the parameter that controls the degree of the internal polynomial P. Two lines of work have so far been able to distinguish HFE systems from random MQ systems. One line of work, proposed in [8], targets so called differential properties of HFE functions and was able to produce a distinguisher with proven complexity for all parameters (q, n, D). The other line of work, proposed in [4,9,15], directly targets the difficulty of the preimage problem on HFE systems. It produced experimental evidence that for some parameters the preimage problem is much easier on HFE systems than on random MQ instances [12]. Since the difficulty of the preimage problem on HFE systems is ultimately the issue, one wishes to clarify what property is disclosed by the methods used in the second line of work and how this property depends on the parameters (q, n, D). So far, the available information has been the following.

1. The experimental evidence has been obtained by using algorithms for computing Gröbner bases [12,17]. These algorithms proceed through combinations with polynomial coefficients of a given set of polynomials and generate additional polynomials that can be used to solve the system.
2. The attacks have only concerned systems over $\mathbb{F}_2$. Experiments for various values of n and D evidenced that the degree of combinations needed to compute a Gröbner basis (for a graded ordering of terms) on HFE systems only depends on D for large enough n [12]. Unfortunately, no extension of this property to larger values of q has been reported. In fact, some authors [7]

argued that the size of the field should have a strong negative impact the computation and observed it on experiments using the Magma package [18].
3. On the theoretical side, a qualitative account was given in [9] on how the combinations performed on the public polynomials correspond to related operations on the internal polynomial. Although this clearly initiated a way of investigating HFE systems, it has not been followed with the computation of theoretical complexity bounds. Nevertheless, the authors in [15] showed that when $q = 2$, complexity bounds can be heuristically derived from results on overdetermined MQ systems.

We note that quantitative information has only been obtained from experiments and on systems over $\mathbb{F}_2$. The theoretical connections have not permitted to derive quantitative information beyond practical reach. Notably, how the phenomenon that is observed experimentally varies as q increases has remained unknown. The gain of potential enhancements also has, incidentally, remained unclear.

Our contribution. Recent studies on the complexity of Gröbner basis algorithms focus on the notion of degree of regularity of a system of polynomials [2,1]. Roughly speaking, the degree of regularity is the smallest degree at which a non-trivial degree fall among algebraic combinations of the input polynomials occurs. The degree of regularity of HFE systems over $\mathbb{F}_2$ was experimentally found within some parameter range in [9] and asymptotically upper bounded in [15] using the results of [2,1]. In this paper, we give a way to compute a numerical bound on the degree of regularity of HFE systems over any field and for any parameters. This is achieved by using previous ideas and methods present in [9,15,1,6] in combination with an apparently unnoticed additional property of HFE systems which is increasingly significant as the size of the ground field grows.

Organization of the paper. In Section 2, we define the degree of regularity of a system of polynomials and relate this notion to the computation of a Gröbner basis. In Section 3, we define HFE systems in greater detail and set a few notations. In Section 4, we map the problem of computing the degree of regularity to some other multivariate ring where the algebraic structure of HFE systems is apparent. This is only a more precise statement of a property used in [9,15] and our upper bound derives from the same observation that the degree of regularity is upper bounded by the degree of regularity of any subsystem. In Section 5, we show how to compute the degrees of regularity of these subsystems by using classical methods such as used in [1,6] but with the specific properties of the polynomials at hand. We deduce numerical bounds for many parameters. In Section 6, we derive estimates on the complexity of algebraic attacks on HFE.

2 Algebraic Properties of a System of Polynomials

We first give an informal presentation of the notions that will be used in the sequel and then give precise definitions and statements for our particular setting.

2.1 Solving a System of Multivariate Equations

Suppose we face the problem of finding the common roots of a system of polynomials $p_1, \ldots, p_k$ in a multivariate ring R over a field. Would this system be in few enough variables to be tried by hand, one would probably try to combine the given polynomials to derive "simpler" ones, that is, that make it easier to discover the space of solutions. For instance, one may try to obtain a polynomial in fewer variables, or with a smaller total degree. In any case, combining the given polynomials always comes down to consider polynomials of the shape $m_1p_1 + \cdots + m_kp_k$ for some polynomial multiples $m_1, \ldots, m_k$. Hence, these polynomials are linear combinations of $p_1, \ldots, p_k$ with coefficients in R. And the goal is then to find such a linear combination within some target subspace of R.

To do this mechanically, one may consider two main strategies. Either one chooses *a priori* search spaces for the m_i (for instance, polynomials with degree under some bound) and one performs linear algebra on their coefficients. (This is the basic idea of XL algorithms [5,19].) Or one defines a priority list among terms to be eliminated (called an ordering) and one performs systematic leading term reductions on polynomials $p_1, \ldots, p_k$ and the new polynomials that are generated by this process, until it can be predicted that any further combination will reduce to zero. (This is the basic idea of Gröbner bases algorithms [3,14,10,11].) These two strategies are not as different as it could seem. Indeed, to reduce the head terms of polynomials $p_1, \ldots, p_k$ the ones by the others, one determines the respective sets of multipliers $\{m_1\}, \ldots, \{m_k\}$ that are needed to do so. Then it remains to perform linear algebra on the resulting combinations and iterate with polynomials with new head terms that may be found in this process. Both strategies therefore have a clear intersection although Gröbner bases algorithms are natively more careful with the number of combinations to be dealt with.

In any case, it is convenient to arrange the available combinations with respect to their total degree. For any integer $d \geq 1$, let V_d denote the set of combinations of degree d multiples of $p_1, \ldots, p_k$. It is a linear subspace of all polynomials of degree at most d. This paper focuses on an intrinsic parameter of polynomials, which we call degree of regularity. This parameter was introduced in [2,1]. It is commonly considered as the main complexity parameter for the following intuitive reasons. Let $\mathcal{A}$ be an algorithm that computes such combinations, and indexing its execution steps by t, one may consider the subspace $V_d[\mathcal{A}(t)]$ of combinations of degree d multiples that are computed through $\mathcal{A}$ up to t. Obviously, $V_d[\mathcal{A}(t)] \subseteq V_d \subseteq R_{\leq d}$. Now, choose a target subspace $W_d \subseteq R_{\leq d}$. There exists an element of W_d among combinations in degree d when the intersection of V_d and W_d is not zero and such a combination is found by the algorithm $\mathcal{A}$ before step t if $V_d[\mathcal{A}(t)] \cap W_d \neq \{0\}$. When the polynomials $p_1, \ldots, p_k$ are not too specific, the intersection of V_d and W_d is expected to be non-zero only when the sum of their respective dimensions exceeds the dimension of $R_{\leq d}$ itself. In this case, any algorithm $\mathcal{A}$ can just consider combinations in degree d to find a non-zero element of W_d. It is assured to find one at step t if $V_d[\mathcal{A}(t)] = V_d$. On the other hand, should the intersection of V_d and W_d be non-zero at a significantly lower degree than expected for a random subspace V_d would suggest that the

polynomials $p_1, \ldots, p_k$ are not random. Interesting choices of a target subspace W_d are polynomials of low degree. For instance, one may consider whether there exists a non-zero polynomial of degree strictly lower than d among combinations in degree d. Such a combination is called a degree fall and the smallest degree at which such a degree fall occurs is essentially the degree of regularity. A precise definition will be given in the sequel. An algorithm $\mathcal{A}$ finds the degree of regularity when at some step t its subspace of combinations in degree d contains a degree fall. At this point, it is worth noting that when using a Gröbner basis algorithm it is best to use an ordering that refines the degree. Indeed in this case new head terms are confined among the smallest degree monomials.

The degree of regularity permits to distinguish a system of polynomials from random. Furthermore, any degree fall can give a new whole set of multiples in degree d or even below, which can be further combined with the existing combinations. Moreover, the dimension of $V_d^{\mathcal{A}}$ usually takes large steps as one increments d and then, many degree falls appear at once. These degree falls in turn help the appearance of new degree falls in lower degrees. Either these degree falls are low enough to solve the system (e.g. linear polynomials) or one pushes the computation until obtaining a complete Gröbner basis.

2.2 Systems with Field Equations over a Finite Field

In the setting of cryptographic schemes, the coefficient field is a finite field $\mathbb{F}_q$ (with q elements) and the solutions are searched with coordinates in this finite field. Let $x_1, \ldots, x_n$ denote the variables of R. Then one actually searches for the solutions of the system $\{p_1 = 0, \ldots, p_k = 0\}$ with the additional equations $\{x_1^q - x_1 = 0, \ldots, x_n^q - x_n = 0\}$. Equivalently, since the x_i describe values in $\mathbb{F}_q$, all monomials in R can be reduced according to the rules $x_i^q = x_i$, $i = 1, \ldots, n$. Then, all combinations of the polynomials $p_1, \ldots, p_k$ can be considered in the reduced ring $R_q = \mathbb{F}_q[x_1, \ldots, x_n]/\{x_1^q - x_1, \ldots, x_n^q - x_n\}$.

While in the sequel we compute the degree of regularity of underdetermined systems ($k \leq n$) in a reduced ring, it serves in upperbounding the degree of regularity of a public HFE system with exactly n polynomials. In this case, the expected number N of solutions is hardly more than one and it can be shown that any Gröbner basis for any ordering that refines the degree contains at least $n - N$ linearly independent degree-1 polynomials (*cf* full version). Hence, our setting makes it particularly easy to derive the solutions from a Gröbner basis.

Since in the sequel we only encounter systems of quadratic polynomials, for convenience sake we specialize the following definitions to this case. Let $p_1, \ldots, p_k$ be a system of quadratic polynomials in R_q. For any integer $d \geq 2$, consider the subspace of combinations $m_1 p_1 + \cdots + m_k p_k$ where the m_i have degree at most $d - 2$ in R_q. By definition, it is the image space of the map

$$\sigma_d(p_1, \ldots, p_k) : (m_1, \ldots, m_k) \in ((R_q)_{\leq d-2})^k \longmapsto m_1 p_1 + \cdots + m_k p_k.$$

An important observation is that the kernel of $\sigma_d(p_1, \ldots, p_k)$ always contains predictible non-zero tuples called *trivial syzygies*. Examples of trivial syzygies are the combinations over R_q of the k-tuples with $m_i = p_j$, $m_j = -p_i$ for some i, j

and 0 otherwise. A formal definition of **trivial syzygies** is the following. For indeterminates $y_1, \ldots, y_k$, let $T_q(y_1, \ldots, y_k)$ denote the set of k-tuples $(m_1, \ldots, m_k)$ over $R_q[y_1, \ldots, y_k]/\{y_1^q - y_1, \ldots, y_k^q - y_k\}$ such that $m_1 y_1 + \cdots + m_k y_k = 0$. For any polynomials $p_1, \ldots, p_k$ over R_q, we call trivial syzygies of $p_1, \ldots, p_k$ the evaluations of the k-tuples in $T_q(y_1, \ldots, y_k)$ at $(p_1, \ldots, p_k)$.

When searching for degree falls, we are only interested in the subspace V_d spanned by the highest degree homogeneous part of the image of $\sigma_d(p_1, \ldots, p_k)$. This subspace is spanned by the degree d homogeneous parts of the combinations $m_1 p_1 + \cdots + m_k p_k$ where $m_1, \ldots, m_k$ are homogeneous polynomials of degree $d-2$. We define a **degree fall** in degree d of $p_1, \ldots, p_k$ as a k-tuple $(m_1, \ldots, m_k)$ of degree $d-2$ homogeneous polynomials such that the degree d homogeneous part of $m_1 p_1 + \cdots + m_k p_k$ is zero. The degree $d-2$ homogeneous parts of the trivial syzygies of $p_1, \ldots, p_k$ in degree $d-2$ are trivially degree falls and we call them *trivial degree falls*. We call the **degree of regularity** of $p_1, \ldots, p_k$ the smallest d such that a non trivial degree fall of $p_1, \ldots, p_k$ exists in degree d.

3 Definition of HFE Systems

The construction of HFE systems is based on the linear isomorphism between $(\mathbb{F}_q)^n$ and $\mathbb{F}_{q^n}$ over $\mathbb{F}_q$. Recall that $\mathbb{F}_{q^n}$ is a degree n polynomial extension over $\mathbb{F}_q$ and as a consequence is an n dimensional vector space over $\mathbb{F}_q$. Any choice of a basis of $\mathbb{F}_{q^n}$ defines a linear bijection S from $(\mathbb{F}_q)^n$ to $\mathbb{F}_{q^n}$, and extends to a linear bijection ψ_S from functions on $\mathbb{F}_{q^n}$ to functions on $(\mathbb{F}_q)^n$ by:

$$\psi_S : P \mapsto S^{-1} \circ P \circ S$$

Recall that functions on $(\mathbb{F}_q)^n$ are uniquely represented by n-tuples of polynomials in $R_q = \mathbb{F}_q[x_1, \ldots, x_n]/\{x_1^q - x_1, \ldots, x_n^q - x_n\}$ and that functions on $\mathbb{F}_{q^n}$ are uniquely represented by polynomials in $\mathbb{F}_{q^n}[X]/\{X^{q^n} - X\}$. This gives an expression of ψ_S on polynomials: $\psi_S : \mathbb{F}_{q^n}[X]/\{X^{q^n} - X\} \longrightarrow (R_q)^n$. Also recall that raising to a power of q is linear over $\mathbb{F}_q$ and that the n distinct q-powerings on $\mathbb{F}_{q^n}$ are called the **Frobenius maps**. More generally, for any power function X^a in $\mathbb{F}_{q^n}[X]/\{X^{q^n} - X\}$, we call **$q$-degree** of X^a the sum $a_0 + \cdots + a_{n-1}$, where $(a_0, a_1, \ldots, a_{n-1})$ is the decomposition of a in base q. In particular, constants have q-degree 0 and Frobenius maps have q-degree 1. Since any function in $\mathbb{F}_{q^n}[X]/\{X^{q^n} - X\}$ is a linear combination of power functions, we define q-degree as the maximal q-degree of its terms. The following proposition ensures that ψ_S maps q-degree in $\mathbb{F}_{q^n}[X]/\{X^{q^n} - X\}$ to degree in $(R_q)^n$.

Proposition 1. *Let S be an arbitrary linear bijection from $(\mathbb{F}_q)^n$ to $\mathbb{F}_{q^n}$. For any integer $d \geq 0$, ψ_S defines a bijection from polynomials in $\mathbb{F}_{q^n}[X]/\{X^{q^n} - X\}$ with q-degree d to n-tuples over R_q with degree d.*

Please refer to the full version for a proof. We are now ready to define HFE systems. Recall from the introduction that an HFE public key is the data of the n coordinate polynomials of a composition $T \circ P \circ S$ where S is a linear bijection

from $(\mathbb{F}_q)^n$ to $\mathbb{F}_{q^n}$, T is a linear bijection from $\mathbb{F}_{q^n}$ to $(\mathbb{F}_q)^n$ and P is a function on $\mathbb{F}_{q^n}$ which as a polynomial in $\mathbb{F}_{q^n}[X]/\{X^{q^n} - X\}$ has the shape

$$P(X) = \sum_{i,j \leq D} p_{ij} X^{q^i+q^j} + \sum_{k \leq D} \lambda_k X^{q^k} + c$$

where D is a parameter of the scheme. For any linear bijection S, we call **HFE systems** the systems in $(R_q)^n$ which are the images by ψ_S of the polynomials $P(X)$ of the above shape. We see from the above proposition that HFE systems are quadratic and that their only particularity in this class is to correspond to a polynomial $P(X)$ of degree upper bounded by $2q^D$. Since T is a linear bijection, an HFE public key has all the algebraic properties of an HFE system.

4 Combinations of HFE Polynomials

In this section, we map combinations of HFE systems to related operations on the defining polynomial in $\mathbb{F}_{q^n}[X]/(X^{q^n} - X)$. This mapping was outlined in [9] and is made precise here. Incidentally, it is independent of the particular shape of HFE defining polynomials and hence is valid for any cryptosystem following a similar construction. To lighten the notation, we now denote $\mathcal{R}_{q^n} = \mathbb{F}_{q^n}[X]/(X^{q^n} - X)$. This section is a chain of technical points which are necessary to make the mapping complete. For a quick reading, one may jump directly to subsection 4.4.

4.1 From Combinations in R_q to Combinations in $\mathcal{R}_{q^n}$

Let P be any polynomial in $\mathcal{R}_{q^n}$ and $(p_1, \ldots, p_n) = \psi_S(P)$. We have defined combinations of $p_1, \ldots, p_n$ as linear combinations of $p_1, \ldots, p_n$ with coefficients in R_q. Hence, n-tuples of linear combinations over R_q are products by $n \times n$ matrices over R_q. Proposition 1 implies that ψ_S^{-1} is a linear bijection from linear maps on $(\mathbb{F}_q)^n$ to linear combinations over $\mathbb{F}_{q^n}$ of the Frobenius maps. We extend this result when coefficients are in R_q and $\mathcal{R}_{q^n}$ instead of $\mathbb{F}_q$ and $\mathbb{F}_{q^n}$.

Proposition 2. *Let S be an arbitrary linear bijection from $(\mathbb{F}_q)^n$ to $\mathbb{F}_{q^n}$. There exists an $\mathbb{F}_q$-linear bijection ψ_S^* from $(\mathcal{R}_{q^n})^n$ to $n \times n$ matrices over R_q, such that for any $M_0, \ldots, M_{n-1}$ and P in $\mathcal{R}_{q^n}$,*

$$\psi_S^*(M_0, \ldots, M_{n-1})\psi_S(P) = \psi_S(M_0 P^{q^0} + \cdots + M_{n-1} P^{q^{n-1}}). \tag{1}$$

Proof. We simply construct ψ_S^* by hand by considering the above identity over the set of constant functions $P = a$ with a in $\mathbb{F}_{q^n}$. Since ψ_S is linear we only need to consider $P = a$ for a over a basis of $\mathbb{F}_{q^n}$. For any $i = 1, \ldots, n$, let $e_i \in \mathbb{F}_{q^n}$ denote the image by S of the i-th canonical vector of $(\mathbb{F}_q)^n$. For any $M_0, \ldots, M_{n-1}$, $\psi_S^*(M_0, \ldots, M_{n-1})\psi_S(e_i)$ is the i-th column of $\psi_S^*(M_0, \ldots, M_{n-1})$ and must be set to $\psi_S(\sum_{k=0}^{n-1} M_k (e_i)^{q^k})$. ψ_S^* is linear by the linearity of ψ_S. Consider $(M_0, \ldots, M_{n-1})$ whose image by ψ_S^* is zero. Then, ψ_S being a linear bijection, for any $i = 1, \ldots, n$, we have $\sum_{k=0}^{n-1} (e_i)^{q^k} . M_k = 0$. The only solution to this invertible system is $M_0 = \cdots = M_{n-1} = 0$, which proves that ψ_S^* is injective. Surjectivity follows ψ_S^* mapping subspaces of identical dimension over $\mathbb{F}_q$. □

Equation (1) over the constants e_i also shows that the q-degree of $(M_0, \ldots, M_{n-1})$ equals the degree of $\psi_S^*(M_0, \ldots, M_{n-1})$. In particular, for any P of q-degree 2 and $d \geq 2$, we define

$$U_{\leq d}(P) = \left\{ M_0 P^{q^0} + \cdots + M_{n-1} P^{q^{n-1}} \,|\, q\text{-deg}(M_i) \leq d-2,\, i = 0, \ldots, n-1 \right\}$$

and on the other hand, for $(p_1, \ldots, p_n) = \psi_S(P)$,

$$V_{\leq d}(p_1, \ldots, p_n) = \{ m_1 p_1 + \cdots + m_n p_n \,|\, \deg(m_i) \leq d-2,\, i = 1, \ldots, n \}.$$

Property 1. For any $d \geq 2$, ψ_S is a bijection from $U_{\leq d}(P)$ to $(V_{\leq d}(p_1, \ldots, p_n))^n$.

Proof. ψ_S^* transforms n-tuples of q-degree $\leq d-2$ to $n \times n$ matrices of degree $\leq d-2$. Both spans have the same dimension over $\mathbb{F}_q$ by Proposition 1, hence ψ_S^* is a bijection from the one to the other. Finally, the property holds by the identity satisfied by ψ_S^* and evaluated at the particular P. □

Since the dimension of $(V_{\leq d})^n$ is n times the dimension of $V_{\leq d}$ and the dimension of $U_{\leq d}$ over $\mathbb{F}_q$ is n times its dimension over $\mathbb{F}_{q^n}$, the property implies

$$\dim_{\mathbb{F}_q} (V_{\leq d}(p_1, \ldots, p_n)) = \dim_{\mathbb{F}_{q^n}} (U_{\leq d}(P)).$$

4.2 From Degree Falls in R_q to q-Degree Falls in $\mathcal{R}_{q^n}$

When considering degree falls, one is really interested in the subspace spanned by the highest degree homogeneous part of a bounded degree combination space. For any quadratic polynomials $p_1, \ldots, p_n$ in R_q and any integer $d \geq 2$, let $V_d^h(p_1, \ldots, p_n)$ denote the subspace generated by the degree d homogeneous parts of polynomials in $V_{\leq d}(p_1, \ldots, p_n)$. Similarly, for any polynomial P of q-degree 2 in $\mathcal{R}_{q^n}$ and any integer $d \geq 2$, let $U_d^h(P)$ denote the subspace of q-degree d homogeneous parts of polynomials in $U_{\leq d}(P)$. Quite expectably, we have:

Property 2. Let P in $\mathcal{R}_{q^n}$ and $(p_1, \ldots, p_n) = \psi_S(P)$. Then, for any $d \geq 2$, there exists an $\mathbb{F}_q$-linear bijection from $U_d^h(P)$ to $(V_d^h(p_1, \ldots, p_n))^n$.

Proof. The highest degree homogeneous part of a polynomial p in R_q with degree $d \geq 2$ is its class mod $(R_q)_{\leq d-1}$. Hence, $V_d^h(p_1, \ldots, p_n)$ is $V_{\leq d}(p_1, \ldots, p_n)$ mod $(R_q)_{\leq d-1}$. Similarly $U_d^h(P)$ is $U_{\leq d}(P)$ mod $(\mathcal{R}_{q^n})_{\leq d-1}$. Let Q and Q' be arbitrary polynomials in $\mathcal{R}_{q^n}$ such that $Q = Q'$ mod $(\mathcal{R}_{q^n})_{\leq d-1}$. Then, $Q - Q'$ has q-degree at most $d-1$. Since ψ_S preserves the degree, $\psi_S(Q - Q')$ has degree at most $d-1$. Hence, since ψ_S is linear, $\psi_S(Q) = \psi_S(Q')$ mod $((R_q)_{\leq d-1})^n$. Therefore, ψ_S induces an $\mathbb{F}_q$-linear map from $\mathcal{R}_{q^n}$ mod $(\mathcal{R}_{q^n})_{\leq d-1}$ to $(R_q)^n$ mod $((R_q)_{\leq d-1})^n$. Since ψ_S is a bijection from $U_{\leq d}(P)$ to $(V_{\leq d}(p_1, \ldots, p_n))^n$, the induced map is a bijection from $U_d^h(P)$ to $(V_d^h(p_1, \ldots, p_n))^n$. □

Let R_q^h denote the set of homogeneous polynomials of R_q. For any polynomial p in R_q and any integer $d \geq 0$, let $[p]_d$ denote the degree d homogeneous part of

p. For any system $p_1, \ldots, p_n$ of quadratic polynomials in R_q and any $d \geq 2$, the degree falls of $p_1, \ldots, p_n$ in degree d are the kernel of the map

$$\sigma_d^h(p_1, \ldots, p_n) : (m_1, \ldots, m_n) \in ((R_q^h)_{d-2})^n \longmapsto [m_1 p_1 + \cdots + m_n p_n]_d.$$

With completely transposed notations, for any P of q-degree 2 in $\mathcal{R}_{q^n}$ and any $d \geq 2$, we define the q-degree falls of P in degree d as the kernel of the map

$$\Sigma_d^h(P) : (M_0, \ldots, M_{n-1}) \in ((\mathcal{R}_{q^n}^h)_{d-2})^n \mapsto [M_0 P + M_1 P^q + \cdots + M_{n-1} P^{q^{n-1}}]_d$$

The image spaces $\sigma_d^h(p_1, \ldots, p_n)$ and $\Sigma_d^h(P)$ respectively are $V_d^h(p_1, \ldots, p_n)$ and $U_d^h(P)$. Property 2 ensures that when $(p_1, \ldots, p_n) = \psi_S(P)$ the image spaces of $(\sigma_d^h(p_1, \ldots, p_n))^n$ and $\Sigma_d^h(P)$ have the same cardinality. Besides, Proposition 1 ensures that the same holds for their input spaces. Therefore, the kernels of $(\sigma_d^h(p_1, \ldots, p_n))^n$ and $\Sigma_d^h(P)$ have the same cardinality. Finally,

$$\dim_{\mathbb{F}_q}(\ker \sigma_d^h(p_1, \ldots, p_n)) = \dim_{\mathbb{F}_{q^n}}(\ker \Sigma_d^h(P)). \tag{2}$$

4.3 Trivial Syzygies and Trivial Degree Falls

Trivial syzygies of $p_1, \ldots, p_n$ are n-tuples over R_q such that $m_1 p_1 + \cdots + m_n p_n = 0$ even when $p_1, \ldots, p_n$ are indeterminates. They are precisely defined the following way. Let $\bar{R}_q$ denote the extension of R_q with additional variables $y_1, \ldots, y_n$, $\bar{R}_q = R_q[y_1, \ldots, y_n]/\{y_1^q - y_1, \ldots, y_n^q - y_n\}$. Let $T_q(y_1, \ldots, y_n)$ denote the set of n-tuples $(m_1, \ldots, m_n)$ over $\bar{R}_q$ such that $m_1 y_1 + \cdots + m_n y_n = 0$. For any polynomials $p_1, \ldots, p_n$ in R_q, we define its trivial syzygies as the evaluations of the n-tuples in $T_q(y_1, \ldots, y_n)$ at $(p_1, \ldots, p_n)$. As a shorthand, let $T_q(p_1, \ldots, p_n)$ denote the set of trivial syzygies of $p_1, \ldots, p_n$.

Elements of $\bar{R}_q$ are polynomials in both $x_1, \ldots, x_n$ and $y_1, \ldots, y_n$. For any monomial in $\bar{R}_q$, let d_x, d_y denote its degrees in $x_1, \ldots, x_n$ and in $y_1, \ldots, y_n$ respectively. Since variables $y_1, \ldots, y_n$ are intended to be specialized at quadratic polynomials $p_1, \ldots, p_n$ in R_q, we define the $(1,2)$**-degree** of a monomial in $\bar{R}_q$ as $d_x + 2d_y$, and the $(1,2)$-degree of a polynomial in $\bar{R}_q$ as the maximum of the $(1,2)$-degree of its monomials. Hence, any element of $T_q(y_1, \ldots, y_n)$ with $(1,2)$-degree d yields an element of $T_q(p_1, \ldots, p_n)$ with degree $\leq d$. We call trivial syzygies of $p_1, \ldots, p_n$ with designed degree d the elements of $T_q(p_1, \ldots, p_n)$ whose corresponding element of $T_q(y_1, \ldots, y_n)$ has $(1,2)$-degree d. The trivial syzygies with designed degree $\leq d$ are denoted by $T_q(p_1, \ldots, p_n)_{\preceq d}$. On the other hand, one may analogously consider the extension of $\mathcal{R}_{q^n}$ with additional variable Y, $\bar{\mathcal{R}}_{q^n} = \mathcal{R}_{q^n}[Y]/(Y^{q^n} - Y)$, and define $\mathcal{T}_{q^n}(Y)$ as the n-tuples $(M_0, \ldots, M_{n-1})$ over $\bar{\mathcal{R}}_{q^n}$ such that $M_0 Y + M_1 Y^q + \cdots + M_{n-1} Y^{q^{n-1}} = 0$. For any P in $\mathcal{R}_{q^n}$, let $\mathcal{T}_{q^n}(P)$ denote the evaluations of the n-tuples in $\mathcal{T}_{q^n}(Y)$ at P. Finally, for any P of q-degree 2 and any $d \geq 0$, we let $\mathcal{T}_{q^n}(P)_{\preceq d}$ denote the elements whose corresponding elements in $\mathcal{T}_{q^n}(Y)$ have $(1,2)$-q-degree d. By a series of simple extensions of the previous results, we can show (*cf* full version)

Property 3. Let P in $\mathcal{R}_{q^n}$ of q-degree 2 and $(p_1, \ldots, p_n) = \psi_S(P)$. For any $d \geq 0$,

$$\dim_{\mathbb{F}_q}(T_q(p_1, \ldots, p_n)_{\preceq d}) = \dim_{\mathbb{F}_{q^n}}(\mathcal{T}_{q^n}(P)_{\preceq d}).$$

The polynomials $p_1, \ldots, p_n$ being quadratic, for any $d \geq 2$, we call trivial degree falls of $p_1, \ldots, p_n$ in degree d the homogeneous parts of (actual) degree $d-2$ of the elements in $T_q(p_1, \ldots, p_n)_{\preceq d-2}$ and denote them (with a slight abuse of notation) by $T_q(p_1, \ldots, p_n)^h_{d-2}$. Similarly, for P of q-degree 2, we call trivial q-degree falls of P in q-degree d the homogeneous parts of q-degree $d-2$ of the elements in $\mathcal{T}_q(P)_{\preceq d-2}$ and denote them by $\mathcal{T}_q(P)^h_{d-2}$. We have (*cf* full version)

Property 4. Let P in $\mathcal{R}_{q^n}$ of q-degree 2 and $(p_1, \ldots, p_n) = \psi_S(P)$. For any $d \geq 2$,

$$\dim_{\mathbb{F}_q}(T_q(p_1, \ldots, p_n)^h_{d-2}) = \dim_{\mathbb{F}_{q^n}}(\mathcal{T}_{q^n}(P)^h_{d-2}).$$

4.4 Mapping the Degree of Regularity from R_q to $\mathcal{R}_{q^n}$

Recall that the degree of regularity of a system of quadratic polynomials $p_1, \ldots, p_n$ is the smallest integer d such that a non-trivial degree fall exists in degree d. With the previous notation, this is the smallest d such that the kernel of $\sigma^h_d(p_1, \ldots, p_n)$ is strictly larger than $T_q(p_1, \ldots, p_n)^h_{d-2}$. Now, let S be an arbitrary linear bijection from $(\mathbb{F}_q)^n$ to $\mathbb{F}_{q^n}$ and P in $\mathcal{R}_{q^n}$ such that $\psi_S(P) = (p_1, \ldots, p_n)$. Then, P has q-degree 2 and, by Equality 2 and Property 3,

Property 5. the degree of regularity of $p_1, \ldots, p_n$ is the smallest d such that the kernel of $\Sigma^h_d(P)$ is strictly larger than $\mathcal{T}_{q^n}(P)^h_{d-2}$.

Hence, we obtain an equivalent characterization of the degree of regularity of $p_1, \ldots, p_n$ in term of the associated polynomial P in $\mathcal{R}_{q^n}$. In the remainder of this section, we slightly modify the above characterization to make it more conveniently usable in the analysis of the next section.

Multivariate representation of $\mathcal{R}_{q^n}$. Our first step is a simple alternative notation for the elements $\mathcal{R}_{q^n}$. This notation was proposed in [9]. As already seen, we can split any power of X according to the decomposition in base q of the exponent. Now simply introduce a distinct notation for the Frobenius of X: for $i = 0, \ldots, n-1$, let X_i denote X^{q^i}. Observe that for any $i = 0, \ldots, n-1$, $X_i^q - X_{i+1} = 0$ where the indices are taken modulo n. Using these relations, any power of X corresponds to a unique multivariate monomial in $X_0, \ldots, X_{n-1}$. It extends trivially to all polynomials in $\mathcal{R}_{q^n}$. Addition and multiplication are compatible with this notation. Therefore, $\mathcal{R}_{q^n}$ identifies as a ring with $\mathbb{F}_{q^n}[X_0, \ldots, X_{n-1}]/\{X_0^q - X_1, \ldots, X_{n-1}^q - X_0\}$. Along with this identification, q-degree becomes degree in the multivariate ring. Also, for any polynomial P in $\mathcal{R}_{q^n}$, let $P_0, \ldots, P_{n-1}$ denote its successsive Frobenius. For any $i = 0, \ldots, n-1$, $P_i^q - P_{i+1} = 0$ where indices are modulo n. When P has q-degree 2, its Frobenius are multivariate quadratic polynomials. Since the P-termed sets really express in terms of the Frobenius of P, they are conveniently rewritten with the above notation. Hence, $\Sigma^h_d(P)$ rewrites to

$$\Sigma^h_d(P_0, \ldots, P_{n-1}) : (M_0, \ldots, M_{n-1}) \in (\mathcal{R}^h_{q^n})_{d-2} \mapsto [M_0 P_0 + \cdots + M_{n-1} P_{n-1}]_d.$$

The ring $\bar{\mathcal{R}}_{q^n} = \mathcal{R}_{q^n}[Y]/(Y^{q^n} - Y)$ rewrites to $\mathcal{R}_{q^n}[Y_0, \ldots, Y_{n-1}]/\{Y_0^q - Y_1, \ldots, Y_{n-1}^q - Y_0\}$. The set $\mathcal{T}_{q^n}(Y)$ rewrites to $\mathcal{T}_{q^n}(Y_0, \ldots, Y_{n-1})$, the n-tuples $(M_0, \ldots, M_{n-1})$ over $\bar{\mathcal{R}}_{q^n}$ such that $M_0Y_0 + \cdots + M_{n-1}Y_{n-1} = 0$. Hence, $\mathcal{T}_{q^n}(P)$ identifies with $\mathcal{T}_{q^n}(P_0, \ldots, P_{n-1})$. And the elements of $\mathcal{T}_{q^n}(P_0, \ldots, P_{n-1})_d^h$ are the degree d homogeneous parts of the elements of $\mathcal{T}_{q^n}(P_0, \ldots, P_{n-1})_{\preceq d}$. Finally, our characterization (Property 5) rewrites to, when $(p_1, \ldots, p_n) = \psi_S(P)$,

Property 6. the degree of regularity of $p_1, \ldots, p_n$ equals the degree of regularity of $P_0, \ldots, P_{n-1}$, the n Frobenius of P in the multivariate representation of $\mathcal{R}_{q^n}$.

At this point, our task is reduced to studying the degree of regularity of the quadratic polynomials $P_0, \ldots, P_{n-1}$ in $\mathcal{R}_{q^n}$, and we do not need to address the polynomials $p_1, \ldots, p_n$ any further. The next paragraph is devoted to refining the characterization of the degree of regularity of $P_0, \ldots, P_{n-1}$.

Characterizing the Degree of Regularity of Systems of $\mathcal{R}_{q^n}$. Our first observation is a simple one: the highest degree terms of combinations in degree d of $P_0, \ldots, P_{n-1}$ only depends on their highest degree terms $\hat{P}_0, \ldots, \hat{P}_{n-1}$.

Property 7. The degree of regularity of quadratic polynomials in $\mathcal{R}_{q^n}$ equals the degree of regularity of their degree 2 homogeneous parts.

Proof. For any degree $d-2$ homogeneous polynomials $M_0, \ldots, M_{n-1}$, the associated combinations of $P_0, \ldots, P_{n-1}$ and $\hat{P}_0, \ldots, \hat{P}_{n-1}$ have the same degree d homogeneous part. Hence, degree falls in degree d are the same for both systems of polynomials. On the other hand, the trivial syzygies of $P_0, \ldots, P_{n-1}$ of designed degree $d-2$ have the same degree $d-2$ homogeneous parts as the trivial syzygies of $\hat{P}_0, \ldots, \hat{P}_{n-1}$ of designed degree $d-2$. The property follows. □

Our second observation is more subtle: when considering combinations of the quadratic homogeneous polynomials $\hat{P}_0, \ldots, \hat{P}_{n-1}$ with degree $d-2$ homogeneous coefficients, terms of degree smaller than d can only appear with reductions modulo the polynomials $X_i^q - X_{i+1}$, $i = 0, \ldots, n-1$. Since all terms with degree smaller than d are discarded, the same result is obtained as when performing combinations in the ring $\boldsymbol{\mathcal{R}}_{q^n} = \mathbb{F}_{q^n}[X_0, \ldots, X_{n-1}]/\{X_0^q, \ldots, X_{n-1}^q\}$. Considering combinations in $\boldsymbol{\mathcal{R}}_{q^n}$ rather than in $\mathcal{R}_{q^n}$, the map $\Sigma_d^h(\hat{P}_0, \ldots, \hat{P}_{n-1})$ simply rewrites to $\boldsymbol{\Sigma}_d^h(\hat{P}_0, \ldots, \hat{P}_{n-1})$:

$$(M_0, \ldots, M_{n-1}) \in ((\boldsymbol{\mathcal{R}}_{q^n}^h)_{d-2})^n \mapsto M_0\hat{P}_0 + M_1\hat{P}_1 + \cdots + M_{n-1}\hat{P}_{n-1}.$$

Furthermore, we can equivalently characterize the trivial degree falls using the ring structure of $\boldsymbol{\mathcal{R}}_{q^n}$. Consider $\bar{\boldsymbol{\mathcal{R}}}_{q^n} = \boldsymbol{\mathcal{R}}_{q^n}[Y_0, \ldots, Y_{n-1}]/\{Y_0^q, \ldots, Y_{n-1}^q\}$ and the associated set $\boldsymbol{\mathcal{T}}_{q^n}(Y_0, \ldots, Y_{n-1})$. For any $d \geq 0$, we can define the sets $\boldsymbol{\mathcal{T}}_{q^n}(\hat{P}_0, \ldots, \hat{P}_{n-1})_{\preceq d}$ and $\boldsymbol{\mathcal{T}}_{q^n}(\hat{P}_0, \ldots, \hat{P}_{n-1})_d^h$, exactly as before.

Property 8. For any $d \geq 0$, the sets $\mathcal{T}_{q^n}(\hat{P}_0, \ldots, \hat{P}_{n-1})_d^h$ and $\boldsymbol{\mathcal{T}}_{q^n}(\hat{P}_0, \ldots, \hat{P}_{n-1})_d^h$ are identical. Therefore, for any $d \geq 2$, the trivial degree falls of $\hat{P}_0, \ldots, \hat{P}_{n-1}$ in degree d are the elements of $\boldsymbol{\mathcal{T}}_{q^n}(\hat{P}_0, \ldots, \hat{P}_{n-1})_{d-2}^h$.

Proof. For any $(M_0, \ldots, M_{n-1})$ in $\boldsymbol{\mathcal{T}}_{q^n}(Y_0, \ldots, Y_{n-1})$, let Q denote the combination $M_0Y_0 + \cdots + M_{n-1}Y_{n-1}$ in $\mathcal{R}_{q^n}[Y_0, \ldots, Y_{n-1}]$. Since Q is zero modulo $Y_0^q, \ldots, Y_{n-1}^q$, any of its term is divisible by at least one of $Y_0^q, \ldots, Y_{n-1}^q$. Since $M_0, \ldots, M_{n-1}$ have degree at most $q-1$ in any Y_i, any term of Q can have degree q in only one single indeterminate and at most $q-1$ in all the others. Therefore, any term of Q exactly has degree q is one indeterminate and at most $q-1$ in all the others. Hence, Q admits a unique decomposition $A_0Y_0^q + \cdots + A_{n-1}Y_{n-1}^q$. Using the unique polynomials $A_0, \ldots, A_{n-1}$ associated to $(M_0, \ldots, M_{n-1})$, we construct an element $(M'_0, \ldots, M'_{n-1})$ of $\mathcal{T}_{q^n}(Y_0, \ldots, Y_{n-1})$ by setting for all $i = 0, \ldots, n-1$, $M'_i = M_i - A_{i-1}$ (indices are modulo n). Now, observe that the terms of $A_0, \ldots, A_{n-1}$ consist of terms of $M_0, \ldots, M_{n-1}$ divided by one indeterminate to the power of $q-1$. As a consequence, each of them has a total degree in the Y_i variables smaller (by $q-1$) than the one it originates from. In particular, when $M_0, \ldots, M_{n-1}$ have $(1,2)$-degree at most d, $M'_0, \ldots, M'_{n-1}$ respectively have the same terms of $(1,2)$-degree d as $M_0, \ldots, M_{n-1}$ because they differ by terms of strictly smaller $(1,2)$-degree. □

Hence, we end up with the following characterization which we use in the sequel.

Property 9. Let $\hat{P}_0, \ldots, \hat{P}_{n-1}$ be homogeneous quadratic polynomials in $\mathcal{R}_{q^n}$. The degree of regularity of $\hat{P}_0, \ldots, \hat{P}_{n-1}$ can be computed in $\boldsymbol{\mathcal{R}}_{q^n}$ as the smallest $d \geq 2$ such that degree $d-2$ homogeneous n-tuples $(M_0, \ldots, M_{n-1})$ satisfying $M_0\hat{P}_0 + \cdots + M_{n-1}\hat{P}_{n-1} = 0$ exist besides the elements of $\boldsymbol{\mathcal{T}}_{q^n}(\hat{P}_0, \ldots, \hat{P}_{n-1})_d^h$.

5 Bounding the Degree of Regularity of HFE Systems

We first describe the proof principle of our upper bound and then perform the combinatorial computations that convey the result.

5.1 Upper Bounding the Degree of Regularity

First consider arbitrary homogeneous quadratic polynomials $\hat{P}_0, \ldots, \hat{P}_{k-1}$ in $\boldsymbol{\mathcal{R}}_{q^n}$ where $k \leq n$. The dimensions of the kernel and the image of the map

$$\begin{aligned} \boldsymbol{\Sigma}_d^h(\hat{P}_0, \ldots, \hat{P}_{k-1}) : \quad ((\boldsymbol{\mathcal{R}}_{q^n})_{d-2}^h)^k &\longrightarrow (\boldsymbol{\mathcal{R}}_{q^n})_d^h \\ (M_0, \ldots, M_{k-1}) &\longmapsto M_0\hat{P}_0 + M_1\hat{P}_1 + \cdots + M_{k-1}\hat{P}_{k-1} \end{aligned}$$

relate to each other by

$$k \dim(\boldsymbol{\mathcal{R}}_{q^n})_{d-2}^h - \dim \ker \boldsymbol{\Sigma}_d^h(\hat{P}_0, \ldots, \hat{P}_{k-1}) = \dim \operatorname{Im}(\boldsymbol{\Sigma}_d^h(\hat{P}_0, \ldots, \hat{P}_{k-1})).$$

Not knowing what the degree of regularity of the system is, one can assume that it is not reached while incrementing d. In this case, the kernel is assumed to contain only the trivial elements of $\boldsymbol{\mathcal{T}}_{q^n}(\hat{P}_0, \ldots, \hat{P}_{k-1})_{d-2}^h$. Since the image is confined in $(\boldsymbol{\mathcal{R}}_{q^n})_d^h$, a contradiction to this assumption appears as soon as

$$k \dim(\boldsymbol{\mathcal{R}}_{q^n})_{d-2}^h - \dim \boldsymbol{\mathcal{T}}_{q^n}(\hat{P}_0, \ldots, \hat{P}_{k-1})_{d-2}^h > \dim(\boldsymbol{\mathcal{R}}_{q^n})_d^h$$

and then we know that the degree of regularity was reached before. The smallest d satisfying this "saturation" condition is therefore an upper bound on the degree of regularity of $\hat{P}_0, \ldots, \hat{P}_{k-1}$. Since it is valid for any homogeneous quadratic polynomials, we refer to it as the *MQ bound.*

We now show how in the case of HFE systems better bounds can be obtained.

5.2 The Case of HFE Systems

It was noted in [15] that when $\hat{P}_0, \ldots, \hat{P}_{n-1}$ are obtained from the Frobenius of an HFE polynomial P, they express over small shifted sets of consecutive variables: $\hat{P}_0$ expresses over X_0 to X_D, $\hat{P}_1$ expresses over X_1 to X_{D+1}, ..., $\hat{P}_{n-1}$ expresses over X_{n-1} to X_{D-1} (indices are modulo n). Then, the authors noted that a consequence of this property is that small subsystems of consecutive polynomials only involve a small subset of the available variables. Consecutive subsystems of a prescribed size being all equivalent up to a cyclic shift, we focus on the n subsystems $\mathcal{S}_k = \{\hat{P}_0, \ldots, \hat{P}_{k-1}\}$ for $k = 1, \ldots, n$. The subsystem $\mathcal{S}_k$ expresses over the first m_k variables, where $m_k = D + k$ for all $k \leq n - D$ and $m_k = n$ beyond. The degree of regularity of $\hat{P}_0, \ldots, \hat{P}_{n-1}$ is upper bounded by the respective degrees of regularity d_k of the subsystems $\mathcal{S}_k$ for all $k = 1, \ldots, n$. Indeed the degree falls of $\mathcal{S}_k$ identify with the degree falls of $\hat{P}_0, \ldots, \hat{P}_{n-1}$ with zero on the last $n-k$ coordinates. On the other hand we will show in Section 5.3 (Property 11) that whenever a degree fall is non-trivial for $\mathcal{S}_k$, its completion with zero on the last $n - k$ coordinates is non-trivial for $\hat{P}_0, \ldots, \hat{P}_{n-1}$. At this point, the authors of [15] estimated the degree of regularity of any subsystem $\mathcal{S}_k$ by using an asymptotic formula from [2]. This needed restricting to $q = 2$ and assuming that the quadratic polynomials $\hat{P}_0, \ldots, \hat{P}_{k-1}, X_0^2, \ldots, X_{m_k-1}^2$ satisfy the condition for which the formula holds. Instead, we use the previous saturation bound: we upper bound the degree of regularity of $\mathcal{S}_k$ by applying the MQ bound to $\hat{P}_0, \ldots, \hat{P}_{k-1}$. Hence it is upper bounded by the smallest d such that

$$k \dim(\boldsymbol{\mathcal{R}}_{q^n|m_k})_{d-2}^h - \dim \boldsymbol{\mathcal{T}}_{q^n|m_k}(\hat{P}_0, \ldots, \hat{P}_{k-1})_{d-2}^h > \dim(\boldsymbol{\mathcal{R}}_{q^n|m_k})_d^h \quad (3)$$

where $\boldsymbol{\mathcal{R}}_{q^n|m_k}$ denotes the restriction of $\boldsymbol{\mathcal{R}}_{q^n}$ to the first m_k variables. Since this upper bound uses a property showed in [15], we refer to it as the *GJS bound.*

We now observe an additional property of HFE systems. Since polynomials $\hat{P}_0, \ldots, \hat{P}_{n-1}$ write over monomials $X_i X_{i+\ell}$ with $\ell \leq D$, combinations of these polynomials necessarily write over the monomials which are divisible by $X_i X_{i+\ell}$ for some i and $\ell \leq D$. Let $\boldsymbol{\mathcal{M}}_q^D$ denote the subspace spanned by such monomials. For any subsystem $\mathcal{S}_k$, we improve the GJS bound by the smallest d such that

$$k \dim(\boldsymbol{\mathcal{R}}_{q^n|m_k})_{d-2}^h - \dim \boldsymbol{\mathcal{T}}_{q^n|m_k}(\hat{P}_0, \ldots, \hat{P}_{k-1})_{d-2}^h > \dim(\boldsymbol{\mathcal{M}}_q^D|m_k)_d^h \quad (4)$$

where $(\boldsymbol{\mathcal{M}}_q^D|m_k)_d^h$ denotes the subspace spanned by degree d monomials of $\boldsymbol{\mathcal{M}}_q^D$ in the first m_k variables. The distinction between $\boldsymbol{\mathcal{M}}_q^D$ and $\boldsymbol{\mathcal{R}}_{q^n}^h$ is increasingly significant as q grows. Indeed, at fixed n and d, the average Hamming weight of multidegrees in degree d decreases as q grows. Then, the proportion of monomials

containing two variables distant by at most D indices (mod n) grows thiner. We call *HFE bound* the upper bound on the degree of regularity of $\hat{P}_0, \ldots, \hat{P}_{n-1}$ obtained from the latter improvement.

We now compute for any $d \geq 2$ and any $k = 1, \ldots, n$, the above dimensions by means of induction formulae and deduce the related numerical upper bound.

5.3 Induction Formulae for Computing Our Upper Bound

We show how to compute the dimensions of $(\boldsymbol{\mathcal{R}}_{q^n|m})_d^h$, $(\boldsymbol{\mathcal{M}}_q^D|m)_d^h$ and $\boldsymbol{\mathcal{T}}_{q^n|m}(\hat{P}_0, \ldots, \hat{P}_{k-1})_d^h$, for any $m, k = 1, \ldots, n$.

The dimension $H(m, d)$ of $(\boldsymbol{\mathcal{R}}_{q^n|m})_d^h$ is simply the number of homogeneous monomials of degree d in m variables, where all exponents are bounded between 0 and $q-1$. Obviously, it equals $H(m, d) = 0$ for $d < 0$, or $d > 0$ and $m \leq 0$, we have $H(m, 0) = 1$ for all m, and when $d > 0, m > 0$ it satisfies the induction $H(m, d) = \sum_{\alpha=0}^{q-1} H(m-1, d-\alpha)$. Equivalently, $H(m, d)$ is the d-th term of the series $((1-z^q)/(1-z))^u$ of term z. In particular, for $q = 2$, $H(u, d) = \binom{u}{d}$.

The Number of Monomials Arising in Combinations of HFE. For any $d \geq 0$, and $u = 1, \ldots, n$, let $C(u, d)$ denote the dimension the complement of $(\boldsymbol{\mathcal{M}}_q^D|u)_d^h$ in $(\boldsymbol{\mathcal{R}}_{q^n|u})_d^h$. This is the number of monomials of degree d in u consecutive variables, with exponents modulo q, such that all variables with non-zero exponents have indices (modulo n) distant by at least $D+1$ positions. First, ignore that distance between indices is taken modulo n, and that we allow for instance X_0 and X_{u-1} to have both a non-zero power. Then, $C'(u, d)$ is given by the simple "Pascal's triangle" formula $C'(u, d) = C'(u-1, d) + \sum_{\alpha=1}^{q-1} C'(u-D-1, d-\alpha)$ for any $u = 1, \ldots, n$, where $C(u, 0) = 1$ and $C(u, d) = 0$ whenever $d < 0$ or $u \leq 0$. When u is lower than $n-D$, then the requested dimension $C(u, d)$ is there equal to $C'(u, d)$ since the last D variables have zero exponents. Otherwise, when $u > n-D$, the distance must be taken modulo n, so we deduce all values of $C(n, d)$ by considering the partitions defined by monomials containing X_0, plus monomials containing X_1, ..., plus monomials containing X_{D-1}, plus monomials containing none of them. Hence, $C(u, d) = C'(u-D, d) + D\sum_{\alpha=1}^{q-1} C'(n-1-2D, d-\alpha)$. Finally, $\dim(\boldsymbol{\mathcal{M}}_q^D|u)_d^h = H(u, d) - C(u, d)$.

The Dimension of Trivial Syzygies in Degree *d*. Simply denote $\boldsymbol{\mathcal{R}}_{q^n|m}$ by $\boldsymbol{\mathcal{R}}_m$. Our first step is to exhibit generators for the module $\boldsymbol{\mathcal{T}}_m(Y_0, \ldots, Y_{k-1})$.

Property 10. An n-tuple $(M_0, \ldots, M_{k-1})$ is an element of $\boldsymbol{\mathcal{T}}_m(Y_0, \ldots, Y_{k-1})$ if and only if it is a combination with polynomial coefficients of the n-tuples

$$\begin{cases} \Gamma_{ij} = (0, \ldots, 0, M_i = Y_j, 0, \ldots, 0, M_j = -Y_i, 0, \ldots, 0), & i, j = 0, \ldots, k-1, \\ \Phi_i = (0, \ldots, 0, M_i = Y_i^{q-1}, 0, \ldots, 0), & i = 0, \ldots, k-1. \end{cases}$$

Proof. For any n-tuple $(M_0, \ldots, M_k)$, decompose M_i into $\bar{M}_i Y_i^{q-1} + M_i'$. An n-tuple $(M_0, \ldots, M_k)$ is an element of $\boldsymbol{\mathcal{T}}_{q^n}(Y_0, \ldots, Y_k)$ if and only if $M_0 Y_0 + \cdots +$

M_kY_k is zero modulo $Y_0^q, \ldots, Y_k^q$. This is equivalent to $M'_0Y_0 + \cdots + M'_kY_k = 0$ (without modulo). We prove that this latter equality implies that $(M'_0, \ldots, M'_k)$ is a combination of (Γ_{ij}). We do this through induction on k. If $k = 1$ then, if M'_0 or M'_1 is zero they are both zero, otherwise $M'_0 = M''Y_1$ and $M'_1 = -M''Y_0$ and $(M'_0, M'_1) = M''(Y_1, -Y_0)$. Assume the property holds up to $k-1$. Then, if M'_k is zero, we fall on the property at $k-1$, otherwise all M'_i, $i = 0, \ldots, k-1$ must contain Y_k and denoting by M''_i the quotient, we have $M'_k = -(M''_0Y_0 + \cdots + M''_{k-1}Y_{k-1})$, from which we get $(M'_0, \ldots, M'_k) = M''_0\Gamma_{0,k} + \cdots + M''_{k-1}\Gamma_{k-1,k}$. Coming back to the main proof, we get$(M_0, \ldots, M_k) = \bar{M}_0\Phi_0 + \cdots + \bar{M}_k\Phi_k + (M'_0, \ldots, M'_k)$where the last n-tuple decomposes over (Γ_{ij})'s. □

Since Γ_{ij}'s and Φ_i's are homogeneous in the variables $Y_0, \ldots, Y_{k-1}$, the $(1,2)$-degree d parts of the elements of $\boldsymbol{\mathcal{T}}_m(Y_0, \ldots, Y_{k-1})$ themselves decompose over these generators. Replacing variables $Y_0, \ldots, Y_{k-1}$ respectively by $\hat{P}_0, \ldots, \hat{P}_{k-1}$, trivial syzygies in degree d of $\hat{P}_0, \ldots, \hat{P}_{k-1}$ write

$$\boldsymbol{\mathcal{T}}_m(\hat{P}_0, \ldots, \hat{P}_{k-1})^h_d = (\boldsymbol{\mathcal{R}}_m)^h_{d-2}\{\Gamma_{ij}\}_{0\le i<j\le k-1} + (\boldsymbol{\mathcal{R}}_m)^h_{d-2(q-1)}\{\Phi_i\}_{0\le i\le k-1},$$

where we again denote by Γ_{ij}'s and Φ_i's their specializations at $(\hat{P}_0, \ldots, \hat{P}_{k-1})$. Unfortunately, decomposition over the above generators is not unique. Therefore, the dimension of $\boldsymbol{\mathcal{T}}_m(\hat{P}_0, \ldots, \hat{P}_{k-1})^h_d$ can not be directly read from the above formula. However, we see that this dimension follows a simple induction.

Let $\partial\Gamma_{d,k}$ denote the subspace spanned by $\Gamma_{i,k}$, $i = 0, \ldots, k-1$ $(k \ge 1)$ and $\partial\Phi_{d,k}$ denote the subspace spanned by Φ_k. Then, for $k \ge 1$,

$$\boldsymbol{\mathcal{T}}_m(\hat{P}_0, \ldots, \hat{P}_k)^h_d = \boldsymbol{\mathcal{T}}_m(\hat{P}_0, \ldots, \hat{P}_{k-1})^h_d + (\partial\Gamma_{d,k} + \partial\Phi_{d,k}). \tag{5}$$

For $k = 1$, we simply have $\boldsymbol{\mathcal{T}}_m(\hat{P}_0)^h_d = \partial\Phi^1_d$. For all $k \ge 1$, the increase of dimension when adding $\partial\Gamma_{d,k} + \partial\Phi_{d,k}$ is the dimension of the quotient space $(\partial\Gamma_{d,k} + \partial\Phi_{d,k}) \bmod \boldsymbol{\mathcal{T}}_m(\hat{P}_0, \ldots, \hat{P}_{k-1})^h_d$. Now we use the following property.

Property 11. For d up to the degree of regularity of $\hat{P}_0, \ldots, \hat{P}_k$,

$$\boldsymbol{\mathcal{T}}_m(\hat{P}_0, \ldots, \hat{P}_k)^h_d \cap \{(*, \ldots, *, 0)\}_d = \boldsymbol{\mathcal{T}}_m(\hat{P}_0, \ldots, \hat{P}_{k-1})^h_d.$$

(Hence, the degree of regularity of $\hat{P}_0, \ldots, \hat{P}_k$ is upper-bounded by the degree of regularity of $\hat{P}_0, \ldots, \hat{P}_{k-1}$ because a cancellation of $\hat{P}_0, \ldots, \hat{P}_{k-1}$ which is non-trivial in the sense of $\hat{P}_0, \ldots, \hat{P}_{k-1}$ is non-trivial in the sense of $\hat{P}_0, \ldots, \hat{P}_k$.)

Proof. First recall that Γ_{ij}'s have degree 2 and ϕ_i's have degree $2(q-1) \ge 2$. As a consequence $\boldsymbol{\mathcal{T}}_m(\hat{P}_0, \ldots, \hat{P}_{k-1})$ has no element in degree 0 or 1.

For any $1 \le \alpha \le q$ and $d \ge 0$, define the set

$$\boldsymbol{\mathcal{T}}^{*\alpha}_m(\hat{P}_0, \ldots, \hat{P}_k)^h_d = \left\{ (M_0, \ldots, M_{k-1}, 0) \;\middle|\; \begin{array}{r} \exists M_k^{(\alpha)}, (M_0, \ldots, M_{k-1}, M_k^{(\alpha)} P_k^{q-\alpha}) \\ \in \boldsymbol{\mathcal{T}}_m(\hat{P}_0, \ldots, \hat{P}_k)^h_d \end{array} \right\}$$

Observe that for $\alpha = 1$, this set is exactly $\boldsymbol{\mathcal{T}}_m(\hat{P}_0, \ldots, \hat{P}_k)^h_d \cap \{(*, \ldots, *, 0)\}_d$. We show that for d up to the degree of regularity of $\hat{P}_0, \ldots, \hat{P}_k$, and $\alpha \le q-1$,

$$\boldsymbol{\mathcal{T}}^{*\alpha}_m(\hat{P}_0, \ldots, \hat{P}_k)^h_d \subseteq \boldsymbol{\mathcal{T}}_m(\hat{P}_0, \ldots, \hat{P}_{k-1})^h_d + P_k\boldsymbol{\mathcal{T}}^{*(\alpha+1)}_m(\hat{P}_0, \ldots, \hat{P}_k)^h_{d-2}. \tag{6}$$

Indeed, let $(M_0, \ldots, M_{k-1}, 0)$ belong to the left handside. By definition, there exists M_k^α such that $(M_0, \ldots, M_{k-1}, M_k^\alpha P_k^\alpha)$ is in $\boldsymbol{\mathcal{T}}_m(\hat{P}_0, \ldots, \hat{P}_k)_d^h$. Refer to the decomposition 5 of this set. Hence there exists an element T_k of $\boldsymbol{\mathcal{T}}_m(\hat{P}_0, \ldots, \hat{P}_{k-1})_d^h$ (with its last coordinate to zero) and coefficients $\mu_0, \ldots, \mu_{k-1}, \nu_k$ such that $(M_0, \ldots, M_{k-1}, M_k^\alpha P_k^\alpha) = T_k + \mu_0 \Gamma_{0k} + \cdots + \mu_{k-1}\Gamma_{k-1,k} + \nu_k \phi_k$. Coordinate-wise identity writes

$$\begin{cases} (M_0, \ldots, M_{k-1}, 0) = T_k + P_k(\mu_0, \ldots, \mu_{k-1}, 0), \\ -M_k^\alpha P_k^\alpha = \mu_0 P_0 + \cdots + \mu_{k-1} P_{k-1} - \nu_k P_k^{q-1}. \end{cases}$$

The second equation implies that $(\mu_0, \ldots, \mu_{k-1}, M_k^\alpha P_k^{\alpha-1} - \nu_k P_k^{q-2})$ lies in $\boldsymbol{\mathcal{T}}_m(\hat{P}_0, \ldots, \hat{P}_k)_{d-2}^h$, which shows (6). Now by using (6), from 1 to $\alpha \leq q-1$,

$$\boldsymbol{\mathcal{T}}_m^{*1}(\hat{P}_0, \ldots, \hat{P}_k)_d^h \subseteq \boldsymbol{\mathcal{T}}_m(\hat{P}_0, \ldots, \hat{P}_{k-1})_d^h + P_k^\alpha \boldsymbol{\mathcal{T}}_m^{*(\alpha+1)}(\hat{P}_0, \ldots, \hat{P}_k)_{d-2\alpha}^h.$$

The second summand is zero as soon as $d - 2\alpha \leq 1$. As α increases to $q-1$, one either encounters this case or ends up with $P_k^{q-1}\boldsymbol{\mathcal{T}}_m^{*q}(\hat{P}_0, \ldots, \hat{P}_k)_{d-2(q-1)}^h$. But again any $(M_0, \ldots, M_{k-1}, 0)$ of the set in factor writes $T_k + P_k(\mu_0, \ldots, \mu_{k-1}, 0)$. In the product set, the second summand vanishes by $P_k^q = 0$. □

By Property 11, two n-tuples of $\boldsymbol{\mathcal{T}}_m(\hat{P}_0, \ldots, \hat{P}_k)_d^h$ are equivalent modulo $\boldsymbol{\mathcal{T}}_m(\hat{P}_0, \ldots, \hat{P}_{k-1})_d^h$ if and only if they have the same $(k+1)$-th coordinate. Hence, the marginal dimension of the second summand in 5 is the dimension of $(\boldsymbol{\mathcal{R}}_m)_{d-2}^h\{\hat{P}_i\}_{0\leq i\leq k-1} + (\boldsymbol{\mathcal{R}}_m)_{d-2(q-1)}^h \hat{P}_k^{q-1}$. Let $\tau_{k,d} = \dim \boldsymbol{\mathcal{T}}_m(\hat{P}_0, \ldots, \hat{P}_{k-1})_d^h$ and let $\delta_{k+1,d}^{q-1}$ be the dimension of the above. So far, $\tau_{k+1,d} = \tau_{k,d} + \delta_{k+1,d}^{q-1}$. Furthermore, iterating this process, we can show (*cf* full version for a proof)

Lemma 1. *For any $1 \leq \alpha \leq q-1$, let $\delta_{k+1,d}^\alpha$ denote the dimension of $(\boldsymbol{\mathcal{R}}_m)_{d-2}^h\{\hat{P}_i\}_{0\leq i\leq k-1} + (\boldsymbol{\mathcal{R}}_m)_{d-2\alpha}^h \hat{P}_k^\alpha$. For d up to the degree of regularity of $\hat{P}_0, \ldots, \hat{P}_k$, this dimension follows the induction*

$$\delta_{k+1,d}^\alpha = k \dim(\boldsymbol{\mathcal{R}}_m)_{d-2}^h - \tau_{k+1,d-2} + \delta_{k+1,d-2}^{\alpha-1},$$

for any $\alpha \geq 2$, and $\delta_{k+1,d}^1 = (k+1)\dim(\boldsymbol{\mathcal{R}}_m)_{d-2}^h - \tau_{k+1,d-2}$.

Using this lemma we finally find the induction defining $\tau_{k,d}$ for any $k \leq n$ and d up to the degree of regularity of $\hat{P}_0, \ldots, \hat{P}_{n-1}$,

$$\tau_{k+1,d} = \tau_{k,d} + \textstyle\sum_{i=1}^{q-1} \left(k \dim(\boldsymbol{\mathcal{R}}_m)_{d-2i}^h - \tau_{k+1,d-2i}\right) + \dim(\boldsymbol{\mathcal{R}}_m)_{d-2(q-1)}^h. \quad (7)$$

5.4 Numerical Computation of the Upper Bounds

We numerically computed the above induction formulas using a dynamic programming approach. A simple complexity analysis can be found in the full version. Figure 1 below represents the upper bounds on the degree of regularity of HFE systems for many parameters q, n. The corresponding value of D was set

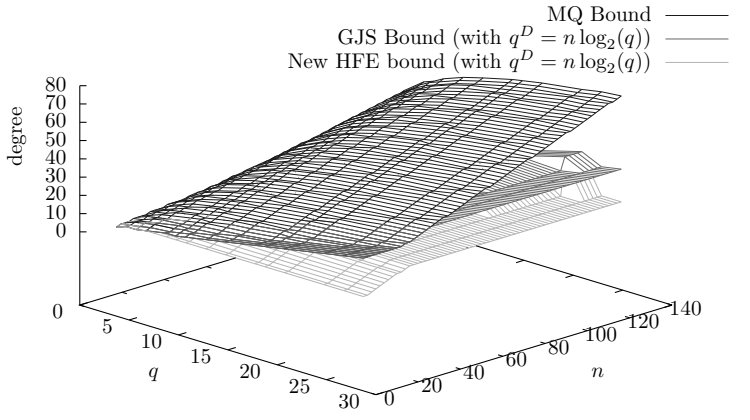


Fig. 1. Overview of the three upper bounds for many HFE parameters: MQ, GJS, HFE

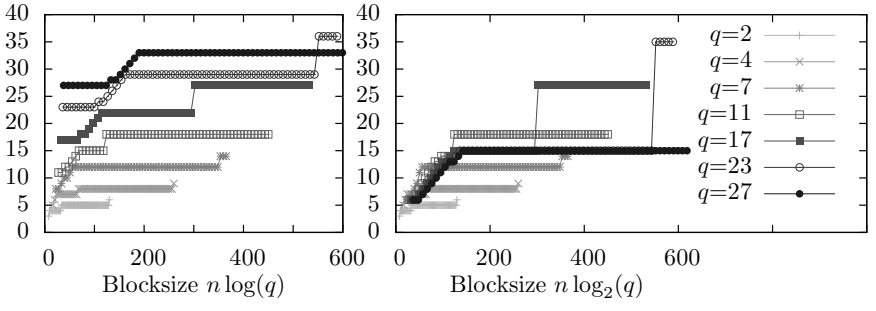


Fig. 2. Comparing the two upper bounds specific to HFE: GJS, HFE bounds

to satisfy $q^D = n \log_2 q$, that is, the degree of the internal HFE polynomial is indexed on the block size. This choice lets schemes operating on the same block size have comparable complexity of the secret operations (roughly $(\log_2 q)^3 n^5$ using the algorithms suggested in [16]). One can note that the surface rendering the GJS bound initially coincides with the MQ surface while our bound ensures a much smaller degree of regularity. Figure 2 below renders (*cf* full version for colorful figures) the improvement of the HFE bound over the GJS bound as q grows. One can perceive the significance of this improvement from the curves being massively pulled down. This is especially true for small blocksizes where the GJS bound is lower bounded by q while the corresponding value of the HFE bound is roughly independent of the value of q.

6 Application to the Security of HFE

The previous discussion has led to the ability to compute an upper bound on the degree of regularity of HFE systems for any parameters. In this section, we describe applications of this parameter to the security of HFE.

6.1 Computing the Degree of Regularity in Practice

We consider a simple algorithm which given the n quadratic polynomials and a prescribed degree $d \geq 2$ computes a generator basis of combinations in degree d of these polynomials (given by monomial multiples of each polynomial) and puts them in row echelon form (any ordering of terms can be used). It is then trivial to obtain the dimension spanned by these combinations. As a consequence, using this algorithm with d incrementing from 2, one can compare the experimental dimension of combinations with the one predicted until the degree of regularity is found. As soon as these dimensions disagree, current d is the degree of regularity of the system. Hence, this simple procedure allows to compute the degree of regularity is practice. Denote by $M_q(n, d)$ the number of monomials of degree d in n variables with exponents modulo q. In degree d, the canonical generator basis has size $M_q(n, d-2)n$. Each such vector has at most $n(n+1)/2$ non-zero coefficients. Computing a row echelon form of these vectors therefore has time complexity about $M_q(n, d-2)^2 n^4$ and space complexity at most $S_q(n, d) = M_q(n, d-2)^2 n^2$. When making d range from 2 to some prescribed d_{max}, the complexity of the iteration is dominated by the complexity at $d = d_{max}$ because $M_q(n, d-2)$ grows exponentially with d. In particular, for HFE(q, n, D) systems, the complexity of computing the degree of regularity is upper bounded by the latter complexity at d set to the HFE bound $\delta(q, n, D)$ computed previously. Since the degree of regularity of random MQ systems is expected very closely tied to the MQ bound (which is much higher for practical parameters), the degree of regularity provides a way to algorithmically distinguish HFE systems from random MQ instances. This distinguisher was already addressed in [4,9,15] and we refer to it as the algebraic distinguisher. Our result makes it possible to compute its complexity for any parameters. Comparing this complexity with the complexity of the differential distinguisher presented in [8], it turns out the latter is almost always far more efficient (*cf* full version of the paper).

6.2 Estimated Upper Bound for Solving HFE Systems

A more critical application uses the heuristic that the degree of regularity originates from the saturation of a subspace of combinations, yielding many degree falls at once. These degree falls in turn contribute to further saturations and further degree falls in smaller degree. When computing a Gröbner basis with a graded ordering, this initiates a process of new head terms appearing with decreasing degree and precipitates the end of the computation. Due to these heuristics, it is commonly taken that the degree of regularity estimates the maximal degree needed in the computation of a Gröbner basis for a graded ordering. In our case, this heuristic is supported by our upper bound on the degree of regularity of HFE closely matching the experimental maximal degree given for $q = 2$ in [12]. As to the complexity of the Gröbner basis computation, it is also commonly estimated as the cost of row echelon form on the combinations matrix at the maximal degree. Although, some algorithms offer improvements to reduce the combinations matrix by removing trivial syzygies [11,13], we keep on

the simple analysis of the precedent paragragh. When a more detailed analysis is available for a particular algorithm our upper bound on the degree of regularity can be readily plugged into it to obtain tighter complexity upper bounds. Figure 3 below represents the obtained upper bound for many HFE parameters, where the degree of the internal parameter is again indexed on the block size by $q^D = n \log_2 q$. Within the limits of the above heuristics, parameters that do not emerge from the 80-bits security level should not be considered secure.

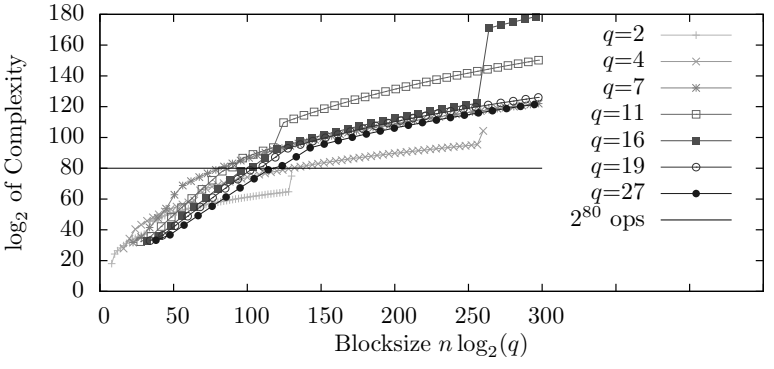


Fig. 3. Estimated Upper Bounds on the Complexity of Algebraic Attacks on HFE

7 Conclusion

In this paper, we provide a rigourous analysis of the degree of regularity of HFE systems. Under commonly used heuristics, this analysis allows to derive estimates for the complexity of algebraic attacks on the public key. In particular, using these estimates, hardly any HFE cryptosystem with block size 80 bits can achieve 80 bits security. HFE over $GF(2)$ with blocksize 128 does not achieve 80 bits security. On the other hand, our work can not be used to infer the security of HFE parameters, because our estimates are only complexity upper bounds and focus on a particular type of attack. Finally, we point out that the first part of our work – shifting the analysis to the internal polynomial – can be used for any cryptosystem following a similar construction to HFE. In particular, it potentially offers a useful framework to the analysis of variations of HFE.

References

1. Bardet, M.: Étude des systémes algébriques surdéterminés. Applications aux codes correcteurs et ála cryptographie. PhD thesis, UniversitéParis 6 (2004)
2. Bardet, M., Faugére, J.-C., Salvy, B.: On the Complexity of Gröbner Basis Computation of Semi-Regular Overdetermined Algebraic Equations. In: ICPSS International Conference on Polynomial System Solving (2004)
3. Buchberger, B.: Ein Algorithmus zum Auffinden der Basiselemente des Restklassnringes nach einem nulldimensionalen Polynomideal. PhD thesis, Innsbruck (1965)

4. Courtois, N.: The Security of Hidden Field Equations (HFE). In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 266–281. Springer, Heidelberg (2001)
5. Courtois, N., Klimov, A., Patarin, J., Shamir, A.: Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 392–407. Springer, Heidelberg (2000)
6. Diem, C.: The xl-algorithm and a conjecture from commutative algebra. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 323–337. Springer, Heidelberg (2004)
7. Ding, J., Schmidt, D., Werner, F.: Algebraic attack on hfe revisited. In: Wu, T.-C., Lei, C.-L., Rijmen, V., Lee, D.-T. (eds.) ISC 2008. LNCS, vol. 5222, pp. 215–227. Springer, Heidelberg (2008)
8. 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)
9. Faugère, J.-C., Joux, A.: Algebraic Cryptanalysis of Hidden Field Equation (HFE) Cryptosystems Using Gröbner Bases. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 44–60. Springer, Heidelberg (2003)
10. Faugére, J.-C.: A New Efficient Algorithm for Computing Gröbner Bases (F4). Journal of Pure and Applied Algebra 139, 61–88 (1999)
11. Faugére, J.-C.: A New Efficient Algorithm for Computing Gröbner Bases without Reductions to Zero $F5$. In: ISSAC, pp. 75–83 (2002)
12. Faugére, J.-C.: Algebraic Cryptanalysis of HFE using Gröbner Bases. Technical Report 4738, INRIA (2003)
13. Kunz-Jacques, S.: Preuves de sécurité et problémes difficiles en cryptologie: étude de cas. PhD thesis, Université Paris 7 (2007)
14. Lazard, D.: Gröbner-Bases, Gaussian Elimination and Resolution of Systems of Algebraic Equations. In: van Hulzen, J.A. (ed.) ISSAC 1983 and EUROCAL 1983. LNCS, vol. 162, pp. 146–156. Springer, Heidelberg (1983)
15. Granboulan, L., Joux, A., Stern, J.: Inverting HFE Is Quasipolynomial. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 345–356. Springer, Heidelberg (2006)
16. 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)
17. Steel, A.: Allan Steel's Groebner Basis Timings Page (2004), magma.maths.usyd.edu.au/users/allan/gb
18. University of Sydney Computational Algebra Group. The MAGMA Computational Algebra System
19. Yang, B.-Y., Chen, J.-M.: All in the xl family: Theory and practice. In: Park, C., Chee, S. (eds.) ICISC 2004. LNCS, vol. 3506, pp. 67–86. Springer, Heidelberg (2005)

Structured Encryption and Controlled Disclosure

Melissa Chase and Seny Kamara

Microsoft Research
{melissac,senyk}@microsoft.com

Abstract. We consider the problem of encrypting structured data (e.g., a web graph or a social network) in such a way that it can be efficiently and privately queried. For this purpose, we introduce the notion of structured encryption which generalizes previous work on symmetric searchable encryption (SSE) to the setting of arbitrarily-structured data.

We present a model for structured encryption, a formal security definition and several efficient constructions. We present schemes for performing queries on two simple types of structured data, specifically lookup queries on matrix-structured data, and search queries on labeled data. We then show how these can be used to construct efficient schemes for encrypting graph data while allowing for efficient neighbor and adjacency queries.

Finally, we consider data that exhibits a more complex structure such as labeled graph data (e.g., web graphs). We show how to encrypt this type of data in order to perform focused subgraph queries, which are used in several web search algorithms. Our construction is based on our labeled data and basic graph encryption schemes and provides insight into how several simpler algorithms can be combined to generate an efficient scheme for more complex queries.

1 Introduction

The most common use of encryption is to provide confidentiality by hiding all useful information about the plaintext. Encryption, however, often renders data useless in the sense that one loses the ability to operate on it. In certain settings this is undesirable and one would prefer encryption schemes that allow for some form of computation over encrypted data.

One example is in the context of remote data storage, or so-called "cloud storage", where a data owner wishes to store *structured* data (e.g., a collection of web pages) on an untrusted server and only retain a constant amount of information locally. To guarantee confidentiality, the owner could encrypt the data before sending it to the server but this approach is unsatisfactory because the data loses its structure and, in turn, the owner loses the ability to query it efficiently.

To address this problem we introduce the notion of *structured encryption*. A structured encryption scheme encrypts structured data in such a way that it can be queried through the use of a query-specific token that can only be generated with knowledge of the secret key. In addition, the query process reveals no useful

M. Abe (Ed.): ASIACRYPT 2010, LNCS 6477, pp. 577–594, 2010.

information about either the query or the data. An important consideration in this context is the efficiency of the query operation on the server side. In fact, in the context of cloud storage, where one often works with massive datasets, even linear time operations can be infeasible.

Roughly speaking, we view structured data as a combination of a data structure δ and a sequence of data items $\mathbf{m} = (m_1, \ldots, m_n)$ such that δ encodes the data's structure and $\mathbf{m}$ represents the actual data. For example, in the case of graph-structured data such as a social network, δ is a graph with n nodes and the ith element of $\mathbf{m}$ is the data associated with node i. To query the data efficiently, one queries δ to recover a set of pointers $I \subseteq [1, n]$ and then retrieves the items in $\mathbf{m}$ indexed by I.

At a high level, a structured encryption scheme takes as input structured data $(\delta, \mathbf{m})$ and outputs an encrypted data structure γ and a sequence of ciphertexts $\mathbf{c} = (c_1, \ldots, c_n)$. Using the private key, a token τ can be constructed for any query such that pointers to the encryptions of $(m_i)_{i \in I}$ can be recovered from γ and τ. Furthermore, given the private key, one can decrypt any ciphertext c_i.

A certain class of symmetric searchable encryption (SSE) schemes [18,11,15] can be viewed as structured encryption schemes for the special purpose of private keyword search over encrypted document collections. Of course, the functionality provided by structured encryption can be achieved using general techniques like oblivious RAMs [20], secure two-party computation [36] and fully-homomorphic encryption (FHE) [17]. In our context, however, we are interested in solutions that are non-interactive and, at worst, linear in the number of data items as opposed to linear in the length of the data. All the schemes described in this work are non-interactive and optimal in that the query time is linear in the number of data items to be returned.

Informally, a basic notion of security for structured encryption guarantees that (1) an encrypted data structure γ and a sequence of ciphertexts $\mathbf{c}$ reveal no partial information about the data $\mathbf{m}$; and that (2) given, in addition, a sequence of tokens $(\tau_1, \ldots, \tau_t)$ for queries $\mathbf{q} = (q_1, \ldots, q_t)$ no information is leaked about either $\mathbf{m}$ or $\mathbf{q}$ beyond what can be inferred from some limited leakage which is a function of δ, $\mathbf{m}$ and $\mathbf{q}$. A stronger notion, introduced in [15], guarantees that (2) holds even when the queries are generated *adaptively*.

All known constructions that can be considered efficient structured encryption schemes (i.e., the index-based SSE schemes [18,11,15]) reveal some limited information about the data items and queries. In particular, for any query they reveal at least (1) the *access pattern*, which consists of the pointers I; and (2) the *query pattern*, which reveals whether two tokens were for the same query[1].

1.1 Applications of Structured Encryption

Private queries on encrypted data. The most immediate application of structured encryption is for performing private queries on encrypted data. In this setting,

[1] While the public-key encryption scheme with keyword search of [7] yields a SSE scheme that hides the access and query patterns, it is interactive.

a client encrypts its (structured) data $(\delta, \mathbf{m})$ resulting in an encrypted data structure γ and a sequence of ciphertexts $\mathbf{c}$. It then sends $(\gamma, \mathbf{c})$ to the server. Whenever the client wishes to query the data, it sends a token τ to the server which the latter uses to recover pointers J to the appropriate ciphertexts. Using a structured encryption scheme in this manner enables the client to store its data remotely while simultaneously guaranteeing confidentiality against the server (in the sense outlined above) and efficient querying and retrieval. While this problem has received considerable attention for the special case of document collections [33,18,5,35,11,1,15,3,31,7], as far as we know, it has never been considered for other kinds of data.

Controlled disclosure for local algorithms. While the original motivation for structured encryption was to perform private queries on encrypted data (or more precisely, private *searches* on encrypted data), we introduce here a new application which we refer to as *controlled disclosure*.

In this setting, the client not only wants to store its data remotely but expects the server (or some third party) to perform some computation over the data. In particular, while the client is willing to reveal the information necessary for the server to perform its task, the client does not want to reveal anything else. Consider, e.g., a client that stores a large-scale social network remotely and that, at some point, needs the server to analyze a small subset of the network. If the social network were encrypted using a classical encryption scheme the client would have to reveal the entire network, leaking extra information to the server. Ideally, what we want in this setting is a mechanism that allows the client to encrypt the data and later disclose the "pieces" of it that are necessary for the server to perform its task.

Another application of controlled disclosure is to the emerging area of (cloud-based) data brokerage services, such as Microsoft's Dallas [14] and Infochimps [23]. Here, the cloud provider acts as a broker between a data provider that wishes to sell access to a massive dataset and a data consumer that needs access to the data. The data is stored "in the cloud" and the cloud operator manages the consumer's access to the provider's data. Using controlled disclosure, the provider could encrypt its data before storing it in the cloud and release tokens to the consumer as appropriate. Such an approach would have several advantages including (1) enabling the producer to get an accurate measure of the consumer's use of the data; and (2) ensuring the producer that the consumer can only access the authorized segments of data, even if the consumer and the cloud operator collude.

Clearly, if the algorithm executed by the server (or the data consumer) is "global", in the sense that it needs to read all the data, then controlled disclosure provides no security. On the other hand, if the algorithm is "local", in that it only needs to read part of the data, then controlled disclosure preserves the confidentiality of the remaining data. There are numerous algorithms that exhibit this kind of local behavior and they are used extensively in practice to solve a variety of problems. For example, many optimization problems like the traveling

salesman problem or vertex cover are handled in practice using local search algorithms (e.g., hill climbing, genetic algorithms or simulated annealing). Several link-analysis algorithms for web search such as Kleinberg's seminal HITS algorithm [26] (and the related SALSA [27] algorithm) are local. Finally, the recent work of Brautbar and Kearns on "jump and crawl" algorithms [10] motivates and proposes several local algorithms for social network analysis, including for finding vertices with high-degree and high clustering coefficient.

Controlled disclosure can be viewed as a compromise between full security on the one hand and efficiency and functionality on the other. In settings where computation needs to be performed on massive datasets and "fully secure" solutions like multi-party computation [36,19,13] and fully-homomorphic encryption [17] are prohibitively expensive, controlled disclosure provides a practical solution without completely compromising security.

1.2 Our Results

Performing private queries on encrypted data is an important goal that is well motivated by the recent trend towards cloud storage. Giving clients the means to encrypt their data without losing the ability to efficiently query and retrieve it provides obvious benefits to the client but also frees the cloud provider from many legal exposures (see [2,24,32] for discussion of these issues). It additionally provides a mechanism by which clients from regulated industries can make use of cloud storage (e.g., to store medical records or financial documents) while remaining compliant.

While the recent work on searchable encryption constitutes an important step towards this goal, we note that a noticeable fraction of the data generated today is *not* text data. Indeed, many large-scale datasets (e.g., image collections, social network data, maps or location information) exhibit a different and sometimes more complex structure that cannot be handled properly using searchable encryption. To address this, we:

1. introduce the notion of structured encryption, which generalizes index-based symmetric searchable encryption [18,11,15] to arbitrarily-structured data and propose a novel application of structured encryption (and therefore of SSE) to the problem of controlled disclosure.
2. extend the adaptive security definition of [15] to the setting of structured encryption,
3. give constructions of adaptively-secure structured encryption schemes for a variety of structures and queries including:
   (a) (lookup queries on matrix-structured data) given a matrix and pair (i, j), return the value stored at row i and column j. This captures, e.g., lookup queries on digital images or retrieval of maps.
   (b) (search queries on labeled data) given a set of labeled items and keyword w, return the items labeled with w. This captures the familiar setting of searchable encryption. We briefly note that our construction exhibits a combination of useful properties that, as far as we know, no previous scheme achieves.

(c) (neighbor queries on graph-structured data) given a graph and a node i, return all the nodes adjacent to i. This captures, e.g., retrieving a user's "friend list" in a social network.
(d) (adjacency queries on graph-structured data) given a graph and two nodes i and j, return 1 if they are adjacent and return 0 otherwise. This captures, e.g., testing whether two users are "friends" in a social network.

While the previous constructions are useful in their own right, an important goal with respect to structured encryption is to construct schemes that are able to encrypt complex structures and to handle expressive queries that take full advantage of the complexity of the data's structure. As an example, consider the case of web graphs (i.e., subgraphs of the Web) which are composed of pages with both text and hyperlinks. Encrypting the pages of a web graph using a searchable encryption scheme will only enable keyword search over the encrypted pages. Web graphs, however, exhibit a much richer structure and we typically want to perform more complex queries on them. Towards this goal, our final contribution is to show how to encrypt web graphs and, more generally, what we refer to as *labeled graph data*. In particular, we:

4. give a structured encryption scheme for labeled graphs that handles *focused subgraph* queries. Roughly speaking, for a given search keywork, a focused subgraph query on a web graph returns a subgraph that encodes enough information about it to yield a good ranking of the pages for that search. These queries are an essential part of Kleinberg's seminal HITS algorithm [26] (and its many successors).

   Our construction uses as building blocks some of the schemes mentioned above. We stress, however, that it is not sufficient to use the schemes "as-is" and we show a novel way of combining structured encryption schemes for simple structures in order to build schemes that handle more complex data and more expressive queries. The approach is general and can be adapted to other complex data types.

2 Related Work

We already mentioned work on oblivious RAMs, secure two-party computation and FHE so we restrict the following discussion to searchable and functional encryption.

Searchable encryption. As mentioned above, structured encryption is a generalization of the notion of a secure index first proposed by Goh [18] for the purpose of building symmetric searchable encryption schemes [33]. In [18], Goh gives a formal security definition for secure indexes and a construction based on Bloom filters. This was followed by [11] and [15], the latter of which gave stronger security definitions and more efficient constructions. Our security definitions for structured encryption in section 4 generalize the ones in [15] to arbitrarily-structured data. Searchable encryption has also been considered in the public-key setting [5,35,1,9,7,8].

Functional encryption. Functional encryption [34] is a recent paradigm that generalizes work on a variety of problems including identity-based encryption [29,6], attribute-based encryption [28,22,4], and predicate encryption [25,30].

Roughly speaking, a structured encryption scheme can be viewed as a functional encryption scheme for which a token can only be used on a single ciphertext. We provide a more detailed comparison between the two approaches in the full version [12].

3 Notation and Preliminaries

Notation. Given a sequence $\mathbf{v}$ of n elements, we refer to its i^{th} element as v_i. If f is a function with domain $\mathcal{U}$ and $S \subseteq \mathcal{U}$, then $f[S]$ refers to the image of S under f. The set of all $\lambda_1 \times \lambda_2$ matrices over a set S is denoted $S^{\lambda_1 \times \lambda_2}$. $\mathcal{G}_n$ and $\boldsymbol{\mathcal{G}}_n$ are the sets of all undirected and directed graphs of size n, respectively. An undirected graph $G = (V, E)$ consists of a set of vertices V and a set of edges $E = \{(i, j)\}$ where $i, j \in V$. We denote by $\deg(i)$ the degree of node i. If G is directed, then the pairs (i, j) are ordered and we refer to i as the tail and to j as the head of the edge. In addition, we denote i's in and out degrees by $\deg^-(i)$ and $\deg^+(i)$, respectively.

Data types. An *abstract data type* is a collection of objects together with a set of operations defined on those objects. For simplicity and visual clarity we define data types as having a single operation but this can be extended to model data types with multiple operations in the natural way. Formally, a data type $\mathscr{T}$ is defined by a universe $\mathcal{U} = \{\mathcal{U}_k\}_{k \in \mathbb{N}}$ and an operation $\mathsf{Query} : \mathcal{U} \times \mathcal{Q} \to \mathcal{R}$, where $\mathcal{Q} = \{\mathcal{Q}_k\}_{k \in \mathbb{N}}$ is the operation's query space and $\mathcal{R} = \{\mathcal{R}_k\}_{k \in \mathbb{N}}$ is its response space. The universe, query and response spaces are ensembles of finite sets indexed by the security parameter k. In this work, we assume the universe is a totally ordered set, and that the response space includes a special element $\perp$ denoting failure.

4 Definitions

In this section we formalize structured encryption schemes and present our main security definition. Before doing so, however, we make explicit two properties of structured encryption which we will make use of throughout this work.

Induced permutation. Unlike previous work on searchable encryption we choose to include the data items (i.e., the documents in the case of searchable encryption) and their encryptions in our definitions. We prefer this approach because explicitly capturing each component of the system can bring to light subtle interactions between them. As an example, consider the correlation between the location of the data items in the sequence $\mathbf{m}$ and the locations of their corresponding ciphertexts in $\mathbf{c}$. More precisely, let π be the permutation over $[n]$ such

that for all $i \in [n]$, $m_i := \mathsf{Dec}_K(c_{\pi(i)})$. We refer to π as the permutation *induced* by $\mathbf{m}$ and $\mathbf{c}$.

The reason most SSE constructions (with the exception of oblvious RAMs) leak the access pattern is because π is the identity function. This means that in order to (efficiently) retrieve items $\{m_i : i \in I\}$ the server must know I. Our constructions hide part of the access pattern essentially because they break this correlation by inducing a (pseudo-)random permutation between $\mathbf{m}$ and $\mathbf{c}$.

Associativity. We also make explicit a property possessed by some constructions (e.g., the non-adaptively secure SSE construction of [15]) that we refer to as *associativity*. Intuitively, a scheme is associative if one can associate an item v_i with data item m_i in such a way that a query operation returns, in addition to the pointers J, the strings $(v_i)_{i \in I}$. We capture this by re-defining the message space of our encryption algorithms to take, in addition to a data structure δ, a sequence $\mathbf{M} = ((m_1, v_1), \ldots, (m_n, v_n))$ of pairs that consist of a private data item m_i and a semi-private[2] item v_i . We sometimes refer to the sequences $(m_1, \ldots, m_n)$ and $(v_1, \ldots, v_n)$ as $\mathbf{m}$ and $\mathbf{v}$, respectively.

Associativity is useful for several reasons. The most direct application is to provide the client the ability to associate some meta-data with the ciphertexts that may be useful to the server (e.g., file name or size). In situations where the client wishes to grant the server access to the data, the semi-private items could even be decryption keys for the associated ciphertexts. As we will see in Section 6, however, associativity can also be used to "chain" structured encryption schemes together in order to construct complex schemes from for simpler ones.

Definition 1 (Private-key structured encryption). *Let $\mathcal{T}$ be an abstract data type supporting operation $\mathsf{Query} : \mathcal{U} \times \mathcal{Q} \to \mathcal{R}$ where $\mathcal{R} = [n]$ for $n \in \mathbb{N}$. An associative private-key structured encryption scheme for $\mathcal{T}$ is a tuple of five polynomial-time algorithms $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Token}, \mathsf{Query}_\mathsf{e}, \mathsf{Dec})$ such that:*

$K \leftarrow \mathsf{Gen}(1^k)$: is a probabilistic algorithm that takes as input a security parameter k and outputs a private key K.
$(\gamma, \mathbf{c}) \leftarrow \mathsf{Enc}(K, \delta, \mathbf{M})$: is a probabilistic algorithm that takes as input a private key K, a data structure δ of type $\mathcal{T}$, and a sequences of private and semi-private data $\mathbf{M}$. It outputs an encrypted data structure γ and a sequence of ciphertexts $\mathbf{c}$. We sometimes write this as $(\gamma, \mathbf{c}) \leftarrow \mathsf{Enc}_K(\delta, \mathbf{M})$.
$\tau \leftarrow \mathsf{Token}(K, q)$: is a (possibly probabilistic) algorithm that takes as input a private key K and a query $q \in \mathcal{Q}$ and outputs a token τ. We sometimes write this as $\tau \leftarrow \mathsf{Token}_K(q)$.
$(J, \mathbf{v}_I) := \mathsf{Query}_\mathsf{e}(\gamma, \tau)$: is a deterministic algorithm that takes as input an encrypted data structure γ and a token τ. It outputs a set of pointers $J \subseteq [n]$ and a sequence of semi-private data $\mathbf{v}_I = (v_i)_{i \in I}$, where $I = \pi^{-1}[J]$.
$m_j := \mathsf{Dec}(K, c_j)$: is a deterministic algorithm that takes as input a secret key K and a ciphertext c_j and outputs a message m_j.

[2] We refer to the items $(v_1, \ldots, v_n)$ as semi-private since, unlike $(m_1, \ldots, m_n)$, they can be recovered given an appropriate token.

We say that Π is correct if for all $k \in \mathbb{N}$, for all K output by $\mathsf{Gen}(1^k)$, for all $\delta \in \mathcal{U}_k$, for all $\mathbf{M}$, for all $(\gamma, \mathbf{c})$ output by $\mathsf{Enc}(K, \delta, \mathbf{M})$, for all $q \in \mathcal{Q}_k$, for all τ output by $\mathsf{Token}(K, q)$, for $(J, \mathbf{v}_I)$ output by $\mathsf{Query}_{\mathsf{e}}(\gamma, \tau)$,

$$J = \pi\left[\mathsf{Query}(\delta, q)\right] \bigwedge \mathsf{Dec}_K(c_j) = m_j \text{ for all } j \in [n],$$

where π is the permutation induced by $\mathbf{m}$ and $\mathbf{c}$.

The intuitive security guarantee we seek is that (1) given an encrypted data structure γ and a sequence of ciphertexts $\mathbf{c}$, no adversary can learn any partial information about $\mathbf{m}$; and that (2) given, in addition, a sequence of tokens $\tau = (\tau_1, \ldots, \tau_t)$ for an adaptively generated sequence of queries $\mathbf{q} = (q_1, \ldots, q_t)$, no adversary can learn any partial information about either $\mathbf{m}$ or $\mathbf{q}$ beyond what is revealed by the semi-private data $(\mathbf{v}_{I_1}, \ldots, \mathbf{v}_{I_t})$.

This exact intuition can be difficult to achieve and in some settings is unnecessarily strong. Consider, e.g., the fact that the number of data items n is immediately revealed to the adversary since it receives the ciphertexts $\mathbf{c}$. Another example is in the setting of SSE where, as discussed earlier, all known efficient and non-interactive schemes [18,11,15] reveal the access and query patterns. We would therefore like to weaken the definition appropriately by allowing some limited information about the messages and the queries to be revealed. On the other hand, it is not clear that such leakage is always necessary in order to achieve efficiency (e.g., the number of data items can be easily hidden by padding) so we prefer not to "hardcode" this leakage in our definition. To formalize this we parameterize the definition with two leakage functions $\mathcal{L}_1$ and $\mathcal{L}_2$ that capture precisely what is being leaked by the ciphertext and the tokens.

We now present our security definition for adaptive adversaries which is a generalization of the definition of [16]. Intuitively, we require that the view of an adversary (i.e., the encrypted data structure, the sequence of ciphertexts, and the sequence of tokens) generated from any adaptive query strategy be simulatable given the leakage information and the semi-private data.

Definition 2 (CQA2-security). *Let $\Sigma = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Token}, \mathsf{Query}_{\mathsf{e}}, \mathsf{Dec})$ be an associative private-key structured encryption scheme for data of type $\mathcal{T}$ supporting operation $\mathsf{Query} : \mathcal{U} \times \mathcal{Q} \to [n]$, for some $n \in \mathbb{N}$, and consider the following probabilistic experiments where $\mathcal{A}$ is an adversary, $\mathcal{S}$ is a simulator and $\mathcal{L}_1$ and $\mathcal{L}_2$ are (stateful) leakage algorithms:*

Real$_{\Sigma,\mathcal{A}}(k)$*: the challenger begins by running $\mathsf{Gen}(1^k)$ to generate a key K. $\mathcal{A}$ outputs a pair $(\delta, \mathbf{M})$ and receives $(\gamma, \mathbf{c}) \leftarrow \mathsf{Enc}_K(\delta, \mathbf{M})$ from the challenger. The adversary makes a polynomial number of adaptive queries and, for each query q, receives a token $\tau \leftarrow \mathsf{Token}_K(q)$ from the challenger. Finally, $\mathcal{A}$ returns a bit b that is output by the experiment.*

Ideal$_{\Sigma,\mathcal{A},\mathcal{S}}(k)$*: $\mathcal{A}$ outputs a tuple $(\delta, \mathbf{M})$. Given $\mathcal{L}_1(\delta, \mathbf{M})$, $\mathcal{S}$ generates and sends a pair $(\gamma, \mathbf{c})$ to $\mathcal{A}$. The adversary makes a polynomial number of adaptive queries and for each query q the simulator is given $(\mathcal{L}_2(\delta, q), \mathbf{v}_I)$, where $I := \mathsf{Query}(\delta, q)$. The simulator returns a token τ. Finally, $\mathcal{A}$ returns a bit b that is output by the experiment.*

We say that Σ is $(\mathcal{L}_1, \mathcal{L}_2)$-secure against adaptive chosen-query attacks if for all PPT *adversaries $\mathcal{A}$, there exists a* PPT *simulator $\mathcal{S}$ such that*

$$|\Pr[\mathbf{Real}_{\Sigma,\mathcal{A}}(k) = 1] - \Pr[\mathbf{Ideal}_{\Sigma,\mathcal{A},\mathcal{S}}(k) = 1]| \leq \mathsf{negl}(k).$$

As previously discussed, the $\mathcal{L}_2$ leakage of our constructions mainly consists of the query and intersection patterns. Intuitively, the query pattern reveals when a query is repeated while the intersection pattern reveals when the same items are accessed. The intersection pattern reveals when the same items are accessed but not *which* items are accessed (i.e., their locations in $\mathbf{m}$). The latter is hidden in our definition below by applying a random permutation to the item's locations in $\mathbf{m}$.

Definition 3 (Query and intersection patterns). *Let $\mathbf{q}$ be a non-empty sequence of queries. For any $q_t \in \mathbf{q}$, the query pattern* $\text{QP}(q_t)$ *is a binary vector of length t with a 1 at location i if $q_t = q_i$ and a 0 otherwise. The intersection pattern* $\text{IP}(q_t)$ *is a sequence of length t with $f[I]$ at location t, where f is a fixed random permutation over $[n]$ and $I := \mathsf{Query}(\delta, q_t)$.*

5 Structured Encryption for Basic Structures

In this Section we present constructions of structured encryption schemes for data with simple structures. In Section 6 we will use some of these as building blocks to design schemes for data that exhibits a more complex structure. We stress, however, that the constructions presented here are of independent interest.

5.1 Lookup Queries on Matrices

We describe a structured encryption scheme for matrix-structured data which consists of an $\lambda_1 \times \lambda_2$ matrix M of pointers into a sequence of n data items $\mathbf{m}$. Here, the matrix type has universe $\mathcal{U} = [n]^{\lambda_1 \times \lambda_2}$ and supports the lookup operation $\mathsf{Lkp} : [n]^{\lambda_1 \times \lambda_2} \times [\lambda_1] \times [\lambda_2] \to [n]$ that takes as input a matrix M and a pair (α, β) and returns $M[\alpha, \beta]$.

Matrix-structured data is ubiquitous and includes any kind of two-dimensional data. Consider, e.g., the case of digital images which can be viewed as a pair $(M, \mathbf{m})$, where M is a matrix such that the cell at location (α, β) points to some m_i that encodes the color of the pixel at location (α, β) in the image.

Our construction, described in Figure 1 below, is associative. At a high level, encryption is done by (1) padding the data items to be of the same length; (2) randomly permuting the location of the data items, (3) randomly permuting the location of the matrix cells using a PRP; and (4) encrypting the contents of the cells (and the semi-private data) using the output of a PRF. The purpose of the last two steps are immediate. Steps (1) and (2) are what allow us hide part of the access pattern by inducing a pseudo-random permutation between $\mathbf{m}$ and $\mathbf{c}$.

Lookup queries are handled by sending the permuted location of a cell (which can be recovered by the client since it stores the key to the PRP) and the output

Let $F : \{0,1\}^k \times \{0,1\}^* \rightarrow \{0,1\}^*$ be a pseudo-random function, $P : \{0,1\}^k \times [\lambda_1] \times [\lambda_2] \rightarrow [\lambda_1] \times [\lambda_2]$ be pseudo-random permutation and $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a private-key encryption scheme. Our encryption scheme $\mathsf{Matrix} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Token}, \mathsf{Lkp_e}, \mathsf{Dec})$ is defined as follows:

- $\mathsf{Gen}(1^k)$: generate two random k-bit strings K_1, K_2 and a key $K_3 \leftarrow \Pi.\mathsf{Gen}(1^k)$. Set $K := (K_1, K_2, K_3)$.
- $\mathsf{Enc}(K, M, \mathbf{M})$: construct a $\lambda_1 \times \lambda_2$ matrix C as follows:
  1. parse $\mathbf{M}$ as $\mathbf{m}$ and $\mathbf{v}$
  2. choose a pseudo-random permutation $G : \{0,1\}^k \times [n] \rightarrow [n]$
  3. sample a k-bit string K_4 uniformly at random
  4. for all $(\alpha, \beta) \in [\lambda_1] \times [\lambda_2]$,
     store $\langle G_{K_4}(i), v_i \rangle \oplus F_{K_1}(\alpha, \beta)$ where $i := M[\alpha, \beta]$, at location $(\alpha', \beta') := P_{K_2}(\alpha, \beta)$ in C.
     If $M[\alpha, \beta] = \perp$, then $\langle G_{K_4}(i), v_i \rangle$ above is replaced with a random string of appropriate length.

  Let $\mathbf{m}^*$ be the sequence that results from padding the elements of $\mathbf{m}$ so that they are of the same length and permuting them according to G_{K_4}. For $1 \leq j \leq n$, let $c_j \leftarrow \Pi.\mathsf{Enc}_{K_3}(m_j^*)$. Output $\gamma := C$ and $\mathbf{c} = (c_1, \ldots, c_n)$.
- $\mathsf{Token}(K, \alpha, \beta)$: output $\tau := (s, \alpha', \beta')$, where $s := F_{K_1}(\alpha, \beta)$ and $(\alpha', \beta') := P_{K_2}(\alpha, \beta)$.
- $\mathsf{Lkp_e}(\gamma, t)$: parse τ as (s, α', β'); compute and output $(j, v) := s \oplus C[\alpha', \beta']$.
- $\mathsf{Dec}(K, c_j)$: return $m_j := \Pi.\mathsf{Dec}_{K_3}(c_j)$.

Fig. 1. An associative structured encryption scheme for matrices

of the PRF used to encrypt the contents (which can also be recovered since the client stores the key to the PRF).

In Theorem 1 below we show that the construction above is secure against adaptive chosen-query attacks.

Theorem 1. *If F, P and G are pseudo-random and if Π is CPA-secure then* Matrix *is $(\mathcal{L}_1, \mathcal{L}_2)$-secure against adaptive chosen-query attacks, where $\mathcal{L}_1(M, \mathbf{M}) = (\lambda_1, \lambda_2, n, \ell)$ and $\mathcal{L}_2(M, \alpha, \beta) = (\text{QP}(\alpha, \beta), \text{IP}(\alpha, \beta))$.*

The proof is omitted due to lack of space but appears in [12].

5.2 Search Queries on Labeled Data

We now present a structured encryption scheme for labeled data which consists of a "labeling" L and a sequence of data items $\mathbf{m}$. Informally, a labeling just associates a set of keywords to each data item. More formally, the labeling data type has as universe $\mathcal{U}$ the set of all binary relations between $[n]$ and W, where W is a set of keywords. In addition, it supports the operation $\mathsf{Search} : \mathcal{U} \times W \rightarrow 2^{[n]}$ that takes as input a labeling and a keyword w and returns the set $L(w) = \{i \in [n] : (i, w) \in L\}$.

Let $F : \{0,1\}^k \times W \to \{0,1\}^*$ and $P : \{0,1\}^k \times W \to \{0,1\}^k$ be pseudo-random functions and $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a private-key encryption scheme. Our scheme $\mathsf{Label} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Token}, \mathsf{Search}_\mathsf{e}, \mathsf{Dec})$ is defined as follows:

- $\mathsf{Gen}(1^k)$: sample two random k-bit keys K_1, K_2, and generate a key $K_3 \leftarrow \Pi.\mathsf{Gen}(1^k)$. Set $K := (K_1, K_2, K_3)$.
- $\mathsf{Enc}(K, L, \mathbf{M})$: construct a dictionary T as follows:
  1. parse $\mathbf{M}$ as $\mathbf{m}$ and $\mathbf{v}$.
  2. choose a pseudo-random permutation $G : \{0,1\}^k \times [n] \to [n]$
  3. sample a k-bit string K_4 uniformly at random
  4. for each $w \in W$ such that $L(w) \neq \emptyset$, let $\kappa_w := P_{K_2}(w)$ and store $\langle (G_{K_4}(i), v_i)_{i \in L(w)} \rangle \oplus F_{K_1}(w)$ in T with search key κ_w. Use padding to ensure that the strings $\langle (G_{K_4}(i), v_i)_{i \in L(w)} \rangle$ are all of the same length.

  Let $\mathbf{m}^*$ be the sequence that results from padding the elements of $\mathbf{m}$ so that they are of the same length and permuting them according to G_{K_4}. For $1 \leq j \leq n$, let $c_j \leftarrow \Pi.\mathsf{Enc}_{K_3}(m_j^*)$. Output $\gamma := T$ and $\mathbf{c} = (c_1, \ldots, c_n)$.
- $\mathsf{Token}(K, w)$: output $\tau := (F_{K_1}(w), P_{K_2}(w))$.
- $\mathsf{Search}_\mathsf{e}(\gamma, \tau)$: parse τ as (α, β) and compute $s := T(\beta) \oplus \alpha$, where $T(\beta)$ refers to the value stored in T with search key β. If β is not in T then output $J = \emptyset$ and $v_I = \perp$. Otherwise parse s as $\langle (j_1, v_{i_1}), \ldots, (j_t, v_{i_t}) \rangle$ and output $J = (j_1, \ldots, j_t)$ and $\mathbf{v}_I = (v_{i_1}, \ldots, v_{i_t})$.
- $\mathsf{Dec}(K, c_j)$: output $m_j := \Pi.\mathsf{Dec}_{K_3}(c_j)$.

Fig. 2. An associative structured encryption scheme for labeled data

Our construction, described in Figure 2, is efficient, associative and adaptively secure and, as far as we know, is the first scheme to achieve all three properties. It is based on the first scheme of [15] (SSE-1) which is efficient and associative but not adaptively secure[3]. The second scheme of [15], on the other hand, is adaptively secure but is inefficient and not associative.

Our construction makes use of a dictionary which is a data structure that stores pairs (a, b) such that given a, the corresponding value b can be recovered efficiently. We refer to a as the "search key" and to b as the value. Dictionaries can be implemented in a variety of ways, including using search trees or hash tables. Intuitively, encryption proceeds as follows in our scheme. As in our previous construction, we pad and permute the data items with a PRP G. For each keyword w an array is constructed where each cell stores (1) a pointer j from the set $L^*(w) = G_K[L(w)]$ and (2) the corresponding semi-private item v_i. The array is then padded up to a standard length, and encrypted using the output of a PRF and is stored in a dictionary using as search key the output of another PRF on the keyword. Search queries are handled by sending the search key

[3] While our scheme achieves the same efficiency as SSE-1 with respect to search time, SSE-1 is more efficient with respect to storage.

(which can be recovered by the client using the key to the second PRF) and the output of the PRF used to encrypt the array (which can be recovered using the key to the first PRF). The efficiency of our search operation depends on how the underlying dictionary is implemented but in this context any solution based on hash tables is appropriate and will give search time that is $O(|I|)$, which is optimal.

Theorem 2. *If F, P and G are pseudo-random and if Π is CPA-secure then* Label *is $(\mathcal{L}_1, \mathcal{L}_2)$-secure against adaptive chosen-query attacks, where $\mathcal{L}_1(L, \mathbf{M}) = (|W|, n, \ell)$ and $\mathcal{L}_2(L, w) = (|I|, \text{QP}(w), \text{IP}(w))$.*

The proof is omitted due to lack of space but appears in [12].

5.3 Neighbor Queries on Graphs

We now consider encryption of graph-structured data and, in particular, of graphs that support neighbor queries. Formally, the graph type we consider has universe $\mathcal{U} = \mathcal{G}_n$ and supports the neighbor operation $\mathsf{Neigh} : \mathcal{G}_n \times [n] \to 2^{[n]}$ that takes as input an undirected graph G with n nodes and a node i and returns the nodes adjacent to i.

Our approach here is to encode the graph as a labeling and to apply a structured encryption scheme for labeled data (such as the one described in the previous Section). Given some graph-structured data $(G, \mathbf{m})$, where $G = (V, E)$, we construct the labeled data $(L, \mathbf{m})$ such that L assigns to each data item m_i a label set corresponding to the set of nodes adjacent to the ith node. Neighbor queries are handled by sending a token for "keyword" $i \in V$ which allows the server to recover pointers to all the data items associated with i by the labeling. Our construction is described in detail in Figure 3 below.

Let $\mathsf{Label} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Token}, \mathsf{Search}_\mathsf{e}, \mathsf{Dec})$ be an associative structured encryption scheme for labeled data. Our scheme $\mathsf{Graph} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Token}, \mathsf{Neigh}_\mathsf{e}, \mathsf{Dec})$ is defined as follows:

- $\mathsf{Gen}(1^k)$: generate and output $K \leftarrow \mathsf{Label.Gen}(1^k)$.
- $\mathsf{Enc}(K, G, \mathbf{M})$: parse $\mathbf{M}$ as $\mathbf{m}$ and $\mathbf{v}$ and construct a labeling L that associates to each m_i the set $\{j \in [n] : (i, j) \in E\}$, where E is the set of edges in G. Output $(\gamma, \mathbf{c}) \leftarrow \mathsf{Label.Enc}_K(L, \mathbf{M})$.
- $\mathsf{Token}(K, i)$: compute and output $\tau \leftarrow \mathsf{Label.Token}_K(i)$.
- $\mathsf{Neigh}_\mathsf{e}(\gamma, \tau)$: output $J := \mathsf{Label.Search}(\gamma, \tau)$.
- $\mathsf{Dec}(K, c_j)$: output $m_j := \mathsf{Label.Dec}_K(c_j)$.

Fig. 3. A structured encryption scheme for graphs supporting neighbor queries

Theorem 3. *If* Label *is $(\mathcal{L}_1, \mathcal{L}_2)$-secure against adaptive chosen-query attacks, then* Graph *is $(\mathcal{L}_1, \mathcal{L}_2)$-secure against adaptive chosen-query attacks as well.*

The theorem follows by construction. Note that if Label is instantiated with the scheme from Section 5.2, then $\mathcal{L}_1$ leaks the size of the graph, the number of data items and the length of the largest data item while $\mathcal{L}_2$ leaks the degree of the node and the query and intersection patterns.

We now discuss a slight variation of this construction to handle incoming and outgoing neighbor queries on directed graphs. This will be useful as a building block for the construction we describe in Section 6. An incoming neighbor query is: given a node i return all the nodes that point to it; and an outgoing neighbor query is: given a node i return all the nodes that it points to. We stress that the changes we describe do not affect security in any way.

Consider the scheme $\mathsf{Graph}^+ = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Token}, \mathsf{Neigh_e}, \mathsf{Dec})$ defined exactly as Graph except that the Enc algorithm constructs the labeling in the following manner: instead of associating a data item m_i to the set of nodes adjacent to node i, associate m_i to the nodes that are pointed to by node i. Similarly, a scheme Graph^- can be constructed by associating to data item m_i the set of nodes that point to node i.

5.4 Adjacency Queries on Graphs

In this Section we give a simple scheme to encrypt graphs supporting adjacency queries based on any matrix encryption scheme. The approach is straightforward and, at a high level, consists of encrypting the graph's adjacency matrix. Given data $(G, \mathbf{m})$, where $G = (V, E)$ is a directed graph of size n and each data item m_i is assigned to some edge in E, encryption proceeds as follows. We create a matrix M that holds at location (α, β) a pointer to the data item associated with edge $(\alpha, \beta) \in V$ (or $\perp$ when there is no such edge). We then use the matrix encryption scheme on $(M, \mathbf{m})$. Our construction is described in detail in Figure 4.

Let $\mathsf{Matrix} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Token}, \mathsf{Lkp_e}, \mathsf{Dec})$ be an associative encryption scheme for matrix-structured data. Our scheme $\mathsf{Graph} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Token}, \mathsf{Adj_e}, \mathsf{Dec})$ is defined as follows:

- $\mathsf{Gen}(1^k)$: generate and output $K \leftarrow \mathsf{Matrix.Gen}(1^k)$.
- $\mathsf{Enc}(K, G, \mathbf{M})$: construct a matrix M as follows: if $(\alpha, \beta) \in V$, then $M[\alpha, \beta]$ stores a pointer to the item assigned to edge (α, β); if $(\alpha, \beta) \notin V$ then $M[\alpha, \beta] = \perp$. Output $(\gamma, \mathbf{c}) \leftarrow \mathsf{Matrix.Enc}_K(M, \mathbf{M})$.
- $\mathsf{Token}(K, i, j)$: compute and output $\tau \leftarrow \mathsf{Matrix.Token}_K(i, j)$.
- $\mathsf{Adj_e}(\gamma, \tau)$: output $J := \mathsf{Matrix.Lkp_e}(\gamma, \tau)$.
- $\mathsf{Dec}(K, c_j)$: output $m_j := \mathsf{Matrix.Dec}_K(c_j)$.

Fig. 4. A structured encryption scheme for graphs supporting adjacency queries

Theorem 4. *If Matrix is $(\mathcal{L}_1, \mathcal{L}_2)$-secure against adaptive chosen-query attacks, then so is Graph.*

Again, the theorem follows by construction. If Matrix is instantiated with the construction from Section 5.1, then $\mathcal{L}_1$ leaks the size of the graph, the number of edges[4] the number of data items and the length of the largest data item. $\mathcal{L}_2$ leaks the query and intersection patterns.

6 Structured Encryption for Labeled Graphs

In this Section we describe an adaptively secure structured encryption scheme for data that is both labeled and associated with a graph-structure. As an example, consider a web graph where each page is labeled with a set of keywords (which could be the set of all the words in the page) and points to a set of other pages. Another example is social network data which consists of user profiles (with some associated meta-data) that link to other users.

While the constructions from the previous Section can be used to encrypt this type of data, the queries they support (i.e., keyword search, adjacency, and neighbor queries) are limited in this setting since they are only relevant to part of the data's structure. Indeed, if we were to encrypt a web graph using a scheme for labeled data, then we could only perform keyword search. Similarly, if we were to use a graph encryption scheme that supports only neighbor queries then we could only retrieve pages that are linked from a particular page. But web graphs, and labeled graph data in general, exhibit a much richer structure and ideally we would like to design schemes that support more complex queries that take advantage of this structure.

Focused subgraph queries. One example of complex queries on web graphs are focused subgraph queries. These queries are an essential part of a certain class of search engine algorithms which includes Kleinberg's seminal HITS algorithm [26] and the SALSA algorithm [27]. At a high level, they work as follows. Given a keyword w a keyword search is performed over the web pages. This results in a subset of pages called the *root* graph. A focused subgraph is then constructed by adding all the pages that either link to pages in the root graph or are linked from pages in the root graph. An iterative algorithm is then applied to the focused subgraph which returns, for each page, a score that quantifies its relevance with respect to keyword w. The key property of these "link-analysis" algorithms (and the reason for their success) is that they take advantage not only of the information provided by the keywords associated with the pages, but also of the implicit information embedded in the graph structure (i.e., the links) of the web graph.

Our approach. At a high level, our approach is to decompose the complex structure into simpler structures (e.g., in the case of a web graph into its graph and its labeling) and then use different structured encryption schemes to handle each "sub-structure". We note, however, that the sub-structures cannot be handled

[4] The number of edges can be hidden by padding $\mathbf{m}$ with $n^2 - |E|$ random strings whose lengths are distributed similarly to real data items.

in isolation. In particular, for this approach to work the individual schemes have to be combined in a particular way. This is where we make essential use of associativity, which will allow us to "chain" the schemes together in order to obtain the functionality we want (this technique will be illustrated in our discussion below).

Our construction. We now illustrate our second approach for the case of web graphs but note that our construction applies to any labeled graph data. A detailed description of our construction is given in Figure 5. We note that it is not associative. A web graph will be viewed as a tuple $(G, L, \mathbf{m})$, which consists of a directed graph $G \in \boldsymbol{\mathcal{G}}_n$ of size n, a labeling L over a keyword space W, and text pages $\mathbf{m}$. The graph G encodes the link structure of the web graph and the labeling assigns keywords to each page[5]. The focused subgraph operation $\mathsf{Subgraph} : \boldsymbol{\mathcal{G}}_n \times W \rightarrow \mathcal{G}_{\leq n}$ takes as input a directed graph G of size n and a keyword w and returns the subgraph $G(w)$ that consists of (1) the nodes i in $L(w)$; (2) any node that links to the nodes in $L(w)$; and (3) any node that is linked from the nodes in $L(w)$.

Our construction makes use of three structured encryption schemes: Label that supports search over labeled data, Graph^- that supports incoming neighbor queries over graph-structured data, and Graph^+ that supports outgoing neighbor queries over graph-structured data. We stress that Label must be associative. Given a web graph $(G, L, \mathbf{m})$ we encrypt $(G, \mathbf{m})$ using both Graph^+ and Graph^-, resulting in ciphertexts $\mathbf{c}^+$ and $\mathbf{c}^-$. Now, for each node i in G, we generate a pair of tokens (τ_i^+, τ_i^-). We then use Label to encrypt $(L, \mathbf{m})$ using the token pairs (τ_i^+, τ_i^-) as semi-private data (recall that Label is associative). We then output the encryption $\mathbf{c}^{\mathsf{l}}$ of $(L, \mathbf{m})$.

A focused subgraph query on keyword w is handled as follows. A token $\tau^{\mathsf{l}} \leftarrow \mathsf{Label.Token}_K(w)$ is generated and sent to the server. When used with the ciphertext $\mathbf{c}^{\mathsf{l}}$, this token will reveal to the server (1) pointers to all the (encrypted) web pages labeled with keyword w; and (2) for each of these encrypted pages c_j, the semi-private information which consists of tokens (τ_j^+, τ_j^-). For each encrypted page, the server can then use the token pairs with ciphertexts $\mathbf{c}_j^+$ and $\mathbf{c}_j^-$ to recover pointers to any incoming and outgoing neighbors of page c_j.

Theorem 5. *If* Label, Graph^+ *and* Graph^- *are respectively (stateless)* $(\mathcal{L}_1^l, \mathcal{L}_2^l)$*-secure,* $(\mathcal{L}_1^+, \mathcal{L}_2^+)$*-secure and* $(\mathcal{L}_1^-, \mathcal{L}_2^-)$*-secure against adaptive chosen query attacks, then the scheme described above is* $(\mathcal{L}_1, \mathcal{L}_2)$*-secure against adaptive chosen-query attacks, where*

$$\mathcal{L}_1(G, L, \mathbf{m}) = (\mathcal{L}_1^l(L, \mathbf{m}), \mathcal{L}_1^+(G, \mathbf{m}), \mathcal{L}_2^-(G, \mathbf{m}))$$

and

$$\mathcal{L}_2(G, L, w) = \left(\mathcal{L}_2^l(L, w), \left(\mathcal{L}_2^+(G, i)\right)_{i \in |R(w)|}, \left(\mathcal{L}_2^-(G, i)\right)_{i \in |R(w)|}\right).$$

The proof is omitted due to lack of space but appears in [12].

[5] If we wish to perform full-text search then the labeling can simply assign a page to all of its words.

Let $\mathsf{Label} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Token}, \mathsf{Search}_\mathsf{e}, \mathsf{Dec})$ be an encryption scheme for labeled data, $\mathsf{Graph}^+ = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Token}, \mathsf{Neigh}_\mathsf{e}, \mathsf{Dec})$ and $\mathsf{Graph}^- = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Token}, \mathsf{Neigh}_\mathsf{e}, \mathsf{Dec})$ be graph encryption schemes that support neighbor queries. Our scheme $\mathsf{LabGraph} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Token}, \mathsf{Subgraph}_\mathsf{e}, \mathsf{Dec})$ is defined as follows:

- $\mathsf{Gen}(1^k)$: generate three keys $K_1 \leftarrow \mathsf{Graph}^+.\mathsf{Gen}(1^k)$, $K_2 \leftarrow \mathsf{Graph}^-.\mathsf{Gen}(1^k)$ and $K_3 \leftarrow \mathsf{Label}.\mathsf{Gen}(1^k)$. Let $K = (K_1, K_2, K_3)$.
- $\mathsf{Enc}(K, G, \mathbf{m})$:
  1. compute $(\gamma^+, \mathbf{c}^+) \leftarrow \mathsf{Graph}^+.\mathsf{Enc}_{K_1}(G, \mathbf{m})$,
  2. compute $(\gamma^-, \mathbf{c}^-) \leftarrow \mathsf{Graph}^-.\mathsf{Enc}_{K_2}(G, \mathbf{m})$,
  3. for $1 \leq i \leq n$,
     (a) compute $\tau_i^+ \leftarrow \mathsf{Graph}^+.\mathsf{Token}_{K_1}(i)$,
     (b) compute $\tau_i^- \leftarrow \mathsf{Graph}^-.\mathsf{Token}_{K_2}(i)$,
  4. let L be the labeling generated from all the words in $\mathbf{m}$ (i.e., each m_i is labeled with the words it contains) and let $\mathbf{v} = \{(t_i^+, t_i^-)_i\}$,
  5. compute $(\gamma^\mathsf{l}, \mathbf{c}^\mathsf{l}) \leftarrow \mathsf{Label}.\mathsf{Enc}_{K_3}(L, \mathbf{M})$, where $\mathbf{M}$ is composed of $\mathbf{m}$ and $\mathbf{v}$,
  6. output $\gamma = (\gamma^+, \gamma^-, \gamma^\mathsf{l})$ and $\mathbf{c} = (\mathbf{c}^+, \mathbf{c}^-, \mathbf{c}^\mathsf{l})$.
- $\mathsf{Token}(K, w)$: output $\tau \leftarrow \mathsf{Label}.\mathsf{Token}_{K_3}(w)$.
- $\mathsf{Subgraph}_\mathsf{e}(\gamma, \tau)$:
  1. compute $(J^\mathsf{l}, \mathbf{v}_I) := \mathsf{Label}.\mathsf{Search}(\gamma^\mathsf{l}, \tau)$
  2. for all $j \in J^\mathsf{l}$,
     (a) compute $J_j^+ := \mathsf{Graph}^+.\mathsf{Neigh}(\gamma^+, \tau_j^+)$,
     (b) compute $J_j^- := \mathsf{Graph}^-.\mathsf{Neigh}(\gamma^-, \tau_j^-)$,
  3. output $J = \left(J^\mathsf{l}, \left(J_j^+, J_j^-\right)_{j \in J^\mathsf{l}}\right)$.
- $\mathsf{Dec}(K, c_j)$: return $m_j := \Pi.\mathsf{Dec}_{K_3}(c_j)$.

Fig. 5. A structured encryption scheme for web graphs supporting focused subgraph queries

7 Conclusions and Future Directions

Several interesting future directions are suggested by this work. The most immediate is whether efficient and non-interactive structured encryption can be achieved while leaking less than the query and intersection pattern. The construction of efficient *dynamic* structured encryption schemes (i.e., that allow for updates to the encrypted data) is another direction left open by this work. Of course, the construction of schemes that handle other types of structured data and more complex queries on the data types considered here would also be interesting.

Acknowledgements

We are grateful to Kristin Lauter for encouragement during the early stages of this work, to Sherman Chow and Satya Lokam for useful discussions regarding graph encryption and to Susan Hohenberger for insisting on a thorough

comparison with functional encryption. We are also grateful to Adam O'Neill for several helpful discussions on functional encryption. Finally, we thank Emily Shen and Charalampos Papamanthou for useful feedback on the manuscript and the anonymous reviewers for helpful suggestions.

References

1. Abdalla, M., Bellare, M., Catalano, D., Kiltz, E., Kohno, T., Lange, T., Lee, J.M., Neven, G., Paillier, P., Shi, H.: Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 205–222. Springer, Heidelberg (2005)
2. Bardin, J., Callas, J., Chaput, S., Fusco, P., Gilbert, F., Hoff, C., Hurst, D., Kumaraswamy, S., Lynch, L., Matsumoto, S., O'Higgins, B., Pawluk, J., Reese, G., Reich, J., Ritter, J., Spivey, J., Viega, J.: Security guidance for critical areas of focus in cloud computing. Technical report, Cloud Security Alliance (April 2009)
3. 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)
4. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: IEEE Symp. on Security and Privacy 2007, pp. 321–334 (2007)
5. 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)
6. 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)
7. Boneh, D., Kushilevitz, E., Ostrovsky, R., Skeith, W.: Public-key encryption that allows PIR queries. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 50–67. Springer, Heidelberg (2007)
8. Boneh, D., Waters, B.: Conjunctive, subset, and range queries on encrypted data. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 535–554. Springer, Heidelberg (2007)
9. Boyen, X., Waters, B.: Anonymous hierarchical identity-based encryption (without random oracles). In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 290–307. Springer, Heidelberg (2006)
10. Brautbar, M., Kearns, M.: Local algorithms for finding intersting individuals in large networks. In: ICS (2010)
11. Chang, Y., Mitzenmacher, M.: Privacy preserving keyword searches on remote encrypted data. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 442–455. Springer, Heidelberg (2005)
12. Chase, M., Kamara, S.: Structured encryption and controlled disclosure. IACR ePrint report (2010), http://eprint.iacr.org
13. Chaum, D., Crépeau, C., Damgard, I.: Multiparty unconditionally secure protocols. In: STOC 1988, pp. 11–19 (1988)
14. Micrsoft Corp. Codename "Dallas", http://www.microsoft.com/windowsazure/dallas
15. Curtmola, R., Garay, J., Kamara, S., Ostrovsky, R.: Searchable symmetric encryption: Improved definitions and efficient constructions. In: ACM CCS 2006, pp. 79–88 (2006)

16. Curtmola, R., Garay, J., Kamara, S., Ostrovsky, R.: Searchable symmetric encryption: Improved definitions and efficient constructions. Journal version (under submission) (2010)
17. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: ACM STOC 2009, pp. 169–178 (2009)
18. Goh, E-J.: Secure indexes. Technical Report 2003/216, IACR ePrint Cryptography Archive (2003), http://eprint.iacr.org/2003/216
19. Goldreich, O., Micali, S., Wigderson, A.: How to play ANY mental game. In: ACM STOC 1987, pp. 218–229 (1987)
20. Goldreich, O., Ostrovsky, R.: Software protection and simulation on oblivious RAMs. Journal of the ACM 43(3), 431–473 (1996)
21. Goldwasser, S., Micali, S.: Probabilistic encryption. Journal of Computer and System Sciences 28(2), 270–299 (1984)
22. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: ACM CCS 2006, pp. 89–98 (2006)
23. Infochimps, http://www.infochimps.org
24. Kamara, S., Lauter, K.: Cryptographic cloud storage. In: Sion, R. (ed.) FC 2010 LNCS, vol. 6054, pp. 136–149. Springer, Heidelberg (2010)
25. Katz, J., Sahai, A., Waters, B.: Predicate encryption supporting disjunctions, polynomial equations, and inner products. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 146–162. Springer, Heidelberg (2008)
26. Kleinberg, J.: Authoritative sources in a hyperlinked environment. In: SODA 1998, pp. 668–677 (1998)
27. Lempel, R., Moran, S.: SALSA: The stochastic approach for link-structure analysis. ACM Transactions on Information Systems 19(2), 131–160 (2001)
28. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)
29. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)
30. Shen, E., Shi, E., Waters, B.: Predicate privacy in encryption systems. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 457–473. Springer, Heidelberg (2009)
31. Shi, E., Bethencourt, J., Chan, T., Song, D., Perrig, A.: Multi-dimensional range query over encrypted data. In: IEEE Symp. on Security and Privacy 2007, pp. 350–364 (2007)
32. Soghoian, C.: Caught in the cloud: Privacy, encryption, and government back doors in the web 2.0 era. Journal on Telecommunications and High Technology Law 8(2) (2010)
33. Song, D., Wagner, D., Perrig, A.: Practical techniques for searching on encrypted data. In: IEEE Symp. on Research in Security and Privacy 2000, pp. 44–55 (2000)
34. Waters, B.: Functional encryption: An overview and survey slides. Presented at Crypto in the Clouds Workshop. MIT, Cambridge (2009)
35. Waters, B., Balfanz, D., Durfee, G., Smetters, D.: Building an encrypted and searchable audit log. In: NDSS 2004 (2004)
36. Yao, A.: Protocols for secure computations. In: FOCS 1982, pp. 160–164 (1982)

Leakage Resilient ElGamal Encryption

Eike Kiltz[1] and Krzysztof Pietrzak[2]

[1] Ruhr-Universität Bochum, Germany*
eike.kiltz@rub.de
[2] CWI, Amsterdam, The Netherlands
pietrzak@cwi.nl

Abstract. Blinding is a popular and well-known countermeasure to protect public-key cryptosystems against side-channel attacks. The high level idea is to randomize an exponentiation in order to prevent multiple measurements of the same operation on different data, as such measurements might allow the adversary to learn the secret exponent. Several variants of blinding have been proposed in the literature, using additive or multiplicative secret-sharing to blind either the base or the exponent. These countermeasures usually aim at preventing particular side-channel attacks (mostly power analysis) and come without any formal security guarantee.

In this work we investigate to which extend blinding can provide *provable security* against a *general class* of side-channel attacks. Surprisingly, it turns out that in the context of public-key encryption some blinding techniques are more suited than others. In particular, we consider a *multiplicatively blinded* version of ElGamal public-key encryption where

- we *prove* that the scheme, instantiated over bilinear groups of prime order p (where $p-1$ is not smooth) is leakage resilient in the generic-group model. Here we consider the model of *chosen-ciphertext security* in the presence of *continuous leakage*, i.e., the scheme remains chosen-ciphertext secure even if with *every* decryption query the adversary can learn a bounded amount (roughly $\log(p)/2$ bits) of arbitrary, adversarially chosen information about the computation.
- we *conjecture* that the scheme, instantiated over arbitrary groups of prime order p (where $p-1$ is not smooth) is leakage resilient.

Previous to this work no encryption scheme secure against continuous leakage was known. Constructing a scheme that can be *proven* secure in the *standard model* remains an interesting open problem.

1 Introduction

Side-channel attacks are cryptanalytic attacks against physical implementations of cryptosystems that exploit some kind of information leakage from the cryptodevice during execution. Traditional security notions (such as chosen-ciphertext security for encryption schemes) do not provide any security guarantee against such

* Part of the work conducted while the author was at CWI, Amsterdam.

M. Abe (Ed.): ASIACRYPT 2010, LNCS 6477, pp. 595–612, 2010.

attacks, and many implementations of provably secure cryptosystems were broken by side-channel attacks exploiting side-channels such as running-time [36], electromagnetic radiation [45,26], power consumption [38], fault detection [8,6] and many more (see, e.g., [46,43]).

Countermeasures against side channel attacks can either be algorithmic, or on the hardware level. In the latter case, one generally tries to build hardware that leaks as few information as possible (e.g., by shielding electromagnetic radiation.) Algorithmic countermeasures means that one designs algorithms, such that their mere description already provides security against side channel attacks. (E.g., one can protect against timing attacks by making sure that the running time of the algorithm is independent of the secret.) Traditionally, such algorithmic countermeasures (such as masking or blinding, cf. [43] for a list of relevant papers) are mostly ad-hoc in the sense that they defend against some specific and known attacks.

Leakage Resilient Cryptography. Recently, formal models were proposed where one does not assume any particular side-channel against which to protect, but only requires that potential side-channels are in some sense "resource bounded." In the model of *leakage resilience* [23], one considers adversaries which, on each invocation of the cryptographic primitive, can learn a bounded amount of arbitrary information about the secret internal state that was accessed during invocation. Since the overall amount of leaked information is unbounded (and may be much larger than the size of the secret state), this model is also often referred to as *continuous leakage* (e.g., [15,9]). As we will discuss below, this is in sharp contrast to the model of "memory leakage" (e.g., [2,41,4,3,16]) which has the inherent limitation that the amount of leaked information is a-priory bounded and therefore cannot exceed the size of the secret state.)

An *implementation* of a leakage resilient primitive will then be secure against every side-channel attack that fits our general model, i.e., as long as the *amount* of information that is leaked on each invocation is sufficiently bounded, and moreover the device adheres the "only computation leaks information" axiom from [40], which states that memory content that is not accessed during an invocation, does not leak. Security in this bounded leakage model hence means that the hardware implementation of the cryptosystem only has to be protected to fit the above model; once that is done, the proof provides security of the scheme. Using bounded leakage is inspired by the bounded retrieval model [13,20,19,10,22,4] which in turn was inspired by the bounded-storage model [39,21,53,30].

So far most theoretical research has focused on preventing memory leakage [13,20,19,10,22,4] and the only known leakage resilient primitives (in our sense of security against continuous leakage) are stream-ciphers [23,44], digital signatures [25] and — in a weaker "non-adaptive" model — pseudorandom functions and permutations [18]. Recently, general compilers have been proposed which turn *any* circuit into a leakage-resilient one [28,33]. Currently, these general compilers are just a proof of concept and too inefficient to be used in practice, relying on fully homomorphic encryption [33] or requiring one full encryption per gate [28].

In this paper, we address the problem of leakage resilient public-key encryption (PKE). The standard security notion for PKE is indistinguishability under a chosen plaintext attack (IND-CPA) or the stronger notion of indistinguishability under a chosen ciphertext attack (IND-CCA).[1]

Modelling Leakage Resilience. Consider some cryptosystem CS, let S_0 denote its initial internal state and S_i its state after the ith invocation. On the ith invocation of CS, the adversary chooses some input X_i and gets Y_i where $(S_{i+1}, Y_i) \leftarrow \mathsf{CS}(S_i, X_i)$.

In the original definition of leakage resilience [23], the adversary gets the additional power to choose, besides the regular input X_i, some leakage function f_i whose range is bounded to some fixed $\lambda \in \mathbb{N}$ bits with every query. After the ith invocation she not only gets the regular output Y_i, but additionally the leakage $\Lambda_i \leftarrow f_i(S_i^+, R)$ where R is the randomness that CS used during its computation, and S_i^+ is the subset of the state S_i that was accessed (i.e., read and/or written) during computation. Note that to be leakage resilient, a primitive must be stateful (i.e. $S_i \neq S_{i-1}$), as otherwise one can just leak the state λ bits at a time.

In this paper we will use a more fine-grained notion of leakage resilience, where an invocation of CS (which will be a decryption query) is split in two phases, and those two phases leak individually. More precisely, the computation of a decryption can syntactically be split into two phases $\mathsf{Dec1}^*$ and $\mathsf{Dec2}^*$, which are executed in a sequential order to decrypt the message. As in a CCA attack, the adversary can make decryption queries with respect to a ciphertext C, and can furthermore specify two (efficiently computable) leakage functions, f and g, whose range is bounded by λ bits. (λ is the leakage parameter.) In addition to the decryption of C the adversary also obtains the output of f and g applied to all the inputs of $\mathsf{Dec1}^*$ and $\mathsf{Dec2}^*$, respectively, including the algorithm's internal random coin tosses.

On Bounded Range and Domain. Summing up, leakage resilience considers attackers who, with every invocation, can adaptively choose a leakage function f and then get the output of f applied to the internal secret state (if the system is probabilistic also all internal coin tosses) of the cryptosystem. The function f can be arbitrary, but is restricted in its input domain and range:

Bounded range: The range of f is $\{0,1\}^\lambda$ for some parameter $\lambda \in \mathbb{N}$.
Bounded domain: f gets as input only the secret state that is actually *accessed* during this invocation.

A mathematical model of side-channel leakage is only useful if it captures (and thus implies security against) leakage that occurs in practice. As f gets the

[1] In a CPA the adversary only gets the public-key and then has to distinguish the encryptions of two different messages. In a CCA [47] the adversary can also ask for decryptions of ciphertexts of her choice. We distinguish between CCA1 and the stronger CCA2 security, in the latter the adversary can make decryption queries also after she got the challenge ciphertext.

same input as the cryptosystem CS, it can simulate the computation of CS on any conceivable hardware (e.g., all the values carried by wires on a computing circuit), and thus also compute any kind of leakage that might occur. Though, the restriction on bounded range might not allow f to actually output the entire leakage, and the restriction on bounded domain might make it impossible to simulate leakage that depends on earlier invocations, we discuss this points below.

Bounded range. In practice, it seems hard to quantify how much information actual hardware (like a smart-card) actually leaks. In most side-channel attacks the adversary measures large amounts of data, e.g., an entire power-consumption curve. So at a glance this assumption might seem unreasonable, but this is a bit overly pessimistic.

Even though side-channel leakage may contain lots of data, only a *small fraction* can actually be exploited in each measurement. The model of leakage resilience allows only for the leakage of a small number λ of bits, but this leakage is "worst case" in the sense that the adversary may choose the leakage function which outputs the most useful information. Below we outline two ways in which this observation can be made precise. The first shows that side-channel attacks used in practice are captured by leakage resilience as they only exploit few bits of information from each actual measurement. The second is a relaxation of bounded leakage which can reasonably be assumed to be satisfied in practice.

Side-Channel Attacks Exploit Few Bits. Many side-channel attacks first measure large amounts of leakage $\Lambda_1, \Lambda_2, \ldots$ from every invocation, like a power consumption curve. Then, in a first step, each leakage Λ_i is preprocessed in order to extract some "useful" information Λ'_i (this Λ'_i could, e.g., be a list of the most likely sub-keys.) The attack then proceeds by trying to recover the secret key from $\Lambda'_1, \Lambda'_2, \ldots$. Such attacks are covered by leakage resilience whenever the amount of extracted data $|\Lambda'_i|$ is at most the amount of leakage λ allowed per invocation.

Relaxing Bounded Range. By inspecting the proofs of our constructions (as well as the ones from [23,44,25]), one sees that a restriction on the leakage functions is required which is considerably weaker than restricting the range to λ bits: it is only required that the leakage $f(S^+)$ does not decrease the HILL-pseudoentropy [31,5][2] the adversary has about the active state S^+ by more than λ bits. (More details will be given in the full version.) Thus, although it may be unreasonable to assume that no more than λ bits leak per invocation of a smart-card, assuming that this leakage will only degrade the HILL-pseudoentropy by λ bits seems much more realistic in practice.

Bounded domain. The *bounded domain* restriction is a very mild restriction. Unlike for bounded range, it is non-trivial to even imagine a remotely realistic side-channel attack which would break a scheme by not adhering to it. This

[2] HILL-pseudoentropy is a computational analogue of min-entropy. As for min-entropy, λ bits of information cannot decrease it (in expectation) by more than λ bits.

restriction (on how leakage functions are mathematically modeled) is implied by the "only computation leaks information" axiom (which states something about physical properties of devices) of [40]. But it also covers other practical attacks which do not satisfy this axiom. For example note that an adversary can learn any linear function $f(S)$ of the *entire* state S (which is split in, say, two parts S_1, S_2 that are accessed individually) by specifying leakage functions f_1, f_2 such that $f_1(a) + f_2(b) = f(a, b)$ (the adversary can ask to learn $f_1(S_1)$ and $f_2(S_2)$ as S_1 and S_2 are accessed respectively, and then compute $f(S)$ locally.) This simple observation already shows that claims made in the literature arguing that the bounded range & domain restrictions do not cover attacks like "cold-boot attacks" [29] or static leakage (as claimed in [51]) are not well-founded.[3] As argued by Dziembowski,[4] this restriction not only covers all linear function $f(a, b) = f_1(a) + f_2(b)$, but in fact any function $f(a, b)$ which has a communication complexity of at most λ. A good candidate for an actual leakage function that does invalidate this assumption[5] is the inner product $f(a, b) = \sum_i a_i \cdot b_i \bmod 2$ which has linear communication complexity.

1.1 ElGamal Encryption

The ElGamal encryption scheme [24] over a cyclic group $\mathbb{G}$ of prime order p works as follows. The public key consists of a generator g of $\mathbb{G}$ and $X = g^x$, where $x \in \mathbb{Z}_p$ is the secret key. Encryption defines the ciphertext as $C = g^r$ and uses the symmetric key $K = X^r$ to blind the message. Decryption reconstructs the key by computing $K = C^x$. In its hybrid version, ElGamal encryption is contained in many standard bodies (e.g., [48,32,50]) and it is (using the name Elliptic Curve Integrated Encryption System, "ECIES") commonly considered to be the *standard method* to encrypt over elliptic curves. At this point it may be instructive to see why the ElGamal encryption scheme is not leakage resilient. An adversary, in the ith decryption query, can specify a leakage function that outputs the i-th bit of the secret key x. Therefore, after $q = |x|$ queries to the leakage oracle the entire secret key can be reconstructed. As we already pointed out, the inherent reason why the above attack works is that decryption is stateless.

Let's first look at a straight forward (but unsuccessful) attempt to make the ElGamal scheme leakage resilient. To this end we make decryption stateful and

[3] In the above argument we implicitly assumed that ultimately the entire secret state will be touched, although this seems obvious (after all, why would one save a secret state if it's not supposed to be ever read), the tokens used in the construction of one-time programs [27] are an example where exactly this happens. For such primitives obeying the "only computation leaks information" axiom in its original physical sense is necessary.

[4] At the workshop "Provable security against physical attacks", February 2010, Leiden.

[5] And thus might be used to construct an actual real world counterexample where the security of an implementation gets broken because the bounded domain restriction is invalidated.

split it into two parts $\mathsf{Dec1}^*$ and $\mathsf{Dec2}^*$. The secret key is *additively* shared into $x = \sigma_0 + \sigma'_0$ by setting $\sigma_0 = x - r_0$ and $\sigma'_0 = x + r_0$. Decryption works as follows. The first part $\mathsf{Dec1}^*$ computes $\sigma_i = \sigma_{i-1} + r_i \bmod p$, $K' = C^{\sigma_i}$ and passes K' as input to the second part. $\mathsf{Dec2}^*$ computes $\sigma'_i = \sigma'_{i-1} - r_i \bmod p$ and then $K = K' \cdot C^{\sigma'_i}$. Note that the state information is randomly re-shared subject to $\sigma_i + \sigma'_i = x$. However, this scheme is not leakage resilient since an attacker can adaptively learn certain bits of $\sigma_i = x + R_i$ and $\sigma'_i = x - R_i$ (where $R_i = \sum_{j=0}^{i} r_j$) that enable him to fully reconstruct the secret key x.[6]

1.2 Our Results

Conjectured leakage resilient ElGamal encryption. We consider a practical randomization method to make the ElGamal PKE scheme (or one if its standardized hybrid variants) leakage resilient under chosen-ciphertext attacks in the above sense. In the context of leakage resilience this method (or variants thereof) were already proposed in [12,37,52]. The central idea is to use *multiplicative secret sharing* to share the secret key x, i.e., x is shared as $\sigma_i = xR_i^{-1} \bmod p$ and $\sigma'_i = R_i \bmod p$, for some random $R_i \in \mathbb{Z}_p^*$. More precisely, the first part of decryption computes $\sigma_i = \sigma_{i-1} r_i^{-1} \bmod p$ and $K' = C^{\sigma_i}$. The second part computes $\sigma'_i = \sigma'_{i-1} r_i \bmod p$ and then $K = K'^{\sigma'_i}$. Again note that the state information is randomly reshared subject to $\sigma_i \cdot \sigma'_i = x$. We remark that our method does not modify ElGamal's encryption algorithm, it only modifies the way ciphertexts are decrypted. In particular, public-keys and ciphertexts are the same as in ElGamal encryption and therefore our method offers an attractive way to update existing ElGamal-based systems with algorithmic security against side-channel attacks. Unfortunately, we are not able to prove that the above method is provable leakage resilient and therefore we can only state the scheme's security as a conjecture.

Provable leakage resilient ElGamal encryption. We also propose to apply multiplicative secret sharing to the ElGamal encryption scheme instantiated over *bilinear groups*. Our main theorem (Theorem 1) states that this scheme is leakage resilient against CCA1 attack in the generic group model. The key observation is that the secret key is a group element X and decryption performs a pairing operation with X as one fixed base. This allows us to multiplicatively share the secret key as a *group element*, i.e., $X = \sigma_i \cdot \sigma'_i \in \mathbb{G}$. Intuitively, we use the fact that in the generic group model some bits of the representation of σ_i and σ'_i essentially look random and therefore are useless to the leakage adversary. To formally prove this intuition, however, turns out to be surprisingly difficult.

We also mention that a proof in the generic group model has its obvious weaknesses. (See, e.g., [35].) In particular in connection with side channel attacks

[6] Since $x = \sigma_i + \sigma'_i \bmod p$, the first $t \approx \lambda$ least significant bits of x can be computed as $(\sigma_i \bmod 2^t) + (\sigma'_i \bmod 2^t) \bmod 2^t$, minus an additive factor $p \bmod 2^t$ in case there is an overflow mod$p$. (The latter can be checked from the high order bits of σ_i and σ'_i.) This process can be iterated to learn the entire secret key.

the generic group model may "abstract away" too much important information an adversary may obtain in a real implementation of the scheme. This should be taken into account when interpreting our formal security statement. However, our result seems to be the first PKE scheme that is provably leakage resilient. Furthermore, the scheme is very practical. Another possible interpretation of our result is that when protecting the exponentiation function against (a large class of) side-channel attacks, multiplicative secret sharing techniques seem more suitable than additive ones.

Leakage Resilient Exponentiation and Pairing Operation. Speaking more generally, our above mentioned methods how to secure ElGamal against side-channel attacks show that one can possibly make *discrete exponentiation* and *a pairing operation* leakage resilient. Let $\mathbb{G}$ be a group of prime order p and g be a generator of $\mathbb{G}$. In discrete exponentiation one wants to take public group elements Y_i to some fixed secret power x (which is only leaked through g^x). We propose to share x as $x = x' \cdot x'' \bmod p$ and compute the values $K_i = Y_i^x$ in two iterative steps as $K_i' = Y_i^{x'}$ followed by $K_i = (K_i')^{x''}$. After each such computation x' and x'' get randomly reshared subject to $x = x' \cdot x'' \bmod p$. In a pairing operation one is given public group elements Y_i and want to compute $e(Y_i, X)$, for some fixed secret group element X (which is only leaked though $e(g, X)$). Here $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a bilinear pairing. Again we propose to share X as $X = X' \cdot X'' \in \mathbb{G}$ and compute the values $K_i = e(Y_i, X)$ in three iterative steps as $K_i' = e(Y_i, X')$, $K_i'' = e(Y_i, X'')$, and $K_i = K_i' \cdot K_i'' \in \mathbb{G}_T$, followed by a resharing of $X = X' \cdot X'' \in \mathbb{G}$. Our main result (Theorem 1) shows that our method to perform a pairing operation is provable leakage resilient in the generic group model.

Difficulty to prove leakage resilience against CCA2 attacks. It is well known that the ElGamal encryption scheme, where the key K is hashed and the one-time pad is replaced with a chosen-ciphertext secure symmetric cipher, is secure against CCA2 attacks [1]. We remark that this scheme is not leakage resilient against CCA2 attack since an adversary can adaptively obtain some bits about the unhashed symmetric key of the challenge ciphertext. Indeed, building a PKE scheme that is (provably) leakage resilient against CCA2 attacks remains a challenging open problem.

1.3 Related Work

In the hardware community the usefulness of secret-sharing in the context of side-channel countermeasures is well known. In particular, secret-sharing has been proposed as a countermeasure against "differential power analysis attacks" for exponentiation algorithms in [11,12,12,37,52], but without any formal analysis.

Most works on side-channel countermeasures, including the ones just mentioned, consider countermeasures against particular side-channel attacks. Micali and Reyzin [40] in their work on "physically observable cryptography" proposed an influential theoretical framework to capture side-channel attacks on a more general level.

Besides leakage resilience, there are several other models that consider cryptosystems which remain secure even if a function $f(sk)$ (chosen by the adversary from a very broad class of functions) of the secret key sk is leaked. We shortly mention these models below. The main difference to leakage resilience is that those models consider stateless cryptosystems, and thus cannot tolerate any kind of "continuous" leakage (an exception is the very recent work on "continuous memory attacks.") On the other hand, the leakage function in those works gets the entire state as input, and not just the part of the state that was accessed.

Memory Attacks. Akavia et al. [2] introduce the model of "security against memory attacks," where one requires that the scheme remains secure even if a function $f(sk)$ of the secret sk is leaked *once*, where the only restriction on $f(\cdot)$ one makes is its bounded output length. (Clearly the bound must satisfy $|f(sk)| \ll |sk|$. This model is a restricted version of the BRM model discussed below.) [2,41] construct public-key encryption schemes in this model, Katz and Vaikuntanathan [34] constructs digital signatures.

Bounded Retrieval Model. The bounded retrieval model (BRM) [13,19,20,10,22,4] is a generalization of the previous model, where one requires that the secret key can be made huge, while the scheme still remains efficient. Such schemes can provide security against malware like viruses or Trojans, which temporarily take control over a computer, but do not have enough "bandwidth" to leak the entire artificially huge key. Most works on intrusion resilient crypto consider symmetric primitives, but after the first success in constructing public-key cryptosystems secure against memory attacks (mentioned above), Alwen et al. achieved public-key crypto also in the BRM model. In particular authentication and signature schemes [4] and public-key encryption [3].

Auxiliary Input. Dodis et al. construct symmetric [17] and public-key [14] encryption schemes in a model where the range of $f(\cdot)$ may be unbounded, but one only requires that it is hard to recover sk from $f(sk)$. (i.e. any polynomial time adversary should output sk with exponentially small probability.)

Continuous Memory Attacks. Very recently, Dodis, Haralambiev, Lopez-Alt, and Wichs [15] and Brakerski, Kalai, Katz and Vaikuntanathan [9] introduce the model of "continuous memory attacks." This model generalizes the notion of memory attacks. Also here the adversary can learn a bounded amount, λ bits say, of leakage about the (entire) secret key. But now there's an additional "refresh" procedure which takes the secret key sk and outputs a new secret key sk'. The adversary can learn λ bits (where λ is $c|sk|$ for some constant $c > 0$) in-between any two refresh phases, but the refreshing itself has to be completely leak-free [15] or leak at most a logarithmic number of bits [9]. Remarkably, in this model [15] construct authentication and signature schemes, [9] obtain get public-key encryption. Both papers work in the standard model, the underlying assumption in both papers is the linear assumption over bilinear groups. The models of leakage resilience and continuous memory attacks are incomparable: leakage resilience assumes "only computation leaks" whereas continuous memory attacks need an (almost) leak-free refresh phase. As mentioned, the constructions [15,9] are proven secure in the

standard model, whereas we use a strong idealized model. On the positive side, our scheme is very efficient (only about two times slower than standard ElGamal) whereas, e.g., [9] needs a constant number of pairings to encrypt a single bit.

2 Definitions

If A is a deterministic algorithm we write $y \leftarrow \mathsf{A}(x)$ to denote that A outputs y on input x. If A is randomized we write $y \xleftarrow{*} \mathsf{A}(x)$ or, $y \xleftarrow{r} \mathsf{A}(x)$ if we want to make the randomness r used by the algorithm explicit (for future reference).

Key Encapsulation Mechanisms. A key encapsulation mechanism (KEM) is defined similarly to a public-key encryption scheme, except that the encryption algorithm (called encapsulation) does not take any input, but rather outputs the encryption of a random key K, which then can be used with as a key in any symmetric encryption scheme to encrypt the actual message.

Formally, a key-encapsulation mechanism KEM consists of three algorithms $\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec}$. $\mathsf{KG} : \{0,1\}^* \rightarrow \mathcal{PK} \times \mathcal{SK}$ is the probabilistic key-generation algorithm, which on input a security parameter κ outputs a public/secret-key pair. The probabilistic encapsulation algorithm $\mathsf{Enc} : \mathcal{PK} \rightarrow \mathcal{K} \times \mathcal{C}$ and decapsulation algorithm $\mathsf{Dec} : \mathcal{SK} \times \mathcal{C} \rightarrow \mathcal{K} \cup \perp$ satisfy the following correctness property for all κ

$$\Pr[K = K' \mid (pk, sk) \xleftarrow{*} \mathsf{KG}(\kappa);\ (C, K) \xleftarrow{*} \mathsf{Enc}(pk);\ K' \leftarrow \mathsf{Dec}(sk, C)] = 1$$

The CCA1 security (aka. security against lunchtime attacks) of a key-encapsulation mechanism KEM is defined by the following experiment.

Experiment $\mathbf{Exp}^{\mathrm{cca1}}_{\mathsf{KEM}}(\mathcal{F}, \kappa)$
$(pk, sk) \xleftarrow{*} \mathsf{KG}(\kappa)$
$w \xleftarrow{*} \mathcal{F}^{\mathcal{O}_{sk}(\cdot)}(pk)$
$b \xleftarrow{*} \{0,1\}$
$(C^*, K_0) \xleftarrow{*} \mathsf{Enc}(pk)$
$K_1 \xleftarrow{*} \mathcal{K}$
$b' \xleftarrow{*} \mathcal{F}(w, C^*, K_b)$

Oracle $\mathcal{O}^{\mathrm{cca1}}_{sk}(C)$
$K \leftarrow \mathsf{Dec}(sk, C)$
Return K

Let μ denote the probability that $b = b'$ in the above experiment, then we define the advantage of $\mathcal{F}$ as $\mathbf{Adv}^{\mathrm{cca1}}_{\mathsf{KEM}}(\mathcal{F}, \kappa) = 2|1/2 - \mu|$. In CCA2 security, the adversary is additionally allowed to query the decryption oracle in its second (guess) stage.

Stateful key encapsulation and leakage resilience. To formally define our notion of leakage resilience we consider stateful key encapsulation mechanisms $\mathsf{KEM}^* = (\mathsf{KG}^*, \mathsf{Enc}^*, \mathsf{Dec1}^*, \mathsf{Dec2}^*)$ in which decapsulation is stateful and can formally split into two sequential stages $\mathsf{Dec} = (\mathsf{Dec1}^*, \mathsf{Dec2}^*)$. The input/output behavior will stay exactly the same as in a standard KEM.

More formally, the key generation algorithm $\mathsf{KG}^*(\kappa)$ generates a public key and and two initial states, σ_0 and σ'_0. Intuitively, the states shares the secret key of the scheme and will be used by the stateful decapsulation algorithms $\mathsf{Dec1}^*, \mathsf{Dec2}^*$.

On the ith invocation of decapsulation, the decapsulated key K_i is computed as follows

$$(\sigma_i, w_i) \xleftarrow{r_i} \mathsf{Dec1}^*(\sigma_{i-1}, C_i) \; ; \; (\sigma'_i, K_i) \xleftarrow{r'_i} \mathsf{Dec2}^*(\sigma'_{i-1}, w_i) \tag{1}$$

Here r_i and r'_i is the explicit randomness of the two randomized algorithms, σ_i and σ'_i are the updated states and w_i is some state information that is passed from $\mathsf{Dec1}^*$ to $\mathsf{Dec2}^*$.

We now define leakage resilience. Let $\lambda \in \mathbb{N}$ be some leakage parameter. We will consider attacks, where the adversary can not only query its oracle for the decapsulated values $K_i = \mathsf{Dec}(sk, C_i)$, but additionally gets leakage from the computation of those values. That is, in the security experiment the adversary can, additionally to the input C_i, specify two efficiently computable leakage functions f_i, g_i with bounded range $\{0,1\}^\lambda$, and additionally to the regular output K_i also gets Λ_i, Λ'_i computed as

$$\Lambda_i = f_i(\sigma_{i-1}, r_i) \; ; \; \Lambda'_i = g_i(\sigma'_{i-1}, w_i, r'_i) \; ,$$

where the notation is as in (1). So the functions f_i, g_i get as input exactly the same data as $\mathsf{Dec1}^*/\mathsf{Dec2}^*$.[7] We define the CCLA1 (chosen ciphertext with leakage attack) security of KEM by the experiment below. (Note that now we not only have to specify the security parameter k, but also a leakage bound λ.)

Experiment $\mathbf{Exp}^{\mathrm{ccla}}_{\mathsf{KEM}}(\mathcal{F}, \kappa, \lambda)$	**Oracle** $\mathcal{O}^{\mathrm{ccla1}}(C, f, g)$
$(pk, \sigma_0, \sigma'_0) \xleftarrow{*} \mathsf{KG}(\kappa)$	If range of f or g is $\neq \{0,1\}^\lambda$ return $\perp$
$w \xleftarrow{*} \mathcal{F}^{\mathcal{O}^{\mathrm{ccla1}}(\cdot)}(pk)$	$i \leftarrow i + 1$
$b \xleftarrow{*} \{0,1\}$	$(\sigma_i, w_i) \xleftarrow{r_i} \mathsf{Dec1}^*(\sigma_{i-1}, C)$
$(C^*, K_0) \xleftarrow{*} \mathsf{Enc}(pk)$	$(\sigma'_i, K_i) \xleftarrow{r'_i} \mathsf{Dec2}^*(\sigma'_{i-1}, w_i)$
$K_1 \xleftarrow{*} \mathcal{K}$	$\Lambda_i \leftarrow f_i(\sigma_{i-1}, r_i)$
$i \leftarrow 0$	$\Lambda'_i \leftarrow g_i(\sigma'_{i-1}, w_i, r'_i)$
$b' \xleftarrow{*} \mathcal{F}(w, C^*, K_b)$	Return $(K_i, \Lambda_i, \Lambda'_i)$

Let μ denote the probability that $b = b'$ in the above experiment, then we define the advantage of $\mathcal{F}$ as $\mathbf{Adv}^{\mathrm{ccla}}_{\mathsf{KEM}}(\mathcal{F}, \kappa, \lambda) = 2|1/2 - \mu|$.

It is well-known that a CCA1 secure KEM plus a one-time secure symmetric cipher (such as a one-time pad) yields a CCA1-secure PKE scheme. For trivial reasons the same statement is also true for CCLA1 secure KEMs so for our purpose it is sufficient to build a CCLA1 secure KEM. On the other hand we remark that the respective composition theorem is wrong in general for CCLA2 secure KEMs.

[7] Note that C_i need not be explicitly given to f_i as the adversary chooses f_i and C_i together, and thus can "hard-code" C_i into f_i.

That is, a CCLA2 secure KEM and a CCA secure DEM will in general not yield a CCLA2 secure PKE scheme.[8]

Bilinear Groups. We assume the existence of a bilinear group generator BGen which is a randomized algorithm that outputs a bilinear group $\mathbb{PG} = (\mathbb{G}, \mathbb{G}_T, g, e, p)$ such that the following properties hold.

1. $\mathbb{G}$ and $\mathbb{G}_T$ are (multiplicative) cyclic groups of prime order p.
2. g is a generator of $\mathbb{G}$.
3. e is a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ which is
   (a) bilinear: for all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}, e(u^a, v^b) = e(u, v)^{ab}$.
   (b) non-degenerate: $e(g, g) \neq 1$.

We say that $\mathbb{G}$ is a bilinear group if there exists a group $\mathbb{G}_T$ and a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ as above, where e and the group action in $\mathbb{G}$ and $\mathbb{G}_T$ can be computed efficiently. We will use $\circ$ and $\star$ for the group operation in $\mathbb{G}$ and $\mathbb{G}_T$ respectively.

Generic Bilinear Groups. In the generic group model [42,49] one encodes the group elements by unique, randomly chosen strings. This enforces that the only property which can be tested by an adversary is equality.

In the generic bilinear model (GBG) [7] the encoding is given by randomly chosen injective functions $\xi : \mathbb{Z}_p \to \Xi$ and $\xi_T : \mathbb{Z}_p \to \Xi_T$ which give the representations of the elements in the base and target group respectively (w.l.o.g. we will assume that $\Xi \cap \Xi_T = \emptyset$). The group operation and the bilinear map are performed by three public oracles $\mathcal{O}, \mathcal{O}_T, \mathcal{O}_e$, where for any $a, b \in \mathbb{Z}_p$

- $\mathcal{O}(\xi(a), \xi(b)) \to \xi(a + b \bmod p)$ (group operation on base group).
- $\mathcal{O}_T(\xi_T(a), \xi_T(b)) \to \xi_T(a + b \bmod p)$ (group operation on target group).
- $\mathcal{O}_e(\xi(a), \xi(b)) \to \xi_T(a \cdot b \bmod p)$ (bilinear map).

All oracles output $\perp$ when queried on an input outside of their domain. For a fixed generator g of $\mathbb{G}$ and $g_T \stackrel{\mathsf{def}}{=} e(g, g)$, one can think of $\xi(a)$ as an encoding of g^a, $\xi_T(a)$ as an encoding of g_T^a and $\xi_e(a, b)$ as an encoding of $g_T^{a \cdot b} = e(g^a, g^b)$. Of course one also must provide some means of computing the group representation $\xi(a)$ or $\xi_T(a)$ for any $a \in \mathbb{Z}_p$, say by providing oracles to do so. We can get away without additional oracles, by providing $\xi(1)$ and observing that then $\xi(a)$ can be computed making $\leq 2 \log p$ queries to $\mathcal{O}$ (by square and multiply). $\xi_T(1)$ (and thus any $\xi_T(a)$) can be computed by $\xi_T(1) \leftarrow \mathcal{O}_e(\xi(1), \xi(1))$.

3 Leakage Resilient ElGamal Encryption

In this section we present a general method to secure ElGamal encryption against leakage attacks. First, we present a modification of the standard ElGamal cryptosystem over any cyclic group of prime order. Unfortunately, we are not able to

[8] An attacker may make a number of decryption queries only modifying the symmetric part of the challenge ciphertext. The decryption algorithm (internally) uses the challenge symmetric key that can be learned (bit-by-bit) through the leakage function.

formally prove the leakage resilience of this scheme so we state its security as a conjecture. Next, we move to the ElGamal scheme over Bilinear Groups. Here we are able to prove that our method leads to a leakage resilient public-key encryption scheme (in the sense of CCLA1) in the generic group model.

3.1 ElGamal Key Encapsulation

Let Gen be a randomized algorithm that outputs a cyclic group $\mathbb{G}$ of order p where p is a strong prime. The ElGamal key-encapsulation mechanism $\mathsf{EG} = (\mathsf{KG}_{\mathsf{EG}}, \mathsf{Enc}_{\mathsf{EG}}, \mathsf{Dec}_{\mathsf{EG}})$ is defined as follows.

- $\mathsf{KG}_{\mathsf{EG}}(\kappa)$: Compute $(\mathbb{G}, p) \xleftarrow{*} \mathsf{Gen}(\kappa)$ and choose random $g \xleftarrow{*} \mathbb{G}$ and random $x \xleftarrow{*} \mathbb{Z}_p$. Set $X = g^x$. The public key is $pk = (\mathbb{G}, p, X)$ and the secret key is $sk = x$.
- $\mathsf{Enc}_{\mathsf{EG}}(pk)$: choose random $r \xleftarrow{*} \mathbb{Z}_p$. Set $C \leftarrow g^r \in \mathbb{G}$ and $K \leftarrow X^r \in \mathbb{G}$. The ciphertext is C and the key is K.
- $\mathsf{Dec}_{\mathsf{EG}}(sk, C)$: Compute the key as $K = C^x \in \mathbb{G}$.

As mentioned in the introduction, EG (or any other stateless scheme) cannot be leakage resilient since in the CCLA1 experiment an adversary can simply obtain the successive bits of the secret key x.

We will now describe a leakage resilient stateful key encapsulation mechanism $\mathsf{EG}^* = (\mathsf{KG}^*_{\mathsf{EG}}, \mathsf{Enc}^*_{\mathsf{EG}}, \mathsf{Dec1}^*_{\mathsf{EG}}, \mathsf{Dec2}^*_{\mathsf{EG}})$, which is derived from EG. As described in Section 2, the decapsulation algorithm is stateful and split in two parts.

- $\mathsf{KG}^*_{\mathsf{EG}}(\kappa)$: Run $(sk, pk) \xleftarrow{*} \mathsf{KG}_{\mathsf{EG}}(\kappa)$. (Recall that $sk = x$ and $pk = (\mathbb{G}, p, X = g^x)$.) Choose random $\sigma_0 \xleftarrow{*} \mathbb{Z}^*_p$ and set $\sigma'_0 = x\sigma_0^{-1} \bmod p$. The public key is pk and the two secret states are σ_0 and σ'_0.
- $\mathsf{Enc}^*_{\mathsf{EG}}(pk)$: the same as $\mathsf{Enc}_{\mathsf{EG}}(pk)$.
- $\mathsf{Dec1}^*_{\mathsf{EG}}(\sigma_{i-1}, C)$: choose random $r_i \xleftarrow{*} \mathbb{Z}^*_p$, set $\sigma_i = \sigma_{i-1} r_i^{-1} \bmod p$, $K' = C^{\sigma_i}$ and return (r_i, K').
- $\mathsf{Dec2}^*_{\mathsf{EG}}(\sigma'_{i-1}, (r_i, K'))$: set $\sigma'_i = \sigma'_{i-1} r_i^{-1} \bmod p$, and $K = K'^{\sigma'_i}$. The symmetric key is K and the updated state information is σ_i and σ'_i.

We cannot formally prove CCLA1 security of the scheme so we have to resort to the following conjecture.

Conjecture 1. EG^* is CCLA1 secure if $p-1$ has a large prime factor (say, $p-1 = 2p'$ for a prime p').[9]

[9] The reason we require p to be not smooth is to prevent the leakage functions to possibly compute discrete logarithms in $\mathbb{Z}_{p-1}$, as otherwise the multiplicative sharing σ, σ' (where $\sigma \cdot \sigma' = x$) can be efficiently turned into an additive sharing (of the discrete log of the secret key) $\sigma = h^{\Sigma}, \sigma' = h^{\Sigma'}$ where $x = h^X$ and $X = \Sigma + \Sigma'$. As described in Section 1.1, an additive sharing cannot give a leakage resilient scheme. The above also hints the inherent difficulty of proving this conjecture. Let us mention that already in [37] it is suggested to use a prime p where (p-1)/2 is prime in a very similar context. Our result can be seen as a formal justification for this choice.

One can furthermore make ElGamal key-encapsulation CCA2 secure (without leakage) under the strong Diffie-Hellman assumption in the random oracle model by hashing the symmetric key symmetric key K [1]. This Hashed ElGamal scheme is contained in many standard bodies, e.g. [48,32,50]. Hashing the symmetric key clearly does not affect its CCLA1 security and therefore Hashed ElGamal is CCLA1 and CCA2 secure.

However, as we will explain now, in our leakage resilience setting hashing K will not make the scheme CCLA2 secure. The (unhashed) EG scheme is not CCA2 secure since it is malleable. (An adversary, given the challenge ciphertext C (enciphering a key K), can ask for a decryption of $C^2 \neq C$ to obtain K^2 from which it can reconstruct K.) Without considering leakage, hashing the key prevents this attack as now the adversary only sees a hashed key $\mathcal{H}(K^2)$. Unfortunately, in the leakage setting hashing will not help at all because the adversary can specify a leakage function which outputs λ bits of the unhashed key K^2. By asking for the decryption of the same ciphertext C^2 several times, leaking λ different bits of K^2 on each invocation, will ultimately reveal the entire K^2.

3.2 Bilinear ElGamal Key Encapsulation

The Bilinear ElGamal key-encapsulation mechanism $\mathsf{BEG} = (\mathsf{KG}_{\mathsf{BEG}}, \mathsf{Enc}_{\mathsf{BEG}}, \mathsf{Dec}_{\mathsf{BEG}})$ is defined as follows.

- $\mathsf{KG}_{\mathsf{BEG}}(\kappa)$: Compute $\mathbb{PG} = (\mathbb{G}, \mathbb{G}_T, p, e) \xleftarrow{*} \mathsf{BGen}(\kappa)$ and choose random $g \xleftarrow{*} \mathbb{G}$ and random $x \xleftarrow{*} \mathbb{Z}_p$. Set $X = g^x$ and $X_T = e(g,g)^x$. The public key is $pk = (\mathbb{PG}, g, X_T)$ and the secret key is $sk = X$.
- $\mathsf{Enc}_{\mathsf{BEG}}(pk)$: choose random $r \xleftarrow{*} \mathbb{Z}_p$. Set $C \leftarrow g^r \in \mathbb{G}$ and $K \leftarrow X_T^r \in \mathbb{G}_T$. The ciphertext is C and the key is K.
- $\mathsf{Dec}_{\mathsf{BEG}}(sk, C)$: Compute the key as $K = e(C, X) \in \mathbb{G}_T$.

Note that correctness follows from the bilinear property $X_T^r = e(g,g)^{xr} = e(g^r, g^x) = e(g^r, X)$.

We will now describe a leakage resilient key encapsulation $\mathsf{BEG}^* = (\mathsf{KG}^*_{\mathsf{BEG}}, \mathsf{Enc}^*_{\mathsf{BEG}}, \mathsf{Dec1}^*_{\mathsf{BEG}}, \mathsf{Dec2}^*_{\mathsf{BEG}})$, which is derived from BEG.

- $\mathsf{KG}^*_{\mathsf{BEG}}(\kappa)$: Run $(sk, pk) \xleftarrow{*} \mathsf{KG}_{\mathsf{BEG}}(\kappa)$. (Recall that $sk = X = g^x$ and $pk = (\mathbb{PG}, g, X_T = e(g,g)^x$.) Choose random $r_0 \xleftarrow{*} \mathbb{Z}_p^*$ and set $\sigma_0 \leftarrow g^{r_0}, \sigma_0' \leftarrow g^{x-r_0}$. The public key is pk and the secret states are σ_0, σ_0'.
- $\mathsf{Enc}^*_{\mathsf{BEG}}(pk)$: the same as $\mathsf{Enc}_{\mathsf{BEG}}(pk)$.
- $\mathsf{Dec1}^*_{\mathsf{BEG}}(\sigma_{i-1}, C)$: choose random $r_i \xleftarrow{*} \mathbb{Z}_p$, set $\sigma_i \leftarrow \sigma_{i-1} \circ g^{r_i}$, $K' \leftarrow e(\sigma_i, C)$ and return (r_i, K').
- $\mathsf{Dec2}^*_{\mathsf{BEG}}(\sigma'_{i-1}, (r_i, K'))$: set $\sigma'_i \leftarrow \sigma'_{i-1} \circ g^{-r_i}$ and $K'' \leftarrow e(\sigma'_i, C)$. The symmetric key is $K \leftarrow K' \star K'' \in \mathbb{G}_T$.

Note that for every i, $R_i \stackrel{\mathsf{def}}{=} \sum_{j=0}^{i} r_j$, we have $\sigma_i \circ \sigma'_i = g^{R_i} \circ g^{x-R_i} = g^x$, so the σ_i, σ'_i are a secret sharing of the secret key.

Theorem 1. *In the bilinear generic group model the scheme* BEG^* *is CCLA1 secure: the advantage of a* q*-query adversary who gets* λ *bits of leakage per invocation of* $\mathsf{Dec1}^*_{\mathsf{BEG}}$ *and* $\mathsf{Dec2}^*_{\mathsf{BEG}}$*, respectively, is at most* $\frac{2^{2\lambda+1}\cdot q^3}{p}$.

Thus, for a statistical security parameter n, we can tolerate $\lambda = \log(p)/2 - 3\log(q) - n/2$ bits of leakage. For space reasons, here we only can give a proof outline. The complete proof appears in the full version of this paper.

Proof Outline. For technical reasons, we will consider a setting where the generic bilinear group is extended with an additional oracle $\mathcal{O}_{DL} : \Xi \cup \Xi_T \to \mathbb{Z}_p \cup \perp$, we will call this the *extended* generic bilinear group model. Intuitively, $\mathcal{O}_{DL}$ is an oracle for the discrete log problem, but only works on inputs that have not yet appeared since the oracles $\mathcal{O}, \mathcal{O}_e, \mathcal{O}_T$ have been initialized.

The proof outline is as follows. We will first show that the discrete logarithm problem (DL) is hard in the (base group of the) *extended* GBG model. We then give a reduction which shows how any adversary that can break the CCA1 security (without leakage) of BEG^* in the (normal) GBG model, can solve the discrete log problem in the extended GBG model. Next, we extend this proof to get our main result, namely a reduction of the CCLA1 security of BEG^* to the discrete log problem.

CCA1 security of BEG^*. Let $\mathcal{F}$ be an adversary that can break the CCA1 security of BEG^*. We construct an adversary $\mathcal{G}$ for DL (using $\mathcal{F}$ as a black-box) by letting $\mathcal{G}$ simulate the $\mathbf{Exp}^{\mathrm{cca1}}_{\mathsf{BEG}^*}(\mathcal{F}, p)$ experiment, where in this experiment $\mathcal{G}$ uses its DL challenge $\xi(y)$ as either the secret key $\xi(x)$ or the challenge encapsulated key $\xi(s)$ with probability $1/2$ respectively.

During the CCA1 experiment, $\mathcal{F}$ (which initially gets $\xi_T(x)$, and after the last decapsulation query gets $\xi(s)$) will learn the representation of elements $\xi_T(e_1)$, $\xi_T(e_2), \ldots$ from the target group. One can show that just from observing $\mathcal{F}$'s oracle queries, $\mathcal{G}$ can assign to each e_i an equation $e_i = a_i + b_i \cdot x + c_i \cdot s + d_i \cdot s^2$ where it knows the coefficients $a_i, b_i, c_i, d_i \in \mathbb{Z}_p$. Similarly, for representations $\xi(e_1), \xi(e_2), \ldots$ of elements in the base group that $\mathcal{F}$ learns, $\mathcal{G}$ can extract a_i, b_i such that $e_i = a_i + b_i \cdot s$. To get an idea why this is the case, consider, e.g., the case where $\mathcal{F}$ makes a query $\xi_T(v \cdot w) \leftarrow \mathcal{O}_e(\xi(v), \xi(w))$. If $\xi(v)$ (same for $\xi(w)$) was never seen before, $\mathcal{G}$ first calls $\mathcal{O}_{DL}(\xi(v))$ to learn v. (Recall that $\mathcal{G}$ is in the extended GBG model.) Now $\mathcal{G}$ knows a, b, a', b' s.t. $v = a + b \cdot s$ and $w = a' + b' \cdot s$, which implies $v \cdot w = a'' + b'' \cdot x + c'' \cdot s + d'' \cdot s^2$ with $a'' = a + a', b'' = 0, c'' = a \cdot b' + a' \cdot b, d'' = b \cdot b'$.

Recall that $\mathcal{F}$'s goal is to distinguish the decapsulated key $\xi_T(x \cdot s)$ from a random element. If $\mathcal{F}$ has advantage ϵ in doing so, it actually must compute the element $\xi_T(x \cdot s)$ with probability ϵ. Which means we learn a, b, c, d such that

$$x \cdot s = a + b \cdot x + c \cdot s + d \cdot s^2, \tag{2}$$

this can be solved for s or x (or both). Thus $\mathcal{G}$ will learn the discrete log of $\xi(y)$ with probability at least $\varepsilon/2$ (as we initially randomly set $\xi(y) = s$ or $\xi(y) = x$).

CCLA1 security of BEG^*. The proof in the case of leakage attacks is more delicate. In a CCLA1 attack, with the ith decapsulation query, the adversary $\mathcal{F}$ also

learns the output of the leakage functions f_i, g_i. If we had no upper bound on the output length of those functions, then f_i and g_i could just leak $\xi(R_i)$ and $\xi(x-R_i)$ respectively, from which $\mathcal{F}$ then could first compute the secret key $\xi(x)$ and then $\xi_T(x \cdot s)$. In this case, the reduction $\mathcal{G}$ does not learn an equation of the form eq. (2), but only the trivial equality $x \cdot s = x \cdot s$. We will prove that if the leakage bound $\lambda \ll \log p/2$, then the leakage functions will not leak any representation of an element to $\mathcal{F}$ that $\mathcal{F}$ could not efficiently compute itself.

To see this, let us first make the simplifying assumption that the leakage functions f_i, g_i are not given access to the group oracles $\mathcal{O}, \mathcal{O}_e, \mathcal{O}_T$. Then all the leakage functions can try to do, is to leak some element they get as input. Consider any such element, say $\xi(R_i)$. As $\xi(R_i)$ is only given as input to f_{i-1} and f_i, at most 2λ bits about this element can leak. If $2\lambda \ll \log p$, then $\mathcal{F}$ will have high min-entropy about $\xi(R_i)$ even given this 2λ bits of leakage. Thus it is very unlikely that it can guess $\xi(R_i)$.

Now consider the general case, where the leakage functions can use the group oracles. Now the leakage functions can trivially leak the representation of some group element, say $f_1, f_2, \ldots$ all use $\mathcal{O}$ to compute $\xi(z)$ for some fixed z and each leaks λ bit of $\xi(z)$ until $\mathcal{F}$ learns the entire $\xi(a)$. Now $\mathcal{F}$ does get the representation of an element $\xi(a)$ without receiving it from the group oracles, but that is no problem, as $\mathcal{G}$ will know an a, b such that $a + b \cdot s = z$ (namely $a = z$ and $b = 0$), and that's all we care about.

Now the f_i leakage function (similarly for g_i) can use their input $\xi(R_{i-1})$ to compute elements $\xi(z)$ where $\mathcal{G}$ only knows a, b (where $b \neq 0$) such that $z = a + b \cdot r_0$. We call such a representation "bound" (as opposed to "free" representations $\xi(z)$ where $\mathcal{G}$ trivially learns z by just observing f_i's oracle queries). It would be a problem if a bound representation could leak to $\mathcal{F}$. As said before, the f_i's can trivially leak 2λ bits about a bound element, as, e.g., f_{i-1} and f_i have access to $\xi(R_i)$ (recall that $R_i = \sum_{j=0}^{i} r_j$ where each r_j is uniformly random). But it is not clear how any other leakage function f_j ($j \notin \{i-1, i\}$) would compute the element $\xi(R_i)$ or any other element derived from it; since the sharings are randomized during each invocation, the values $\xi(R_{j-1}), r_j$ that f_j has are completely independent of R_i (and thus $\xi(R_i)$). In fact, we show that if $\mathcal{F}$ manages to choose leakage functions such that the same bound element is computed by f_i and f_j (where $j > i+1$) with probability ϵ, then $\mathcal{F}$ can be used to solve the discrete logarithm problem with probability $\epsilon/2^{2\lambda}q$. The idea is to use the discrete logarithm challenge $\xi(y)$ as $\xi(r_j)$ for a random j. Note that to simulate the experiment, $\mathcal{G}$ only needs $\xi(r_j)$ not r_j, except to compute the 2λ bits of leakage from the jth decapsulation query. (As here the leakage functions f_j, g_j expect r_j as input.) We let $\mathcal{G}$ randomly guess this leakage, which will be correct with probability $2^{-2\lambda}$. Now assume we have two identical bound elements $\xi(z)$ computed by $f_{i'}$ and $f_{i''}$ where $i'' > i' + 1$. As this query was made by $f_{i'}$, and up to this point $\mathcal{G}$ only used $r_0, \ldots, r_{i'}$ that it sampled himself, he will know z. As this query was also made by i'', $\mathcal{G}$ learns $a, b \neq 0$ such that $z = a + b \cdot r_j$, and thus can solve this equality to get r_j.

References

1. Abdalla, M., Bellare, M., Rogaway, P.: The oracle Diffie-Hellman assumptions and an analysis of DHIES. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, p. 143. Springer, Heidelberg (2001)
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, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 36–54. Springer, Heidelberg (2009)
5. Barak, B., Shaltiel, R., Wigderson, A.: Computational analogues of entropy. In: RANDOM-APPROX, pp. 200–215 (2003)
6. Biham, E., Shamir, A.: Differential fault analysis of secret key cryptosystems. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 513–525. Springer, Heidelberg (1997)
7. 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)
8. Boneh, D., De Millo, R.A., Lipton, R.J.: On the importance of checking cryptographic protocols for faults (extended abstract). In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 37–51. Springer, Heidelberg (1997)
9. Brakerski, Z., Kalai, Y.T., Katz, J., Vaikuntanathan, V.: Overcoming the hole in the bucket: Public-key cryptography resilient to continual memory leakage. In: 51st FOCS. IEEE Computer Society Press, Los Alamitos (2010)
10. 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)
11. Chari, S., Jutla, C.S., Rao, J.R., 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)
12. Clavier, C., Joye, M.: Universal exponentiation algorithm. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 300–308. Springer, Heidelberg (2001)
13. 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)
14. Dodis, Y., Goldwasser, S., Kalai, Y.T., Peikert, C., Vaikuntanathan, V.: Public-key encryption schemes with auxiliary inputs. In: Micciancio, D. (ed.) Theory of Cryptography. LNCS, vol. 5978, pp. 361–381. Springer, Heidelberg (2010)
15. Dodis, Y., Haralambiev, K., Lopez-Alt, A., Wichs, D.: Cryptography against continuous memory attacks. In: 51st FOCS. IEEE Computer Society Press, Los Alamitos (2010)
16. Dodis, Y., Haralambiev, K., Lopez-Alt, A., Wichs, D.: Efficient public-key cryptography in the presence of key leakage. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 595–612. Springer, Heidelberg (2010)

17. Dodis, Y., Kalai, Y.T., Lovett, S.: On cryptography with auxiliary input. In: 41st ACM STOC. ACM Press, New York (2009)
18. 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)
19. 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)
20. Dziembowski, S.: On forward-secure storage (extended abstract). In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 251–270. Springer, Heidelberg (2006)
21. Dziembowski, S., Maurer, U.M.: Optimal randomizer efficiency in the bounded-storage model. Journal of Cryptology 17(1), 5–26 (2004)
22. Dziembowski, S., Pietrzak, K.: Intrusion-resilient secret sharing. In: 48th FOCS, pp. 227–237. IEEE Computer Society Press, Los Alamitos (2007)
23. Dziembowski, S., Pietrzak, K.: Leakage-resilient cryptography. In: 49th FOCS, pp. 293–302. IEEE Computer Society Press, Los Alamitos (2008)
24. El Gamal, T.: On computing logarithms over finite fields. In: Williams, H.C. (ed.) CRYPTO 1985. LNCS, vol. 218, pp. 396–402. Springer, Heidelberg (1986)
25. Faust, S., Kiltz, E., Pietrzak, K., Rothblum, G.N.: Leakage-resilient signatures. In: Micciancio, D. (ed.) Theory of Cryptography. LNCS, vol. 5978, pp. 343–360. Springer, Heidelberg (2010)
26. 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)
27. Goldwasser, S., Kalai, Y.T., Rothblum, G.N.: One-time programs. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 39–56. Springer, Heidelberg (2008)
28. 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)
29. Alex Halderman, J., 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. ACM Commun. 52(5), 91–98 (2009)
30. Harnik, D., Naor, M.: On everlasting security in the hybrid bounded storage model. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006.Part II LNCS, vol. 4052, pp. 192–203. Springer, Heidelberg (2006)
31. 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)
32. IEEE P1363a Committee. IEEE P1363a / D9 — standard specifications for public key cryptography: Additional techniques (June 2001), http://grouper.ieee.org/groups/1363/index.html/ draft Version 9
33. 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)
34. 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)
35. Koblitz, N., Menezes, A.J.: Another look at generic groups. Advances in Mathematics of Communications 1, 13–28 (2007)
36. 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)

37. Kocher, P.C., Jaffe, J.: Leak-Resistant Cryptographic Method and Apparatus. United States Patent 6304658 B1 (October 16, 2001)
38. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
39. Maurer, U.M.: A provably-secure strongly-randomized cipher. In: Damgård, I. (ed.) EUROCRYPT 1990. LNCS, vol. 473, pp. 361–373. Springer, Heidelberg (1991)
40. Micali, S., Reyzin, L.: Physically observable cryptography (extended abstract). In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 278–296. Springer, Heidelberg (2004)
41. Naor, M., Segev, G.: Public-key cryptosystems resilient to key leakage. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 18–35. Springer, Heidelberg (2009)
42. Nechaev, V.I.: Complexity of a determinate algorithm for the discrete logarithm. Mathematical Notes 55(2), 165–172 (1994)
43. European Network of Excellence (ECRYPT). The side channel cryptanalysis lounge, http://www.crypto.ruhr-uni-bochum.de/en_sclounge.html (retrieved on March 29, 2008)
44. Pietrzak, K.: A leakage-resilient mode of operation. In: Joux, A. (ed.) EUROCRYPT. LNCS, pp. 462–482. Springer, Berlin (2009)
45. Quisquater, J.-J., Samyde, D.: Electromagnetic analysis (ema): Measures and counter-measures for smart cards. In: E-smart, pp. 200–210 (2001)
46. Quisquater, J.-J., Koene, F.: Side channel attacks: State of the art (October 2002) [43]
47. 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)
48. Certicom research, standards for efficient cryptography group (SECG) — sec 1: Elliptic curve cryptography (September 20, 2000), http://www.secg.org/secg_docs.htm version 1.0
49. Shoup, V.: Lower bounds for discrete logarithms and related problems. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 256–266. Springer, Heidelberg (1997)
50. Shoup, V.: ISO 18033-2: An emerging standard for public-key encryption (December 2004), http://shoup.net/iso/std6.pdf (final Committee Draft)
51. 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/
52. Trichina, E., Bellezza, A.: Implementation of elliptic curve cryptography with built-in counter measures against side channel attacks. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 98–113. Springer, Heidelberg (2003)
53. 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)

Efficient Public-Key Cryptography in the Presence of Key Leakage

Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs

New York University
dodis@cs.nyu.edu, kkh@cs.nyu.edu, lopez@cs.nyu.edu, wichs@cs.nyu.edu

Abstract. We study the design of cryptographic primitives resistant to a large class of side-channel attacks, called "memory attacks", where an attacker can repeatedly and adaptively learn information about the secret key, subject *only* to the constraint that the *overall amount* of such information is bounded by some parameter ℓ. Although the study of such primitives was initiated only recently by Akavia et al. [2], subsequent work already produced many such "leakage-resilient" primitives [48, 4, 42], including signature, encryption, identification (ID) and authenticated key agreement (AKA) schemes. Unfortunately, every existing scheme, — for any of the four fundamental primitives above, — fails to satisfy at least one of the following desirable properties:

- **Efficiency.** While the construction may be generic, it should have some *efficient* instantiations, based on standard cryptographic assumptions, and without relying on random oracles.
- **Strong Security**. The construction should satisfy the strongest possible definition of security (even in the presence of leakage). For example, encryption schemes should be secure against chosen *ciphertext* attack (CCA), while signatures should be *existentially* unforgeable.
- **Leakage Flexibility.** It should be possible to set the scheme parameters so that the leakage bound ℓ can come arbitrarily close to the secret-key size.

In this work we design the first signature, encryption, ID and AKA schemes which overcome these limitations, and satisfy all the properties above. Moreover, all our constructions are generic, in several cases elegantly simplifying and generalizing the prior constructions (which did not have any efficient instantiations). We also introduce several tools of independent interest, such as the abstraction (and constructions) of *true-simulation extractable* NIZK arguments, and a new *deniable* DH-based AKA protocol based on any CCA-secure encryption.

1 Introduction

Traditionally, the security of cryptographic schemes has been analyzed in an idealized setting, where an adversary only sees the specified "input/output behavior" of a scheme, but has no other access to its internal secret state. Unfortunately, in the real world, an adversary may often learn some partial information about secret state via various *key leakage* attacks. Such attacks come in a large variety and include *side-channel attacks* [43, 10, 7, 44, 54, 27], where the physical realization of a cryptographic primitive

M. Abe (Ed.): ASIACRYPT 2010, LNCS 6477, pp. 613–631, 2010.

can leak additional information, such as the computation-time, power-consumption, radiation/noise/heat emission etc. The cold-boot attack of Halderman et al. [34] is another example of a key-leakage attack, where an adversary can learn (imperfect) information about memory contents of a machine, even after the machine is powered down. Schemes that are proven secure in an idealized setting, without key leakage, may become completely insecure if the adversary learns even a small amount of information about the secret key. Indeed, even very limited leakage attacks have been shown to have devastating consequences for the security of many natural schemes.

Unfortunately, it is unrealistic to assume that we can foresee, let alone block, all of the possible means through which key leakage can occur in real-world implementations of cryptographic schemes. Therefore, the cryptographic community has recently initiated the investigation of increasingly general (formally modeled) classes of leakage attacks, with the aim of constructing *leakage-resilient* cryptographic schemes that remain provably secure even in the presence of such attacks. Of course, if an adversary can get unrestricted information about the secret key, then she can learn the key in its entirety and the security of the system is necessarily compromised. Therefore, we must first place some "upper bound" on the type or amount of information that the adversary can learn. The nature of such bounds varies in the literature, as we survey later. For this work, we only restrict the *amount*, but not the *type*, of information that an adversary can learn through a key-leakage attack. In particular, we will assume that the attacker can learn *any efficiently computable function of the secret key* sk, subject only to the constraint that the total amount of information learned (i.e. the output size of the leakage function) is bounded by ℓ bits, where ℓ is called the "leakage parameter" of the system.[1] Clearly, at this level of generality, the secret-key size s must be strictly greater than the leakage-parameter ℓ.[2] Therefore, the quantity ℓ/s can be thought as the *relative leakage* of the system, with the obvious goal to make it as close to 1 as possible.

Our model of leakage-resilience was recently introduced by Akavia et al. [2], but already attracted a lot of attention from the cryptographic community [48,4,42,3]. In particular, as we survey later, we already know many "leakage-resilient" primitives, including such fundamental primitives as signature schemes, encryption schemes, identification (ID) schemes and authenticated key agreement (AKA) protocols. Unfortunately, we observe that every existing scheme, — for any of the four fundamental primitives above, — fails to satisfy at least one of the following desirable properties:

– **Efficiency.** While the proposed construction may be based on some generic cryptographic primitives, — which is in fact preferable for modular design, — it should have some *efficient* instantiations, based on standard cryptographic assumptions, and without relying on random oracles. We view this property as the main property we will strive to achieve.

[1] More formally, we allow adaptive measurements, as long as the sum of leaked outputs is bounded by ℓ.

[2] In fact, our actual constructions easily extend to the more general "noisy leakage" model of Naor and Segev [48], where the outputs can be longer than s, as long as the "average min-entropy" of sk drops by at most ℓ bits. However, we do not pursue this generalization, in order to keep our notation simple.

– **Strong Security**. The construction should satisfy the strongest possible definition of security (even in the presence of leakage). For example, encryption schemes should be secure against chosen *ciphertext* attack (CCA), while signatures should be *existentially* unforgeable, etc.
– **Leakage Flexibility.** It should be possible to set the parameters of the schemes so that the relative leakage ℓ/s is arbitrarily close to 1. We call such schemes *leakage-flexible*.

1.1 Our Results

In this work we design the first signature, encryption, ID and AKA schemes which simultaneously satisfy the efficiency, strong security and leakage flexibility properties mentioned above. Moreover, all our constructions are generic. This means that the actual construction is modularly defined and explained using natural simpler blocks, and its security against key leakage is also proven no matter how these simpler blocks are (securely) implemented. However, unlike the prior generic constructions, which did not have any known efficient instantiations (at least, with the desired security and flexibility we seek), ours are yet more general, which will allow us to obtain several efficient instantiations. Given this fact, it is not surprising that our contributions can be roughly split into two categories: "conceptual" contributions, allowing us to obtain more general (and, yet, conceptually simpler) leakage-resilient constructions, and "concrete" contributions, allowing us to instantiate our general schemes efficiently.

CONCEPTUAL CONTRIBUTIONS. As we will see, existing schemes (e.g., signature and CCA-encryption) could be largely divided into two categories: potentially efficient schemes, with some *inherent* limitation not allowing them to achieve relative leakage approaching 1 (which also prevents us from using these ideas for our purposes), and more theoretical schemes [48,42], achieving good relative leakage, but relying on the notion of *simulation-sound* non-interactive zero-knowledge (ss-NIZK) [56]. Informally, ss-NIZK proofs remain sound even if the attacker can see simulated proofs of arbitrary (even false) statements. Unfortunately, it appears that the existing cryptographic machinery does not allow us to instantiate non-trivial ss-NIZK proofs efficiently.[3] On the other hand, a recent breakthrough result of Groth-Sahai [33] showed that one can obtain efficient *non-simulation-sound* NIZK proofs for a non-trivial class of languages. While the techniques of [31] could be applied to Groth-Sahai proofs to achiehve ss-NIZKs, it is a non-trivial "exercise" and the resulting proofs are *significantly* less efficient, as the construction involves OR-proofs for Groth-Sahai languages. Therefore, our first idea was to try to generalize the existing constructions sufficiently, making them rely only on regular NIZKs, in the hope that such NIZKs can then be instantiated using the powerful Groth-Sahai techniques.

In the end, this is indeed what we realized. However, in the process we also abstracted away an elegant notion of independent interest: *true-simulation extractable* (tSE) NIZKs. While similar to the notion of simulation-sound extractable NIZKs [31],

[3] The work of [31] constructs ss-NIZK proofs for practical languages and uses them to construct group signatures, but the resulting scheme has signature size of "thousands or perhaps even millions of group elements" [32] despite being constant.

it involves a subtle but rather important difference: whether the adversary has oracle access to simulated proofs for arbitrary (even false) statements or only true ones. Intuitively, both the Naor-Segev's leakage-resilient CCA encryption [48] and Katz-Vaikuntanathan's leakage-resilient signature scheme [42] used the technique of encrypting a witness x for some relation R, and then providing a ss-NIZK proof φ that the ciphertext c indeed contains the encryption of a valid witness x. The main reason for using this technique is to allow the reduction to extract a valid witness from any "new" valid pair (c^*, φ^*) produced by the attacker $\mathcal{A}$ (who saw many such valid pairs earlier). In this paper, we will abstract this property into the tSE notion mentioned above (of which the above mentioned technique is a specific example, where the pair (c, φ) together makes up a single tSE-NIZK proof). Moreover, we show that true-simulation extractability, as we abstract it, is *precisely* the right notion for generalizing and proving the security of the previous constructions. This has two positive effects. First, it makes the generic constructions of CCA-encryption and signatures somewhat more intuitive, both for proving and understanding. For example, the traditional "double-encryption" paradigm of Naor-Yung [49] for designing CCA-secure schemes from chosen-plaintext secure (CPA-secure) schemes, also used by [48] in the context of key leakage, can be stated as "CPA-encrypting message m under two keys and proving plaintext equality". Using our more general "simulation-extractability view", it is now stated as "CPA-encrypting m and proving that one knows the plaintext". We believe that the latter view is not only more general, but also more intuitive as a way of explaining "CPA-to-CCA" transformation. It also follows the original intuition of Rackoff and Simon [55], who combine CPA-encryption with NIZK-POK to achieve CCA-encryption, but in the model where the sender also has a secret key. A similar discussion is true for our signature constructions.

Second, we show a generic way to build tSE-NIZKs which *avoids using (expensive) ss-NIZKs*. Instead, our method uses *regular* NIZKs and *any* CCA-secure encryption scheme.[4] Perhaps surprisingly, given the current state-of-the-art NIZK and CCA schemes, the combination "CCA + NIZK" appears to be much more efficient in practice than the combination "CPA + ss-NIZK".[5] As a result, we were able to provide a general framework for building leakage-flexible signature and CCA-encryption schemes, eventually allowing us to efficiently instantiate our schemes (by avoiding using ss-NIZKs). We summarize our results for signature and CCA-encryption schemes in Tables 1 and 2, also comparing them to the best prior constructions. In all the tables, the "sub-optimal" entries (for efficiency, security, model or relative leakage of prior constructions) are written in italics, and most prior rows are also explained in the related work Section 1.2. For signatures, we stress that no efficient construction in the standard model was known prior to our work, for any non-trivial relative leakage fraction (let alone 1).

Once we have efficient leakage-flexible signature schemes, we can obtain ID and AKA schemes with the same properties. The signature-based AKA protocol is not deniable. However, we also construct a *deniable* AKA protocol based on our construction

[4] This is OK for the signature application, but might appear strange for our CCA-encryption application, as we need "CCA to get CCA". However, as a building block for tSE-NIZKs, we only need *standard* CCA schemes and as a result obtain *leakage-resilient* CCA schemes.

[5] Indirectly, the same realization was made by Groth [32] and Camenisch et al. [11].

Table 1. Previous work on leakage-resilient signatures and results of this work

Reference	Unforgeability	Model	Leakage	Efficient?
[4]	Existential	*Random Oracle*	*1/2*	Yes
[4]	*Entropic*	*Random Oracle*	1	Yes
[42]	Existential	Standard	1	*No*
This Work	Existential	Standard	1	Yes

Table 2. Previous work on leakage-resilient encryption and results of this work

Reference	Attack	Model	Leakage	Efficient?
[2,48]	*CPA*	Standard	1	Yes
[48]	CCA	Standard	*1/6*	Yes
[48]	CCA	Standard	1	*No*
This Work	CCA	Standard	1	Yes

Table 3. Previous work on leakage-resilient identification schemes and results of this work

Reference	Security	Model	Leakage	Efficient?
[4]	*Pre-Impersonation*	Standard	1	Yes
[4]	Anytime	Standard	*1/2*	Yes
[42] (implicit)	Anytime	Standard	1	*No*
This Work	Anytime	Standard	1	Yes

Table 4. Previous work on leakage-resilient AKA and results of this work

Reference	Model	Leakage	Deniable?	Efficient?
[4]	*Random Oracle*	1	*No*	Yes
[4,42]	Standard	1	*No*	*No*
This Work	Standard	1	*No*/Yes*	Yes

* Our first AKA protocol is not deniable; our second — is.

of leakage-flexible CCA-secure encryption. We summarize our results for ID schemes in Table 3 and for AKA protocols in Table 4. See Section 6 for details.

CONCRETE CONTRIBUTIONS. As we explained above, we generically reduce the question of building efficient leakage-flexible ID schemes and AKA protocol to the question of efficiently instantiating our leakage-flexible signature and/or encryption schemes. Such instantiations are given in Section 5. We also explained how the latter instantiations became possible in our work, since we gave generic constructions of both primitives based on the new notion of tSE-NIZK, and then showed that satisfying this notion may be possible using *ordinary* NIZKs for appropriate languages, without relying on the expensive simulation-sound NIZKs. Unfortunately, efficient construction of (even ordinary) NIZKs, due to Groth and Sahai [33], are only known for a pretty restrictive class or languages in bilinear groups. Thus, obtaining a *concrete* efficient instantiation still requires quite a substantial effort.

Specifically, all the building blocks have to be instantiated efficiently, and expressed in a form such that the resulting NP relation satisfies the severe limitations imposed by the Groth-Sahai NIZKs. For example, to build leakage-resilient CCA-encryption, we need to have an efficient leakage-flexible CPA scheme, a CCA scheme supporting labels and a one-time signature scheme, all connected together by an efficient NIZK for a complicated "plaintext equality" relation. Similarly, for leakage-resilient signature schemes, we need an efficient second-preimage resistant (SPR; see Definition 1) relation and a CCA scheme supporting labels, once again connected by an efficient NIZK for a complex relation. Not surprisingly, such tasks cannot typically be done by simply combining "off-the-shelf" schemes from the literature. At best, it requires very careful selection of parameters to make everything "match", followed by a round of further efficiency optimizations. Usually, though, it requires the design of new primitives, which work well with other known primitives, to enable efficient NIZK. For example, in this work, we designed two new SPR relations (see Section 5), since prior SPR relations did not appear to mesh well with our CCA encryption scheme. To emphasize the importance of the new SPR relations, we point out that combining previous constructions with Groth-Sahai proofs would require committing to the witness bit-by-bit in order to achieve full extractability.

Overall, we get two different efficient instantiations of both leakage-resilient signature and CCA encryption schemes in the standard model, based on standard (static and "fixed-length") assumptions in bilinear groups, called external Diffie-Hellman (SXDH) and Decision Linear (DLIN). The high-level idea of these schemes, as well as their efficiency, is described in Section 5. The actual low-level details of how to put "everything together" in the most efficient manner, is described in the full version [18].

1.2 Related Work

LEAKAGE-RESILIENCE AND MEMORY ATTACKS. Our model of leakage, sometimes called memory-attacks, was first proposed by Akavia et al. [2], who also constructed CPA secure PKE and IBE schemes in this model under the *learning with errors (LWE)* assumption. Later Naor and Segev [48] generalized the main ideas behind these constructions to show that all schemes based on *hash proof systems* (see [15]) are leakage-resilient. In particular, this resulted in efficient constructions based on the DDH and K-Linear assumptions, where the relative leakage on the secret key could be made to approach 1. Moreover, [48] showed how to also achieve CCA security in this model by either: (1) relying on the generic (and inefficient) Naor-Yung paradigm where the leakage-rate can be made to approach 1 or (2) using efficient hash proof systems with leakage-rate only approaching $1/6$. Unfortunately, it seems that the hash proof system approach to building CCA encryption is inherently limited to leakage-rates below $1/2$: this is because the secret-key consists of two components (one for verifying that the ciphertext is well-formed and one for decrypting it) and the proofs break down if either of the components is individually leaked in its entirety. The work of [3] generalizes [48] still further by showing how to construct leakage-resilient IBE schemes generically based on *identity-based hash proof systems*, with several instantiations.

Leakage-resilient signature schemes in the model of memory attacks were constructed in the random-oracle model by [4,42], and in the standard model by [42]. The

random-oracle schemes are highly-efficient but suffer from two limitations. Firstly they rely on the Fiat-Shamir [25] transform which is only known to be secure in the Random Oracle model and is not sound in general [30]. Secondly, the schemes can only tolerate leakage which approaches $1/2$ of the secret key. On the other hand, the standard-model schemes allow for relative-leakage approaching 1, but are based on generic simulation-sound NIZKs and do not come with an efficient instantiation.

The work of [4] also constructs ID schemes and AKA protocols. For ID schemes, two notions of security were considered: a weaker notion called pre-impersonation leakage-resilience and a stronger notion called anytime leakage-resilience. Although efficient schemes in the standard model were given for both notions, the leakage resilience could be made to approach 1 only for pre-impersonation leakage while, for anytime leakage, the given schemes can only tolerate a leakage-rate below $1/2$. For AKA schemes, a construction was given based on leakage-resilient signatures (only requiring a weakened notion of security called entropic-unforgeability). Using the appropriate signature schemes, this yielded two types of constructions: efficient constructions in the random-oracle model and generic but inefficient constructions in the standard model (both of which have leakage-rates approaching 1).

OTHER MODELS OF LEAKAGE-RESILIENCE. Several other models of leakage-resilience have appeared in the literature. They differ from the model we described in that they restrict the *type*, as well as *amount*, of information that the adversary can learn. For example, *exposure resilient cryptography* [12,20,41] studies the case where an adversary can only learn some small *subset of the physical bits of the secret key*. Similarly, [38] studies how to implement arbitrary computation in the setting where an adversary can observe a small *subset o the physical wires of a circuit*. Most recently, [24] study a similar problem, where the adversary can observe a low-complexity (e.g. AC^0) function of the wires. Unfortunately, these models fail to capture many meaningful side-channel attacks, such as learning the hamming-weight of the bits or their parity.

In their seminal work, Micali and Reyzin [46] initiated the formal modeling of side-channel attacks under the axiom that *"only computation leaks information"* (OCLI), where each invocation of a cryptographic primitive leaks a function of *only* the bits accessed during that invocation. Several primitives have been constructed in this setting including stream ciphers [22,53] and signatures [23]. More recently, [40] construct a general compiler that can secure *all primitives* in this setting assuming the use of some limited leak-free components and the existence of fully homomorphic encryption. On the positive side, the OCLI model only imposes a bound on the amount of information learned during each invocation of a primitive, but not on the overall amount of information that the attacker can get throughout the lifetime of the system. On the negative side, this model fails to capture many leakage-attacks, such as the cold-boot attack of [34], where *all* memory contents leak information, even if they were never accessed.

Lastly, we mention models of leakage-resilience which are strictly stronger than the memory-attacks model. Firstly, the Bounded-Retrieval Model [16,21,4,3] imposes an additional requirement on leakage-resilient schemes, by insisting that they provide a way to "grow" the secret-key (possibly to many Gigabytes) so as to proportionally increase the amount of tolerated leakage, but without increasing the size of the public-key, the computational or communication efficiency of the scheme, or the lengths of the

ciphertexts or signatures. The work of [4] constructs "entropic" signatures, ID schemes and AKA protocols in this setting, while the work of [3] constructs PKE and IBE schemes in this model. A different strengthening is the auxiliary input model [19,17] where the leakage is not necessarily bounded in length, but it is (only) assumed to be computationally hard to recover the secret-key from the leakage. The work of [19] constructs symmetric-key encryption in this model, under a strengthening of the learning parity with noise (LPN) assumption, while [17] constructs public-key encryption under the DDH and LWE assumptions. Yet another strengthening of the memory-attacks model, proposed by [29], is to require that there is a single scheme (parameterized only by the security parameter) which can tolerate essentially any amount of relative-leakage where the exact-security of the scheme degrades smoothly as the relative-leakage increases. In this model, [29] construct a symmetric-key encryption scheme.

2 Definitions of Leakage-Resilient Primitives

We model leakage attacks by giving the adversary access to a *leakage oracle*, which he can adaptively access to learn leakage on the secret key. A leakage oracle $\mathcal{O}_{sk}^{\lambda,\ell}(\cdot)$ is parametrized by a secret key sk, a leakage parameter ℓ, and a security parameter λ. A query to the leakage oracle consists of a function $h_i : \{0,1\}^* \to \{0,1\}^{\alpha_i}$, to which the oracle answers with $y_i = h_i(sk)$. We only require that the functions h_i be efficiently computable, and the total number of bits leaked is $\sum_i \alpha_i \leq \ell$.

Definition 1 (Leakage Resilient Hard Relation). *A relation R with a randomized PPT sampling algorithm* $\texttt{KeyGen}$ *is an* ℓ-leakage resilient hard relation *if:*

- *For any $(sk, pk) \leftarrow \texttt{KeyGen}(1^\lambda)$, we have $(sk, pk) \in R$.*
- *There is a poly-time algorithm that decides if $(sk, pk) \in R$.*
- *For all PPT adversaries $\mathcal{A}^{\mathcal{O}_{sk}^{\lambda,\ell}(\cdot)}$ with access to the leakage oracle $\mathcal{O}_{sk}^{\lambda,\ell}(\cdot)$:*

$$\Pr\left[R(sk^*, pk) = 1 \;\mid\; (pk, sk) \leftarrow \texttt{KeyGen}(1^\lambda)\,,\; sk^* \leftarrow \mathcal{A}^{\mathcal{O}_{sk}^{\lambda,\ell}(\cdot)}(pk)\right] \leq negl(\lambda)$$

Notice that without loss of generality, we can assume that $\mathcal{A}$ queries $\mathcal{O}_{sk}^{\lambda,\ell}(\cdot)$ only once with a function h whose output is ℓ bits.

Definition 2 (Leakage Resilient Signatures). *A signature scheme* $\mathcal{S} = (\texttt{KeyGen}, \texttt{Sign}, \texttt{SigVer})$ *is* ℓ-leakage resilient *if* $\forall$ *PPT $\mathcal{A}$ we have* $\Pr[\mathcal{A}\ wins] \leq negl(\lambda)$ *in the following game:*

1. **Key Generation:** *The challenger runs $(vk, sk) \leftarrow \texttt{KeyGen}(1^\lambda)$ and gives vk to $\mathcal{A}$.*
2. **Signing and leakage queries:** *$\mathcal{A}^{\mathcal{O}_{sk}^{\lambda,\ell}(\cdot),\mathcal{S}_{sk}(\cdot)}$ is given access to the leakage oracle $\mathcal{O}_{sk}^{\lambda,\ell}(\cdot)$ and the signing oracle $\mathcal{S}_{sk}(\cdot)$. A query to the signing oracle $\mathcal{S}_{sk}(\cdot)$ consists of a message m, to which the oracle responds with $\sigma = \texttt{Sign}_{sk}(m)$.*
3. *$\mathcal{A}$ outputs (m^*, σ^*) and wins if $\texttt{SigVer}_{vk}(m^*, \sigma^*) = 1$ and m^* was not given to $\mathcal{S}_{sk}(\cdot)$ as a signing query.*

Definition 3 (Leakage Resilient CCA-Secure Encryption). *We say that an encryption scheme* $\mathcal{E} = (\texttt{KeyGen}, \texttt{Enc}, \texttt{Dec})$ *is* ℓ-leakage resilient CCA-secure *if* $\forall$ *PPT $\mathcal{A}$ we have* $\Pr[\mathcal{A}\ wins] \leq \frac{1}{2} + negl(\lambda)$ *in the following game:*

1. **Key Generation:** *The challenger runs* $(pk, sk) \leftarrow \mathtt{KeyGen}(1^\lambda)$ *and gives* pk *to* $\mathcal{A}$.
2. **Decryption and leakage queries:** $\mathcal{A}^{\mathcal{O}^{\lambda,\ell}_{sk}(\cdot),\mathcal{D}_{sk}(\cdot)}$ *is given access to the leakage oracle* $\mathcal{O}^{\lambda,\ell}_{sk}(\cdot)$ *and the decryption oracle* $\mathcal{D}_{sk}(\cdot)$. *A query to the decryption oracle* $\mathcal{D}_{sk}(\cdot)$ *consists of a ciphertext* c, *to which the oracle responds with* $m = \mathtt{Dec}_{sk}(c)$.
3. **Challenge generation:** $\mathcal{A}$ *sends plaintexts* m_0, m_1 *to the challenger. The challenger chooses* $b \xleftarrow{\$} \{0,1\}$, *and sends* $c^* \leftarrow \mathtt{Enc}_{pk}(m_b)$ *to* $\mathcal{A}$.
4. **Decryption queries:** $\mathcal{A}^{\mathcal{D}_{sk}(\cdot)}$ *is given access to the decryption oracle* $\mathcal{D}_{sk}(\cdot)$ *with the restriction that* $\mathcal{A}$ *cannot send* c^* *as a decryption query. Notice also that* $\mathcal{A}^{\mathcal{D}_{sk}(\cdot)}$ *is* not *given access to the leakage oracle* $\mathcal{O}^{\lambda,\ell}_{sk}(\cdot)$.
5. $\mathcal{A}$ *outputs* b', *and wins if* $b = b'$.

We refer to a 0-leakage-resilient CCA-secure *as simply* CCA-secure.

Recall that we can define labeled CCA encryption in which a message is encrypted and decrypted according to a public label L. If an encryption scheme $\mathcal{E} = (\mathtt{KeyGen}, \mathtt{Enc}, \mathtt{Dec})$ supports labels, we use the syntax $\mathtt{Enc}^L(m)$ to denote the encryption of message m under label L. Similarly, we use $\mathtt{Dec}^L(c)$ to denote the decryption of ciphertext c under the label L. In this case, we extend the correctness of encryption/decryption to requiring that $\mathtt{Dec}^L(\mathtt{Enc}^L(m)) = m$. The security definition described in Definition 3 can also be easily modified as follows. A query to the decryption oracle now consists of a ciphertext c and a label L, to which the oracle responds with $m = \mathtt{Dec}^L_{sk}(c)$. In the challenge generation stage, $\mathcal{A}$ submits a label L^* as well as messages m_0, m_1 and the challenger computes $c^* \leftarrow \mathtt{Enc}^{L^*}_{pk}(m_b)$ for $b \xleftarrow{\$} \{0,1\}$. Finally, in the second stage of decryption queries we require that the adversary is allowed to ask for decryptions of any ciphertext c under label L only subject to $(L, c) \neq (L^*, c^*)$.

Definition 4 (Leakage Resilient CPA-Secure Encryption). *We say that an encryption scheme* $\mathcal{E} = (\mathtt{KeyGen}, \mathtt{Enc}, \mathtt{Dec})$ *is* ℓ-leakage resilient CPA-secure *if* $\forall$ *PPT* $\mathcal{A}$ *we have* $\Pr[\mathcal{A}\ \textit{wins}] \leq \frac{1}{2} + negl(\lambda)$ *in the game described above with the modification that* $\mathcal{A}$ *does not have access to the decryption oracle* $\mathcal{D}_{sk}(\cdot)$. *If an encryption scheme is* 0-leakage-resilient CPA-secure *we simply refer to it as being* CPA secure.

3 Simulation Extractability

We start by briefly recalling the notion of *non-interactive zero-knowledge (NIZK)* [8]. For our purposes, it will be slightly more convenient to use the notion of *(same-string) NIZK argument* from [57]. Note, however, that the definitions and constructions given in this section can be extended to the case of NIZK proofs.

Let R be an NP relation on pairs (x, y) with corresponding language $L_R = \{y \mid \exists x \text{ s.t. } (x, y) \in R\}$. A *non-interactive zero-knowledge (NIZK) argument* for a relation R consists of three algorithms $(\mathtt{Setup}, \mathtt{Prove}, \mathtt{Verify})$ with syntax:

- $(\mathrm{CRS}, \mathrm{TK}) \leftarrow \mathtt{Setup}(1^\lambda)$: Creates a common reference string (CRS) and a trapdoor key to the CRS.
- $\pi \leftarrow \mathtt{Prove}_{\mathrm{CRS}}(x, y)$: Creates an argument that $R(x, y) = 1$.
- $0/1 \leftarrow \mathtt{Verify}_{\mathrm{CRS}}(y, \pi)$: Verifies whether or not the argument π is correct.

For the sake of clarity, we write $\mathtt{Prove}$, $\mathtt{Verify}$ without the CRS in the subscript when the CRS can be inferred from the context. We require that the following properties hold:

Completeness: For any $(x, y) \in R$, if $(\mathrm{CRS}, \mathtt{TK}) \leftarrow \mathtt{Setup}(1^\lambda)$, $\pi \leftarrow \mathtt{Prove}(x, y)$, then $\mathtt{Verify}(y, \pi) = 1$.

Soundness: For any PPT adversary $\mathcal{A}$, $\Pr[\mathtt{Verify}(y, \pi^*) = 1 \wedge y \notin L_R] \leq negl(\lambda)$, where the probability is taken over $(\mathrm{CRS}, \mathtt{TK}) \leftarrow \mathtt{Setup}(1^\lambda), (y, \pi^*) \leftarrow \mathcal{A}(\mathrm{CRS})$.

Composable Zero-Knowledge: There exists PPT simulator $\mathtt{Sim}$ such that, for any PPT adversary $\mathcal{A}$ we have $\left|\Pr[\mathcal{A} \text{ wins }] - \frac{1}{2}\right| \leq negl(\lambda)$ in the following game:

- The challenger samples $(\mathrm{CRS}, \mathtt{TK}) \leftarrow \mathtt{Setup}(1^\lambda)$ and gives $(\mathrm{CRS}, \mathtt{TK})$ to $\mathcal{A}$.
- $\mathcal{A}$ chooses $(x, y) \in R$ and gives these to the challenger.
- The challenger samples $\pi_0 \leftarrow \mathtt{Prove}(x, y), \pi_1 \leftarrow \mathtt{Sim}(y, \mathtt{TK}), b \leftarrow \{0, 1\}$ and gives π_b to $\mathcal{A}$.
- $\mathcal{A}$ outputs a bit $\tilde{b}$ and wins if $\tilde{b} = b$.

We revisit the notion of simulation extractable NIZK arguments [58,13,50,51,31], and define a new primitive called *true-simulation extractable* NIZK arguments. Apart from satisfying the properties described above, an NIZK argument is simulation extractable if there exists a PPT *extractor* $\mathtt{Ext}$ which given an additional trapdoor to the CRS, extracts a witness x' from any proof π produced by a malicious prover P^*, *even* if P^* has previously seen some *simulated proofs* for other statements. We make an important distinction between our new definition of *true*-simulation extractability, where all simulated proofs seen by P^* are only of *true* statements, and the stronger notion of *any*-simulation extractability, where P^* can also see proofs of *false* statements. As we will see, the former notion is often simpler to construct and sufficient in our applications.

We extend our definition to f*-extractability*, where $\mathtt{Ext}$ only needs to output some function $f(x')$ of a valid witness x'. We further extend this definition to support *labels*, so that the $\mathtt{Prove}$, $\mathtt{Verify}$, $\mathtt{Sim}$, and $\mathtt{Ext}$ algorithms also take a public label L as input, and the correctness, soundness, and zero-knowlegde properties are updated accordingly. If $\Pi = (\mathtt{Setup}, \mathtt{Prove}, \mathtt{Verify})$ is an NIZK argument with simulator $\mathtt{Sim}$ and extractor $\mathtt{Ext}$, we write $\mathtt{Prove}^L, \mathtt{Verify}^L, \mathtt{Sim}^L, \mathtt{Ext}^L$ to denote proof, verification, simulation, and extraction under label L, respectively.

We start by defining a simulation oracle $\mathcal{SIM}_{\mathrm{TK}}(\cdot)$. A query to the simulation oracle consists of a pair (x, y) and a label L. The oracle checks if $(x, y) \in R$. If true, it ignores x and outputs a simulated argument $\mathtt{Sim}^L(\mathtt{TK}, y)$, and otherwise outputs $\perp$. We now give a formal definition of true-simulation extractability.

Definition 5 (True-Simulation f-Extractability). *Let f be a fixed efficiently computable function and let $\Pi = (\mathtt{Setup}, \mathtt{Prove}, \mathtt{Verify})$ be an NIZK argument for a relation R, satisfying the completeness, soundness and zero-knowledge properties above. We say that Π is* true-simulation f-extractable *(f-tSE) with labels if:*

- *Apart from outputting a CRS and a trapdoor key,* $\mathtt{Setup}$ *also outputs an extraction key:* $(\mathrm{CRS}, \mathtt{TK}, \mathrm{EK}) \leftarrow \mathtt{Setup}(1^\lambda)$.
- *There exists a PPT algorithm* $\mathtt{Ext}(y, \varphi, \mathrm{EK})$ *such that for all* P^*, $\Pr[P^* \; wins] \leq negl(\lambda)$ *in the following game:*
  1. **Key Generation:** *The challenger runs* $(\mathrm{CRS}, \mathtt{TK}, \mathrm{EK}) \leftarrow \mathtt{Setup}(1^\lambda)$ *and gives* CRS *to* P^*.
  2. **Simulation queries:** $P^{*\mathcal{SIM}_{\mathrm{TK}}(\cdot)}$ *is given access to the simulation oracle* $\mathcal{SIM}_{\mathrm{TK}}(\cdot)$*, which it can adaptively access.*

3. **Adversary Output:** *P^* outputs a tuple (y^*, L^*, φ^*).*
4. **Extraction:** *The challenger runs $z^* \leftarrow \mathtt{Ext}^{L^*}(y^*, \varphi^*, \mathtt{EK})$.*
5. *P^* wins if (a) the pair (y^*, L^*) was not part of a simulator query, (b) $\mathtt{Verify}^{L^*}(y^*, \varphi^*) = 1$, and (c) for all x' such that $f(x') = z^*$ we have $R(x', y^*) = 0$.*[6]

In the case when f is the identity function, we simply say that Π is true-simulation extractable (tSE).

We give several variations of this new primitive. First, we define *one-time* simulation extractability, in which the adversary P^* is only given *a single* query to the simulation oracle $\mathcal{SIM}_{\mathrm{TK}}(\cdot)$. Second, we define the notion of *strong* simulation extractability by changing the winning condition so that P^* is now required to output a new statement/argument pair instead of a new statement. More formally, condition 5a becomes: the tuple (y^*, L^*, φ^*) is new, that is, either (y^*, L^*) was not part of a simulator query, or if it was, the argument φ^* is different from the one(s) given to P^* by $\mathcal{SIM}_{\mathrm{TK}}(\cdot)$. We observe that we can generically construct strong f-tSE NIZK arguments from (standard) f-tSE NIZK arguments if we additionally use a strongly-secure one-time signature. In particular, the prover now computes the standard f-tSE argument, signs it, and attaches the verification key vk to the public label. To verify, we first check that the signature is valid and then verify the f-tSE argument.

Finally, we say that an NIZK argument Π is *any-simultation f-extractable (f-aSE)* (similar to the notion of simulation-sound extractability of [31]) if the adversary P^* instead has access to a modified simulation oracle $\widetilde{\mathcal{SIM}}_{\mathrm{TK}}(\cdot)$ that responds to all simulation queries without checking that $R(x, y) = 1$ (and hence might also give simulated arguments of false statements). In this work we do not make use of this variation, but state it here because as we will see, this notion has been implicitly used in prior works. However, f-aSE is a stronger notion than f-tSE and is *not needed*, as we will show that f-tSE is sufficient in constructing leakage-resilient signatures and CCA-encryption.

4 Generic Constructions

In this section we give generic constructions of leakage-resilient hard relations, signatures, and CCA-secure encryption. In the latter two we use the f-tSE NIZK primitive that we defined in Section 3. Finally, we give a construction of f-tSE NIZK arguments.

LEAKAGE-RESILIENT HARD RELATIONS. We begin by showing how to generically construct leakage-resilient hard relations from SPR relations. Informally, we say that a relation R is *second-preimage resistant (SPR)* if given a random $(x, y) \in R$ it is difficult to find $x' \neq x$ such that $(x', y) \in R$. We formalize this in the following definition.

Definition 6 (Second-Preimage Resistant (SPR) Relation). *A relation R with a randomized PPT sampling algorithm* KeyGen *is* second-preimage resistant *if:*

- *For any $(x, y) \leftarrow \mathtt{KeyGen}(1^\lambda)$, we have $(x, y) \in R$.*
- *There is a poly-time algorithm that decides if $(x, y) \in R$.*

[6] In other words, the adversary wins if the extractor fails to extract a good value z^* which corresponds to at least one valid witness x'; i.e. $f(x') = z^*$. For the identity function, $f(x) = x$, this corresponds to the statement: $R(z^*, y) = 0$.

– *For any PPT algorithm* $\mathcal{A}$, $\Pr[(x', y) \in R \wedge x' \neq x] \leq negl(\lambda)$, *where the probability is taken over* $(x, y) \leftarrow \mathtt{KeyGen}(1^\lambda), x' \leftarrow \mathcal{A}(x, y)$.

We define the average-case pre-image entropy *of the SPR relation to be* $\mathbf{H}_{avg}(R) = \widetilde{\mathbf{H}}_\infty(X \mid Y)$ *, where random variables* (X, Y) *are distributed according to* $\mathtt{KeyGen}(1^\lambda)$. *(We refer the reader to the full version [18] for the definition of* $\widetilde{\mathbf{H}}_\infty(X \mid Y)$.*)*

Theorem 1. *If* $R(x, y)$ *is an SPR relation, then it is also an* ℓ*-leakage resilient hard relation for* $\ell = \mathbf{H}_{avg}(R) - \omega(\log \lambda)$*, where* λ *is the security parameter.*

LEAKAGE-RESILIENT SIGNATURES. We give a generic construction of leakage-resilient signatures based on leakage-resilient hard relations and tSE-NIZK arguments. Let $R(x, y)$ be an ℓ-leakage resilient hard relation with sampling algorithm $\mathtt{KeyGen}_R(1^\lambda)$. Let $\Pi = (\mathtt{Setup}, \mathtt{Prove}, \mathtt{Verify})$ be a tSE-NIZK argument for relation R supporting labels. Consider the following signature scheme:

– $\mathtt{KeyGen}(1^\lambda)$: Output $sk = x$ and $vk = (\mathrm{CRS}, y)$ where $(x, y) \leftarrow \mathtt{KeyGen}_R(1^\lambda)$, $(\mathrm{CRS}, \mathrm{TK}, \mathrm{EK}) \leftarrow \mathtt{Setup}(1^\lambda)$.
– $\mathtt{Sign}_{sk}(m)$: Output $\sigma = \varphi$ where $\varphi \leftarrow \mathtt{Prove}^m(x, y)$. (Note that m is the *label*.)
– $\mathtt{SigVer}_{vk}(m, \sigma)$: Output $\mathtt{Verify}^m(y, \sigma)$.

Theorem 2. *If* $R(x, y)$ *is an* ℓ*-leakage resilient hard relation and* Π *is a labeled tSE-NIZK argument for* R*, then the above scheme is an* ℓ*-leakage resilient signature scheme.*

LEAKAGE-RESILIENT CCA-SECURE ENCRYPTION. We give a generic construction of leakage-resilient CCA-secure encryption from leakage-resilient CPA-secure encryption and strong f-tSE NIZK arguments. Let $\mathcal{E} = (\mathtt{KeyGen}, \mathtt{Enc}, \mathtt{Dec})$ be an ℓ-LR-CPA secure encryption scheme and let $\Pi = (\mathtt{Setup}, \mathtt{Prove}, \mathtt{Verify})$ be a one-time strong f-tSE NIZK argument for the relation $R_{enc} = \{(\,(m, r), (pk, c)\,) \mid c = \mathtt{Enc}_{pk}(m; r)\}$. where $f(m, r) = m$ (i.e. the extractor only needs to extract the message m, but not the randomness r of encryption). We show how to use $\mathcal{E}, \Pi$ to construct an ℓ-LR-CCA encryption scheme $\mathcal{E}^*$. Define $\mathcal{E}^* = (\mathtt{KeyGen}^*, \mathtt{Enc}^*, \mathtt{Dec}^*)$ by:

– $\mathtt{KeyGen}^*(1^\lambda)$: Output $pk = (pk_0, \mathrm{CRS})$, $sk = sk_0$ where $(pk_0, sk_0) \leftarrow \mathtt{KeyGen}(1^\lambda)$, $(\mathrm{CRS}, \mathrm{TK}, \mathrm{EK}) \leftarrow \mathtt{Setup}(1^\lambda)$.
– $\mathtt{Enc}^*_{pk}(m; r)$: Output $C = (c, \pi)$ where $c \leftarrow \mathtt{Enc}_{pk_0}(m; r)$, $\pi \leftarrow \mathtt{Prove}_{\mathrm{CRS}}((pk_0, c), (m, r))$.
– $\mathtt{Dec}^*_{sk}(C)$: Parse $C = (c, \pi)$. If π verifies output $\mathtt{Dec}_{sk}(c)$, else output $\perp$.

Theorem 3. *Assume that* $\mathcal{E}$ *is* ℓ*-LR-CPA secure, and* Π *is a strong one-time* f*-tSE NIZK argument for the relation* R_{enc} *where, for any witness* (m, r)*, we define* $f(m, r) = m$*. Then the scheme* $\mathcal{E}^*$ *defined above is* ℓ*-LR-CCA secure.*

We note that if the tSE NIZK construction allows labels, then we can naturally extend our construction above to yield a ℓ-LR-CCA encryption *with labels*, by simply putting the encryption labels into the NIZK proofs (and using them to verify the proofs).

TRUE-SIMULATION f-EXTRACTABLE (f-TSE) NIZK. Let f be any efficiently computable function, and let $R(x, y)$ be an NP relation. We show how to construct an f-tSE NIZK argument Ψ from any labeled CCA-secure encryption scheme, and (standard)

NIZK arguments. Let $\mathcal{E} = (\mathtt{KeyGen}, \mathtt{Enc}, \mathtt{Dec})$ be a CCA-secure encryption scheme supporting labels, and let $\Pi = (\mathtt{Setup}_\Pi, \mathtt{Prove}_\Pi, \mathtt{Verify}_\Pi)$ be an NIZK argument for the relation $R_\Pi = \{(\ (x,r),(y,c,pk,L)\ ) \mid R(x,y) = 1 \wedge c = \mathtt{Enc}^L_{pk}(f(x);r)\}$. We define f-tSE NIZK argument Ψ (supporting labels) as follows:

- $\mathtt{Setup}(1^\lambda)$: Output $\mathrm{CRS} = (\mathrm{CRS}_\Pi, pk)$, $\mathrm{TK} = \mathrm{TK}_\Pi$, $\mathrm{EK} = sk$ where $(pk, sk) \leftarrow \mathtt{KeyGen}(1^\lambda)$, $(\mathrm{CRS}_\Pi, \mathrm{TK}_\Pi) \leftarrow \mathtt{Setup}_\Pi(1^\lambda)$.
- $\mathtt{Prove}^L(x, y; r)$: Output $\varphi = (c, \pi)$ where $c \leftarrow \mathtt{Enc}^L_{pk}(f(x); r)$, $\pi \leftarrow \mathtt{Prove}_\Pi((x,r),(y,c,pk,L))$.
- $\mathtt{Verify}^L(y, \varphi)$: Parse $\varphi = (c, \pi)$ and run $\mathtt{Verify}_\Pi((y,c,pk,L),\pi)$.

Theorem 4. *If $\mathcal{E}$ is a labeled CCA-secure encryption scheme and Π is an NIZK argument for relation R_Π, then Ψ is a f-tSE NIZK argument for relation R.*

COMPARISON OF OUR GENERIC CONSTRUCTIONS TO PRIOR WORK. The idea of using an SPR relation to construct leakage-resilient hard relations was implicit in [4,42], and explicitly described in [5] for the case of leakage-resilient one-way functions.

Our constructions of leakage-resilient CCA encryption and signatures from tSE NIZKs bear significant resemblance to prior constructions. In particular, we observe that an alternate construction of tSE NIZK could be achieved by using a CPA-encryption scheme instead of a CCA one, and a ss-NIZK argument system [56] instead of a standard one. In fact, the resulting construction would yield an *any*-simulation extractable (aSE) NIZK argument. This instantiation of aSE NIZKs is implicitly used by [42] in their construction of leakage-resilient signatures. It is also used implicitly in the Naor-Yung "double-decryption" paradigm [49,55,56,45] for CCA security, which was later used in [48] to construct leakage-resilient CCA-encryption. However, as we have seen, tSE is sufficient for constructing *both* leakage-resilient signatures and CCA-encryption and thus, the stronger notion of aSE is not needed. Furthermore, given the current state of efficient encryption schemes and NIZK, the difference in efficiency between ss-NIZK and standard NIZK is *significantly* greater than the difference between CCA and CPA-secure encryption[7], thus making tSE superior in both simplicity and efficiency.

We note that our construction of tSE NIZKs (based on CCA-encryption and standard NIZKs) was implicitly used by [31] to construct signatures of group elements, and by [11] to construct efficient CCA-encryption with key-dependent message (KDM) security from KDM-secure CPA-encryption. Still, the abstraction of tSE has not been explicitly defined in prior work despite its apparent usefulness.

5 Instantiations

ASSUMPTIONS. We review several standard hardness assumptions on which we will base our constructions.

Decisional Diffie-Hellman (DDH). Let $\mathbb{G}$ be a group of prime order q. Let $g_1, g_2 \xleftarrow{\$} \mathbb{G}$ and $r, r_1, r_2 \xleftarrow{\$} \mathbb{Z}_q$. The decisional Diffie-Hellman (DDH) assumption states that the

[7] Informally, the difference between CCA and CPA-secure encryption is only 2 group elements, whereas the size of a ss-NIZK proof is *more than twice* the size of a standard NIZK proof.

following two distributions are computationally indistinguishable: $(\mathbb{G}, g_1, g_2, g_1^{r_1}, g_2^{r_2})$ and $(\mathbb{G}, g_1, g_2, g_1^{r}, g_2^{r})$.

Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ be groups of prime order q and let $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ be a non-degenerate efficiently computable bilinear map.

Symmetric External Diffie-Hellman (SXDH) [59,9,6,26,61]. The symmetric external Diffie-Hellman assumption (SXDH) is that the DDH problem is hard in *both* $\mathbb{G}_1$ and $\mathbb{G}_2$. The assumption is clearly invalid for symmetric pairings (when $\mathbb{G}_1 = \mathbb{G}_2$), but is believed to hold when there is no efficiently computable mapping between $\mathbb{G}_1$ and $\mathbb{G}_2$.

K-Linear [37,60] and DLIN [9]. Let $\mathbb{G}$ be a group of primer order q and let $K \geq 1$ be constant. Let $g_0, g_1, \ldots, g_K \xleftarrow{\$} \mathbb{G}$ and $x_0, x_2, \ldots, x_K \xleftarrow{\$} \mathbb{Z}_q$. The K-Linear assumption states that the following two distributions are computationally indistinguishable: $(\mathbb{G}, g_0, g_1, \ldots, g_K, g_1^{x_1}, \ldots, g_K^{x_K}, g_0^{x_0})$, and $(\mathbb{G}, g_0, g_1, \ldots, g_K, g_1^{x_1}, \ldots, g_K^{x_K}, g_0^{X})$, with $X = \sum_{i=1}^{K} x_i$. Note that for $K = 1$, the K-Linear is the same as DDH, and that it does not hold when working with symmetric pairings. In that setting, the 2-Linear assumption is usually assumed to hold, and is often referred to as the Decisional Linear (DLIN) assumption. *Throughout this paper we assume the K-Linear assumption holds in both $\mathbb{G}_1$ and $\mathbb{G}_2$, which is the case when working with symmetric pairings, and slightly abuse notation when $K = 1$ and assume SXDH holds in that case.*

OUR INSTANTIATIONS. We show efficient instantiations of the leakage-resilient signature and CCA-secure encryption constructions described in Section 4. For each scheme, we give two instantiations based on bilinear maps: one secure under the SXDH assumption, and a second, secure under the DLIN assumption. The first can be used with asymmetric pairings, while the second applies to the case of symmetric pairings. We give details of all instantiations in the full version [18] but give a high-level idea below.

Signatures. Recall that to instantiate the signature scheme from Section 4, we need a leakage-resilient hard relation R (which we will derive from an SPR relation) and a true-simulation extractable (tSE) NIZK argument, which we build from CCA-secure encryption and a standard NIZK argument for the relation $\{((x, r), (y, c, pk, L)) \,|R(x, y) = 1 \wedge c = \mathtt{Enc}_{pk}^{L}(f(x); r)\}$. We show our choice of instantiations for these components:

- *CCA-Secure Encryption:* Under both the SXDH and DLIN assumptions, we use efficient encryption schemes in the style of Cramer-Shoup [14,60].
- *NIZK Argument:* We use the Groth-Sahai proof system [33], which can be instantiated both under SXDH and DLIN.
- *SPR Relation:* Previous constructions of leakage-resilient primitives use the SPR function $g_1^{x_1} g_2^{x_2} \ldots g_n^{x_n}$. However, this function has the problem that the witness lies in the exponent. This means that we cannot combine it with an encryption scheme for elements in $\mathbb{G}$ (unless each witness component is committed bit by bit which, among other things, results in proofs growing linearly with the security parameter), and unfortunately encryption schemes for messages in $\mathbb{Z}_q$ cannot be combined with the Groth-Sahai system. We therefore construct two new SPR relations based on pairing-product equations. For our SXDH instantiation, we use the relation $e(h_1, x_1)\ e(h_2, x_2) \ldots e(h_n, x_n) = e(y, \tilde{g})$, where $\tilde{g}$ is a generator of $\mathbb{G}_2$. We prove that this relation is SPR under the SXDH assumption. In the DLIN case, we

use the relation: $e(h_1, x_1)\ e(h_2, x_2) \ldots e(h_n, x_n) = e(y_1, g)\ ,\ e(\hbar_1, x_1)\ e(\hbar_2, x_2) \ldots e(\hbar_n, x_n) = e(y_2, g)$, where g is a generator of $\mathbb{G}$. We prove that this relation is SPR under the DLIN assumption. To achieve a $(1 - \epsilon)$ leakage ratio, we let n (the number of witness components) in the SPR relation be inversely proportional to ϵ.

Theorem 5. *Let $\mathbb{G}_1, \mathbb{G}_2$ be groups of primer order q. For any $\epsilon > 0$, there exists a $(1 - \epsilon)|sk|$-leakage resilient signature scheme, secure under the SXDH assumption, using signatures consisting of $(9/\epsilon)(1 + \omega(\log \lambda)/\log q) + 24$ group elements and 2 elements in $\mathbb{Z}_q$. Similarly, for any $\epsilon > 0$, there exists a $(1 - \epsilon)|sk|$-leakage resilient signature scheme, secure under the DLIN assumption, using signatures consisting of $(19/\epsilon)(2 + \omega(\log \lambda)/\log q) + 70$ group elements and 6 elements in $\mathbb{Z}_q$.*

CCA-Secure Encryption. Recall that for leakage-resilient encryption, we need leakage-resilient CPA-secure encryption, standard CCA-secure encryption and strong tSE NIZK, which we can get from combining regular tSE NIZK with a strong one-time signature. We build regular tSE NIZK from CCA-secure encryption and regular NIZK. We describe our choices for each of these below.

- *LR-CPA-Secure Encryption:* We construct a new leakage-resilient CPA-secure encryption scheme for our purpose in the style of ElGamal (similar to ones used in [48,11] but making it more efficient). The leakage that our new CCA-secure encryption tolerates is the same as the leakage tolerated by the CPA-secure scheme. Informally, we achieve a $(1-\epsilon)$ leakage ratio in the CPA-secure scheme by increasing the number of generators used in the public key and ciphertext. This number will be inversely proportional to ϵ.
- *CCA-Secure Encryption:* Under both the SXDH and DLIN assumptions, we use efficient encryption schemes in the style of Cramer-Shoup [14,60].
- *NIZK Argument:* We use the Groth-Sahai proof system [33], which can be instantiated both under SXDH and DLIN.
- *One-Time Signature:* We observe that *any* strong one-time signature secure under these assumptions can be used. Here, we opt for the scheme of [31], secure under the Discrete Log assumption (implied by both SDXH and DLIN), because its signature size is small, namely 2 elements in $\mathbb{Z}_q$.

Theorem 6. *Let $\mathbb{G}_1, \mathbb{G}_2$ be groups of primer order q. For any $\epsilon > 0$, there exists a $(1 - \epsilon)|sk|$-leakage resilient encryption scheme, secure under the SXDH assumption, using ciphertexts consisting of $(2/\epsilon)(2 + \lambda/\log q) + 15$ group elements and 2 elements in $\mathbb{Z}_q$. Similarly, for any $\epsilon > 0$, there exists a $(1 - \epsilon)|sk|$-leakage resilient encryption scheme, secure under the DLIN assumption, using ciphertexts consisting of $(3/\epsilon)(3 + \lambda/\log q) + 34$ group elements and 2 elements in $\mathbb{Z}_q$.*

6 Other Applications

Once we have efficient leakage-flexible signature schemes, we observe that the standard signature-based ID scheme, where the verifier asks the prover to sign a random message, easily extends to the leakage setting. Moreover, the resulting actively secure ID scheme inherits its relative leakage from the corresponding signature scheme, and satisfies the

strongest notion of "anytime-leakage" [4], where the leakage can occur even during the impersonation attack. Although our method is pretty simple, we notice that the other two popular methods of building ID schemes — the use of Σ-protocols for hard relations analyzed in [4] (see first two rows of Tables 3), and the use of CCA-secure encryption (where the prover decrypts a random challenge ciphertext) — inherently do not allow us to obtain optimal results, even when instantiated with leakage-flexible hard relations or CCA-encryption schemes.

Finally, we obtain two efficient leakage-flexible AKA protocols. First, similarly to the case of ID schemes, we can obtain leakage-resilient AKA schemes from any leakage-resilient signature scheme, as formally explained in [4]. The idea is to essentially sign every flow of a standard Diffie-Hellman-based protocol, but with a leakage-resilient signature scheme. We notice, though, that the resulting protocol is not *deniable*. Namely, the transcript of the protocol leaves irrefutable evidence that the protocol took place. Motivated by this deficiency, we design another general AKA protocol based on CCA-encryption. The details are given in the full version [18], but, intuitively, the parties encrypt the flows of the standard Diffie-Hellman-based protocol, effectively proving their identities by successfully re-encrypting the appropriate flows. Although we do not formalize this, this protocols is "deniable", because the transcript of the protocol can be simulated without the knowledge of parties' secret keys. To the best of our knowledge, this protocol was not suggested and analyzed even in the leakage-free setting, where it appears interesting already. Here we actually show that our (new) deniable AKA protocol works even in the presence of leakage.

Aknowledgments

We would like to thank Victor Shoup for helpful discussions, as well as the anonymous reviewers for their useful suggestions.

References

1. In: Proceedings of 44th Symposium on Foundations of Computer Science (FOCS 2003), IEEE Computer Society, Los Alamitos (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. Gilbert [28], pp. 113–134 (2010)
4. Alwen, J., Dodis, Y., Wichs, D.: Leakage-resilient public-key cryptography in the bounded-retrieval model. Halevi [35], pp. 36–54 (2009)
5. Alwen, J., Dodis, Y., Wichs, D.: Survey: Leakage resilience and the bounded retrieval model. In: Kurosawa, K. (ed.) Information Theoretic Security. LNCS, vol. 5973, pp. 1–18. Springer, Heidelberg (2010)
6. Ballard, L., Green, M., de Medeiros, B., Monrose, F.: Correlation-resistant storage via keyword-searchable encryption. Cryptology ePrint Archive, Report 2005/417 (2005)
7. Biham, E., Shamir, A.: Differential fault analysis of secret key cryptosystems. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 513–525. Springer, Heidelberg (1997)

8. Blum, M., Feldman, P., Micali, S.: Non-interactive zero-knowledge and its applications (extended abstract). In: STOC, pp. 103–112. ACM, New York (1988)
9. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)
10. Boneh, D., DeMillo, R.A., Lipton, R.J.: On the importance of checking cryptographic protocols for faults (extended abstract). In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 37–51. Springer, Heidelberg (1997)
11. Camenisch, J., Chandran, N., Shoup, V.: A public key encryption scheme secure against key dependent chosen plaintext and adaptive chosen ciphertext attacks. Joux [39], pp. 351–368 (2009)
12. Canetti, R., Dodis, Y., Halevi, S., Kushilevitz, E., Sahai, A.: Exposure-resilient functions and all-or-nothing transforms. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 453–469. Springer, Heidelberg (2000)
13. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: STOC, pp. 494–503 (2002)
14. 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)
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. Halevi and Rabin [36], pp. 225–244 (2006)
17. Dodis, Y., Goldwasser, S., Kalai, Y.T., Peikert, C., Vaikuntanathan, V.: Public-key encryption schemes with auxiliary inputs. Micciancio [47], pp. 361–381 (2010)
18. Dodis, Y., Haralambiev, K., Lopez-Alt, A., Wichs, D.: Efficient public-key cryptography in the presence of key leakage. Cryptology ePrint Archive, Report 2010/154 (2010)
19. Dodis, Y., Kalai, Y.T., Lovett, S.: On cryptography with auxiliary input. In: Mitzenmacher, M. (ed.) STOC, pp. 621–630. ACM, New York (2009)
20. Dodis, Y., Sahai, A., Smith, A.: On perfect and adaptive security in exposure-resilient cryptography. Pfitzmann [52], pp. 301–324 (2001)
21. Dziembowski, S.: Intrusion-resilience via the bounded-storage model. Halevi and Rabin [36], pp. 207–224 (2006)
22. Dziembowski, S., Pietrzak, K.: Leakage-resilient cryptography. In: FOCS, pp. 293–302. IEEE Computer Society, Los Alamitos (2008)
23. Faust, S., Kiltz, E., Pietrzak, K., Rothblum, G.N.: Leakage-resilient signatures. Micciancio [47], pp. 343–360 (2010)
24. Faust, S., Rabin, T., Reyzin, L., Tromer, E., Vaikuntanathan, V.: Protecting circuits from leakage: the computationally-bounded and noisy cases. Gilbert [28], pp. 135–156 (2010)
25. 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)
26. Galbraith, S.D., Rotger, V.: Easy decision-diffie-hellman groups. LMS Journal of Computation and Mathematics 7 (2004)
27. 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)
28. Gilbert, H. (ed.): EUROCRYPT 2010. LNCS, vol. 6110. Springer, Heidelberg (2010)
29. Goldwasser, S., Kalai, Y., Peikert, C., Vaikuntanathan, V.: Robustness of the learning with errors assumption. In: Innovations in Computer Science, ICS (2010)

30. Goldwasser, S., Kalai, Y.T.: On the (in)security of the fiat-shamir paradigm. FOCS [1], p. 102 (2003)
31. Groth, J.: Simulation-sound nizk proofs for a practical language and constant size group signatures. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 444–459. Springer, Heidelberg (2006)
32. Groth, J.: Fully anonymous group signatures without random oracles. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 164–180. Springer, Heidelberg (2007)
33. 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)
34. 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. ACM Commun. 52(5), 91–98 (2009)
35. Halevi, S. (ed.): CRYPTO 2009. LNCS, vol. 5677. Springer, Heidelberg (2009)
36. Halevi, S., Rabin, T. (eds.): TCC 2006. LNCS, vol. 3876. Springer, Heidelberg (2006)
37. 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)
38. 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)
39. Joux, A. (ed.): EUROCRYPT 2009. LNCS, vol. 5479. Springer, Heidelberg (2009)
40. Juma, A., Vahlis, Y.: Protecting cryptographic keys against continual leakage. In: Rabin, T. (ed.) Advances in Cryptology – CRYPTO 2010. LNCS, vol. 6223, pp. 41–58. Springer, Heidelberg (2010)
41. Kamp, J., Zuckerman, D.: Deterministic extractors for bit-fixing sources and exposure-resilient cryptography. FOCS [1], pp. 92–101 (2003)
42. 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)
43. 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)
44. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M.J. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
45. Lindell, Y.: A simpler construction of cca2-secure public-keyencryption under general assumptions. J. Cryptology 19(3), 359–377 (2006)
46. Micali, S., Reyzin, L.: Physically observable cryptography (extended abstract). In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 278–296. Springer, Heidelberg (2004)
47. Micciancio, D. (ed.): TCC 2010. LNCS, vol. 5978. Springer, Heidelberg (2010)
48. Naor, M., Segev, G.: Public-key cryptosystems resilient to key leakage. Halevi [35], pp. 18–35 (2009)
49. Naor, M., Yung, M.: Public-key cryptosystems provably secure against chosen ciphertext attacks. In: STOC, pp. 427–437. ACM, New York (1990)
50. Pass, R., Rosen, A.: Concurrent non-malleable commitments. In: FOCS, pp. 563–572. IEEE Computer Society, Los Alamitos (2005)
51. Pass, R., Rosen, A.: New and improved constructions of non-malleable cryptographic protocols. In: Gabow, H.N., Fagin, R. (eds.) STOC, pp. 533–542. ACM, New York (2005)
52. Pfitzmann, B. (ed.): EUROCRYPT 2001. LNCS, vol. 2045. Springer, Heidelberg (2001)
53. Pietrzak, K.: A leakage-resilient mode of operation. Joux [39], pp. 462–482 (2009)
54. Quisquater, J.-J., Samyde, D.: Electromagnetic analysis (ema): Measures and countermeasures for smart cards. In: Attali, I., Jensen, T.P. (eds.) E-smart 2001. LNCS, vol. 2140, pp. 200–210. Springer, Heidelberg (2001)

55. 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)
56. Sahai, A.: Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In: FOCS, pp. 543–553 (1999)
57. Santis, A.D., Crescenzo, G.D., Ostrovsky, R., Persiano, G., Sahai, A.: Robust non-interactive zero knowledge. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 566–598. Springer, Heidelberg (2001)
58. Santis, A.D., Persiano, G.: Zero-knowledge proofs of knowledge without interaction (extended abstract). In: FOCS, pp. 427–436. IEEE, Los Alamitos (1992)
59. Scott, M.: Authenticated id-based key exchange and remote log-in with simple token and pin number. Cryptology ePrint Archive, Report 2002/164 (2002)
60. Shacham, H.: A cramer-shoup encryption scheme from the linear assumption and from progressively weaker linear variants. Cryptology ePrint Archive, Report 2007/074 (2007)
61. Verheul, E.R.: Evidence that xtr is more secure than supersingular elliptic curve cryptosystems. J. Cryptology 17(4), 277–296 (2004)

Author Index