

Mitsuru Matsui (Ed.)

LNCS 5912

Advances in Cryptology – ASIACRYPT 2009

15th International Conference on the Theory
and Application of Cryptology and Information Security
Tokyo, Japan, December 2009, Proceedings



 Springer

Commenced Publication in 1973

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison

Lancaster University, UK

Takeo Kanade

Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler

University of Surrey, Guildford, UK

Jon M. Kleinberg

Cornell University, Ithaca, NY, USA

Alfred Kobsa

University of California, Irvine, CA, USA

Friedemann Mattern

ETH Zurich, Switzerland

John C. Mitchell

Stanford University, CA, USA

Moni Naor

Weizmann Institute of Science, Rehovot, Israel

Oscar Nierstrasz

University of Bern, Switzerland

C. Pandu Rangan

Indian Institute of Technology, Madras, India

Bernhard Steffen

University of Dortmund, Germany

Madhu Sudan

Microsoft Research, Cambridge, MA, USA

Demetri Terzopoulos

University of California, Los Angeles, CA, USA

Doug Tygar

University of California, Berkeley, CA, USA

Gerhard Weikum

Max-Planck Institute of Computer Science, Saarbruecken, Germany

Mitsuru Matsui (Ed.)

Advances in Cryptology – ASIACRYPT 2009

15th International Conference on the Theory
and Application of Cryptology and Information Security
Tokyo, Japan, December 6-10, 2009
Proceedings



Springer

Volume Editor

Mitsuru Matsui

Information Technology R&D Center, Mitsubishi Electric Corporation

Kamakura, Kanagawa, 247-8501, Japan

E-mail: Matsui.Mitsuru@ab.MitsubishiElectric.co.jp

Library of Congress Control Number: 2009938634

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

LNCS Sublibrary: SL 4 – Security and Cryptology

ISSN 0302-9743

ISBN-10 3-642-10365-0 Springer Berlin Heidelberg New York

ISBN-13 978-3-642-10365-0 Springer Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

springer.com

© Springer-Verlag Berlin Heidelberg 2009

Printed in Germany

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

Printed on acid-free paper SPIN: 12796120 06/3180 5 4 3 2 1 0

Preface

ASIACRYPT 2009, the 15th International Conference on the Theory and Application of Cryptology and Information Security was held in Tokyo, Japan, during December 6–10, 2009. The conference was sponsored by the International Association for Cryptologic Research (IACR) in cooperation with the Technical Group on Information Security (ISEC) of the Institute of Electronics, Information and Communication Engineers (IEICE). ASIACRYPT 2009 was chaired by Eiji Okamoto and I had the honor of serving as the Program Chair.

The conference received 300 submissions from which two papers were withdrawn. Each paper was assigned at least three reviewers, and papers co-authored by Program Committee members were assigned at least five reviewers. We spent eight weeks for the review process, which consisted of two stages. In the first four-week stage, each Program Committee member individually read and evaluated assigned papers (individual review phase), and in the second four-week stage, the papers were scrutinized with an extensive discussion (discussion phase). The review reports and discussion comments reached a total of 50,000 lines.

Finally, the Program Committee decided to accept 42 submissions, of which two submissions were merged into one paper. As a result, 41 presentations were given at the conference. The authors of the accepted papers had four weeks to prepare final versions for these proceedings. These revised papers were not subject to editorial review and the authors bear full responsibility for their contents. Unfortunately there were a number of good papers that could not be included in the program due to this year's tough competition.

Tatsuaki Okamoto delivered the 2009 IACR Distinguished Lecture. The Program Committee decided to give the Best Paper Award of ASIACRYPT 2009 to the following paper: “Improved Generic Algorithms for 3-Collisions” by Antoine Joux and Stefan Lucks. They received an invitation to submit a full version to the *Journal of Cryptology*. In addition to the papers included in this volume, the conference also featured a rump session, a forum for short and entertaining presentations on recent works of both a technical and non-technical nature.

There are many people who contributed to the success of ASIACRYPT 2009. First I would like to thank all authors for submitting their papers to the conference. I am deeply grateful to the Program Committee for giving their time, expertise and enthusiasm in order to ensure that each paper received a thorough and fair review. Thanks also to 303 external reviewers, listed on the following pages, for contributing their time and expertise. Finally, I would like to thank Shai Halevi for maintaining his excellent Web Submission and Review Software. Without this system, which covers all processes from paper submission to preparation of the proceedings, I could not have handled 300 papers so smoothly.

ASIACRYPT 2009

December 6–10, 2009, Tokyo, Japan

Sponsored by

The International Association for Cryptologic Research

in cooperation with the

Technical Group on Information Security (ISEC) of the Institute of
Electronics, Information and Communication Engineers (IEICE)

General Chair

Eiji Okamoto University of Tsukuba, Japan

Program Chair

Mitsuru Matsui Mitsubishi Electric Corporation

Program Committee

Masayuki Abe	NTT, Japan
Josh Benaloh	Microsoft Research, USA
Daniel J. Bernstein	University of Illinois at Chicago, USA
Xavier Boyen	Stanford, USA
Claude Carlet	University of Paris 8, France
Kim-Kwang Raymond Choo	Australian Institute of Criminology, and ARC Center of Excellence in Policing and Security / Australian National University, Australia
Claus Diem	University of Leipzig, Germany
Stefan Dziembowski	University of Rome “La Sapienza”, Italy
Serge Fehr	CWI, The Netherlands
Jun Furukawa	NEC Corporation, Japan
Henri Gilbert	Orange Labs, France
Jens Groth	University College London, UK
Shai Halevi	IBM, USA
Goichiro Hanaoka	AIST, Japan
Helena Handschuh	Katholieke Universiteit Leuven, Belgium
Tetsu Iwata	Nagoya University, Japan
Thomas Johansson	Lund University, Sweden
Marc Joye	Thomson R&D, France
Lars R. Knudsen	DTU Mathematics, Denmark
Xuejia Lai	Shanghai Jiao Tong University, China

Dong Hoon Lee	Korea University , Korea
Arjen Lenstra	Ecole Polytechnique Fédérale de Lausanne, Switzerland, and Alcatel-Lucent Bell Laboratories, USA
Keith Martin	Royal Holloway, University of London, UK
Phong Nguyen	INRIA and ENS, France
Kaisa Nyberg	Helsinki University of Technology and Nokia, Finland
Elisabeth Oswald	University of Bristol, UK
Pascal Paillier	Gemalto Security Labs, France
Josef Pieprzyk	Macquarie University, Australia
David Pointcheval	ENS, CNRS and INRIA, France
Manoj Prabhakaran	University of Illinois at Urbana-Champaign, USA
Bart Preneel	Katholieke Universiteit Leuven, Belgium
Vincent Rijmen	Katholieke Universiteit Leuven, Belgium and Graz University of Technology, Austria
Phillip Rogaway	University of California, Davis, USA
Rei Safavi-Naini	University of Calgary, Canada
Berry Schoenmakers	TU Eindhoven, The Netherlands
Francois-Xavier Standaert	Université catholique de Louvain, Belgium
Serge Vaudenay	Ecole Polytechnique Fédérale de Lausanne, Switzerland
Ivan Visconti	University of Salerno, Italy

External Reviewers

Michel Abdalla	Jean-Luc Beuchat	Nishanth Chandran
Hadi Ahmadi	Olivier Billet	Melissa Chase
Martin Albrecht	Carlo Blundo	Jianhong Chen
Davide Alessio	Julia Borghoff	Jung Hee Cheon
Joel Francis Alwen	Joppe Bos	Céline Chevalier
Elena Andreeva	Arnaud Boscher	Joo Yeon Cho
Prasanth	Charles Bouillaguet	Kwan Tae Cho
Anthapadmanabhan	Niek Bouman	Kyu Young Choi
Toshinori Araki	Christina Boura	Sherman S. M. Chow
Mina Askari	Emmanuel Bresson	Ji Young Chun
Gilles Van Assche	Billy Bob Brumley	Carlos Cid
Nuttapong Attrapadung	Jin Wook Byun	Christophe Clavier
Man Ho Au	Jan Camenisch	Baudoin Collard
Joonsang Baek	Sébastien Canard	Jean-Sebastien Coron
Aurélie Bauer	Dario Catalano	Nicolas Courtois
Mira Belenkiy	Julien Cathalo	Yang Cui
Mihir Bellare	Pierre-Louis Cayrel	Paolo D'Arco
Thierry Berger	Rafik Chaabouni	Joan Daemen

Ed Dawson	Dennis Hofheinz	Mario Lamberger
Cécile Delerablée	Susan Hohenberger	Tanja Lange
Yi Deng	Xuan Hong	Sven Laur
Alex Dent	Sebastiaan de Hoogh	Gregor Leander
Jérémie Detrey	Nick Howgrave-Graham	Ji-Seon Lee
Cunsheng Ding	Alex Huang	Hyung Tae Lee
Jintai Ding	Qiong Huang	Kwangsue Lee
Ning Ding	Jung Yeon Hwang	Gaëtan Leurent
Christophe Doche	Toshiyuki Isshiki	Wei Li
Yevgeniy Dodis	Thomas Icart	Benoit Libert
Ming Duan	Sebastiaan Indestege	Peter van Liesdonk
Alfredo Rial Duran	Vincenzo Iovino	Huijia Lin
Xiwen Fang	Yuval Ishai	Yehuda Lindell
Sebastian Faust	Malika Izabachčne	Helger Lipmaa
Anna Lisa Ferrara	Stanislaw Jarecki	Moses Liskov
Pierre-Alain Fouque	Ik Rae Jeong	Yu Long
Georg Fuchsbauer	Ellen Jochemsz	Steve Lu
Thomas Fuhr	Otto Johnston	Xianhui Lu
Eiichiro Fujisaki	Antoine Joux	Yi Lu
Guillaume Fumaroli	Jorge Nakahara Jr.	Yiyuan Luo
Teddy Furon	Pascal Junod	Yuan Luo
Martin Gagné	Kimmo Järvinen	Anna Lysyanskaya
Sebastian Gajek	Emilia Käsper	Vadim Lyubashevsky
Steven Galbraith	Özgül Küçük	Ducas Léo
Nicolas Gama	Marcelo Kaihara	Kay Maggaard
Juan Garay	Bhavana Kanukurthi	Gilles Macario-Rat
Pierrick Gaudry	Alexandre Karlov	Hemanta Maji
Praveen Gauravaram	Mohamed Karroumi	Stefan Mangard
Rosario Gennaro	Jonathan Katz	Stéphane Manuel
Hossein Ghodosi	Shahram Khazaei	Mark Manulis
Benedikt Gierlichs	Aggelos Kiayias	Atefeh Mashatan
Marc Girault	Eike Kiltz	Krystian Matusiewicz
Zheng Gong	Bum Han Kim	Alexander May
Louis Goubin	Chong Hee Kim	James McKee
Aline Gouget	Jihye Kim	Wilfried Meidl
Vipul Goyal	Ki Tak Kim	Florian Mendel
Gaurav Gupta	Thorsten Kleinjung	Alfred Menezes
Robbert de Haan	Francois Koeune	Kazuhiko Minematsu
Risto Hakala	Yuichi Komano	Kunihiko Miyazaki
Kristijan Haralambiev	Woo Kwon Koo	Kirill Morozov
Martin Hell	Hugo Krawczyk	Nicky Mouha
Jens Hermans	Ted Krovetz	Ciaran Mullan
Miia Hermelin	Kaoru Kurosawa	Tomislav Nad
Clemens Heuberger	Jeong Ok Kwon	Kris Narayan
Viet Tung Hoang	Fabien Laguillaumie	Anderson Nascimento

Jesper Buus Nielsen	Tom Ristenpart	Jean-Pierre Tillich
Svetla Nikova	Matthew Robshaw	Elmar Tischhauser
Ryo Nishimaki	Markku-Juhani O.	
Adam O'Neill	Saarinen	Deniz Toz
Satoshi Obana	Bagus Santoso	Jacques Traoré
Miyako Ohkubo		Elena Trichina
Tatsuaki Okamoto	Juraj Sarinay	Eran Tromer
Claudio Orlandi	Palash Sarkar	Ashraful Tuhin
Dag Arne Osvik	Takakazu Satoh	Michael Tunstall
Khaled Ouafi	Ruediger Schack	Dominique Unruh
Onur Ozen	Martin Schlaeffler	Vinod Vaikuntanathan
Dan Page	Thomas Schneider	Frédéric Valette
Omkant Pandey	Daniel Schreiber	Jukka Valkonen
Hyun-A Park	Mike Scott	Kerem Varici
Jong Hwan Park	Gautham Sekar	Carmine Ventre
Vijayakrishnan	Reza Sepahi	Frederik Vercauteren
Pasupathinathan	Pouyan Sepehrdad	Damien Vergnaud
Kenny Paterson	Yannick Seurin	Nicolas Veyrat-Charvillon
Maura Paterson	Abhi Shelat	José Villegas
Christopher J. Peikert	Emily Shen	Nguyen Vo
Sylvain Pelissier	Elaine Shi	Martin Vuagnoux
Olivier Pereira	Hongsong Shi	Christian Wachsmann
Ludovic Perret	SeongHan Shin	Zhongmei Wan
Giuseppe Persiano	Tom Shrimpton	Peishun Wang
Christiane Peters	Alice Silverberg	Shengbao Wang
Thomas Peyrin	Thomas Sirvent	Brent Waters
Duong Hieu Phan	Boris Skoric	Hoeteck Wee
Laurent Philippe	Nigel Smart	Benne de Weger
Krzysztof Pietrzak	Adam Smith	Jian Weng
Benny Pinkas	Kannan Srinathan	Christopher Wolf
Vinod Prabhakaran	Colin Stahlke	Ronald de Wolf
Christine Priplata	John Steinberger	Bo-Yin Yang
Bartosz Przydatek	Ron Steinfeld	Kan Yasuda
Tal Rabin	Marc Stevens	Qingsong Ye
Mariana Raykova	Xiaorui Sun	Yu Yu
Christian Rechberger	Koutarou Suzuki	Erik Zenner
Francesco Regazzoni	Christophe Tartary	Rui Zhang
Mathieu Renaud	Joseph Chee Ming Teo	Jinmin Zhong
Leonid Reyzin	Isamu Teranishi	Hong-Sheng Zhou
Hyun Sook Rhee	Soeren S. Thomsen	

Table of Contents

Block Ciphers

Related-Key Cryptanalysis of the Full AES-192 and AES-256	1
<i>Alex Biryukov and Dmitry Khovratovich</i>	
The Key-Dependent Attack on Block Ciphers	19
<i>Xiaorui Sun and Xuejia Lai</i>	
Cascade Encryption Revisited	37
<i>Peter Gaži and Ueli Maurer</i>	

Quantum and Post-Quantum

Quantum-Secure Coin-Flipping and Applications	52
<i>Ivan Damgård and Carolin Lunemann</i>	
On the Power of Two-Party Quantum Cryptography	70
<i>Louis Salvail, Christian Schaffner, and Miroslava Sotáková</i>	
Security Bounds for the Design of Code-Based Cryptosystems	88
<i>Matthieu Finiasz and Nicolas Sendrier</i>	

Hash Functions I

Rebound Attack on the Full LANE Compression Function	106
<i>Krystian Matusiewicz, María Naya-Plasencia, Ivica Nikolić, Yu Sasaki, and Martin Schläffer</i>	
Rebound Distinguishers: Results on the Full Whirlpool Compression Function	126
<i>Mario Lamberger, Florian Mendel, Christian Rechberger, Vincent Rijmen, and Martin Schläffer</i>	
MD5 Is Weaker Than Weak: Attacks on Concatenated Combiners	144
<i>Florian Mendel, Christian Rechberger, and Martin Schläffer</i>	
The Intel AES Instructions Set and the SHA-3 Candidates	162
<i>Ryad Benadjila, Olivier Billet, Shay Gueron, and Matt J.B. Robshaw</i>	

Encryption Schemes

Group Encryption: Non-interactive Realization in the Standard Model	179
<i>Julien Cathalo, Benoît Libert, and Moti Yung</i>	
On Black-Box Constructions of Predicate Encryption from Trapdoor Permutations	197
<i>Jonathan Katz and Arkady Yerukhimovich</i>	
Hierarchical Predicate Encryption for Inner-Products	214
<i>Tatsuaki Okamoto and Katsuyuki Takashima</i>	
Hedged Public-Key Encryption: How to Protect against Bad Randomness	232
<i>Mihir Bellare, Zvika Brakerski, Moni Naor, Thomas Ristenpart, Gil Segev, Hovav Shacham, and Scott Yilek</i>	

Multi Party Computation

Secure Two-Party Computation Is Practical	250
<i>Benny Pinkas, Thomas Schneider, Nigel P. Smart, and Stephen C. Williams</i>	
Secure Multi-party Computation Minimizing Online Rounds	268
<i>Seung Geol Choi, Ariel Elbaz, Tal Malkin, and Moti Yung</i>	
Improved Non-committing Encryption with Applications to Adaptively Secure Protocols	287
<i>Seung Geol Choi, Dana Dachman-Soled, Tal Malkin, and Hoeteck Wee</i>	

Cryptographic Protocols

Non-malleable Statistically Hiding Commitment from Any One-Way Function	303
<i>Zongyang Zhang, Zhenfu Cao, Ning Ding, and Rong Ma</i>	
Proofs of Storage from Homomorphic Identification Protocols	319
<i>Giuseppe Ateniese, Seny Kamara, and Jonathan Katz</i>	
Simple Adaptive Oblivious Transfer without Random Oracle	334
<i>Kaoru Kurosawa and Ryo Nojima</i>	

Hash Functions II

Improved Generic Algorithms for 3-Collisions	347
<i>Antoine Joux and Stefan Lucks</i>	

A Modular Design for Hash Functions: Towards Making the Mix-Compress-Mix Approach Practical	364
<i>Anja Lehmann and Stefano Tessaro</i>	

How to Confirm Cryptosystems Security: The Original Merkle-Damgård Is Still Alive!	382
<i>Yusuke Naito, Kazuki Yoneyama, Lei Wang, and Kazuo Ohta</i>	

Models and Frameworks I

On the Analysis of Cryptographic Assumptions in the Generic Ring Model	399
<i>Tibor Jager and Jörg Schwenk</i>	

Zero Knowledge in the Random Oracle Model, Revisited	417
<i>Hoeteck Wee</i>	

A Framework for Universally Composable Non-committing Blind Signatures	435
<i>Masayuki Abe and Miyako Ohkubo</i>	

Cryptanalysis: Square and Quadratic

Cryptanalysis of the Square Cryptosystems	451
<i>Olivier Billet and Gilles Macario-Rat</i>	

Factoring pq^2 with Quadratic Forms: Nice Cryptanalyses	469
<i>Guilhem Castagnos, Antoine Joux, Fabien Laguillaumie, and Phong Q. Nguyen</i>	

Attacking Power Generators Using Unravelling Linearization: When Do We Output Too Much?	487
<i>Mathias Herrmann and Alexander May</i>	

Models and Frameworks II

Security Notions and Generic Constructions for Client Puzzles	505
<i>Liqun Chen, Paul Morrissey, Nigel P. Smart, and Bogdan Warinschi</i>	

Foundations of Non-malleable Hash and One-Way Functions	524
<i>Alexandra Boldyreva, David Cash, Marc Fischlin, and Bogdan Warinschi</i>	

Hash Functions III

Improved Cryptanalysis of Skein	542
<i>Jean-Philippe Aumasson, Çağdaş Çalk, Willi Meier, Onur Özen, Raphael C.-W. Phan, and Kerem Varici</i>	

Linearization Framework for Collision Attacks: Application to
 CubeHash and MD6 560
Eric Brier, Shahram Khazaei, Willi Meier, and Thomas Peyrin

Preimages for Step-Reduced SHA-2 578
*Kazumaro Aoki, Jian Guo, Krystian Matusiewicz, Yu Sasaki, and
 Lei Wang*

Lattice-Based

Fiat-Shamir with Aborts: Applications to Lattice and Factoring-Based
 Signatures 598
Vadim Lyubashevsky

Efficient Public Key Encryption Based on Ideal Lattices
 (Extended Abstract) 617
Damien Stehlé, Ron Steinfeld, Keisuke Tanaka, and Keita Xagawa

Smooth Projective Hashing and Password-Based Authenticated Key
 Exchange from Lattices 636
Jonathan Katz and Vinod Vaikuntanathan

Side Channels

PSS Is Secure against Random Fault Attacks 653
Jean-Sébastien Coron and Avradip Mandal

Cache-Timing Template Attacks 667
Billy Bob Brumley and Risto M. Hakala

Memory Leakage-Resilient Encryption Based on Physically Unclonable
 Functions 685
*Frederik Armknecht, Roel Maes, Ahmad-Reza Sadeghi,
 Berk Sunar, and Pim Tuyls*

Signature Schemes with Bounded Leakage Resilience 703
Jonathan Katz and Vinod Vaikuntanathan

Author Index 721

Related-Key Cryptanalysis of the Full AES-192 and AES-256

Alex Biryukov and Dmitry Khovratovich

University of Luxembourg

Abstract. In this paper we present two related-key attacks on the full AES. For AES-256 we show the first key recovery attack that works for all the keys and has $2^{99.5}$ time and data complexity, while the recent attack by Biryukov-Khovratovich-Nikolić works for a weak key class and has much higher complexity. The second attack is the first cryptanalysis of the full AES-192. Both our attacks are boomerang attacks, which are based on the recent idea of finding *local collisions in block ciphers* and enhanced with the *boomerang switching* techniques to gain free rounds in the middle.

Keywords: AES, related-key attack, boomerang attack.

The extended version of this paper is available at <http://eprint.iacr.org/2009/317.pdf>.

1 Introduction

The Advanced Encryption Standard (AES) [9] — a 128-bit block cipher, is one of the most popular ciphers in the world and is widely used for both commercial and government purposes. It has three variants which offer different security levels based on the length of the secret key: 128, 192, 256-bits. Since it became a standard in 2001 [1], the progress in its cryptanalysis has been very slow. The best results until 2009 were attacks on 7-round AES-128 [10,11], 10-round AES-192 [5,13], 10-round AES-256 [5,13] out of 10, 12 and 14 rounds respectively. The two last results are in the related-key scenario.

Only recently there was announced a first attack on the full AES-256 [6]. The authors showed a related-key attack which works with complexity 2^{96} for one out of every 2^{35} keys. They have also shown practical attacks on AES-256 (see also [7]) in the chosen key scenario, which demonstrates that AES-256 can not serve as a replacement for an ideal cipher in theoretically sound constructions such as Davies-Meyer mode.

In this paper we improve these results and present the first related-key attack on AES-256 that works for all the keys and has a better complexity ($2^{99.5}$ data and time). We also develop the first related key attack on the full AES-192. In both attacks we minimize the number of active S-boxes in the key-schedule (which caused the previous attack on AES-256 to work only for a fraction of all

Table 1. Best attacks on AES-192 and AES-256

Attack	Rounds	# keys	Data	Time	Memory	Source
192						
Partial sums	8	1	$2^{127.9}$	2^{188}	?	[10]
Related-key rectangle	10	64	2^{124}	2^{183}	?	[5,13]
Related-key amplified boomerang	12	4	2^{123}	2^{176}	2^{152}	Sec. 6
256						
Partial sums	9	256	2^{85}	2^{226}	2^{32}	[10]
Related-key rectangle	10	64	2^{114}	2^{173}	?	[5,13]
Related-key differential	14	2^{35}	2^{131}	2^{131}	2^{65}	[6]
Related-key boomerang	14	4	$2^{99.5}$	$2^{99.5}$	2^{77}	Sec. 5

keys) by using a boomerang attack [15] enhanced with *boomerang switching* techniques. We find our boomerang differentials by searching for *local collisions* [8,6] in a cipher. The complexities of our attacks and a comparison with the best previous attacks are given in Table 1.

This paper is structured as follows: In Section 3 we develop the idea of local collisions in the cipher and show how to construct optimal related-key differentials for AES-192 and AES-256. In Section 4 we briefly explain the idea of a boomerang and an amplified boomerang attack. In Sections 5 and 6 we describe an attack on AES-256 and AES-192, respectively.

2 AES Description and Notation

We expect that most of our readers are familiar with the description of AES and thus point out only the main features of AES-256 that are crucial for our attack.

AES rounds are numbered from 1 to 14 (12 for AES-192). We denote the i -th 192-bit subkey (do not confuse with a 128-bit round key) by K^i , i.e. the first (whitening) subkey is the first four columns of K^0 . The last subkey is K^7 in AES-256 and K^8 in AES-192. The difference in K^i is denoted by ΔK^i . Bytes of a subkey are denoted by $k_{i,j}^l$, where i, j stand for the row and column index, respectively, in the standard matrix representation of AES, and l stands for the number of the subkey. Bytes of the plaintext are denoted by $p_{i,j}$, and bytes of the internal state after the SubBytes transformation in round r are denoted by $a_{i,j}^r$, with A^r depicting the whole state. Let us also denote by $b_{i,j}^r$ byte in position (i, j) after the r -th application of MixColumns.

Features of AES-256. AES-256 has 14 rounds and a 256-bit key, which is two times larger than the internal state. Thus the key schedule consists of only 7 rounds. One key schedule round consists of the following transformations:

$$\begin{aligned}
 k_{i,0}^{l+1} &\leftarrow S(k_{i+1,7}^l) \oplus k_{i,0}^l \oplus C^l, & 0 \leq i \leq 3; \\
 k_{i,j}^{l+1} &\leftarrow k_{i,j-1}^{l+1} \oplus k_{i,j}^l, & 0 \leq i \leq 3, 1 \leq j \leq 3; \\
 k_{i,4}^{l+1} &\leftarrow S(k_{i,3}^{l+1}) \oplus k_{i,4}^l, & 0 \leq i \leq 3; \\
 k_{i,j}^{l+1} &\leftarrow k_{i,j-1}^{l+1} \oplus k_{i,j}^l, & 0 \leq i \leq 3, 5 \leq j \leq 7,
 \end{aligned}$$

where $S()$ stands for the S-box, and C^l — for the round-dependent constant. Therefore, each round has 8 S-boxes.

Features of AES-192. AES-192 has 12 rounds and a 192-bit key, which is 1.5 times larger than the internal state. Thus the key schedule consists of 8 rounds. One key schedule round consists of the following transformations:

$$\begin{aligned}
 k_{i,0}^{l+1} &\leftarrow S(k_{i+1,5}^l) \oplus k_{i,0}^l \oplus C^l, & 0 \leq i \leq 3; \\
 k_{i,j}^{l+1} &\leftarrow k_{i,j-1}^{l+1} \oplus k_{i,j}^l, & 0 \leq i \leq 3, 1 \leq j \leq 5.
 \end{aligned}$$

Notice that each round has only four S-boxes.

3 Local Collisions in AES

The notion of a local collision comes from the cryptanalysis of hash functions with one of the first applications by Chabaud and Joux [8]. The idea is to inject a difference into the internal state, causing a *disturbance*, and then to *correct* it with the next injections. The resulting difference pattern is spread out due to the message schedule causing more disturbances in other rounds. The goal is to have as few disturbances as possible in order to reduce the complexity of the attack.

In the related-key scenario we are allowed to have difference in the key, and not only in the plaintext as in the pure differential cryptanalysis. However the attacker can not control the key itself and thus the attack should work for any key pair with a given difference.

Local collisions in AES-256 are best understood on a one-round example (Fig. 1), which has one active S-box in the internal state, and five non-zero byte differences in the two consecutive subkeys. This differential holds with probability 2^{-6} if we use an optimal differential for an S-box:

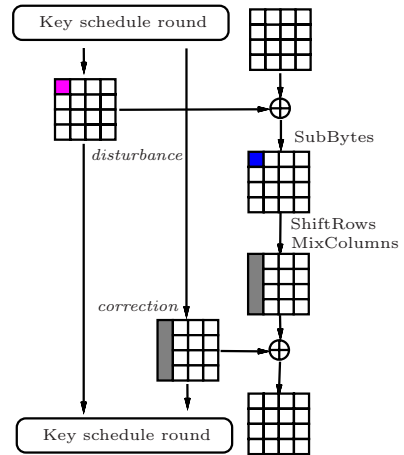


Fig. 1. A local collision in AES-256

$$0x01 \xrightarrow{\text{SubBytes}} 0x1f; \quad \begin{pmatrix} 0x1f \\ 0 \\ 0 \\ 0 \end{pmatrix} \xrightarrow{\text{MixColumns}} \begin{pmatrix} 0x3e \\ 0x1f \\ 0x1f \\ 0x21 \end{pmatrix}$$

Due to the key schedule the differences spread to other subkeys thus forming the *key schedule difference*. The resulting key schedule difference can be viewed as a set of local collisions, where the expansion of the disturbance (also called *disturbance vector*) and the correction differences compensate each other. The probability of the full differential trail is then determined by the number of active S-boxes in the key-schedule and in the internal state. The latter is just the number of the non-zero bytes in the disturbance vector.

Therefore, to construct an optimal trail we have to construct a minimal-weight disturbance expansion, which will become a part of the full key schedule difference. For the AES key schedule, which is mostly linear, this task can be viewed as building a low-weight codeword of a linear code. Simultaneously, correction differences also form a codeword, and the key schedule difference codeword is the sum of the disturbance and the correction codewords. In the simplest trail the correction codeword is constructed from the former one by just shifting four columns to the right and applying the S-box–MixColumns transformation.

An example of a good key-schedule pattern for AES-256 is depicted in Figure 2 as a 4.5-round codeword. In the first four key-schedule rounds the disturbance codeword has only 9 active bytes (red cells in the picture), which is the lower bound. We want to avoid active S-boxes in the key schedule as long as possible, so we start with a single-byte difference in byte $k_{0,0}^4$ and go backwards. Due to a slow diffusion in the AES key schedule the difference affects only one more byte per key schedule round. The correction (grey) column should be positioned four columns to the right, and propagates backwards in the same way. The last column in the first subkey is active, so all S-boxes of the first round are active as well, which causes an unknown difference in the first (green) column. This “alien” difference should be canceled by the plaintext difference.

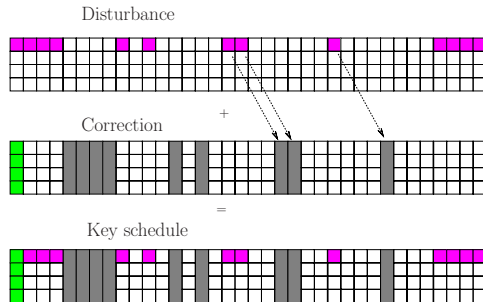


Fig. 2. Full key schedule difference (4.5 key-schedule rounds) for AES-256

4 Related Key Boomerang and Amplified Boomerang Attacks

In this section we describe two types of boomerang attacks in the related-key scenario.

A basic boomerang distinguisher [15] is applied to a cipher $E_K(\cdot)$ which is considered as a composition of two sub-ciphers: $E_K(\cdot) = E_1 \circ E_0$. The first sub-cipher is supposed to have a differential $\alpha \rightarrow \beta$, and the second one to have a differential $\gamma \rightarrow \delta$, with probabilities p and q , respectively. In the further text the differential trails of E_0 and E_1 are called *upper* and *lower* trails, respectively.

In the boomerang attack a plaintext pair results in a *quartet* with probability p^2q^2 . The amplified boomerang attack [12] (also called rectangle attack [4]) works in a chosen-plaintext scenario and constructs $N^2p^2q^22^{-n}$ quartets of N plaintext pairs. We refer to [15,12] for the full description of the attacks.

In the original boomerang attack paper by Wagner [15] it was noted that the number of good ciphertext quartets is actually higher, since an attacker may consider other β and γ (with the same α and δ). This observation can be applied to both types of boomerang attacks. As a result, the number Q of good quartets is expressed via *amplified probabilities* \hat{p} and \hat{q} as follows:

$$Q = \hat{p}^2\hat{q}^22^{-n}N^2,$$

where

$$\hat{p} = \sqrt{\sum_{\beta} P[\alpha \rightarrow \beta]^2}; \quad \hat{q} = \sqrt{\sum_{\gamma} P[\gamma \rightarrow \delta]^2}. \quad (1)$$

4.1 Related-Key Attack Model

The related-key attack model [3] is a class of cryptanalytic attacks in which the attacker knows or chooses a relation between several keys and is given access to encryption/decryption functions with all these keys. The goal of the attacker is to find the actual secret keys. The relation between the keys can be an arbitrary bijective function R (or even a family of such functions) chosen in advance by the attacker (for a formal treatment of the general related key model see [2,14]). In the simplest form of this attack, this relation is just a XOR with a constant: $K_2 = K_1 \oplus C$, where the constant C is chosen by the attacker. This type of relation allows the attacker to trace the propagation of XOR differences induced by the key difference C through the key schedule of the cipher. However, more complex forms of this attack allow other (possibly non-linear) relations between the keys. For example, in some of the attacks described in this paper the attacker chooses a desired XOR relation in the second subkey, and then defines the implied relation between the actual keys as: $K_2 = F^{-1}(F(K_1) \oplus C) = R_C(K_1)$ where

F represents a single round of the AES-256 key schedule, and the constant C is chosen by the attacker.¹

Compared to other cryptanalytic attacks in which the attacker can manipulate only the plaintexts and/or the ciphertexts the choice of the relation between secret keys gives additional degree of freedom to the attacker. The downside of this freedom is that such attacks might be harder to mount in practice. Still, designers usually try to build “ideal” primitives which can be automatically used without further analysis in the widest possible set of applications, protocols, or modes of operation. Thus resistance to such attacks is an important design goal for block ciphers, and in fact it was one of the stated design goals of the Rijndael algorithm, which was selected as the Advanced Encryption Standard.

In this paper we use boomerang attacks in the related-key scenario. In the following sections we denote the difference between subkeys in the upper trail by ΔK^i , and in the lower part by ∇K^i .

4.2 Boomerang Switch

Here we analyze the transition from the sub-trail E_0 to the sub-trail E_1 , which we call the *boomerang switch*. We show that the attacker can gain 1-2 middle rounds for free due to a careful choice of the top and bottom differentials. The position of the switch is a tradeoff between the sub-trail probabilities, that should minimize the overall complexity of the distinguisher. Below we summarize the switching techniques that can be used in boomerang or amplified boomerang attacks on any block cipher.

Ladder switch. By default, a cipher is decomposed into rounds. However, such decomposition may not be the best for the boomerang attack. We propose not only to further decompose the round into simple operations but also to exploit the existing parallelism in these operations. For example some bytes may be independently processed. In such case we can switch in one byte before it is transformed and in another one after it is transformed, see Fig. 3 for an illustration.

An example is our attack on AES-192. Let us look at the differential trails (see Fig. 8). There is one active S-box in round 7 of the lower trail in byte $b_{0,2}^7$. On the other hand, the S-box in the same position is not active in the upper trail. If we would switch after ShiftRows in round 6, we would “pay” the probability in round 7 afterwards. However, we switch all the state except $b_{0,2}$ after MixColumns, and switch the remaining byte after the S-box application in round 7, where it is not active. We thus do not pay for this S-box.

Feistel switch. Surprisingly, a Feistel round with an arbitrary function (e.g., an S-box) can be passed for free in the boomerang attack (this was first observed in the attack on cipher Khufu in [15]). Suppose the internal state (X, Y) is

¹ Note that due to low nonlinearity of AES-256 key schedule such subkey relation corresponds to a fixed XOR relation in 28 out of 32 bytes of the secret key, and a simple S-box relation in the four remaining bytes.

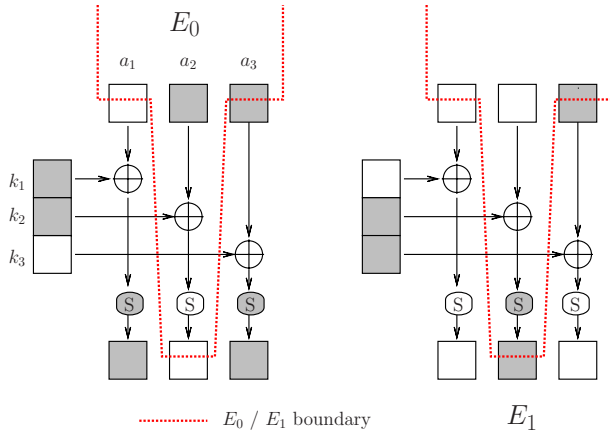


Fig. 3. The ladder switch in a toy three S-box block. A switch either before or after the S-box layer would cost probability, while the ladder does not.

transformed to $(Z = X \oplus f(Y), Y)$ at the end of E_0 . Suppose also that the E_0 difference before this transformation is (Δ_X, Δ_Y) , and that the E_1 difference after this transformation is (Δ_Z, Δ_Y) .

As a result, variable Y in the four iterations of a boomerang quartet takes two values: Y_0 and $Y_0 \oplus \Delta_Y$ for some Y_0 . Then the f transformation is guaranteed to have the same output difference Δ_f in the quartet. Therefore, the decryption phase of the boomerang creates the difference Δ_X in X at the end of E_0 “for free”. This trick is used in the switch in the subkey in the attack on AES-192.

S-box switch. This is similar to the Feistel switch, but costs probability only in one of the directions. Suppose that E_0 ends with an S-box $Y \leftarrow S(X)$ with difference Δ . If the output of an S-box in a cipher has difference Δ and if the same difference Δ comes from the lower trail, then propagation through this S-box is for free on one of the faces of the boomerang. Moreover, the other direction can use amplified probability since specific value of the difference Δ is not important for the switch².

5 Attack on AES-256

In this section we present a related key boomerang attack on AES-256.

5.1 The Trail

The boomerang trail is depicted in Figure 7, and the actual values are listed in Tables 3 and 2. It consists of two similar 7-round trails: the first one covers rounds

² This type of switch was used in the original version of this paper, but is not needed now due to change in the trails. We describe it here for completeness, since it might be useful in other attacks.

Table 2. Key schedule difference in the AES-256 trail

ΔK^2					
0	? 00 00 00 3e 3e 3e 3e	1	00 00 00 00 3e 00 3e 00	2	00 00 00 00 3e 3e 00 00
	? 01 01 01 ? 21 21 21		00 01 00 01 21 00 21 00		00 01 01 00 21 21 00 00
	? 00 00 00 1f 1f 1f 1f		00 00 00 00 1f 00 1f 00		00 00 00 00 1f 1f 00 00
	? 00 00 00 1f 1f 1f 1f		00 00 00 00 1f 00 1f 00		00 00 00 00 1f 1f 00 00
3	00 00 00 00 3e 00 00 00	4	00 00 00 00 3e 3e 3e 3e		
	00 01 00 00 21 00 00 00		00 01 01 01 ? ? ? ?		
	00 00 00 00 1f 00 00 00		00 00 00 00 1f 1f 1f 1f		
	00 00 00 00 1f 00 00 00		00 00 00 00 1f 1f 1f 1f		
∇K^2					
0	? ? ? ? ? ? ? 00	1	? 01 ? 00 ? ? 00 00	2	? ? 00 00 ? 00 00 00
	X X X X 1f 1f 1f 00		X 00 X 00 1f 1f 00 00		X X 00 00 1f 00 00 00
	? ? ? ? 1f 1f 1f 00		? 00 ? 00 1f 1f 00 00		? ? 00 00 1f 00 00 00
	? ? ? ? 21 21 21 00		? 00 ? 00 21 21 00 00		? ? 00 00 21 00 00 00
3	? 01 01 01 3e 3e 3e 3e	4	01 00 01 00 3e 00 3e 00	5	01 01 00 00 3e 3e 00 00
	X 00 00 00 1f 1f 1f 1f		00 00 00 00 1f 00 1f 00		00 00 00 00 1f 1f 00 00
	? 00 00 00 1f 1f 1f 1f		00 00 00 00 1f 00 1f 00		00 00 00 00 1f 1f 00 00
	? 00 00 00 21 21 21 21		00 00 00 00 21 00 21 00		00 00 00 00 21 21 00 00
6	01 00 00 00 3e 00 00 00	7	01 01 01 01 ? ? ? ?		
	00 00 00 00 1f 00 00 00		00 00 00 00 1f 1f 1f 1f		
	00 00 00 00 1f 00 00 00		00 00 00 00 1f 1f 1f 1f		
	00 00 00 00 21 00 00 00		00 00 00 00 21 21 21 21		

1–8, and the second one covers rounds 8–14. The trails differ in the position of the disturbance bytes: the row 1 in the upper trail, and the row 0 in the lower trail. This fact allows the Ladder switch.

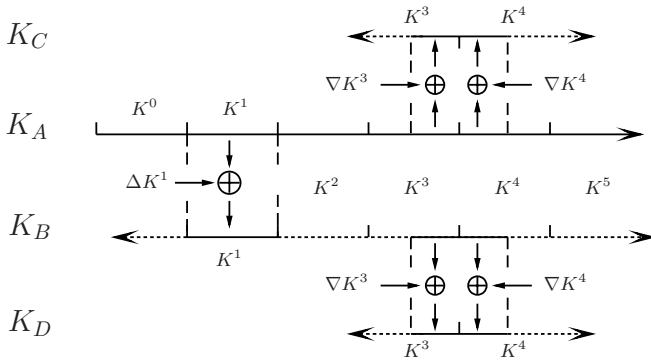
The switching state is the state A^9 (internal state after the SubBytes in round 9) and a special key state K_S , which is the concatenation of the last four columns of K^3 and the first four columns of K^4 . Although there are active S-boxes in the first round of the key schedule, we do not impose conditions on them. As a result, the difference in column 0 of K^0 is unknown yet.

Related Keys. We define the relation between four keys as follows (see also Figure 4). For a secret key K_A , which the attacker tries to find, compute its second subkey K_A^1 and apply the difference ΔK^1 to get a subkey K_B^1 , from which the key K_B is computed. The relation between K_A and K_B is a constant XOR relation in 28 bytes out of 32 and is computed via a function $k'_{i,0} = k_{i,0} \oplus S(k_{i+1,7}) \oplus S(k_{i+1,7} \oplus c_{i+1,7})$, $i=0,1,2,3$, with constant $c_{i+1,7} = \Delta k_{i+1,7}^0$ for the four remaining bytes.

The switch into the keys K_C, K_D happens between the 3rd and the 4th subkeys in order to avoid active S-boxes in the key-schedule using the *Ladder switch* idea described above. We compute subkeys K^3 and K^4 for both K_A and K_B . We add the difference ∇K^3 to K_A^3 and compute the upper half (four columns) of K_C^3 . Then we add the difference ∇K^4 to K_A^4 and compute the lower half (four

Table 3. Non-zero internal state differences in the AES-256 trail

ΔP	? 00 00 00 ? ? ? ? ? 00 ? 00 ? 00 00 ?	ΔA^1	? 00 00 00 1f ? 1f 1f 00 00 ? 00 00 00 00 ?	ΔA^3	00 00 00 00 00 1f 00 1f 00 00 00 00 00 00 00 00	ΔA^5	00 00 00 00 00 1f 1f 00 00 00 00 00 00 00 00 00
ΔA^7	00 00 00 00 00 1f 00 00 00 00 00 00 00 00 00 00	∇A^7	1f 1f 1f 1f 00 00 00 00 00 00 00 00 00 00 00 00	∇A^9	1f 00 1f 00 00 00 00 00 00 00 00 00 00 00 00 00	∇A^{11}	1f 1f 00 00 00 00 00 00 00 00 00 00 00 00 00 00
∇A^{13}	1f 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	ΔC	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00				


Fig. 4. AES-256: Computing K_B , K_C , and K_D from K_A

columns) of K_C^4 . From these eight consecutive columns we compute the full K_C . The key K_D is computed from K_B in the same way.

Finally, we point out that difference between K_C and K_D can be computed in the backward direction deterministically since there would be no active S-boxes till the first round. The secret key K_A , and the three keys K_B , K_C , K_D computed from K_A as described above form a proper related key quartet. Moreover, due to a slow diffusion in the backward direction, as a bonus we can compute some values in ∇K^i even for $i = 0, 1, 2, 3$ (see Table 2). Hence given the byte value $k_{i,j}^l$ for K_A we can partly compute K_B , K_C and K_D .

Internal State. The plaintext difference is specified in 9 bytes. We require that all the active S-boxes in the internal state should output the difference $0x1f$ so that the active S-boxes are passed with probability 2^{-6} . The only exception is the first round where the input difference in nine active bytes is not specified.

Let us start a boomerang attack with a random pair of plaintexts that fit the trail after one round. Active S-boxes in rounds 3–7 are passed with probability 2^{-6} each, so the overall probability is 2^{-30} .

We switch the internal state in round 9 with the *Ladder switch* technique: the row 1 is switched before the application of S-boxes, and the other rows are switched after the S-box layer. As a result, we do not pay for active S-boxes at all in this round.

The second part of the boomerang trail is quite simple. Three S-boxes in rounds 10–14 contribute to the probability, which is thus equal to 2^{-18} . Finally we get one boomerang quartet after the first round with probability $2^{-30-30-18-18} = 2^{-96}$.

5.2 The Attack

The attack works as follows. Do the following steps $2^{25.5}$ times:

1. Prepare a structure of plaintexts as specified below.
2. Encrypt it on keys K_A and K_B and keep the resulting sets S_A and S_B in memory.
3. XOR Δ_C to all the ciphertexts in S_A and decrypt the resulting ciphertexts with K_C . Denote the new set of plaintexts by S_C .
4. Repeat previous step for the set S_B and the key K_D . Denote the set of plaintexts by S_D .
5. Compose from S_C and S_D all the possible pairs of plaintexts which are equal

	c	c	c
	c	c	c
	c	c	c

in 56 bits .

6. For every remaining pair check if the difference in $p_{i,0}$, $i > 1$ is equal on both sides of the boomerang quartet (16-bit filter). Note that $\nabla k_{i,7}^0 = 0$ so $\Delta k_{i,0}^0$ should be equal for both key pairs (K_A, K_B) and (K_C, K_D) .
7. Filter out the quartets whose difference can not be produced by active S-boxes in the first round (one-bit filter per S-box per key pair) and active S-boxes in the key schedule (one-bit filter per S-box), which is a $2 \cdot 2 + 2 = 6$ -bit filter.
8. Gradually recover key values and differences simultaneously filtering out the wrong quartets.

Each structure has all possible values in column 0 and row 0, and constant values in the other bytes. Of 2^{72} texts per structure we can compose 2^{144} ordered pairs. Of these pairs $2^{144-8 \cdot 9} = 2^{72}$ pass the first round. Thus we expect one right quartet per $2^{96-72} = 2^{24}$ structures, and three right quartets out of $2^{25.5}$ structures.

Let us now compute the number of noisy quartets. About $2^{144-56-16} = 2^{72}$ pairs come out of step [6](#). The next step applies a 6-bit filter, so we get $2^{72+25.5-6} = 2^{91.5}$ candidate quartets in total.

The remainder of this section deals with gradual recovering of the key and filtering wrong quartets. The key bytes are recovered as shown in [Figure 5](#).

5						0
2	3	1	1	^{3D} ₄		
0D		5				⁰ ₄
0D			5			0

Fig. 5. Gradual key recovery. Digits stand for the steps, 'D' means difference.

1. First, consider 4-tuples of related key bytes in each position $(1, j), j < 4$. Two differences in a tuple are known by default. The third difference is unknown but is equal for all tuples (see Table 2, where it is denoted by X) and gets one of 2^7 values. We use this fact for key derivation and filtering as follows. Consider key bytes $k_{2,2}^0$ and $k_{2,3}^0$. The candidate quartet proposes 2^2 candidates for both 4-tuples of related-key bytes, or 2^4 candidates in total. Since the differences are related with the X-difference, which is a 9-bit filter, this step reveals two key bytes and the value of X and reduces the number of quartets to $2^{91.5-5} = 2^{86.5}$.
2. Now consider the value of $\Delta k_{1,0}^0$, which is unknown yet and might be different in two pairs of related keys. Let us notice that it is determined by the value of $k_{2,7}^0$, and $\nabla k_{2,7}^0 = 0$, so that $\Delta k_{1,0}^0$ is the same for both related key pairs and can take 2^7 values. Each guess of $\Delta k_{1,0}^0$ proposes key candidates for byte $k_{2,0}^0$, where we have a 8-bit filter for the 4-tuple of related-key bytes. We thus derive the value of $k_{1,0}^0$ in all keys and reduce the number of candidate quartets to $2^{85.5}$.
3. The same trick holds for the unknown $\Delta k_{1,4}^0$, which can get 2^7 possible values and can be computed for both key pairs simultaneously. Each of these values proposes four candidates for $k_{1,1}^0$, which are filtered with an 8-bit filter. We thus recover $k_{1,1}^0$ and $\Delta k_{1,4}^0$ and reduce the number of quartets to $2^{79.5}$.
4. Finally, we notice that $\Delta k_{1,4}^0$ is completely determined by $k_{1,0}^0, k_{1,1}^0, k_{1,2}^0, k_{1,3}^0$, and $k_{2,7}^0$. There are at most two candidates for the latter value as well as for $\Delta k_{1,4}^0$, so we get a 6-bit filter and reduce the number of quartets to $2^{72.5}$.
5. Each quartet also proposes two candidates for each of key bytes $k_{0,0}^0, k_{2,2}^0$, and $k_{3,3}^0$. Totally, the number of key candidates proposed by each quartet is 2^6 .

The key candidates are proposed for 11 bytes of each of four related keys. However, these bytes are strongly related so the number of independent key bytes on which the voting is performed is significantly smaller than 11×4 . At least, bytes $k_{0,0}^0, k_{1,1}^0, k_{2,2}^0$ and $k_{3,3}^0$ of K_A and K_C are independent so we recover 15 key bytes with $2^{78.5}$ proposals. The probability that three wrong quartets propose the same candidates does not exceed 2^{-80} .

We thus estimate the complexity of the filtering step as $2^{77.5}$ time and memory. We recover $3 \cdot 7 + 8 \cdot 8 = 85$ bits of K_A (and 85 bits of K_C) with $2^{99.5}$ data and time and $2^{77.5}$ memory.

The remaining part of the key can be found with many approaches. One is to relax the condition on one of the active S-boxes in round 3 thus getting four

more active S-boxes in round 2, which in turn leads to a full-difference state in round 1. The condition can be actually relaxed only for the first part of the boomerang (the key pair (K_A, K_B)) thus giving a better output filter. For each candidate quartet we use the key bytes, that were recovered at the previous step, to compute ΔA^1 and thus significantly reduce the number of keys that are proposed by a quartet. We then rank candidates for the first four columns of K_A^0 and take the candidate that gets the maximal number of votes. Since we do not make key guesses, we expect that the complexity of this step is smaller than the complexity of the previous step ($2^{99.5}$). The right quartet also provide information about four more bytes in the right half of K_A^0 that correspond to the four active S-boxes in round 2. The remaining 8 bytes of K_A can be found by exhaustive search.

6 Attack on AES-192

The key schedule of AES-192 has better diffusion, so it is hard to avoid active S-boxes in the subkeys. We construct a related-key boomerang attack with two sub-trails of 6 rounds each. The attack is an amplified-boomerang attack because we have to deal with truncated differences in both the plaintext and the ciphertext, the latter would be expensive to handle in a plain boomerang attack.

6.1 The Trail

The trail is depicted in Figure 8, and the actual values are listed in Tables 4 and 5. The key schedule codeword is depicted in Figure 6.

Table 4. Internal state difference in the AES-192 trail

ΔP	? ? 3e ? 1f 1f ? 1f 1f 1f 1f ? ? 21 21 21	ΔA^1	1f ? 00 1f 00 00 ? 00 00 00 00 ? ? 00 00 00	ΔA^2	00 1f 00 00 00 00 00 00 00 00 00 00 00 00 00 00	ΔA^3	00 1f 1f 00 00 00 00 00 00 00 00 00 00 00 00 00
ΔA^4	00 00 00 1f 00 00 00 00 00 00 00 00 00 00 00 00	ΔA^5	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	ΔA^6	00 1f 1f 1f 00 00 00 00 00 00 00 00 00 00 00 00	ΔA^7	00 00 00 1f 00 00 00 00 00 00 00 00 00 00 00 00
∇A^6	1f 1f 1f 1f 00 00 00 00 00 00 00 00 00 00 00 00	∇A^7	00 00 1f 00 00 00 00 00 00 00 00 00 00 00 00 00	∇A^8	1f 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	∇A^9	1f 1f 00 00 00 00 00 00 00 00 00 00 00 00 00 00
∇A^{10}	00 00 1f 00 00 00 00 00 00 00 00 00 00 00 00 00	∇A^{11}	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	∇A^{12}	? ? ? ? 00 00 00 00 00 00 00 00 00 00 00 00	ΔC	? ? ? ? 1f 1f 1f 1f 1f 1f 1f 1f ? ? ? ?

Table 5. Key schedule difference in the AES-192 trail

ΔK^0	00 3e 3e 3f 3e 01 00 1f 1f 1f 1f 00 00 1f 1f 1f 1f 00 ? 21 21 21 21 00	ΔK^1	00 3e 00 3f 01 00 00 1f 00 1f 00 00 00 1f 00 1f 00 00 00 21 00 21 00 00	ΔK^2	00 3e 3e 01 00 00 00 1f 1f 00 00 00 00 1f 1f 00 00 00 00 21 21 00 00 00
ΔK^3	00 3e 00 01 01 01 00 1f 00 00 00 00 00 1f 00 00 00 00 00 21 00 00 00 00	ΔK^4	00 3e 3e 3f 3e 3f 00 1f 1f 1f 1f 1f 00 1f 1f 1f 1f 1f ? ? ? ? ? ?		
∇K^0	? ? ? 3e 3f 3e ? ? ? 1f 1f 1f ? ? ? 1f 1f 1f ? ? ? ? 21 21	∇K^1	? ? 3f 01 3e 00 ? ? 1f 00 1f 00 ? ? 1f 00 1f 00 ? ? ? 00 21 00	∇K^2	? 3e 01 00 3e 3e ? 1f 00 00 1f 1f ? 1f 00 00 1f 1f ? ? 00 00 21 21
∇K^3	3e 00 01 01 3f 01 1f 00 00 00 1f 00 1f 00 00 00 1f 00 ? 00 00 00 21 00	∇K^4	3e 3e 3f 3e 01 00 1f 1f 1f 1f 00 00 1f 1f 1f 1f 00 00 21 21 21 21 00 00	∇K^5	3e 00 3f 01 00 00 1f 00 1f 00 00 00 1f 00 1f 00 00 00 21 00 21 00 00 00
∇K^6	3e 3e 01 00 00 00 1f 1f 00 00 00 00 1f 1f 00 00 00 00 21 21 00 00 00 00	∇K^7	3e 00 01 01 01 01 1f 00 00 00 00 00 1f 00 00 00 00 00 21 00 00 00 00 00	∇K^8	3e 3e 3f 3e 3f 3e 1f 1f 1f 1f 1f 1f 1f 1f 1f 1f 1f 1f ? ? ? ? ? ?

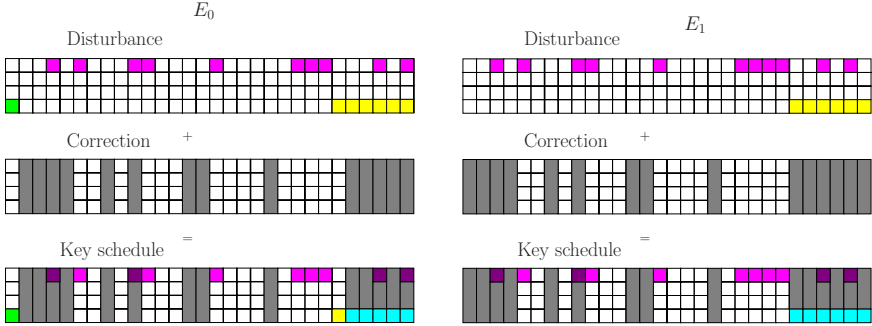


Fig. 6. AES-192 key schedule codeword

Related Keys. We define the relation between four keys similarly to the attack on AES-256. Assume we are given a key K_A , which the attacker tries to find. We compute its subkey K_A^1 and apply the difference ΔK^1 to get the subkey K_B^1 , from which the key K_B is computed. Then we compute the subkeys K_A^4 and K_B^4 and apply the difference ∇K^4 to them. We get subkeys K_C^4 and K_D^4 , from which the keys K_C and K_D are computed.

Now we prove that keys $K_A, K_B, K_C,$ and K_D form a quartet, i.e. the subkeys of K_C and K_D satisfy the equations $K_C^l \oplus K_D^l = \Delta K^l, l = 1, 2, 3$. The only active S-box is positioned between K^3 and K^4 , whose input is $k_{0,5}^3$. However, this

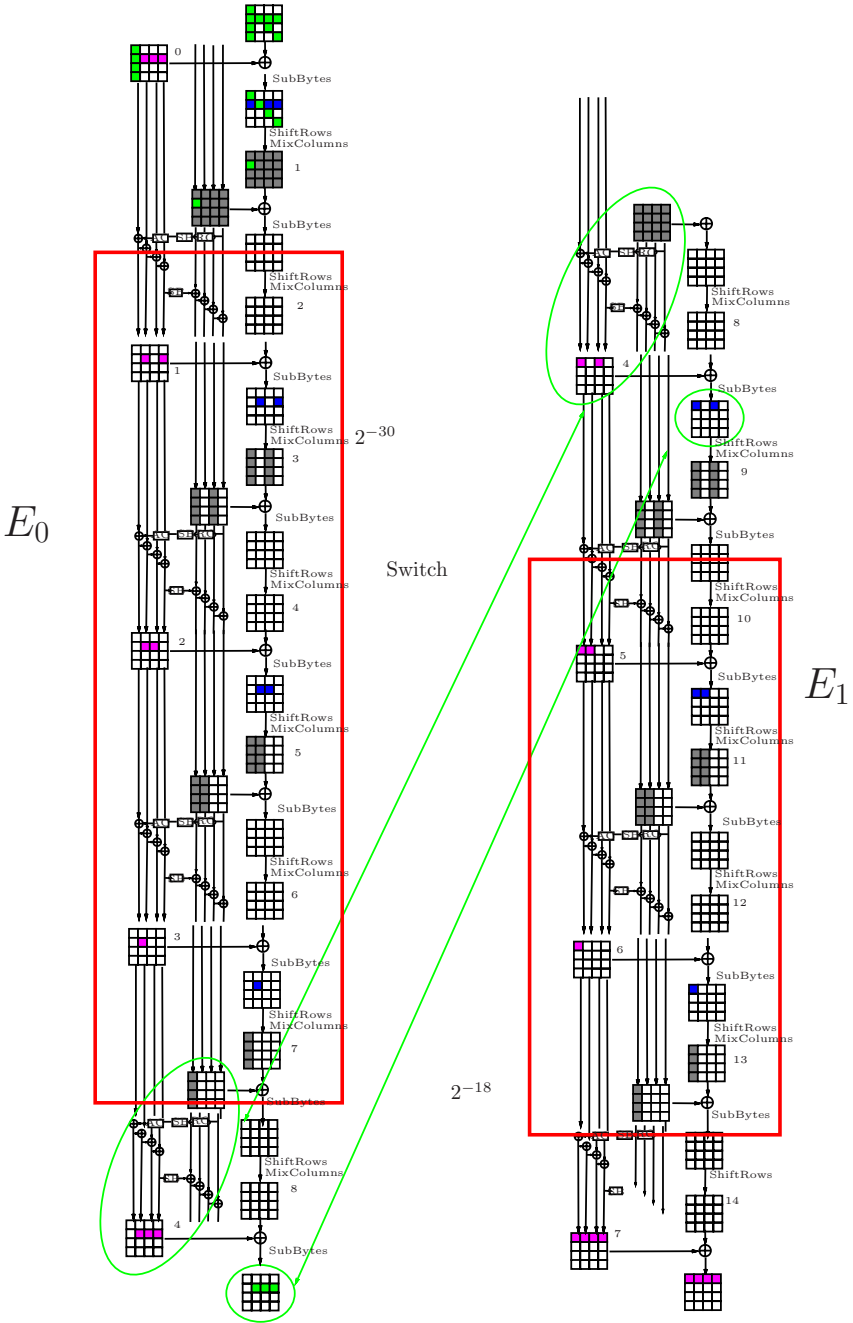


Fig. 7. AES-256 E_0 and E_1 trails. Green ovals show an overlap between the two trails where the switch happens.

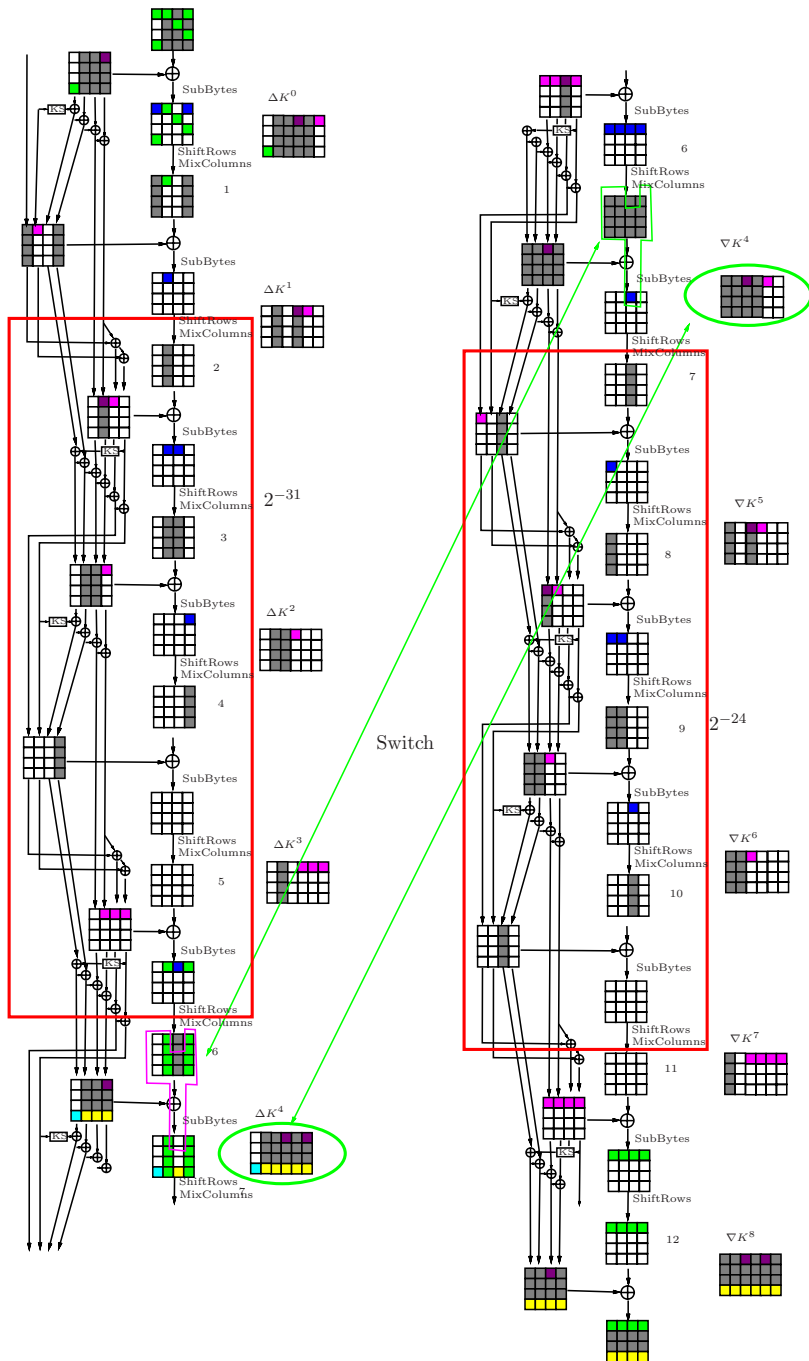



Fig. 8. AES-192 trail


S-box gets the same pair of inputs in both key pairs (see the “*Feistel switch*” in Sec. 4.2). Indeed, if we compute $\nabla k_{0,5}^3$ from ΔK^4 , then it is equal to $\Delta k_{0,5}^3 = 0x01$. Therefore, if the active S-box gets as input α and $\alpha \oplus 1$ in K_A and K_B , respectively, then it gets $a \oplus 1$ and a in K_C and K_D , respectively. As a result, $K_C^3 \oplus K_D^3 = \Delta K^3$, the further propagation is linear, so the four keys form a quartet.

Due to a slow diffusion in the backward direction, we can compute some values in ∇K^l even for small l (Table 5). Hence given $k_{i,j}^l$ for K_A we can partly compute K_B , K_C and K_D , which provides additional filtration in the attack.

Internal State. The plaintext difference is specified in 10 bytes , the difference in the other six bytes not restricted. The three active S-boxes in rounds 2–4 are passed with probability 2^{-6} each. In round 6 (the switching round) we ask for the fixed difference only in $a_{0,2}^6$, the other two S-boxes can output any difference such that it is the same as in the second related-key pair. Therefore, the amplified probability of round 6 equals to $2^{-6-2\cdot 3.5} = 2^{-13}$. We switch between the two trails before the key addition in round 6 in all bytes except $b_{0,2}^6$, where we switch after the S-box application in round 7 (the *Ladder switch*). This trick allows us not to take into account the only active S-box in the lower trail in round 7. The overall probability of the rounds 3–6 is $2^{-3\cdot 6-13} = 2^{-31}$.

The lower trail has 8 active S-boxes in rounds 8–12. Only the first four active S-boxes are restricted in the output difference, which gives us probability 2^{-24} for the lower trail. The ciphertext difference is fully specified in the middle two rows, and has 35 bits of entropy in the other bytes. More precisely, each $\nabla c_{0,*}$ is taken from a set of size 2^7 , and all the $\nabla c_{3,*}$ should be the same on both sides of the boomerang and again should belong to a set of size 2^7 . Therefore, the ciphertext difference gives us a 93-bit filter.

6.2 The Attack

We compose 2^{73} structures of type  with 2^{48} texts each. Then we encrypt all the texts with the keys K_A and K_C , and their complements w.r.t. ΔP on K_B and K_D . We keep all the data in memory and analyze it with the following procedure:

1. Compose all candidate plaintext pairs for the key pairs (K_A, K_B) and (K_C, K_D) .
2. Compose and store all the candidate quartets of the ciphertexts.
3. For each guess of the subkey bytes: $k_{0,3}^0$, $k_{2,3}^0$, and $k_{0,5}^0$ in K_A ; $k_{0,5}^7$ in K_A and K_B :
 - (a) Derive values for these bytes in all the keys from the differential trail. Derive the yet unknown key differences in ΔK^0 and ∇K^8 .
 - (b) Filter out candidate quartets that contradict ∇K^8 .
 - (c) Prepare counters for the yet unknown subkey bytes that correspond to active S-boxes in the first two rounds and in the last round: $k_{0,0}^0$, $k_{0,1}^0$,

$k_{1,2}^0, k_{3,0}^0$ — in keys K_A and K_C , $k_{0,0}^8, k_{0,1}^8, k_{0,2}^8, k_{0,3}^8$ — in keys K_A and K_B , i.e. 16 bytes in total.

- (d) For each candidate quartet derive possible values for these unknown bytes and increase the counters.
- (e) Pick the group of 16 subkey bytes with the maximal number of votes.
- (f) Try all possible values of the yet unknown 9 key bytes in K^0 and check whether it is the right key. If not then go to the first step.

Right quartets. Let us first count the number of right quartets in the data. Evidently, there exist 2^{128} pairs of internal states with the difference ΔA^2 . The inverse application of 1.5 rounds maps these pairs into structures that we have defined, with 2^{48} pairs per structure. Therefore, each structure has 2^{48} pairs that pass 1.5 rounds, and 2^{73} structures have 2^{121} pairs. Of these pairs $2^{(121-31)\cdot 2-128} = 2^{52}$ right quartets can be composed after the switch in the middle. Of these quartets $2^{52-2\cdot 24} = 16$ right quartets come out of the last round.

Now we briefly describe the attack. Full details will be published in the extended version. In steps 1 and 2 we compose 2^{152} candidate quartets. The guess of five key bytes gives a 32-bit filter in step 3, so we leave with 2^{120} candidate quartets, which are divided according to $\nabla c_{3,0}$ into 2^{14} groups. Then we perform key ranking in each group and recover 16 more key bytes. The exhaustive search for the remaining 9 key bytes can be done with the complexity 2^{72} . The overall time complexity is about 2^{176} , and the data complexity is 2^{123} .

7 Conclusions

We presented related-key boomerang attacks on the full AES-192 and the full AES-256. The differential trails for the attacks are based on the idea of finding local collisions in the block cipher. We showed that optimal key-schedule trails should be based on low-weight codewords in the key schedule. We also exploit various boomerang-switching techniques, which help us to gain free rounds in the middle of the cipher. However, both our attacks are still mainly of theoretical interest and do not present a threat to practical applications using AES.

Acknowledgements. The authors thank Vincent Rijmen and anonymous reviewers for their valuable comments, which helped to improve the paper. Dmitry Khovratovich is supported by PRP "Security & Trust" grant of the University of Luxembourg.

References

1. FIPS-197: Advanced Encryption Standard (November 2001), <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>
2. Bellare, M., Kohno, T.: A theoretical treatment of related-key attacks: Rka-prps, rka-prfs, and applications. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 491–506. Springer, Heidelberg (2003)

3. Biham, E.: New types of cryptanalytic attacks using related keys. *J. Cryptology* 7(4), 229–246 (1994)
4. Biham, E., Dunkelman, O., Keller, N.: The rectangle attack - rectangling the Serpent. In: Pfitzmann, B. (ed.) *EUROCRYPT 2001*. LNCS, vol. 2045, pp. 340–357. Springer, Heidelberg (2001)
5. Biham, E., Dunkelman, O., Keller, N.: Related-key boomerang and rectangle attacks. In: Cramer, R. (ed.) *EUROCRYPT 2005*. LNCS, vol. 3494, pp. 507–525. Springer, Heidelberg (2005)
6. Biryukov, A., Khovratovich, D., Nikolić, I.: Distinguisher and related-key attack on the full AES-256. In: Halevi, S. (ed.) *CRYPTO 2009*. LNCS, vol. 5677, pp. 231–249. Springer, Heidelberg (2009)
7. Biryukov, A., Khovratovich, D., Nikolić, I.: Examples of differential multicollisions for 13 and 14 rounds of AES-256 (2009), <http://eprint.iacr.org/2009/242.pdf>
8. Chabaud, F., Joux, A.: Differential collisions in SHA-0. In: Krawczyk, H. (ed.) *CRYPTO 1998*. LNCS, vol. 1462, p. 56. Springer, Heidelberg (1998)
9. Daemen, J., Rijmen, V.: *The Design of Rijndael. AES — the Advanced Encryption Standard*. Springer, Heidelberg (2002)
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: *AES Candidate Conference*, pp. 230–241 (2000)
12. Kelsey, J., Kohno, T., Schneier, B.: Amplified boomerang attacks against reduced-round MARS and Serpent. In: Schneier, B. (ed.) *FSE 2000*. LNCS, vol. 1978, pp. 75–93. Springer, Heidelberg (2001)
13. Kim, J., Hong, S., Preneel, B.: Related-key rectangle attacks on reduced AES-192 and AES-256. In: Biryukov, A. (ed.) *FSE 2007*. LNCS, vol. 4593, pp. 225–241. Springer, Heidelberg (2007)
14. Lucks, S.: Ciphers secure against related-key attacks. In: Roy, B., Meier, W. (eds.) *FSE 2004*. LNCS, vol. 3017, pp. 359–370. Springer, Heidelberg (2004)
15. Wagner, D.: The boomerang attack. In: Knudsen, L.R. (ed.) *FSE 1999*. LNCS, vol. 1636, pp. 156–170. Springer, Heidelberg (1999)

Disclaimer on colors. We intensively use colors in our figures in order to provide better understanding on the trail construction. In figures, different colors refer to different values, which is hard to depict in black and white. However, we also list all the trail differences in the tables, so all the color information is actually dubbed.

Trail details. By ΔA^i we denote the upper trail difference in the internal state after the S-box layer, and by ∇A^i the same for the lower trail.

The Key-Dependent Attack on Block Ciphers^{*}

Xiaorui Sun and Xuejia Lai

Department of Computer Science
Shanghai Jiao Tong University
Shanghai, 200240, China

sunsirius@sjtu.edu.cn, lai-xj@cs.sjtu.edu.cn

Abstract. In this paper, we formalize an attack scheme using the key-dependent property, called key-dependent attack. In this attack, the intermediate value, whose distribution is key-dependent, is considered. The attack determines whether a key is right by conducting statistical hypothesis test of the intermediate value. The time and data complexity of the key-dependent attack is also discussed.

We also apply key-dependent attack on reduced-round IDEA. This attack is based on the key-dependent distribution of certain items in Biryukov-Demirci Equation. The attack on 5.5-round variant of IDEA requires 2^{21} chosen plaintexts and $2^{112.1}$ encryptions. The attack on 6-round variant requires 2^{49} chosen plaintexts and $2^{112.1}$ encryptions. Compared with the previous attacks, the key-dependent attacks on 5.5-round and 6-round IDEA have the lowest time and data complexity, respectively.

Keywords: Block Cipher, Key-Dependent Attack, IDEA.

1 Introduction

In current cryptanalysis on block ciphers, widespread attacks use special probability distributions of certain intermediate values. These probability distributions are considered as invariant under different keys used. For example, differential cryptanalysis [7] makes use of the probability of the intermediate differential with high probability. Its value is assumed not to vary remarkably with different keys. Linear cryptanalysis [23] is based on the bias of the linear approximation, which is also generally constant for different keys.

Instead of concentrating on the probability distribution which is invariant for different keys, Ben-Aroya and Biham first proposed the key-dependent property in [2]. Key-dependent property means that the probability distribution of intermediate value varies for different keys. In [2], an attack on Lucifer using key-dependent differential was presented. Knudsen and Rijmen also used similar idea to attack DFC in [20].

^{*} This work was supported by NSFC Grant No.60573032, 60773092 and 11th PRP of Shanghai Jiao Tong University.

In this paper, we consider the key-dependent property further. The distribution of intermediate value which is key-dependent is called *key-dependent distribution*. Assume that there are some randomly chosen encryptions. For the intermediate values calculated from these encryptions with the actual key, they should conform to key-dependent distribution. On the other hand, if we use a wrong key to calculate the intermediate values, they are assumed to conform to random distribution. Basing on key-dependent distribution, we formalize a scheme of discovering the actual key by performing statistical hypothesis test [17] on possible keys, and we call this scheme *key-dependent attack*. For a given key, the null hypothesis of the test is that the intermediate value conforms to the key-dependent distribution determined by the key. The samples of the test are the intermediate values calculated from a few encryptions. If the test is passed, the given key is concluded to be the actual key, otherwise it is discarded. For the keys that share the same key-dependent distribution and the same intermediate value calculation, the corresponding hypothesis tests can be merged to reduce the time needed. By this criterion, the whole key space is divided into several *key-dependent subsets*.

Due to the scheme of the key-dependent attack, the time complexity of the attack is determined by the time for distinguishing between the random distribution and the key-dependent distribution. The time needed relies on the entropy of the key-dependent distribution: the closer the key-dependent distribution is to the random distribution, the more encryptions are needed. For each key-dependent subset, the number of encryptions and the criteria of rejecting hypothesis can be chosen so that the attack on this subset is optimized. The expected time of the attack on each subset is also obtained.

The total expected time complexity can be calculated from the expected time on each key-dependent subset. Different orders of the key-dependent subsets attacked have different expected time complexities. The order with minimal expected time complexity is presented. The total expected time complexity is also minimized in this way if the actual key is supposed to be chosen uniformly from the whole key space.

This paper also presents a key-dependent attack on block cipher IDEA. The block cipher IDEA (International Data Encryption Algorithm) was proposed in [21,22]. The cryptanalysis of IDEA was discussed in [1,3,4,5,6,8,9,11,12,13,14,15,16,18,19,24,25], and no attack on full version IDEA is faster than exhaustive search so far. We investigate the Biryukov-Demirci Equation, which is widely used in recent attacks on IDEA [1,5,6,13,16,18]. We find that particular items of Biryukov-Demirci Equation satisfy key-dependent distribution under some specific constraints. This makes it possible to perform the key-dependent attack on IDEA. Biryukov-Demirci Equation is used to recover the intermediate values from encryptions.

Our key-dependent attack on 5.5-round variant of IDEA requires 2^{21} chosen plaintexts and has a time complexity of $2^{112.1}$ encryptions. Our key-dependent attack on the 6-round variant of IDEA requires 2^{49} chosen plaintexts and has a time complexity of $2^{112.1}$ encryptions. These attacks use both fewer chosen

Table 1. Selected Results of attacks on IDEA

Rounds	Attack type	Data	Time	Ref.
4.5	Impossible Differential	2^{64} CP	2^{112}	[3]
4.5	Linear	16 CP	2^{103}	[5]
5^\dagger	Meet-in-the-Middle	2^{24} CP	2^{126}	[13]
5^\dagger	Meet-in-the-Middle	$2^{24.6}$ CP	2^{124}	[11]
5	Linear	$2^{18.5}$ KP	2^{103}	[6]
5	Linear	2^{19} KP	2^{103}	[5]
5	Linear	16 KP	2^{114}	[6]
5.5	Higher-Order Differential-Linear	2^{32} CP	$2^{126.85}$	[6]
6	Higher-Order Differential-Linear	$2^{64} - 2^{52}$ KP	$2^{126.8}$	[6]
5^\dagger	Key-Dependent	2^{17} CP	$2^{125.5}$	Section 5.3
5^\dagger	Key-Dependent	2^{64} KP	$2^{115.3}$	Section 5.3
5.5	Key-Dependent	2^{21} CP	$2^{112.1}$	Section 5.1
6	Key-Dependent	2^{49} CP	$2^{112.1}$	Section 5.2

CP - Chosen Plaintext, KP - Known Plaintext.

\dagger Attack on IDEA starting from the first round.

plaintexts and less time than all the previous corresponding attacks. We also give two key-dependent attacks on 5-round IDEA starting from the first round. One requires 2^{17} chosen plaintexts and needs $2^{125.5}$ encryptions. The other one requires 2^{64} known plaintexts and needs $2^{115.3}$ encryptions. We summarize our attacks and previous attacks in Table [1](#), where the data complexity is measured in the number of plaintexts and the time complexity is measured in the number of encryptions needed in the attack.

The paper is organized as follows: In Section [2](#) we give a general view of the key-dependent attack. In Section [3](#) we give a brief description of IDEA block cipher. In Section [4](#) we show that the probability distribution of some items of the Biryukov-Demirci Equation is a key-dependent distribution. In Section [5](#) we present two key-dependent attacks on reduced-round IDEA. Section [6](#) concludes this paper.

2 The Key-Dependent Attack

In [\[2\]](#), Ben-Aroya and Biham first proposed the key-dependent property and implemented a key-dependent differential attack on Lucifer. Knudsen and Rijmen also used similar idea to attack DFC in [\[20\]](#).

In this section, we formalize a scheme of identifying the actual key using the following key-dependent property (with high success probability).

Definition 1. *For a block cipher, if the probability distribution of an intermediate value varies for different keys under some specific constraints, then this probability distribution is defined as key-dependent distribution.*

Consider some randomly chosen encryptions satisfying the specific constraints. If one uses the actual key to calculate the intermediate value, it should conform to key-dependent distribution. If one uses a wrong key to calculate the intermediate value, it is assumed to be randomly distributed. With such a property, determining whether a given key is right can be done by distinguishing which distribution the intermediate value conforms to, the key-dependent distribution or the random distribution.

We propose an attack scheme, called *key-dependent attack*, using key-dependent distribution. The attack uses statistical hypothesis test, whose idea is also used in differential and linear attack [17], to distinguish between key-dependent distribution and random distribution. For a key, the null hypothesis of the test is that the intermediate value conforms to the key-dependent distribution determined by the key. Then the attack uses some samples to determine whether the hypothesis is right. The samples of the statistical hypothesis test are the intermediate values obtained from the encryptions satisfying the specific constraints. If the key passes the hypothesis test, the attack concludes that the key is right, otherwise the key is judged to be wrong.

For the keys that share the same key-dependent distribution and the same intermediate value calculation, the corresponding hypothesis tests can be merged. Hence the whole key space is divided into several key-dependent subsets. (Similar idea is proposed in [2].)

Definition 2. A key-dependent subset is a tuple (P, U) , where P is a fixed key-dependent distribution of intermediate value, and U is a set of keys that share the same key-dependent distribution P and the same intermediate value calculation.

Definition 3. The key fraction (f) of a key-dependent subset is the ratio between the size of U and the size of the whole key space.

The key-dependent attack determines which key-dependent subset the actual key is in by conducting hypothesis tests on each key-dependent subset. Such process on a key-dependent subset (P, U) , called *individual attack*, can be described as the following four phases:

1. **Parameter Determining Phase** Determine the size of the samples and the criteria of rejecting the hypothesis that the intermediate values conform to P .
2. **Data Collecting Phase** Randomly choose some encryptions according to the specific constraints.¹
3. **Judgement Phase** Calculate the intermediate values from the collected encryptions. If the results satisfy the criteria of rejection, then discard this key-dependent subset, otherwise enter the next phase.
4. **Exhaustive Search Phase** Exhaustively search U to find the whole key. If the exhaustive search does not find the whole actual key, then start another individual attack on the next key-dependent subset.

¹ Though each individual attack chooses encryptions randomly, one encryption can be used for many individual attacks thus to reduce the total data complexity.

The time complexity of the key-dependent attack is determined by the time complexity of each individual attack and the order of performing these individual attacks.

For a key-dependent subset (P, U) , the time needed for individual attacks relies on the entropy of P : the closer P is to the random distribution, the more difficult the attack is—to ensure the same probability of making the right judgement, the attack needs more encryptions. This indicates that individual attacks for different key-dependent subsets have different time complexities. The time complexity of each individual attack is determined by corresponding key-dependent distribution P . For each key-dependent subset, the number of encryptions and the criteria of rejecting hypothesis are then chosen to minimize the time complexity of this individual attack.

To minimize the time complexity of an individual attack, the attack should consider the probability of committing two types of errors: Type I error and Type II error. Type I error occurs when the hypothesis is rejected for a key-dependent subset while in fact the actual key is in U , and the attack will fail to find the actual key in this case. The probability of Type I error is also defined as significant level, denoted as α . Type II error occurs when the test is passed while in fact it is not right, and in this case the attack will come into the exhaustive search phase, but will not find the actual key. The probability of Type II error is denoted as β . With a fixed size of samples (denoted as N) and the significance level α , the criteria of rejecting the hypothesis is determined, and the probability of Type II error β is also fixed. For a fixed size of samples, it is impossible to reduce both α and β simultaneously. In order to reduce both α and β , the attack has to use a larger size of samples, but time and data complexity will increase. Hence, an individual attack needs to balance between the size of samples, and the probability of making wrong judgement.

For a key-dependent subset (P, U) , if the actual key is not in this subset, the expected time complexity (measured by the number of encryptions) of the individual attack on this subset is

$$W = N + \beta|U| \quad (1)$$

If the actual key is in this subset, the expected time of the individual attack on this subset is

$$R = N + (1 - \alpha) \frac{|U|}{2}$$

Since the time complexity is dominated by attacking on wrong key-dependent subsets (there is only one key-dependent subset containing the actual key), the attack only needs to minimize the time complexity of the individual attack for each wrong key-dependent subset to minimize the total time complexity. Although α does not appear in Equation (1), α affects the success probability of the attack, so α should also be considered. We set one upper bound of α to ensure that the success probability is above a fixed value, and then choose such size of samples that Equation (1) is minimized, in order to minimize the time complexity of individual attacks.

In addition, it is entirely possible that some key-dependent distributions is so close to random distribution that the expected time for performing hypothesis tests is longer than directly searching the subsets. For these key-dependent subsets, the attack exhaustively searches the subset directly instead of using statistical hypothesis test method.

On the other hand, the time complexity of the key-dependent attack is also affected by the order of performing individual attacks on different key-dependent subsets. Because the expected time complexities of individual attacks are different, different sequences of performing individual attacks result in different total expected time complexity. Assume that a key-dependent attack performs individual attacks on m key-dependent subsets in the order of $(P_1, U_1), \dots, (P_m, U_m)$. Let R_i denote the expected time for (P_i, U_i) if the actual key is in U_i , and W_i denote the expected time if the actual key is not in U_i . We have following result:

Theorem 1. *The expected time for the whole key-dependent attack is minimal if the following condition is satisfied*

$$\frac{f_1}{W_1} \geq \frac{f_2}{W_2} \geq \dots \geq \frac{f_m}{W_m}$$

Proof. The expected time of the attack in the order of $(P_1, U_1), \dots, (P_m, U_m)$ is

$$\begin{aligned} \Phi &= f_1[R_1 + \alpha(W_2 + W_3 + \dots + W_m)] + f_2[W_1 + R_2 + \alpha(W_3 + \dots + W_m)] \\ &\quad + f_3[W_1 + W_2 + R_3 + \alpha(W_4 + \dots + W_m)] + \dots + f_m(W_1 + W_2 + \dots + W_{m-1} + R_m) \\ &= \sum_{i=1}^m f_i R_i + \sum_{i=1}^m (f_i \sum_{j=1}^{i-1} W_j) + \alpha \sum_{i=1}^m (f_i \sum_{j=i+1}^m W_j) \end{aligned} \tag{2}$$

If the attack is performed in the order of $(P_{s_1}, U_{s_1}), (P_{s_2}, U_{s_2}), \dots, (P_{s_m}, U_{s_m})$, where s_1, s_2, \dots, s_m is a permutation of $1, 2, \dots, m$. The expected time is

$$\Phi' = \sum_{i=1}^m f_{s_i} R_{s_i} + \sum_{i=1}^m (f_{s_i} \sum_{j=1}^{i-1} W_{s_j}) + \alpha \sum_{i=1}^m (f_{s_i} \sum_{j=i+1}^m W_{s_j})$$

$f_i W_j + \alpha f_j W_i$ occurs in Φ if and only if $j < i$ and occurs in Φ' if and only if $j' < i'$ where $s_{i'} = i$ and $s_{j'} = j$. Hence

$$\Phi - \Phi' = \sum_{j < i \text{ and } j' > i'} (f_i W_j + \alpha f_j W_i - f_j W_i - \alpha f_i W_j)$$

Since $\alpha \leq 1$ and $f_i W_j - f_j W_i \leq 0$ for $j < i$, $\Phi - \Phi' \leq 0$ for any permutation s_1, s_2, \dots, s_m . \square

In the following sections of this paper, we present a concrete key-dependent attack on the block cipher IDEA.

3 The IDEA Block Cipher

In this section, we give a brief introduction of IDEA and notations used later in this paper.

IDEA block cipher encrypts a 64-bit plaintext with a 128-bit key by an 8.5-round encryption. The fifty-two 16-bit subkeys are generated from the 128-bit key Z by key-schedule algorithm. The subkeys are generated in the order $Z_1^1, Z_2^1, \dots, Z_6^1, Z_1^2, \dots, Z_6^8, Z_1^9, \dots, Z_4^9$. The key Z is partitioned into eight 16-bit words which are used as the first eight subkeys. The key Z is then cyclically shifted to the left by 25 bits, and then generate the following eight subkeys. This process is repeated until all the subkeys are obtained. In Table 2 the correspondence between the subkeys and the key Z is directly given.

The block cipher partitions the 64-bit plaintext into four 16-bit words and uses three different group operations on pairs of 16-bit words: exclusive OR, denoted by \oplus ; modular addition 2^{16} , denoted by \boxplus and modular multiplication $2^{16} + 1$ (0 is treated as 2^{16}), denoted by \odot .

As Figure 1, each round of IDEA contains three layers: KA layer, MA layer and Permutation layer. We denote the 64-bit input of round i by $X^i = (X_1^i, X_2^i, X_3^i, X_4^i)$. In the KA layer, the first and the fourth words are modular multiplied with Z_1^i and Z_4^i respectively. The second and the third words are modular added with Z_2^i and Z_3^i respectively. The output of the KA layer is denoted by $Y^i = (Y_1^i, Y_2^i, Y_3^i, Y_4^i)$.

In the MA layer, two intermediate values $p^i = Y_1^i \oplus Y_3^i$ and $q^i = Y_2^i \oplus Y_4^i$ are computed first. These two values are processed to give u^i and t^i ,

$$u^i = (p^i \odot Z_5^i) \boxplus t^i$$

$$t^i = ((p^i \odot Z_5^i) \boxplus q^i) \odot Z_6^i$$

We denote s^i the intermediate value $p^i \oplus Z_5^i$ for convenience. The output of the MA layer is then permuted to give the output of this round $(Y_1^i \oplus u^i, Y_3^i \oplus u^i, Y_2^i \oplus t^i, Y_4^i \oplus t^i)$, which is also the input of round $i + 1$, denoted by $(X_1^{i+1}, X_2^{i+1}, X_3^{i+1}, X_4^{i+1})$. The complete diffusion, which means every bit of $(X_1^{i+1}, X_2^{i+1}, X_3^{i+1}, X_4^{i+1})$ is affected by every bit of $(Y_1^i, Y_2^i, Y_3^i, Y_4^i)$, is obtained in the MA layer.

Table 2. The Key-Schedule of IDEA

Round	Z_1^i	Z_2^i	Z_3^i	Z_4^i	Z_5^i	Z_6^i
1	0-15	16-31	32-47	48-63	64-79	80-95
2	96-111	112-127	25-40	41-56	57-72	73-88
3	89-104	105-120	121-8	9-24	50-65	66-81
4	82-97	98-113	114-1	2-17	18-33	34-49
5	75-90	91-106	107-122	123-10	11-26	27-42
6	43-58	59-74	100-115	116-3	4-19	20-35
7	36-51	52-67	68-83	84-99	125-12	13-28
8	29-44	45-60	61-76	77-92	93-108	109-124
9	22-37	38-53	54-69	70-85		

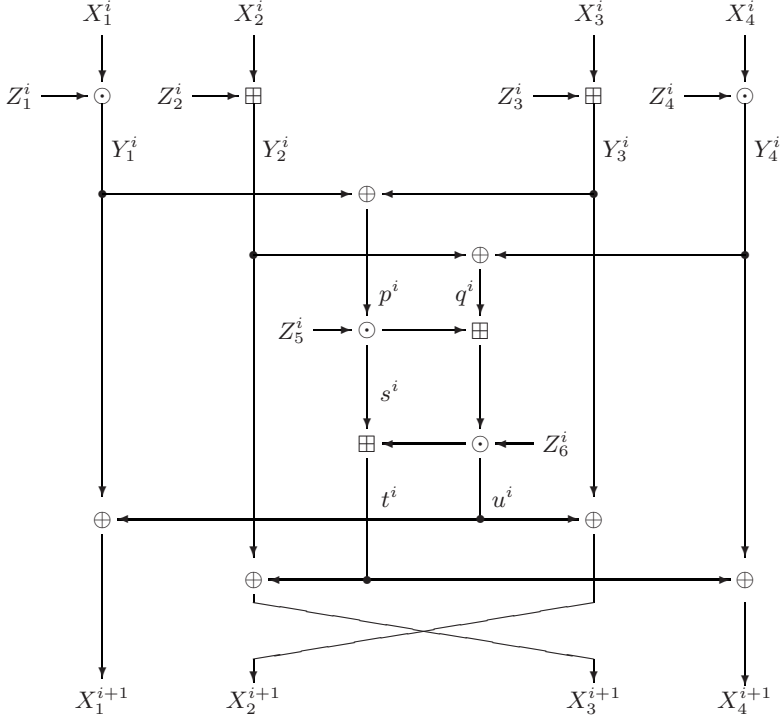


Fig. 1. Round i of IDEA

In this paper, we will use $P = (P_1, P_2, P_3, P_4)$ and $P' = (P'_1, P'_2, P'_3, P'_4)$ to denote a pair of plaintexts, where P_i and P'_i are 16-bit words. $C = (C_1, C_2, C_3, C_4)$ and $C' = (C'_1, C'_2, C'_3, C'_4)$ are their ciphertexts respectively. We also use the symbol $'$ to distinguish the intermediate values corresponding to P' from to P . For example, s^i is obtained from plaintext P and P' will generate s'^i . The notation Δ will denote the XOR difference, for instance, Δs^i is equal to $s^i \oplus s'^i$.

4 The Key-Dependent Distribution of IDEA

In this section, we describe the key-dependent distribution of the block cipher IDEA, which will be used in our attack later. The notations used are the same as in [6].

The Biryukov-Demirci relation was first proposed by Biryukov [16] and Demirci [13]. Many papers have discussed attacking on IDEA using this relation, such as [15, 6, 13, 16, 18]. The relation can be written in following form (LSB denotes the least significant bit)

$$\begin{aligned}
LSB(C_2 \oplus C_3) = & LSB(P_2 \oplus P_3 \oplus Z_2^1 \oplus Z_3^1 \oplus s^1 \oplus Z_2^2 \oplus Z_3^2 \oplus s^2 \\
& \oplus Z_2^3 \oplus Z_3^3 \oplus s^3 \oplus Z_2^4 \oplus Z_3^4 \oplus s^4 \oplus Z_2^5 \oplus Z_3^5 \oplus s^5 \\
& \oplus Z_2^6 \oplus Z_3^6 \oplus s^6 \oplus Z_2^7 \oplus Z_3^7 \oplus s^7 \oplus Z_2^8 \oplus Z_3^8 \oplus s^8 \\
& \oplus Z_2^9 \oplus Z_3^9)
\end{aligned} \tag{3}$$

It is shown in [5] that, for two pairs of plaintext and ciphertext (P, C) and (P', C') , XOR their corresponding Biryukov-Demirci relation, we will obtain from Equation (3)

$$\begin{aligned}
LSB(C_2 \oplus C_3 \oplus C'_2 \oplus C'_3) = & LSB(P_2 \oplus P_3 \oplus P'_2 \oplus P'_3 \oplus \Delta s^1 \oplus \Delta s^2 \\
& \oplus \Delta s^3 \oplus \Delta s^4 \oplus \Delta s^5 \oplus \Delta s^6 \oplus \Delta s^7 \oplus \Delta s^8)
\end{aligned} \tag{4}$$

We call Equation (4) *Biryukov-Demirci Equation*.

The following theorem shows that the probability distribution of $LSB(\Delta s^i)$ in Biryukov-Demirci Equation is a key-dependent distribution.

Theorem 2. *Consider round i of IDEA. If one pair of intermediate value (p^i, p'^i) satisfies $\Delta p^i = 8000_x$, then the probability of $LSB(\Delta s^i) = LSB(8000_x \odot Z_5^i)$ is*

$$Prob(LSB(\Delta s^i) = LSB(8000_x \odot Z_5^i)) = \frac{\#W}{2^{15}} \tag{5}$$

where W is the set of all such 16-bit words w that $1 \leq w \leq 8000_x$ and that

$$(w * Z_5^i) + (8000_x * Z_5^i) < 2^{16} + 1$$

where $*$ is defined as

$$a * b = \begin{cases} a \odot b & \text{if } a \odot b \neq 0 \\ 2^{16} & \text{if } a \odot b = 0 \end{cases}$$

Proof. Consider every intermediate pair (p^i, p'^i) which satisfies $\Delta p^i = 8000_x$, excluding $(0, 8000_x)$. We have $p'^i = p^i + 8000_x$ or $p^i = p'^i + 8000_x$. Without losing generality, assume $p'^i = p^i + 8000_x$, where $1 \leq p^i < 8000_x$ and $8000_x < p'^i < 2^{16}$.

If we consider only the least significant bit, $LSB(s^i) = LSB(p^i * Z_5^i)$. The following equations also hold

$$\begin{aligned}
LSB(s^i) = & LSB(p'^i \odot Z_5^i) \\
= & LSB(p'^i * Z_5^i) \\
= & LSB((p^i + 8000_x) * Z_5^i) \\
= & LSB(((p^i * Z_5^i) + (8000_x * Z_5^i)) \pmod{2^{16} + 1})
\end{aligned} \tag{6}$$

In the special case when (p^i, p'^i) is $(0, 8000_x)$, let $p^i = 8000_x$, and $p'^i = 0$. The Equations (6) also holds, because $p'^i = 0$ is actually treated as 2^{16} for inputs of \odot and $*$.

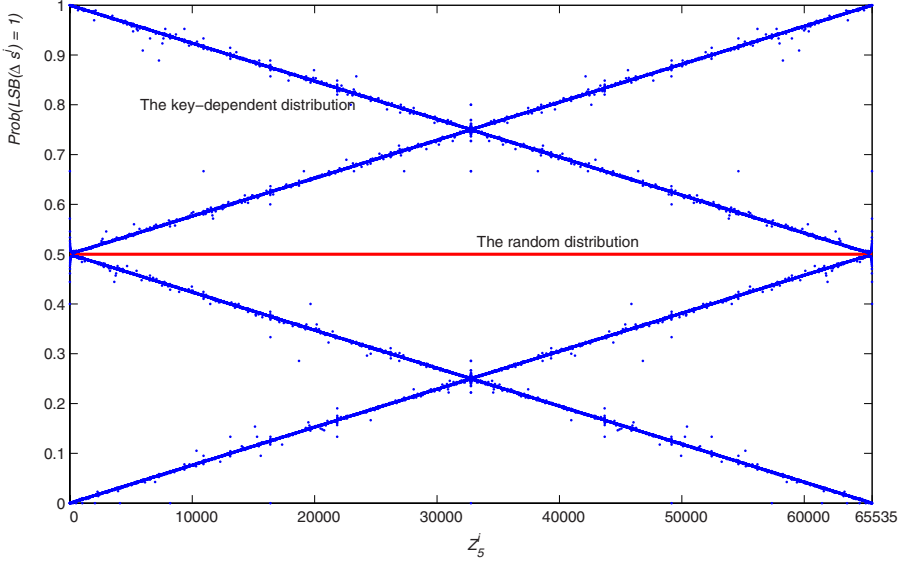


Fig. 2. The key-dependent distribution of $Prob(LSB(\Delta s) = 1)$ on the value of Z_5^i

If $(p^i * Z_5^i) + (8000_x * Z_5^i)$ is smaller than $2^{16} + 1$, then $LSB(s'^i) = LSB(s^i) \oplus LSB(8000_x * Z_5^i)$ holds because of the equivalence of XOR and modular addition for the least significant bit. Moreover, $LSB(\Delta s^i) = LSB(8000_x * Z_5^i)$ is satisfied, which means $LSB(\Delta s^i) = LSB(8000_x \odot Z_5^i)$.

Otherwise, $LSB(s'^i)$ is equal to $LSB(s^i) \oplus LSB(8000_x * Z_5^i) \oplus 1$ because of the carry. So $LSB(\Delta s^i)$ equals to $LSB(8000_x \odot Z_5^i) \oplus 1$.

Therefore, we may conclude that $LSB(\Delta s^i) = LSB(8000_x \odot Z_5^i)$ if and only if the pair (p^i, p'^i) satisfies $(w * Z_5^i) + (8000_x * Z_5^i) < 2^{16} + 1$, where w is either p^i or p'^i , whichever between 1 and 8000_x . And there are at most 2^{15} such w , hence Equation (6) holds. This completes the proof. \square

Remark 1. Figure 2 plots the relation between the subkey Z_5^i and the probability of $LSB(\Delta s^i) = 1$. As shown in Figure 2, for most Z_5^i , the probability of $LSB(\Delta s^i) = 1$ is different from random distribution. Hence, it is possible to perform key-dependent attack on IDEA using this key-dependent distribution.

For most Z_5^i , there are general four cases for the probability of $LSB(\Delta s^i) = 1$ as Z_5^i grows from 0 to $2^{16} - 1$, which can be roughly approximated as following:

$$Prob(LSB(\Delta s^i) = 1) \approx \begin{cases} \frac{Z_5^i}{2^{17}} & \text{last two bits of } Z_5^i = 00 \\ 0.5 - \frac{Z_5^i}{2^{17}} & \text{last two bits of } Z_5^i = 01 \\ 1.0 - \frac{Z_5^i}{2^{17}} & \text{last two bits of } Z_5^i = 10 \\ 0.5 + \frac{Z_5^i}{2^{17}} & \text{last two bits of } Z_5^i = 11 \end{cases} \quad (7)$$

From Equation (7), following approximation also holds for most Z_5^i

$$\min\{Prob(LSB(\Delta s^i) = 0), Prob(LSB(\Delta s^i) = 1)\} \approx \begin{cases} \frac{Z_5^i}{2^{17}}, & LSB(Z_5^i) = 0 \\ 0.5 - \frac{Z_5^i}{2^{17}}, & LSB(Z_5^i) = 1 \end{cases} \quad (8)$$

Calculation shows that, for only 219 out of all 2^{16} possible Z_5^i , the difference between the approximation (Equation (7) or (8)) and the accurate provability is larger than 0.01.

Equation (8) indicates that we can approximate left hand side of Equation (8) by fixing several most significant bits and the least significant bit. In following sections, we will show that we only need to distinguish the approximate probability distribution from random distribution. Hence, for most Z_5^i , this approximation is close enough to the accurate value. For Z_5^i that can not be approximated in this way, we use other methods to deal with this situation.

5 The Key-Dependent Attack on IDEA

In this section, we will present two key-dependent attacks on reduced-round IDEA. In Section 5.1, we will give a basic attack on the 5.5-round variant of IDEA and then extend it to 6-round variant in Section 5.2. We also give two key-dependent attacks on 5-round IDEA starting from the first round in Section 5.3.

5.1 The Attack on 5.5-Round Variant of IDEA

We first present one key-dependent attack on the 5.5-round variant of IDEA. The attack starts from the third round and ends before the MA layer of the eighth round. The main idea of this attack is to perform key-dependent attack based on the key-dependent distribution of Δs^4 described in Theorem 2.

Consider the 5.5-round variant of IDEA starting from the third round, the Biryukov-Demirci Equation can be rewritten as

$$LSB(\Delta s^4) = LSB(P_2 \oplus P_3 \oplus P_2' \oplus P_3' \oplus C_2 \oplus C_3 \oplus C_2' \oplus C_3' \oplus \Delta s^3 \oplus \Delta s^5 \oplus \Delta s^6 \oplus \Delta s^7) \quad (9)$$

Where P and P' are equivalent to X^3 and X'^3 , C and C' are equivalent to Y^8 and Y'^8 by the variant of IDEA.

We first construct a pair of plaintexts satisfying the specific constraint $\Delta p^4 = 8000x$. The construction is based on the following lemma.

Lemma 1. *For any α , if two 16-bit words x and x' have the same least 15 significant bits, then*

- $x \oplus \alpha$ and $x' \oplus \alpha$ have the same least 15 significant bits,
- $x \boxplus \alpha$ and $x' \boxplus \alpha$ have the same least 15 significant bits.

Based on Lemma 1, the following proposition can be obtained.

Proposition 1. *If a pair of intermediate values Y^3 and Y'^3 satisfy the following conditions:*

- a. $\Delta Y_1^3 = \Delta Y_3^3 = 0$
- b. $\Delta Y_2^3 = 8000_x$
- c. $Y_2^3 \oplus Y_4^3 = Y_2'^3 \oplus Y_4'^3$

then $\Delta s^3 = 0$ and the probability of $LSB(\Delta s^4) = 0$ can be determined by Equation (5).

Proof. From Condition (a), $\Delta Y_1^3 = \Delta Y_3^3 = 0$, p^3 is equal to q'^3 . Then $\Delta s^3 = 0$ is quite straightforward.

From Condition (c), q^3 is equal to q'^3 . If p^3 and q^3 are fixed, u^3 and t^3 are also fixed with respect to any Z_5^3 and Z_6^3 . It indicates that $X_1^4 = Y_1^3 \oplus u^3 = X_1'^4$. Note that Y_1^4 and $Y_1'^4$ are the results of modular-multiplying X_1^4 and $X_1'^4$ with the same Z_1^4 , hence Y_1^4 is equal to $Y_1'^4$.

On the other hand, $\Delta Y_2^3 = 8000_x$ means that the least significant 15 bits of Y_2^3 are equal to those of $Y_2'^3$ and the most significant bit of Y_2^3 and that of $Y_2'^3$ are different. Because u^3 is fixed, by Lemma 1, the least significant 15 bits of X_3^4 are equal to those of $X_3'^4$. Then ΔX_3^4 is equal to 8000_x and $\Delta Y_3^4 = 8000_x$ is obtained by modular addition with the same Z_3^4 . From $\Delta Y_1^4 = 0$ and $\Delta Y_3^4 = 8000_x$, Δp^4 is 8000_x . By Theorem 2, the conclusion is obtained. \square

In our attack, we use the plaintext pairs satisfying Proposition 1. We obtain Condition (a) by letting $\Delta P_1 = \Delta P_3 = 0$. By Lemma 2, P_2 and P_2' are fixed to have the same least significant 15 bits, and hence $\Delta Y_2^1 = 8000_x$. In order to fulfill Condition (c), we have to guess Z_4^3 and then according to this guess, to choose P_4 and P_4' which satisfy $\Delta Y_4^3 = 8000_x$.

By Proposition 1, Δs^3 is equal to zero. In order to get the right hand side of Equation (9), we still need to get Δs^5 , Δs^6 , Δs^7 . We need to guess Z_5^5 , Z_1^6 , Z_2^6 , Z_5^6 , Z_6^6 , Z_1^7 , Z_2^7 , Z_3^7 , Z_4^7 , Z_5^7 , Z_6^7 , Z_1^8 , Z_2^8 , Z_3^8 , Z_4^8 . As shown in [6], one can partially decrypt one pair of encryptions using these 15 subkeys to calculate the values of Δs^5 , Δs^6 , Δs^7 . These 15 subkeys only take key bits 125-99 and also cover the subkey Z_4^3 . Hence, for one guessed 103 key bits, we can calculate the value of Δs^4 from a special pair of encryptions.

We also note that these 103 bits also cover the key Z_5^4 , which determine the key-dependent distribution on Δs^4 according to Theorem 2. Therefore, we can perform the key-dependent attack on 5.5-round variant of IDEA. As described in Section 2, the key space can be divided into 2^{103} key-dependent subsets by the 103 key bits, each contains 2^{25} keys.

For a key-dependent subset (P, U) , let p denote the probability of $LSB(\Delta s^4) = LSB(8000_x \odot Z_5^4)$. For simplicity, in the following analysis, we assume that $p \leq 0.5$, the case when $p > 0.5$ is similar. Assume the size of the samples is n pairs of encryptions that satisfy the specific constraint on this key-dependent subset, and t of them satisfy $LSB(\Delta s^4) = LSB(8000_x \odot Z_5^4)$. The criteria for not rejecting the hypothesis is that t is smaller or equal to a fixed value k . The probability of

Type I error is

$$\alpha = \sum_{i=k+1}^n \binom{n}{i} p^i (1-p)^{n-i}$$

Type II error is

$$\beta = \sum_{i=0}^k \binom{n}{i} 0.5^n$$

If (P, U) is a wrong key-dependent subset, the expected time complexity of checking this subset is

$$W = 2n + 2^{25}\beta \quad (10)$$

As shown in Section 2, the attack sets α smaller than or equal to 0.01 to ensure that the probability of the false rejection will not exceed 0.01. Under this precondition, the attack chooses n and β so that $\alpha < 0.01$ and minimizes Equation (10) to minimize the time complexity on each key-dependent subset (P, U) . By Section 2, we minimize the total expected time complexity with this method. Because this choice is related only to the key Z_5^4 , so we only need to get n and k for 2^{16} different values.

For example, for a key-dependent subset (P, U) with $Z_5^4 = 8001_x$, p is about 0.666687. The attack checks every possible n and k to find the minimized expected time complexity of the individual attack for this subset. As shown in Section 2, the expected time complexity for each subset is upper bounded by exhaustive search on the subset, which is 2^{25} in this attack. Hence, the attack only

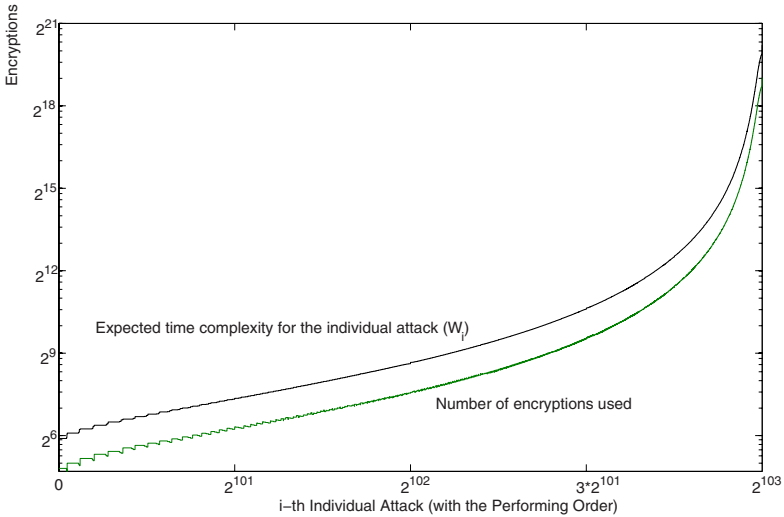


Fig. 3. The number of encryptions used and expected time complexity for individual attacks

checks all the n and k smaller than 2^{25} . The expected time is minimized with precondition $\alpha < 0.01$ when $n = 425$ and $k = 164$. In this case, $\alpha = 0.009970$, $\beta = 0.000001$ and $W = 899.094678$.

Since all the key-dependent subsets have the same key fraction, the order of performing individual attacks with minimal expected time complexity becomes the ascending order of W for all key-dependent subsets due to Theorem [1](#). Figure [3](#) plots the number of encryptions used and expected time complexity for all the individual attacks.

The total expected time complexity of the attack, described as Equation [\(2\)](#), becomes

$$\begin{aligned}
\Phi &= \sum_{i=1}^m f_i R_i + \sum_{i=1}^m (f_i \sum_{j=1}^{i-1} W_j) + \alpha \sum_{i=1}^m (f_i \sum_{j=i+1}^m W_j) \\
&= \frac{1}{2^{103}} \left(\sum_{i=1}^{2^{103}} R_i + \sum_{i=1}^{2^{103}} \sum_{j=1}^{i-1} W_j + 0.01 \sum_{i=1}^{2^{103}} \sum_{j=i+1}^{2^{103}} W_j \right) \\
&\leq \frac{1}{2^{103}} \left(\sum_{i=1}^{2^{103}} 2^{26} + \sum_{i=1}^{2^{103}} \sum_{j=1}^{i-1} W_j + 0.01 \sum_{i=1}^{2^{103}} \sum_{j=i+1}^{2^{103}} W_j \right) \\
&= \frac{1}{2^{103}} \left(2^{103} \cdot 2^{26} + \sum_{i=1}^{2^{103}} (2^{103} - i + 0.01i) W_i \right) \\
&\approx 2^{112.1}
\end{aligned}$$

with 99% success probability if the attack chooses n and β for each key-dependent set and determines the order of performing individual attacks as shown above. The number of pairs needed in one test is about 2^{19} in the worst case. The attack uses a set of 2^{21} plaintexts, which can provide 2^{20} plaintext pairs satisfying the conditions in Proposition 1 for each key-dependent subset.

The attack is summarized as follows:

1. For every possible Z_5^4 , calculate the corresponding number of plaintext pairs needed n and the criteria of not rejecting the hypothesis k .
2. Suppose S is an empty set. Randomly enumerate a 16-bit word s , insert s and $s \oplus 8000_x$ into the set S . Repeat this enumeration until set S contains 2^5 different words. Ask for the encryption of all the plaintexts of the form (A, B, C, D) , where A and C are fixed to two arbitrary constants, B takes all the values in S and D takes all the 16-bit possible values.
3. Enumerate the key-dependent sets in ascending order of W :
 - (a) Randomly choose a set of plaintext pairs with cardinality n from the known encryptions. The plaintext pairs must satisfy the requirements of Proposition [1](#).
 - (b) Partially decrypt all the selected encryption pairs and count the occurrence of $LSB(\Delta_{s_4}) = 1$.
 - (c) Test the hypothesis. If the hypothesis is not rejected, perform exhaustive search for the remaining 25 key bits.

5.2 The Attack on 6-Round Variant of IDEA

We now extend the 5.5-round attack to an attack on the 6-round variant of IDEA starting before the MA layer of the second round. The data complexity of the attack is 2^{49} and the time complexity is $2^{112.1}$.

As shown in [6], Z_5^2 and Z_6^2 are included in the 103 key bits in the 5.5-round attack. Hence, we can add this half round to the 5.5-round attack without enlarging the time complexity.

It is more difficult to construct right plaintext pairs satisfying Proposition 1. Consider a pair of intermediate values X^3 and X'^3 before the third round, which satisfy Proposition 1. If we partially decrypt X^3 and X'^3 using any possible Z_5^2 and Z_6^2 , the only fact we know is that all the results have the same XOR of the first and third words. The attack hence selects all the plaintexts P where the least 15 significant bits of $P_1 \oplus P_3$ are fixed to an arbitrary 15-bit constant. The total number of selected plaintexts is 2^{49} . It is possible to provide 2^{48} plaintext pairs satisfying the conditions in Proposition 1 in the test for any Z_5^2 , Z_6^2 and Z_4^3 . This number is sufficient in any situation.

5.3 Two Key-Dependent Attacks on 5-Round IDEA Starting from the First Round

We apply the key-dependent attack to the 5-round IDEA starting from the first round. Biryukov-Demirci Equation is reduced to

$$\begin{aligned} LSB(\Delta s^2) = & LSB(P_2 \oplus P_3 \oplus P'_2 \oplus P'_3 \oplus C_2 \\ & \oplus C_3 \oplus C'_2 \oplus C'_3 \oplus \Delta s^1 \oplus \Delta s^3 \oplus \Delta s^4 \oplus \Delta s^5) \end{aligned} \quad (11)$$

We choose the plaintext pairs to satisfy Proposition 1 before the first round by guessing Z_4^1 , and then Δs^1 is equal to 0 as shown in Section 5.1. In order to determine the right hand side of Equation (11), we need to know Z_5^3 , Z_1^4 , Z_2^4 , Z_5^4 , Z_6^4 , Z_1^5 , Z_2^5 , Z_3^5 , Z_4^5 , Z_5^5 , Z_6^5 . These 12 subkeys take the bits 75-65 from key Z . These 119 bits only cover the most significant nine bits of Z_5^2 , which determines the probability distribution of $LSB(\Delta s^2)$. It is not necessary to guess the complete subkey Z_5^2 . The attack continues to guess the least significant bit of Z_5^2 (the 72nd bit of Z), and estimates the probability of $LSB(\Delta s^2) = 1$ by Remark 1 instead. Hence, the attack divides the key space into 2^{120} key-dependent subsets by the 120 key bits, and performs the individual attacks on each key-dependent subset. The attack uses statistical hypothesis test method to determine which subset the actual key is in. For the subkeys Z_5^2 of which $Prob(LSB(\Delta s^2) = 1)$ can not be approximated by Remark 1 as shown in Section 4, the attack exhaustively searches the remaining key bits.

In this attack, it is possible that the expected time of individual attacks are larger than exhaustively search directly for some key-dependent subsets, which means

$$2n + \beta \cdot 2^8 \geq 2^8$$

Under this condition, the attack also uses exhaustive key search to determine the remaining eight key bits to make sure the time needed not exceed exhaustive search.

This attack also choose $\alpha \leq 0.01$ to ensure that the attack successes with 99% probability. In this case, the total expected time complexity is $2^{125.5}$ encryptions.

Our experiment shows that the attack needs at most 75 pairs of encryptions for one test. We ask for 2^{17} encryptions which can provide 2^{16} pairs of encryptions, which is sufficient for the test. This data complexity(2^{17}) is the least out of all the known attacks on the 5-round IDEA starting from the first round.

In the second attack, we try to obtain the plaintext pairs satisfying Proposition [11](#) before the second round. In order to determine $LSB(\Delta s^3)$, we need to know the least significant bits of Δs^1 , Δs^2 , Δs^4 and Δs^5 . Hence, the subkeys we need to know are Z_1^1 , Z_2^1 , Z_3^1 , Z_4^1 , Z_5^1 , Z_6^1 , Z_4^2 , Z_5^2 , Z_5^3 , Z_5^4 , Z_1^5 , Z_2^5 , Z_5^5 and Z_6^5 . These 13 subkeys only cover 107 bits of key Z (0-106). For every guessed 107 key bits, we use similar technique as before. The expected time complexity is $2^{115.3}$, which is the least time complexity out of all the known attacks on the 5-round IDEA starting from the first round.

Because it is not possible to predict the plaintext pairs which produces the intermediate pairs satisfying Proposition [11](#) before the second round, the encryptions of all the 2^{64} plaintexts are required.

6 Conclusions

In this paper, we formalized a scheme of identifying the actual key using the key-dependent distribution, called key-dependent attack. How to minimize the time complexity of the key-dependent attack was also discussed. With the key-dependent attack, we could improve known cryptanalysis results and obtain more powerful attacks. We presented two key-dependent attacks on IDEA. Our attack on 5.5-round and 6-round variant of IDEA has the least time and data complexities compared with the previous attacks.

We only implemented a tentative exploration of the key-dependent distribution. How to make full use of the key-dependent distribution, especially how to use the key-dependent distribution to improve existing attacks, is worth further studying.

The attack on IDEA makes use of the relation between XOR, modular addition and modular multiplication. We believe that the operation XOR and modular multiplication have more properties that can be explored further [\[10\]](#). Similar relations among other operations are also valuable to research. The way of making full use of the Biryukov-Demirci Equation to improve attacks on IDEA is also interesting.

References

1. Ayaz, E.S., Selçuk, A.A.: Improved DST Cryptanalysis of IDEA. In: Biham, E., Youssef, A.M. (eds.) SAC 2006. LNCS, vol. 4356, pp. 1–14. Springer, Heidelberg (2007)
2. Ben-Aroya, I., Biham, E.: Differential Cryptanalysis of Lucifer. J. Cryptology 9(1), 21–34 (1996)

3. Biham, E., Biryukov, A., Shamir, A.: Miss in the Middle Attacks on IDEA and Khufu. In: Knudsen, L.R. (ed.) FSE 1999. LNCS, vol. 1636, pp. 124–138. Springer, Heidelberg (1999)
4. Biham, E., Dunkelman, O., Keller, N.: Related-Key Boomerang and Rectangle Attacks. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 507–525. Springer, Heidelberg (2005)
5. Biham, E., Dunkelman, O., Keller, N.: New Cryptanalytic Results on IDEA. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 412–427. Springer, Heidelberg (2006)
6. Biham, E., Dunkelman, O., Keller, N.: A New Attack on 6-Round IDEA. In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 211–224. Springer, Heidelberg (2007)
7. 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)
8. Biryukov, A., Nakahara Jr., J., Preneel, B., Vandewalle, J.: New Weak-Key Classes of IDEA. In: Deng, R.H., Qing, S., Bao, F., Zhou, J. (eds.) ICICS 2002. LNCS, vol. 2513, pp. 315–326. Springer, Heidelberg (2002)
9. Borst, J., Knudsen, L.R., Rijmen, V.: Two Attacks on Reduced IDEA. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 1–13. Springer, Heidelberg (1997)
10. Contini, S., Rivest, R.L., Robshaw, M.J.B., Yin, Y.L.: Improved Analysis of Some Simplified Variants of RC6. In: Knudsen, L.R. (ed.) FSE 1999. LNCS, vol. 1636, pp. 1–15. Springer, Heidelberg (1999)
11. Daemen, J., Govaerts, R., Vandewalle, J.: Weak keys for IDEA. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 224–231. Springer, Heidelberg (1994)
12. Demirci, H.: Square-like Attacks on Reduced Rounds of IDEA. In: Nyberg, K., Heys, H.M. (eds.) SAC 2002. LNCS, vol. 2595, pp. 147–159. Springer, Heidelberg (2003)
13. Demirci, H., Selçuk, A.A., Türe, E.: A New Meet-in-the-Middle Attack on the IDEA Block Cipher. In: Matsui, M., Zuccherato, R.J. (eds.) SAC 2003. LNCS, vol. 3006, pp. 117–129. Springer, Heidelberg (2004)
14. Hawkes, P.: Differential-Linear Weak Key Classes of IDEA. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 112–126. Springer, Heidelberg (1998)
15. Hawkes, P., O'Connor, L.: On Applying Linear Cryptanalysis to IDEA. In: Kim, K.-c., Matsumoto, T. (eds.) ASIACRYPT 1996. LNCS, vol. 1163, pp. 105–115. Springer, Heidelberg (1996)
16. Nakahara Jr., J., Preneel, B., Vandewalle, J.: The Biryukov-Demirci Attack on Reduced-Round Versions of IDEA and MESH Ciphers. In: Wang, H., Pieprzyk, J., Varadharajan, V. (eds.) ACISP 2004. LNCS, vol. 3108, pp. 98–109. Springer, Heidelberg (2004)
17. Junod, P.: On the Optimality of Linear, Differential, and Sequential Distinguishers. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 17–32. Springer, Heidelberg (2003)
18. Junod, P.: New Attacks Against Reduced-Round Versions of IDEA. In: Gilbert, H., Handschuh, H. (eds.) FSE 2005. LNCS, vol. 3557, pp. 384–397. Springer, Heidelberg (2005)
19. Kelsey, J., Schneier, B., Wagner, D.: Key-Schedule Cryptoanalysis of IDEA, GDES, GOST, SAFER, and Triple-DES. In: Kobitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 237–251. Springer, Heidelberg (1996)

20. 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)
21. Lai, X.: On the Design and Security of Block Ciphers. ETH Series in Information Processing. Hartung-Gorre Verlag, Konstanz (1992)
22. Lai, X., Massey, J.L.: A Proposal for a New Block Encryption Standard. In: Damgård, I.B. (ed.) EUROCRYPT 1990. LNCS, vol. 473, pp. 389–404. Springer, Heidelberg (1991)
23. Matsui, M.: Linear Cryptanalysis Method for DES Cipher. In: Helleseeth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 386–397. Springer, Heidelberg (1994)
24. Meier, W.: On the Security of the IDEA Block Cipher. In: Helleseeth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 371–385. Springer, Heidelberg (1994)
25. Raddum, H.: Cryptanalysis of IDEA-X/2. In: Johansson, T. (ed.) FSE 2003. LNCS, vol. 2887, pp. 1–8. Springer, Heidelberg (2003)

Cascade Encryption Revisited

Peter Gaži^{1,2} and Ueli Maurer¹

¹ ETH Zürich, Switzerland

Department of Computer Science

{gazipete,maurer}@inf.ethz.ch

² Comenius University, Bratislava, Slovakia

Department of Computer Science

Abstract. The security of cascade blockcipher encryption is an important and well-studied problem in theoretical cryptography with practical implications. It is well-known that double encryption improves the security only marginally, leaving triple encryption as the shortest reasonable cascade. In a recent paper, Bellare and Rogaway showed that in the ideal cipher model, triple encryption is significantly more secure than single and double encryption, stating the security of longer cascades as an open question.

In this paper, we propose a new lemma on the indistinguishability of systems extending Maurer’s theory of random systems. In addition to being of independent interest, it allows us to compactly rephrase Bellare and Rogaway’s proof strategy in this framework, thus making the argument more abstract and hence easy to follow. As a result, this allows us to address the security of longer cascades. Our result implies that for blockciphers with smaller key space than message space (e.g. DES), longer cascades improve the security of the encryption up to a certain limit. This partially answers the open question mentioned above.

Keywords: cascade encryption, ideal cipher model, random system, indistinguishability.

1 Introduction

The cascade encryption is a simple and practical construction used to enlarge the key space of a blockcipher without the need to switch to a new algorithm. Instead of applying the blockcipher only once, it is applied l times with l independently chosen keys. A prominent and widely used example of this construction is the Triple DES encryption [2][3][4].

Many results investigating the power of the cascade construction have been published. It is well-known that double encryption does not significantly improve the security over single encryption due to the meet-in-the-middle attack [7]. The marginal security gain achieved by double encryption was described in [1]. Even and Goldreich [8] show that a cascade of ciphers is at least as strong as the strongest of the ciphers against attacks that are restricted to operating on full blocks. In contrast, Maurer and Massey [11] show that for the most general

attack model, where it is for example possible that an attacker might obtain only half the ciphertext block for a chosen message block, the cascade is only at least as strong as the *first* cipher of the cascade.

In a recent paper [4], Bellare and Rogaway have claimed a lower bound on the security of triple encryption in the ideal cipher model. Their bound implies that for a blockcipher with key length k and block length n , triple encryption is indistinguishable from a random permutation as long as the distinguisher is allowed to make not more than roughly $2^{k+\frac{1}{2}\min\{n,k\}}$ queries. This bound is significantly higher than the known upper bound on the security of single and double encryption, proving that triple encryption is the shortest cascade that provides a reasonable security improvement over single encryption. Since a longer cascade is at least as secure as a shorter one, their bound applies also to longer cascades. They formulate as an interesting open problem to determine whether the security improves with the length of the cascade also for lengths $l > 3$. However, the proof in [4] contains a few bugs, which we describe in the appendix of this paper. The first part of our contribution is to fix these errors and to reestablish the lower bound on the security of triple encryption up to a constant factor.

Second, we have rephrased the proof into the random systems framework introduced in [10]. Our goal here is to simplify the proof and express it on the most abstract level possible, thus making the main line of reasoning easy to follow and clearly separated from the two technical arguments required. To achieve this, we extend the random systems framework by a new lemma. This lemma is a generalization of both Lemma 7 from [10] and hence also of its special case for the game-playing scenario, the Fundamental lemma of game-playing. This was introduced in [4] and subsequently used as an important tool in the game-playing proofs (see for example [15,3,5]). We illustrate the use of this new lemma in our proof of the security of cascade encryption. Apart from the simplification, this also gives us an improvement of the result by a constant factor.

Finally, our reformulation makes it natural to consider also the security of longer cascades. The lower bound we prove improves with the length of the cascade l for all blockciphers where $k < n$ and for moderate values of l . With increasing cascade length, the bound approaches very roughly the value $2^{k+\min\{n/2,k\}}$ (the exact formula can be found in Theorem [1]). The condition $k < n$ is satisfied for example for the DES blockcipher, where the length of the key is 56 bits and the length of one block is 64 bits. For these parameters, the result from [4] that we reestablish proves that the triple encryption is secure up to 2^{78} queries, but our result shows that a cascade of length 5 is secure up to 2^{83} queries. The larger the difference $n - k$, the more a longer cascade can help. This partially answers the open question from [4].

2 Preliminaries

2.1 Basic Notation

Throughout the paper, we denote sets by calligraphic letters (e.g. \mathcal{S}). For a finite set \mathcal{S} , we denote by $|\mathcal{S}|$ the number of its elements. A k -tuple is denoted as

$u^k = (u_1, \dots, u_k)$, and the set of all k -tuples of elements of \mathcal{U} is denoted as \mathcal{U}^k . The composition of mappings is interpreted from left to right, i.e., $f \circ g$ denotes the mapping $g(f(\cdot))$. The set of all permutations of $\{0, 1\}^n$ is denoted by $\text{Perm}(n)$ and id represents the identity mapping, if the domain is implicitly given. The notation $x^{\underline{n}}$ represents the falling factorial power, i.e., $x^{\underline{n}} = x(x-1) \cdots (x-n+1)$. The symbol $p_{coll}(n, k)$ denotes the probability that k independent random variables with uniform distribution over a set of size n contain a collision, i.e., that they are not all distinct. It is well-known that $p_{coll}(n, k) < k^2/2n$. By $\text{CS}(\cdot)$ we shall denote the set of all *cyclic shifts* of a given tuple, in other words, $\text{CS}(\pi_1, \pi_2, \dots, \pi_r) = \{(\pi_1, \pi_2, \dots, \pi_r), (\pi_2, \pi_3, \dots, \pi_r, \pi_1), \dots, (\pi_r, \pi_1, \dots, \pi_{r-1})\}$.

We usually denote random variables and concrete values they can take on by capital and small letters, respectively. For events A and B and random variables U and V with ranges \mathcal{U} and \mathcal{V} , respectively, we denote by $\mathbb{P}_{U|V}$ the corresponding conditional probability distribution, seen as a function $\mathcal{U} \times \mathcal{V} \rightarrow \langle 0, 1 \rangle$. Here the value $\mathbb{P}_{U|V}(u, v)$ is well-defined for all $u \in \mathcal{U}$ and $v \in \mathcal{V}$ such that $\mathbb{P}_V(v) > 0$ and undefined otherwise. Two probability distributions \mathbb{P}_U and $\mathbb{P}_{U'}$ on the same set \mathcal{U} are equal, denoted $\mathbb{P}_U = \mathbb{P}_{U'}$, if $\mathbb{P}_U(u) = \mathbb{P}_{U'}(u)$ for all $u \in \mathcal{U}$. Conditional probability distributions are equal if the equality holds for all arguments for which both of them are defined. To emphasize the random experiment \mathcal{E} in consideration, we sometimes write it in the superscript, e.g. $\mathbb{P}_{U|V}^{\mathcal{E}}(u, v)$. The expected value of the random variable X is denoted by $\mathbb{E}[X] = \sum_{x \in \mathcal{X}} (x \cdot \mathbb{P}[X = x])$. The complement of an event A is denoted by \overline{A} .

2.2 Random Systems

In this subsection, we present the basic notions of the random systems framework, as introduced in [10], along with some new extensions of the framework. The input-output behavior of any discrete system can be described by a *random system* in the spirit of the following definition.

Definition 1. An $(\mathcal{X}, \mathcal{Y})$ -random system \mathbf{F} is a (generally infinite) sequence of conditional probability distributions $\mathbb{P}_{Y_i|X^i Y^{i-1}}^{\mathbf{F}}$ for all $i \geq 1$.

The behavior of the random system is specified by the sequence of conditional probabilities $\mathbb{P}_{Y_i|X^i Y^{i-1}}^{\mathbf{F}}(y_i, x^i, y^{i-1})$ (for $i \geq 1$) of obtaining the output $y_i \in \mathcal{Y}$ on query $x_i \in \mathcal{X}$ given the previous $i - 1$ queries $x^{i-1} = (x_1, \dots, x_{i-1}) \in \mathcal{X}^{i-1}$ and their corresponding outputs $y^{i-1} = (y_1, \dots, y_{i-1}) \in \mathcal{Y}^{i-1}$. A random system can also be defined by a sequence of conditional probability distributions $\mathbb{P}_{Y^i|X^i}^{\mathbf{F}}$ for $i \geq 1$. This description is often convenient, but is not minimal.

We shall use boldface letters (e.g. \mathbf{F}) to denote both a discrete system and a random system corresponding to it. This should cause no confusion. We emphasize that although the results of this paper are stated for random systems, they hold for arbitrary systems, since the only property of a system that is relevant here is its input-output behavior. It is reasonable to consider two discrete systems equivalent if their input-output behaviors are the same, even if their internal structure differs.

Definition 2. Two systems \mathbf{F} and \mathbf{G} are equivalent, denoted $\mathbf{F} \equiv \mathbf{G}$, if they correspond to the same random system, i.e., if $\mathbf{P}_{Y_i|X^iY^{i-1}}^{\mathbf{F}} = \mathbf{P}_{Y_i|X^iY^{i-1}}^{\mathbf{G}}$ for all $i \geq 1$.

We shall usually define a system (and hence also the corresponding random system) by a description of its internal working, as long as the transition to the probability distributions is straightforward. Examples of random systems that we consider in the following are the *uniform random permutation* $\mathbf{P} : \{0, 1\}^n \rightarrow \{0, 1\}^n$, which realizes a function randomly chosen from $\text{Perm}(n)$; and the *ideal blockcipher* $\mathbf{E} : \{0, 1\}^k \times \{0, 1\}^n \rightarrow \{0, 1\}^n$, which realizes an independent uniformly random permutation for each key $K \in \{0, 1\}^k$. In this paper we assume that both \mathbf{P} and \mathbf{E} can be queried in both directions.

We can define a *distinguisher* \mathbf{D} for an $(\mathcal{X}, \mathcal{Y})$ -random system as a $(\mathcal{Y}, \mathcal{X})$ -random system which is one query ahead, i.e., it is defined by the conditional probability distributions $\mathbf{P}_{X_i|X^{i-1}Y^{i-1}}^{\mathbf{D}}$ for all $i \geq 1$. In particular, the first query of \mathbf{D} is determined by $\mathbf{P}_{X_1}^{\mathbf{D}}$. After a certain number of queries (say q), the distinguisher outputs a bit W_q depending on the transcript (X^q, Y^q) . For a random system \mathbf{F} and a distinguisher \mathbf{D} , let \mathbf{DF} be the random experiment where \mathbf{D} interacts with \mathbf{F} . Then for two $(\mathcal{X}, \mathcal{Y})$ -random systems \mathbf{F} and \mathbf{G} , the *distinguishing advantage* of \mathbf{D} in distinguishing systems \mathbf{F} and \mathbf{G} by q queries is defined as $\Delta_q^{\mathbf{D}}(\mathbf{F}, \mathbf{G}) = |\mathbf{P}^{\mathbf{DF}}(W_q = 1) - \mathbf{P}^{\mathbf{DG}}(W_q = 1)|$. We are usually interested in the maximal distinguishing advantage over all such distinguishers, which we denote by $\Delta_q(\mathbf{F}, \mathbf{G}) = \max_{\mathbf{D}} \Delta_q^{\mathbf{D}}(\mathbf{F}, \mathbf{G})$.

For a random system \mathbf{F} , we often consider an internal *monotone condition* defined on it. Such a condition is initially satisfied (true), but once it gets violated, it cannot become true again. We characterize such a condition by a sequence of events $\mathcal{A} = A_0, A_1, \dots$ such that A_0 always holds, and A_i holds if the condition holds after query i . The probability that a distinguisher \mathbf{D} issuing q queries makes a monotone condition \mathcal{A} fail in the random experiment \mathbf{DF} is denoted by $\nu^{\mathbf{D}}(\mathbf{F}, \overline{A}_q) = \mathbf{P}^{\mathbf{DF}}(\overline{A}_q)$ and we are again interested in the maximum over all distinguishers, denoted by $\nu(\mathbf{F}, \overline{A}_q) = \max_{\mathbf{D}} \nu^{\mathbf{D}}(\mathbf{F}, \overline{A}_q)$. For a random system \mathbf{F} with a monotone condition $\mathcal{A} = A_0, A_1, \dots$ and a random system \mathbf{G} , we say that \mathbf{F} *conditioned on \mathcal{A} is equivalent to \mathbf{G}* , denoted $\mathbf{F}|\mathcal{A} \equiv \mathbf{G}$, if $\mathbf{P}_{Y_i|X^iY^{i-1}A_i}^{\mathbf{F}} = \mathbf{P}_{Y_i|X^iY^{i-1}}^{\mathbf{G}}$ for $i \geq 1$, for all arguments for which $\mathbf{P}_{Y_i|X^iY^{i-1}A_i}^{\mathbf{F}}$ is defined. The following claim was proved in [10].

Lemma 1. *If $\mathbf{F}|\mathcal{A} \equiv \mathbf{G}$ then $\Delta_q(\mathbf{F}, \mathbf{G}) \leq \nu(\mathbf{F}, \overline{A}_q)$.*

Let \mathbf{F} be a random system with a monotone condition \mathcal{A} . Following [12], we define \mathbf{F} *blocked by \mathcal{A}* to be a new random system that behaves exactly like \mathbf{F} while the condition \mathcal{A} is satisfied. Once \mathcal{A} is violated, it only outputs a special blocking symbol \perp not contained in the output alphabet of \mathbf{F} . More formally, the following mapping is applied to the i^{th} output of \mathbf{F} :

$$y_i \mapsto \begin{cases} y_i & \text{if } A_i \text{ holds} \\ \perp & \text{otherwise.} \end{cases}$$

The following new lemma relates the optimal advantage in distinguishing two random systems to the optimal advantage in distinguishing their blocked counterparts.

Lemma 2. *Let \mathbf{F} and \mathbf{G} be two random systems with monotone conditions \mathcal{A} and \mathcal{B} defined on them, respectively. Let \mathbf{F}^\perp denote the random system \mathbf{F} blocked by \mathcal{A} and let \mathbf{G}^\perp denote \mathbf{G} blocked by \mathcal{B} . Then for every distinguisher \mathbf{D} we have $\Delta_q^{\mathbf{D}}(\mathbf{F}, \mathbf{G}) \leq \Delta_q(\mathbf{F}^\perp, \mathbf{G}^\perp) + \nu^{\mathbf{D}}(\mathbf{F}, \overline{A}_q)$.*

Proof. Let \mathbf{D} be an arbitrary distinguisher for \mathbf{F} and \mathbf{G} . Let \mathbf{D}' be a distinguisher that works as follows: it simulates \mathbf{D} , but whenever it receives an answer \perp to its query, it aborts and outputs 1. Then we have $\mathsf{P}^{\mathbf{D}\mathbf{G}}[W_q = 1] \leq \mathsf{P}^{\mathbf{D}'\mathbf{G}^\perp}[W_q = 1]$ and $\mathsf{P}^{\mathbf{D}'\mathbf{F}^\perp}[W_q = 1] \leq \mathsf{P}^{\mathbf{D}\mathbf{F}}[W_q = 1] + \nu^{\mathbf{D}}(\mathbf{F}, \overline{A}_q)$.

First, let us assume that $\mathsf{P}^{\mathbf{D}\mathbf{G}}[W_q = 1] \geq \mathsf{P}^{\mathbf{D}\mathbf{F}}[W_q = 1]$. Then, using the definition of advantage and the above inequalities, we get

$$\begin{aligned} \Delta_q^{\mathbf{D}}(\mathbf{F}, \mathbf{G}) &= |\mathsf{P}^{\mathbf{D}\mathbf{G}}[W_q = 1] - \mathsf{P}^{\mathbf{D}\mathbf{F}}[W_q = 1]| \\ &= \mathsf{P}^{\mathbf{D}\mathbf{G}}[W_q = 1] - \mathsf{P}^{\mathbf{D}\mathbf{F}}[W_q = 1] \\ &\leq \mathsf{P}^{\mathbf{D}'\mathbf{G}^\perp}[W_q = 1] - (\mathsf{P}^{\mathbf{D}'\mathbf{F}^\perp}[W_q = 1] - \nu^{\mathbf{D}}(\mathbf{F}, \overline{A}_q)) \\ &\leq \Delta_q(\mathbf{F}^\perp, \mathbf{G}^\perp) + \nu^{\mathbf{D}}(\mathbf{F}, \overline{A}_q), \end{aligned}$$

which proves the lemma in this case. On the other hand, if $\mathsf{P}^{\mathbf{D}\mathbf{G}}[W_q = 1] < \mathsf{P}^{\mathbf{D}\mathbf{F}}[W_q = 1]$, we can easily construct another distinguisher \mathbf{D}^* with the same behavior as \mathbf{D} and the opposite final answer bit. Then we can proceed with the argument as before and since $\Delta_q^{\mathbf{D}}(\mathbf{F}, \mathbf{G}) = \Delta_q^{\mathbf{D}^*}(\mathbf{F}, \mathbf{G})$ and $\nu^{\mathbf{D}}(\mathbf{F}, \overline{A}_q) = \nu^{\mathbf{D}^*}(\mathbf{F}, \overline{A}_q)$, the conclusion is valid also for the distinguisher \mathbf{D} . \square

Lemma 2 is a generalization of both Lemma 7 from [10] and of its special case, the Fundamental lemma of game-playing from [4]. Both these lemmas describe the special case when $\Delta_q(\mathbf{F}^\perp, \mathbf{G}^\perp) = 0$, i.e., when the distinguished systems behave identically until some conditions are violated. Our lemma is useful in the situations where the systems are not identical even while the conditions are satisfied, but their behavior is very similar. A good example of such a situation is presented in the proof of Theorem 1.

A random system \mathbf{F} can be used as a component of a larger system: in particular, we shall consider *constructions* $\mathbf{C}(\cdot)$ such that the resulting random system $\mathbf{C}(\mathbf{F})$ invokes \mathbf{F} as a subsystem. We state the following two observations about the composition of systems.

Lemma 3. *Let $\mathbf{C}(\cdot)$ and $\mathbf{C}'(\cdot)$ be two constructions invoking an internal random system, and let \mathbf{F} and \mathbf{G} be random systems. Then*

- (i) $\Delta_q(\mathbf{C}(\mathbf{F}), \mathbf{C}(\mathbf{G})) \leq \Delta_{q'}(\mathbf{F}, \mathbf{G})$, where q' is the maximum number of invocations of any internal system \mathbf{H} for any sequence of q queries to $\mathbf{C}(\mathbf{H})$, if such a value is defined.
- (ii) There exists a fixed permutation $S \in \text{Perm}(n)$ (represented by a deterministic stateless system) such that $\Delta_q(\mathbf{C}(\mathbf{P}), \mathbf{C}'(\mathbf{P})) \leq \Delta_q(\mathbf{C}(S), \mathbf{C}'(S))$.

Proof. The first claim comes from [10], so here we only prove the second one. Since the random system \mathbf{P} can be seen as a system that picks a permutation uniformly at random from $\text{Perm}(n)$ and then realizes this permutation, we have:

$$\Delta_q(\mathbf{C}(\mathbf{P}), \mathbf{C}'(\mathbf{P})) \leq \frac{1}{(2^n)!} \sum_{S \in \text{Perm}(n)} \Delta_q(\mathbf{C}(S), \mathbf{C}'(S)).$$

If all the values $\Delta_q(\mathbf{C}(S), \mathbf{C}'(S))$ were smaller than $\Delta_q(\mathbf{C}(\mathbf{P}), \mathbf{C}'(\mathbf{P}))$ it would contradict the inequality above, hence there exists a permutation $S \in \text{Perm}(n)$ such that $\Delta_q(\mathbf{C}(\mathbf{P}), \mathbf{C}'(\mathbf{P})) \leq \Delta_q(\mathbf{C}(S), \mathbf{C}'(S))$. \square

2.3 Ideal Blockciphers and Chains

We introduce some specific notions related to the cascade encryption setting. Our terminology follows and extends that in [4].

A *blockcipher* with keyspace $\{0, 1\}^k$ and message space $\{0, 1\}^n$ is a mapping $E: \{0, 1\}^k \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ such that for each $K \in \{0, 1\}^k$, $E(K, \cdot)$ is a permutation on the set $\{0, 1\}^n$. Typically $E_K(x)$ is written instead of $E(K, x)$ and $E_K^{-1}(\cdot)$ refers to the inverse of the permutation $E_K(\cdot)$.

Throughout the paper, we shall work in the *ideal blockcipher model*, which was recently shown to be equivalent to the random oracle model [6]. The ideal blockcipher model is widely used to analyze blockcipher constructions (e.g. [14, 9]) and consists of the assumption that for each key, the blockcipher realizes an independent random permutation.

A blockcipher can be seen as a directed graph consisting of 2^n vertices representing the message space and 2^{n+k} edges. Each vertex x has 2^k outgoing edges pointing to the encryptions of the message x using all possible keys. Each of the edges is labeled by the respective key. For a fixed blockcipher E , we denote by [1]

$$w(E) = \max_{x, y} |\{K \mid E_K(x) = y\}|$$

the maximal number of distinct keys mapping the plaintext x onto the ciphertext y , the maximum taken over all pairs of blocks (x, y) . Intuitively, $w(E)$ is the weight of the heaviest edge in the graph corresponding to E . This also naturally defines a random variable $w(\mathbf{E})$ for the random system \mathbf{E} realizing the ideal blockcipher.

If a distinguisher makes queries to a blockcipher E , let $x \xrightarrow{K} y$ denote the fact that it either made a query $E_K(x)$ and received the encryption y or made a query $E_K^{-1}(y)$ and received the decryption x . An *r -chain for keys* (K_1, \dots, K_r) is an $(r+1)$ -tuple (x_0, K_1, \dots, K_r) for which there exist x_1, \dots, x_r such that $x_0 \xrightarrow{K_1} x_1 \xrightarrow{K_2} \dots \xrightarrow{K_r} x_r$ holds. Similarly, if a fixed permutation S is given and $1 \leq i < r$, then an *i -disconnected r -chain for keys* (K_1, \dots, K_r) with respect to S is an $(r+1)$ -tuple (x_0, K_1, \dots, K_r) for which there exist x_1, \dots, x_r such that we have both

¹ $w(E)$ was denoted as Keys^E in [4].

$x_0 \xrightarrow{K_{r-i+1}} x_1 \xrightarrow{K_{r-i+2}} \dots \xrightarrow{K_r} x_i$ and $S^{-1}(x_i) \xrightarrow{K_1} x_{i+1} \xrightarrow{K_2} \dots \xrightarrow{K_{r-i}} x_r$. When describing chains, we sometimes explicitly refer to the permutations instead of the keys that define them. For disconnected chains, we sometimes omit the reference to the permutation S if it is clear from the context. The purpose of the following definition will be clear from the proof of Theorem [1](#).

Definition 3. *Let S be a fixed permutation. A distinguisher examines the key tuple (K_1, K_2, \dots, K_r) w.r.t. S if it creates either an r -chain or an i -disconnected r -chain w.r.t. S for (K_1, K_2, \dots, K_r) for any $i \in \{1, \dots, r - 1\}$.*

3 The Security of Cascade Encryption

In this section we reestablish the lower bound on the security of triple encryption from [4](#) in a more general setting. Our goal here is to simplify the proof and make it more comprehensible thanks to the level of abstraction provided by the random systems framework. Using Lemma [2](#) we also gain an improvement by a constant factor of 2 (cf. equation (10) in [4](#)). However, in order to fix the problem of the proof in [4](#), a new factor l appears in the security bound.

Although Theorem [1](#) only explicitly states the security of cascades with odd length, we point out that a simple reduction argument proves that longer cascades cannot be less secure than shorter ones, except for a negligible term $l/2^k$. Therefore, our result also implicitly proves any even cascade to be at least as secure as a one step shorter odd-length cascade.

We also point out that our bound is only useful for cascades of reasonable length, for extremely long cascades (e.g. $l \approx 2^{k/2}$) it becomes trivial.

3.1 Proof of the Main Result

Since this subsection aims to address the overall structure of the proof, we shall use two technical lemmas without proof (Lemmas [4](#) and [5](#)). These lemmas correspond to Lemmas 7 and 9 from [4](#), which they improve and generalize. We shall prove them in later subsections.

Let $l \geq 3$ be an odd integer. Let $\mathbf{C}_1(\cdot, \cdot)$ denote a construction which expects two subsystems: a blockcipher E and a permutation P . It chooses in advance l uniformly distinct keys K_1, \dots, K_l . These are not used by the system, their purpose is to make $\mathbf{C}_1(\cdot, \cdot)$ comparable to the other constructions. $\mathbf{C}_1(\cdot, \cdot)$ provides an interface to make forward and backward queries both to the blockcipher E and to the permutation P .

On the other hand, let $\mathbf{C}_2(\cdot)$ denote a construction which expects a blockcipher E as the only subsystem. It chooses in advance l uniformly random keys K_1, \dots, K_l . It provides an interface to make forward and backward queries both to the blockcipher E and to a permutation P , which it realizes as $E_{K_1} \circ \dots \circ E_{K_l}$. To achieve this, $\mathbf{C}_2(\cdot)$ queries its subsystem for all necessary values. Let $\mathbf{C}_2^d(\cdot)$ be the same construction as $\mathbf{C}_2(\cdot)$ except that it chooses the keys K_1, \dots, K_l to be uniformly distinct.

Finally, let $\mathbf{C}_3(\cdot, \cdot)$ denote a construction which again expects two subsystems: a blockcipher E and a permutation P . It chooses in advance l uniformly distinct keys K_1, \dots, K_l . It provides an interface to make forward and backward queries both to the blockcipher E and to the permutation P . However, answers to the blockcipher queries involving the key K_l are modified to satisfy the equation $E_{K_1} \circ \dots \circ E_{K_l} = P$. More precisely, forward queries are realized as $E_{K_l}(x) = P(E_{K_1}^{-1}(\dots E_{K_{l-1}}^{-1}(x) \dots))$ and backward queries are realized as $E_{K_l}^{-1}(y) = E_{K_{l-1}}(E_{K_{l-2}}(\dots E_{K_1}(P^{-1}(y)) \dots))$. To achieve this, $\mathbf{C}_3(\cdot, \cdot)$ queries its subsystems for all necessary values.

Recall that \mathbf{P} and \mathbf{E} denote the uniform random permutation and the ideal blockcipher, respectively. The following theorem bounds $\Delta_q(\mathbf{C}_1(\mathbf{E}, \mathbf{P}), \mathbf{C}_2(\mathbf{E}))$, the advantage in distinguishing cascade encryption of length l from a random permutation, given access to the underlying blockcipher.

Theorem 1. *For the constructions $\mathbf{C}_1(\cdot, \cdot)$, $\mathbf{C}_2(\cdot)$ and random systems \mathbf{E} , \mathbf{P} defined as above we have*

$$\Delta_q(\mathbf{C}_1(\mathbf{E}, \mathbf{P}), \mathbf{C}_2(\mathbf{E})) \leq 2l\alpha^{\lfloor l/2 \rfloor} \frac{q^{\lceil l/2 \rceil}}{(2^k)^{\lfloor l/2 \rfloor}} + 1.9 \left(\frac{lq}{2^{k+n/2}} \right)^{2/3} + \frac{l^2}{2^{k+1}},$$

where $\alpha = \max\{2e2^{k-n}, 2n + k \lfloor l/2 \rfloor\}$.

Proof. First, it is easy to see that $\Delta_q(\mathbf{C}_2(\mathbf{E}), \mathbf{C}_2^d(\mathbf{E})) \leq p_{\text{coll}}(2^k, l) < l^2/2^{k+1}$ and hence we have $\Delta_q(\mathbf{C}_1(\mathbf{E}, \mathbf{P}), \mathbf{C}_2(\mathbf{E})) \leq \Delta_q(\mathbf{C}_1(\mathbf{E}, \mathbf{P}), \mathbf{C}_2^d(\mathbf{E})) + l^2/2^{k+1}$. However, note that $\mathbf{C}_2^d(\mathbf{E}) \equiv \mathbf{C}_3(\mathbf{E}, \mathbf{P})$; this is because in both systems the permutations $E_{K_1}, \dots, E_{K_l}, P$ are chosen randomly with the only restriction that $E_{K_1} \circ \dots \circ E_{K_l} = P$ is satisfied. Now we can use Lemma 3 to substitute the random permutation \mathbf{P} in both $\mathbf{C}_1(\mathbf{E}, \mathbf{P})$ and $\mathbf{C}_3(\mathbf{E}, \mathbf{P})$ for a fixed one. Let S denote the permutation guaranteed by Lemma 3. Then we have

$$\Delta_q(\mathbf{C}_1(\mathbf{E}, \mathbf{P}), \mathbf{C}_2^d(\mathbf{E})) = \Delta_q(\mathbf{C}_1(\mathbf{E}, \mathbf{P}), \mathbf{C}_3(\mathbf{E}, \mathbf{P})) \leq \Delta_q(\mathbf{C}_1(\mathbf{E}, S), \mathbf{C}_3(\mathbf{E}, S)).$$

Since the permutation S is fixed, it makes now no sense for the distinguisher to query this permutation; it can have the permutation S hardwired.

From now on, we shall denote all queries to a blockcipher that involve one of the keys K_1, K_2, \dots, K_l as *relevant queries*. Let us now consider a monotone condition \mathcal{A}^h ($h \in \mathbb{N}$ is a parameter) defined on the random system $\mathbf{C}_1(\mathbf{E}, S)$. The condition \mathcal{A}_q^h is satisfied if the keys (K_1, K_2, \dots, K_l) were not examined w.r.t. S (in the sense of Definition 3) by the first q queries and at most h of these q queries were relevant. Let \mathcal{B}^h be an analogous condition defined on $\mathbf{C}_3(\mathbf{E}, S)$: \mathcal{B}_q^h is satisfied if the first q queries did not form a chain for the tuple (K_1, K_2, \dots, K_l) and at most h of these queries were relevant. Let \mathbf{G} and \mathbf{H} denote the random systems $\mathbf{C}_1(\mathbf{E}, S)$ and $\mathbf{C}_3(\mathbf{E}, S)$ blocked by \mathcal{A}^h and \mathcal{B}^h , respectively. Then by Lemma 2,

$$\Delta_q(\mathbf{C}_1(\mathbf{E}, S), \mathbf{C}_3(\mathbf{E}, S)) \leq \Delta_q(\mathbf{G}, \mathbf{H}) + \nu(\mathbf{C}_1(\mathbf{E}, S), \overline{\mathcal{A}_q^h}).$$

Let us first bound the quantity $\nu(\mathbf{C}_1(\mathbf{E}, S), \overline{A}_q^h)$. We can write A_q^h as $U_q \wedge V_q^h$, where U_q is satisfied if the first q queries did not examine the tuple of keys (K_1, K_2, \dots, K_l) and V_q^h is satisfied if at most h of the first q queries were relevant. Since $\overline{A}_q^h \Leftrightarrow \overline{U}_q \vee \overline{V}_q^h$, the union bound gives us

$$\nu(\mathbf{C}_1(\mathbf{E}, S), \overline{A}_q^h) \leq \nu(\mathbf{C}_1(\mathbf{E}, S), \overline{U}_q) + \nu(\mathbf{C}_1(\mathbf{E}, S), \overline{V}_q^h).$$

We prove in Lemma 4 that $\nu(\mathbf{C}_1(\mathbf{E}, S), \overline{U}_q) \leq 2l\alpha^{\lfloor l/2 \rfloor} q^{\lceil l/2 \rceil} / (2^k)^l$. Since the keys K_1, \dots, K_l do not affect the outputs of $\mathbf{C}_1(\mathbf{E}, S)$, adaptivity does not help when trying to violate the condition \overline{V}_q^h , therefore we can restrict our analysis to nonadaptive strategies for provoking \overline{V}_q^h . The probability that a given query is relevant is $l/2^k$, hence the expected number of relevant queries among the first q queries is $lq/2^k$ and by Markov's inequality we have $\nu(\mathbf{C}_1(\mathbf{E}, S), \overline{V}_q^h) \leq lq/h2^k$. All put together, $\nu(\mathbf{C}_1(\mathbf{E}, S), \overline{A}_q^h) \leq 2l\alpha^{\lfloor l/2 \rfloor} q^{\lceil l/2 \rceil} / (2^k)^l + lq/h2^k$.

It remains to bound $\Delta_q(\mathbf{G}, \mathbf{H})$. These systems only differ in their behavior for the first h relevant queries, so let us make this difference explicit. Let \mathbf{G}_r be a random system that allows queries to l independent random permutations $\pi_1, \pi_2, \dots, \pi_l$, but returns \perp once the queries create an l -chain for any tuple in $\mathbf{CS}(\pi_1, \pi_2, \dots, \pi_l)$. Let \mathbf{H}_r be a random system that allows queries to l random permutations $\pi_1, \pi_2, \dots, \pi_l$ such that $\pi_1 \circ \pi_2 \circ \dots \circ \pi_l = id$, but returns \perp once the queries create an l -chain for the tuple $(\pi_1, \pi_2, \dots, \pi_l)$. Let $\mathbf{C}_{h,S}(\cdot)$ be a construction that allows queries to a blockcipher, let us denote it by E . In advance, it picks l random distinct keys K_1, K_2, \dots, K_l . Then it realizes the queries to $E_{K_1}, E_{K_2}, \dots, E_{K_l}$ as $\pi_1, \pi_2, \dots, \pi_{l-1}$ and $\pi_l \circ S$ respectively, where the permutations π_i for $i \in \{1, \dots, l\}$ are provided by a subsystem. E_K for all other keys K are realized by $\mathbf{C}_{h,S}(\cdot)$ as random permutations. However, $\mathbf{C}_{h,S}(\cdot)$ only redirects the first h relevant queries to the subsystem, after this number is exceeded, it responds to all queries by \perp . Intuitively, the subsystem used is responsible for the answers to the first h relevant queries (hence the subscript "r"). Since the disconnected chains in $\mathbf{C}_{h,S}(\mathbf{G}_r)$ correspond exactly to the ordinary chains in \mathbf{G}_r , we have $\mathbf{C}_{h,S}(\mathbf{G}_r) \equiv \mathbf{G}$ and $\mathbf{C}_{h,S}(\mathbf{H}_r) \equiv \mathbf{H}$. According to Lemma 3 and Lemma 5 below, we have $\Delta_q(\mathbf{G}, \mathbf{H}) \leq \Delta_h(\mathbf{G}_r, \mathbf{H}_r) \leq h^2/2^n$.

Now we can optimize the choice of the constant h . The part of the advantage that depends on h is $f(h) = lq/h2^k + h^2/2^n$. This term is minimal for $h^* = (lq2^{n-k-1})^{1/3}$ and we get $f(h^*) < 1.9 \left(\frac{lq}{2^{k+n/2}} \right)^{2/3}$. This completes the proof. \square

3.2 Examining the Relevant Keys

Here we analyze the probability that the adversary examines the relevant keys (K_1, \dots, K_l) w.r.t. S during its interaction with the random system $\mathbf{C}_1(\mathbf{E}, S)$. This is a generalization of Lemma 7 from 4 to longer cascades, also taking disconnected chains into account.

Lemma 4. *Let the random system $\mathbf{C}_1(\mathbf{E}, S)$ and the condition U_q be defined as in the proof of Theorem 1, with the number of keys l being odd. Then we have $\nu(\mathbf{C}_1(\mathbf{E}, S), \overline{U}_q) \leq 2l\alpha^{\lfloor l/2 \rfloor} q^{\lfloor l/2 \rfloor} / (2^k)^{\lfloor l/2 \rfloor}$, where $\alpha = \max\{2e2^{k-n}, 2n + k\lfloor l/2 \rfloor\}$.*

Proof. Recall that the relevant keys K_1, \dots, K_l are examined by the distinguisher if it creates either an l -chain or an i -disconnected l -chain for the tuple (K_1, K_2, \dots, K_l) for any $i \in \{1, \dots, l-1\}$.

Let $i \in \{1, \dots, l-1\}$ be fixed. We first bound the probability that the distinguisher creates an i -disconnected l -chain. Since the relevant keys do not affect the behavior of the system $\mathbf{C}_1(\mathbf{E}, S)$, this probability is equal to the number of l -tuples of distinct keys for which an i -disconnected l -chain was created, divided by the number of all l -tuples of distinct keys, which is $(2^k)^{\lfloor l/2 \rfloor}$. The numerator can be upper bounded by the number of all i -disconnected l -chains that were created (here we also count those created for non-distinct key tuples). Hence, let $\text{Ch}_{i,l,q}^E$ denote the maximum number of i -disconnected l -chains any distinguisher can create by issuing q queries to a fixed blockcipher E and let $\text{Ch}_{i,l,q}^{\mathbf{E}}$ denote the expected value of $\text{Ch}_{i,l,q}^E$ with respect to the choice of E by \mathbf{E} .

Let G be a directed graph corresponding to a blockcipher E , as described in Subsection 2.3. Let H be the spanning subgraph of G containing only the edges that were queried by the distinguisher. Any i -disconnected l -chain consists of l edges in H , let us denote them as e_1, e_2, \dots, e_l , following the order in which they appear in the chain. Then for each of the odd edges e_1, e_3, \dots, e_l there are q possibilities to choose which of the queries corresponds to this edge. Once the odd edges are fixed, they uniquely determine the vertices x_0, x_1, \dots, x_l such that e_j is $x_{j-1} \rightarrow x_j$ for $j \in \{1, 3, \dots, l\} \setminus \{i+1\}$ and e_{i+1} is $S^{-1}(x_i) \rightarrow x_{i+1}$ if i is even. Since there are at most $w(E)$ possible edges to connect any pair of vertices in G , there are now at most $w(E)$ possibilities to choose each of the even edges e_2, e_4, \dots, e_{l-1} so that e_j is $x_{j-1} \rightarrow x_j$ for $j \in \{2, 4, \dots, l-1\} \setminus \{i+1\}$ and e_{i+1} is $S^{-1}(x_i) \rightarrow x_{i+1}$ if i is odd. Hence, $\text{Ch}_{i,l,q}^E \leq w(E)^{\lfloor l/2 \rfloor} q^{\lfloor l/2 \rfloor}$ and $\text{Ch}_{i,l,q}^{\mathbf{E}} \leq w(\mathbf{E})^{\lfloor l/2 \rfloor} q^{\lfloor l/2 \rfloor}$.

It remains to bound the value $w(\mathbf{E})$. For this, we use the bound from [4], where the inequality $\mathbb{P}[w(\mathbf{E}) \geq \beta] < 2^{2n+1-\beta}$ is proved for any $\beta \geq 2e2^{k-n}$. Using this inequality gives us

$$\begin{aligned} \text{Ch}_{i,l,q}^{\mathbf{E}} &\leq \mathbb{E}[\text{Ch}_{i,l,q}^E \mid w(E) < \alpha] + \mathbb{E}[\text{Ch}_{i,l,q}^E \mid w(E) \geq \alpha] \cdot 2^{2n+1-\alpha} \\ &\leq \alpha^{\lfloor l/2 \rfloor} q^{\lfloor l/2 \rfloor} + 2^{k\lfloor l/2 \rfloor} q^{\lfloor l/2 \rfloor} 2^{2n+1-\alpha} \leq 2\alpha^{\lfloor l/2 \rfloor} q^{\lfloor l/2 \rfloor}, \end{aligned}$$

where the last two inequalities hold since $w(E) \leq 2^k$ and $\alpha \geq 2n + k\lfloor l/2 \rfloor \geq 2$.

Putting all together, we get that the probability of forming an i -disconnected l -chain for the keys (K_1, K_2, \dots, K_l) can be upper bounded by $2\alpha^{\lfloor l/2 \rfloor} q^{\lfloor l/2 \rfloor} / (2^k)^{\lfloor l/2 \rfloor}$. Since this holds for each $i \in \{1, 2, \dots, l-1\}$ and the probability of creating an l -chain for the keys (K_1, \dots, K_l) can be bounded in the same way, by the union bound we get $\nu(\mathbf{C}_1(\mathbf{E}, S), \overline{U}_q) \leq 2l\alpha^{\lfloor l/2 \rfloor} q^{\lfloor l/2 \rfloor} / (2^k)^{\lfloor l/2 \rfloor}$. \square

3.3 Distinguishing Independent and Correlated Permutations

Now we shall improve the bound on $\Delta_h(\mathbf{G}_r, \mathbf{H}_r)$ stated by Lemma 9 in [4]. Using the concept of conditional equivalence from [10], our result is better by a constant factor and is applicable for the general case of l -cascade encryption.

Recall that \mathbf{G}_r is a random system that provides an interface to query l random independent permutations² π_1, \dots, π_l in both directions. However, if the queries of the distinguisher form an l -chain for any tuple of permutations in $\text{CS}(\pi_1, \dots, \pi_l)$, the system \mathbf{G}_r becomes blocked and answers all subsequent queries (including the one that formed the chain) with the symbol \perp . On the other hand, \mathbf{H}_r is a random system that provides an interface to query l random permutations π_1, \dots, π_l such that $\pi_1 \circ \dots \circ \pi_l = id$, again in both directions. Similarly, if an l -chain is created for any tuple in $\text{CS}(\pi_1, \dots, \pi_l)$ (which is in this case equivalent to creating an l -chain for (π_1, \dots, π_l)), \mathbf{H}_r answers all subsequent queries with the symbol \perp . Therefore, the value $\Delta_h(\mathbf{G}_r, \mathbf{H}_r)$ denotes the best possible advantage in distinguishing l independent random permutations from l random permutations correlated in the described way, without forming an l -chain.

Lemma 5. *Let \mathbf{G}_r and \mathbf{H}_r be the random systems defined in the proof of Theorem 7. Then $\Delta_h(\mathbf{G}_r, \mathbf{H}_r) \leq h^2/2^n$.*

Proof. First, let us introduce some notation. In any experiment where the permutations π_1, \dots, π_l are queried, let $\text{dom}_j(\pi_i)$ denote the set of all $x \in \{0, 1\}^n$ such that among the first j queries, the query $\pi_i(x)$ was already answered or some query $\pi_i^{-1}(y)$ was answered by x . Similarly, let $\text{range}_j(\pi_i)$ be the set of all $y \in \{0, 1\}^n$ such that among the first j queries, the query $\pi_i^{-1}(y)$ was already answered or some query $\pi_i(x)$ was answered by y . In other words, $\text{dom}_j(\pi_i)$ and $\text{range}_j(\pi_i)$ denote the domain and range of the partial function π_i defined by the first j answers. For each pair of consecutive permutations³ π_i and π_{i+1} , let $\mathcal{X}_i^{(j)}$ denote the set $\{0, 1\}^n \setminus (\text{range}_j(\pi_i) \cup \text{dom}_j(\pi_{i+1}))$ of fresh, unused values. If $x \xrightarrow{\pi_i} y$ then we call the queries $\pi_i(x)$ and $\pi_i^{-1}(y)$ *trivial* and the queries $\pi_{i+1}(y)$ and $\pi_{i+1}^{-1}(x)$ are said to *extend* a chain if they are not trivial too.

Now we introduce an intermediate random system \mathbf{S} and show how both \mathbf{G}_r and \mathbf{H}_r are conditionally equivalent to \mathbf{S} . This allows us to use Lemma 11 to bound the advantage in distinguishing \mathbf{G}_r and \mathbf{H}_r . The system \mathbf{S} also provides an interface to query l permutations π_1, \dots, π_l . It works as follows: it answers any non-trivial forward query $\pi_i(x)$ with a value chosen uniformly from the set $\mathcal{X}_i^{(j-1)}$ and any non-trivial backward query $\pi_i^{-1}(x)$ with a value chosen uniformly from the set $\mathcal{X}_{i-1}^{(j-1)}$ (assuming it is the j^{th} query). Any trivial queries are answered consistently with previous answers. Moreover, if the queries form an l -chain for any tuple in $\text{CS}(\pi_1, \dots, \pi_l)$, \mathbf{S} also gets blocked and responds with \perp to any further queries. Note that \mathbf{S} is only defined as long as $|\mathcal{X}_i^{(j-1)}| \geq 0$, but if this is not true, we have $h \geq 2^n$ and the lemma holds trivially.

² All permutations considered here are defined on the set $\{0, 1\}^n$.

³ The indexing of permutations is cyclic, e.g. π_{l+1} denotes the permutation π_1 .

Let us now consider the j^{th} query that does not extend an $(l - 1)$ -chain (otherwise both \mathbf{G}_r and \mathbf{S} get blocked). Then the system \mathbf{G}_r answers any non-trivial forward query $\pi_i(x)$ by a random element uniformly chosen from $\{0, 1\}^n \setminus \text{range}_{j-1}(\pi_i)$ or gets blocked if this answer would create an l -chain by connecting two shorter chains. On the other hand, the system \mathbf{S} answers with a random element uniformly chosen from $\mathcal{X}_i^{(j-1)}$, which is a subset of $\{0, 1\}^n \setminus \text{range}_{j-1}(\pi_i)$. The situation for backward queries is analogous. Therefore, let us define a monotone condition \mathcal{K} on \mathbf{G}_r : the event K_j is satisfied if K_{j-1} was satisfied and the answer to the j^{th} query was picked from the set $\mathcal{X}_i^{(j-1)}$ if it was a non-trivial forward query $\pi_i(x)$ or from the set $\mathcal{X}_{i-1}^{(j-1)}$ if it was a non-trivial backward query $\pi_i^{-1}(y)$. Note that as long as \mathcal{K} is satisfied, no l -chain can emerge by connecting two shorter chains. By the previous observations and the definition of \mathcal{K} , we have $\mathbf{G}_r | \mathcal{K} \equiv \mathbf{S}$ which by Lemma [□](#) implies $\Delta_h(\mathbf{G}_r, \mathbf{S}) \leq \nu(\mathbf{G}_r, \overline{K_h})$. The probability that \mathcal{K} is violated by the j^{th} answer is

$$\frac{|\text{dom}_{j-1}(\pi_{i+1}) \setminus \text{range}_{j-1}(\pi_i)|}{|\{0, 1\}^n \setminus \text{range}_{j-1}(\pi_i)|} \leq \frac{|\{0, 1\}^n \setminus \mathcal{X}_i^{(j-1)}|}{|\{0, 1\}^n|} \leq \frac{j-1}{2^n},$$

which gives us $\nu(\mathbf{G}_r, \overline{K_h}) \leq \sum_{j=1}^h (j-1)/2^n \leq h^2/2^{n+1}$.

In the system \mathbf{H}_r , the permutations π_1, \dots, π_l can be seen as 2^n cycles of length l , each of which is formed by the edges connecting the vertices $x, \pi_1(x), \dots, \pi_{l-1}(\dots \pi_1(x) \dots), x$ for some $x \in \{0, 1\}^n$ and labeled by the respective permutations. We shall call such a cycle *used* if at least one of its edges was queried in either direction^{[4](#)}, otherwise we call it *unused*. Let us now define a monotone condition \mathcal{L} on \mathbf{H}_r : the event L_j is satisfied if during the first j queries, any non-trivial query which did not extend an existing chain queried an unused cycle.

We claim that $\mathbf{H}_r | \mathcal{L} \equiv \mathbf{S}$. To see this, let us consider all possible types of queries. If the j^{th} query $\pi_i(x)$ is trivial or it extends an $(l - 1)$ -chain, both systems behave identically. Otherwise, the system \mathbf{H}_r answers with a value y , where $y \notin \text{range}_{j-1}(\pi_i)$ (because π_i is a permutation) and $y \notin \text{dom}_{j-1}(\pi_{i+1})$, since that would mean that \mathcal{L} was violated either earlier (if this query extends an existing chain) or now (if it starts a new chain). All values from $\mathcal{X}_i^{(j-1)}$ have the same probability of being y , because for any $y_1, y_2 \in \mathcal{X}_i^{(j-1)}$, there exists a straightforward bijective mapping between the arrangement of the cycles consistent with $\pi_i(x) = y_1$ or $\pi_i(x) = y_2$ (and all previous answers). Therefore, \mathbf{H}_r answers with an uniformly chosen element from $\mathcal{X}_i^{(j-1)}$ and so does \mathbf{S} . For backward queries, the situation is analogous. By Lemma [□](#) this gives us $\Delta_h(\mathbf{S}, \mathbf{H}_r) \leq \nu(\mathbf{H}_r, \overline{L_h})$.

Let the j^{th} query be a non-trivial forward query $\pi_i(x)$ that does not extend a chain, i.e., $x \in \mathcal{X}_{i-1}^{(j-1)}$. Let u denote the number of elements in $\mathcal{X}_{i-1}^{(j-1)}$ that are in a used cycle on the position between π_{i-1} and π_i . Then since every element in $\mathcal{X}_{i-1}^{(j-1)}$ has the same probability of having this property (for the same reason as above), this query violates the condition \mathcal{L} with probability $u/|\mathcal{X}_{i-1}^{(j-1)}| \leq$

⁴ We consider a separate edge connecting two vertices for each cycle in which they follow each other, hence each query creates at most one used cycle.

$(u + |\text{range}_{j-1}(\pi_{i-1}) \cup \text{dom}_{j-1}(\pi_i)|)/2^n \leq (j-1)/2^n$. Hence $\nu(\mathbf{H}_r, \overline{L}_h) \leq \sum_{j=1}^h (j-1)/2^n \leq h^2/2^{n+1}$.

Putting everything together, we have $\Delta_h(\mathbf{G}_r, \mathbf{H}_r) \leq \Delta_h(\mathbf{G}_r, \mathbf{S}) + \Delta_h(\mathbf{S}, \mathbf{H}_r) \leq h^2/2^n$, which completes the proof. \square

4 Conclusions

In this paper, we have studied the security of the cascade encryption. The most important recent result on this topic [4] contained a few mistakes, which we pointed out and corrected. We have formulated the proof from [4] in the random systems framework, which allows us to describe it on a more abstract level and thus in a more compact argument. This abstraction leads to a minor improvement for the case of triple encryption, as well as a generalization for the case of longer cascades. We prove that for the wide class of blockciphers with smaller key space than message space, a reasonable increase in the length of the cascade improves the encryption security. Our intention here was also to demonstrate the power of the random systems framework as a tool for modelling the behavior and interactions of discrete systems, with a focus towards analyzing their indistinguishability.

Acknowledgements. We would like to thank the anonymous reviewers for useful comments. This research was partially supported by the Swiss National Science Foundation (SNF) project no. 200020-113700/1 and by the grants VEGA 1/0266/09 and UK/385/2009.

References

1. Aiello, W., Bellare, M., Di Crescenzo, G., Venkatesan, R.: Security Amplification by Composition: The case of Doubly-Iterated, Ideal Ciphers. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 499–558. Springer, Heidelberg (1998)
2. ANSI X9.52, Triple Data Encryption Algorithm Modes of Operation (1998)
3. Bellare, M., Namprempre, Ch.: Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm, full version, Cryptology ePrint Archive, Report 2000/025 (2007)
4. Bellare, M., Rogaway, P.: Code-Based Game-Playing Proofs and the Security of Triple Encryption. In: Eurocrypt 2006. LNCS, vol. 4004, pp. 409–426. Springer, Heidelberg (2006), <http://eprint.iacr.org/2004/331>
5. Bellare, M., Ristenpart, T.: Hash Functions in the Dedicated-Key Setting: Design Choices and MPP Transforms. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 399–410. Springer, Heidelberg (2007)
6. Coron, J.S., Patarin, J., Seurin, Y.: The Random Oracle Model and the Ideal Cipher Model are Equivalent. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 1–20. Springer, Heidelberg (2008)
7. Diffie, W., Hellman, M.: Exhaustive Cryptanalysis of the Data Encryption Standard. Computer 10, 74–84 (1977)

8. Even, S., Goldreich, O.: On the Power of Cascade Ciphers. *ACM Transactions on Computer Systems* 3(2), 108–116 (1985)
9. Even, S., Mansour, Y.: A Construction of a Cipher from a Pseudorandom Permutation. In: Matsumoto, T., Imai, H., Rivest, R.L. (eds.) *ASIACRYPT 1991*. LNCS, vol. 739, pp. 210–224. Springer, Heidelberg (1993)
10. Maurer, U.: Indistinguishability of Random Systems. In: Knudsen, L.R. (ed.) *EUROCRYPT 2002*. LNCS, vol. 2332, pp. 110–132. Springer, Heidelberg (2002)
11. Maurer, U., Massey, J.: Cascade Ciphers: the Importance of Being First. *J. of Cryptology* 6(1), 55–61 (1993)
12. Maurer, U., Pietrzak, K., Renner, R.: Indistinguishability Amplification. In: Menezes, A. (ed.) *CRYPTO 2007*. LNCS, vol. 4622, pp. 130–149. Springer, Heidelberg (2007)
13. National Institute of Standards and Technology: FIPS PUB 46-3: Data Encryption Standard (DES) (1999)
14. National Institute of Standards and Technology: Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher, NIST Special Publication 800-67 (2004)
15. Rogaway, P., Shrimpton, T.: Deterministic Authenticated-Encryption. In: Vaudenay, S. (ed.) *EUROCRYPT 2006*. LNCS, vol. 4004, pp. 373–390. Springer, Heidelberg (2006)

A Problems with the Proof in [4]

The proof of a lower bound for the security of triple encryption presented in [4] contains some errors. We describe briefly where these errors come from, assuming the reader is familiar with the terminology and the proof from [4]. We shall be referring to the version 2.3 of the paper published at the online ePrint archive. The proof eventually comes down to bounding the advantage in distinguishing independent random permutations π_0, π_1, π_2 from random permutations π_0, π_1, π_2 such that $\pi_0 \circ \pi_1 \circ \pi_2 = id$ (distinguishing games G and H). This can be done easily if the distinguisher is allowed to extend a 2-chain by his queries, therefore the adversary is not allowed to do that in games G and H . To justify this, before proceeding to this part of the proof, the authors have to argue in a more complex setting (games D_S and R_3) that the probability of extending a 2-chain for the relevant keys is negligible. However, due to the construction of the adversary $B_{S,b}$ from the adversary B , extending a 2-chain by $B_{S,b}$ in the experiment $H^{B_{S,b}}$ does not correspond to extending a 2-chain by B in D_S^B , but to something we call a disconnected chain. The same can be said about the experiments R_3^B and $G^{B_{S,b}}$. Therefore, by bounding the probability of extending a 2-chain for the relevant keys in the experiment R_3^B , the authors do not bound the probability of extending a 2-chain in the experiment $G^{B_{S,b}}$, which they later need.

The second problem of the proof in [4] lies in bounding the probability of creating a chain using the game L . This is done by the equation $\mathbb{P}[R_3^B \text{ sets } x2ch] \leq 3 \cdot 2^{-k} + \mathbb{P}[B^L \text{ sets } bad]$ on page 19, which is also invalid. To see this, note that the game L only considers chains using subsequently the keys (K_0, K_1, K_2) , while the flag $x2ch$ in the experiment R_3^B can also be set by a chain for any cyclic

shift of this triple, e.g. (K_2, K_0, K_1) . This is why a new multiplicative factor l appears in the security bound we have proved.

In the version 3.0 of the paper [4], the second bug mentioned here was fixed, while the first is still present in a different form. Now the games G and H_S can be easily distinguished by forming a disconnected chain, for example by the following trivial adversary B :

Adversary B

```

 $x_1 \xleftarrow{\$} \{0, 1\}^n;$ 
 $x_2 \leftarrow \Pi(1, x_1); x_3 \leftarrow \Pi(2, x_2); x_0 \leftarrow S^{-1}(x_3); x'_1 \leftarrow \Pi(0, x_0);$ 
if  $x_1 = x'_1$  return 1 else return 0;
```

This problem can be fixed by introducing the concept of disconnected chains and bounding the probability of them being constructed by the adversary, as we do for the general case of l -cascades in Lemma [4].

Quantum-Secure Coin-Flipping and Applications

Ivan Damgård and Carolin Lunemann

DAIMI, Aarhus University, Denmark
{ivan, carolin}@cs.au.dk

Abstract. In this paper, we prove classical coin-flipping secure in the presence of quantum adversaries. The proof uses a recent result of Watrous [20] that allows quantum rewinding for protocols of a certain form. We then discuss two applications. First, the combination of coin-flipping with any non-interactive zero-knowledge protocol leads to an easy transformation from non-interactive zero-knowledge to interactive quantum zero-knowledge. Second, we discuss how our protocol can be applied to a recently proposed method for improving the security of quantum protocols [4], resulting in an implementation without set-up assumptions. Finally, we sketch how to achieve efficient simulation for an extended construction in the common-reference-string model.

Keywords. quantum cryptography, coin-flipping, common reference string, quantum zero-knowledge.

1 Introduction

In this paper, we are interested in a standard coin-flipping protocol with classical messages exchange but where the adversary is assumed to be capable of quantum computing. Secure coin-flipping allows two parties Alice and Bob to agree on a uniformly random bit in a fair way, i.e., neither party can influence the value of the coin to his advantage. The (well-known) protocol proceeds as follows: Alice commits to a bit a , Bob then sends bit b , Alice opens the commitment and the resulting coin is the exclusive disjunction of both bits, i.e. $coin = a \oplus b$.

For Alice's commitment to her first message, we assume a classical bit commitment scheme. Intuitively, a commitment scheme allows a player to commit to a value, while keeping it hidden (*hiding property*) but preserving the possibility to later reveal the value fixed at commitment time (*binding property*). More formally, a bit commitment scheme takes a bit and some randomness as input. The hiding property is formalized by the non-existence of a distinguisher able to distinguish with non-negligible advantage between a commitment to 0 and a commitment to 1. The binding property is fulfilled, if it is infeasible for a forger to open one commitment to both values 0 and 1. The hiding respectively binding property holds with unconditional (i.e. perfect or statistical) security in the classical and the quantum setting, if the distinguisher respectively the forger is unrestricted with respect to his (quantum-) computational power. In case of a polynomial-time bounded classical distinguisher respectively forger, the

commitment is computationally hiding respectively binding. The computationally hiding property translates to the quantum world by simply allowing the distinguisher to be quantum. However, the case of a quantum forger can not be handled in such a straightforward manner, due to the difficulties of rewinding in general quantum systems (see e.g. [12,15,20] for discussions).

For our basic coin-flip protocol, we assume the commitment to be *unconditionally binding* and *computationally hiding against a quantum adversary*.¹ Thus, we achieve unconditional security against cheating Alice and quantum-computational security against dishonest Bob. Such a commitment scheme follows, for instance, from any pseudorandom generator [15], secure against a quantum distinguisher. Even though the underlying computational assumption, on which the security of the embedded commitment is based, withstands quantum attacks, the security proof of the entire protocol and its integration into other applications could previously not be naturally translated from the classical to the quantum world. Typically, security against a classical adversary is argued using rewinding of the adversary. But in general, rewinding as a proof technique cannot be directly applied, if Bob runs a quantum computer: First, the intermediate state of a quantum system cannot be copied [21], and second, quantum measurements are in general irreversible. Hence, in order to produce a classical output, the simulator had to (partially) measure the quantum system without copying it beforehand, but then it would become generally impossible to reconstruct all information necessary for correct rewinding. For these reasons, no simple and straightforward security proofs for the quantum case were previously known.

In this paper, we show the most natural and direct quantum analogue of the classical security proof for standard coin-flipping, by using a recent result of Watrous [20]. Watrous showed how to construct an efficient quantum simulator for quantum verifiers for several zero-knowledge proof systems such as graph isomorphism, where the simulation relies on the newly introduced *quantum rewinding theorem*. We now show that his quantum rewinding argument can also be applied to classical coin-flipping in a quantum world.

By calling the coin-flip functionality sequentially a sufficient number of times, the communicating parties can interactively generate a common random string from scratch. The generation can then be integrated into other (classical or quantum) cryptographic protocols that work in the common-reference-string model. This way, several interesting applications can be implemented entirely in a simple manner without any set-up assumptions. Two example applications are discussed in the second part of the paper.

The first application relates to zero-knowledge proof systems, an important building block for larger cryptographic protocols. Recently, Hallgren et al. [13] showed that any honest verifier zero-knowledge protocol can be made zero-knowledge against any classical and quantum verifier. Here we show a related

¹ Recall that unconditionally secure commitments, i.e. unconditionally hiding and binding at the same time, are impossible in both the classical and the quantum world.

result, namely, a simple transformation from non-interactive (quantum) zero-knowledge to interactive quantum zero-knowledge. A non-interactive zero-knowledge proof system can be trivially turned into an interactive *honest verifier* zero-knowledge proof system by just letting the verifier choose the reference string. Therefore, this consequence of our result also follows from [13]. However, our proof is much simpler. In general, the difference between us and [13] is that our focus is on establishing coin-flipping as a stand-alone tool that can be used in several contexts rather than being integrated in a zero-knowledge construction as in [13].

As second application we discuss the interactive generation of a common reference string for the general compiler construction improving the security of a large class of quantum protocols that was recently proposed in [4]. Applying the compiler, it has been shown how to achieve hybrid security in existing protocols for password-based identification [6] and oblivious transfer [1] without significant efficiency loss, such that an adversary must have both large quantum memory *and* large computing power to break the protocol. Here we show how a common reference string for the compiler can be generated from scratch according to the specific protocol requirements in [4].

Finally, we sketch an extended commitment scheme for quantum-secure coin-flipping in the common-reference-string model. This construction can be *efficiently* simulated without the need of rewinding, which is necessary to claim universal composability.

2 Preliminaries

2.1 Notation

We assume the reader's familiarity with basic notation and concepts of quantum information processing as in standard literature, e.g. [16]. Furthermore, we will only give the details of the discussed applications that are most important in the context of this work. A full description of the applications can be found in the referenced papers.

We denote by $\text{negl}(n)$ any function of n , if for any polynomial p it holds that $\text{negl}(n) \leq 1/p(n)$ for large enough n . As a measure of *closeness* of two quantum states ρ and σ , their trace distance $\delta(\rho, \sigma) = \frac{1}{2} \text{tr}(|\rho - \sigma|)$ or square-fidelity $\langle \rho | \sigma | \rho \rangle$ can be applied. A quantum algorithm consists of a family $\{C_n\}_{n \in \mathbb{N}}$ of quantum circuits and is said to run in polynomial time, if the number of gates of C_n is polynomial in n . Two families of quantum states $\{\rho_n\}_{n \in \mathbb{N}}$ and $\{\sigma_n\}_{n \in \mathbb{N}}$ are called *quantum-computationally indistinguishable*, denoted $\rho \stackrel{q}{\approx} \sigma$, if any polynomial-time quantum algorithm has negligible advantage in n of distinguishing ρ_n from σ_n . Analogously, they are *statistically indistinguishable*, denoted $\rho \stackrel{s}{\approx} \sigma$, if their trace distance is negligible in n . For the reverse circuit of quantum circuit Q , we use the standard notation for the transposed, complex conjugate operation, i.e. Q^\dagger . The controlled-NOT operation (CNOT) with a control and a target qubit as input flips the target qubit, if the control qubit is 1. In other words, the value

of the second qubit corresponds to the classical exclusive disjunction (XOR). A phase-flip operation can be described by Pauli operator Z . For quantum state ρ stored in register R we write $|\rho\rangle_R$.

2.2 Definition of Security

We follow the framework for defining security which was introduced in [8] and also used in [4]. Our cryptographic two-party protocols run between player Alice, denoted by A , and player Bob (B). Dishonest parties are indicated by A^* and B^* , respectively. The security against a dishonest player is based on the *real/ideal-world paradigm* that assumes two different worlds: The *real-world* that models the actual protocol Π and the *ideal-world* based on the ideal functionality \mathcal{F} that describes the intended behavior of the protocol. If both executions are indistinguishable, security of the protocol in real life follows. In other words, a dishonest real-world player P^* that attacks the protocol cannot achieve (significantly) more than an ideal-world adversary \hat{P}^* attacking the corresponding ideal functionality.

More formally, the joint input state consists of classical inputs of honest parties and possibly quantum input of dishonest players. A protocol Π consists of an infinite family of interactive (quantum) circuits for parties A and B . A classical (non-reactive) ideal functionality \mathcal{F} is given by a conditional probability distribution $P_{\mathcal{F}(in_A, in_B)|in_A in_B}$, inducing a pair of random variables $(out_A, out_B) = \mathcal{F}(in_A, in_B)$ for every joint distribution of in_A and in_B , where in_P and out_P denote party P 's in- and output, respectively. For the definition of (quantum-) computational security against a dishonest Bob, a polynomial-size (quantum) *input sampler* is considered, which produces the input state of the parties.

Definition 2.1 (Correctness). *A protocol Π correctly implements an ideal classical functionality \mathcal{F} , if for every distribution of the input values of honest Alice and Bob, the resulting common outputs of Π and \mathcal{F} are statistically indistinguishable.*

Definition 2.2 (Unconditional security against dishonest Alice). *A protocol Π implements an ideal classical functionality \mathcal{F} unconditionally securely against dishonest Alice, if for any real-world adversary A^* , there exists an ideal-world adversary \hat{A}^* , such that for any input state it holds that the output state, generated by A^* through interaction with honest B in the real-world, is statistically indistinguishable from the output state, generated by \hat{A}^* through interaction with \mathcal{F} and A^* in the ideal-world.*

Definition 2.3 ((Quantum-) Computational security against dishonest Bob). *A protocol Π implements an ideal classical functionality \mathcal{F} (quantum-) computationally securely against dishonest Bob, if for any (quantum-) computationally bounded real-world adversary B^* , there exists a (quantum-) computationally bounded ideal-world adversary \hat{B}^* , such that for any efficient input sampler, it holds that the output state, generated by B^* through interaction with honest A*

in the real-world, is (quantum-) computationally indistinguishable from the output state, generated by $\hat{\mathbf{B}}^*$ through interaction with \mathcal{F} and \mathbf{B}^* in the ideal-world.

For more details and a definition of indistinguishability of quantum states, see [8]. There, it has also been shown that protocols satisfying the above definitions compose sequentially in a classical environment. Furthermore, note that in Definition 2.2, we do not necessarily require the ideal-world adversary $\hat{\mathbf{A}}^*$ to be efficient. We show in Section 5 how to extend our coin-flipping construction such that we can achieve an efficient simulator.

The coin-flipping scheme in Section 5 as well as the example applications in Sections 4.1 and 4.2 work in the common-reference-string (CRS) model. In this model, all participants in the real-world protocol have access to a classical public CRS, which is chosen before any interaction starts, according to a distribution only depending on the security parameter. However, the participants in the ideal-world interacting with the ideal functionality do not make use of the CRS. Hence, an ideal-world simulator $\hat{\mathbf{P}}^*$ that operates by simulating a real-world adversary \mathbf{P}^* is free to choose a string in any way he wishes.

3 Quantum-Secure Coin-Flipping

3.1 The Coin-Flip Protocol

Let n indicate the security parameter of the commitment scheme which underlies the protocol. We use an *unconditionally binding* and *quantum-computationally hiding* commitment scheme that takes a bit and some randomness r of length l as input, i.e. $com : \{0, 1\} \times \{0, 1\}^l \rightarrow \{0, 1\}^{l+1}$. The unconditionally binding property is fulfilled, if it is impossible for any forger to open one commitment to both 0 and 1, i.e. to compute r, r' such that $com(0, r) = com(1, r')$. Quantum-computationally hiding is ensured, if no quantum distinguisher can distinguish between $com(0, r)$ and $com(1, r')$ for random r, r' with non-negligible advantage. As mentioned earlier, for a specific instantiation we can use, for instance, Naor's commitment based on a pseudorandom generator [15]. This scheme does not require any initially shared secret information and is secure against a quantum distinguisher.²

We let Alice and Bob run the **Coin – Flip Protocol** (see Fig. 1), which interactively generates a random and fair *coin* in one execution and does not require any set-up assumptions. Correctness is obvious by inspection of the protocol: If both players are honest, they independently choose random bits. These bits are then combined via exclusive disjunction, resulting in a uniformly random *coin*.

The corresponding ideal coin-flip functionality $\mathcal{F}_{\text{COIN}}$ is described in Figure 2. Note that dishonest \mathbf{A}^* may refuse to open $com(a, r)$ in the real-world after learning \mathbf{B} 's input. For this case, $\mathcal{F}_{\text{COIN}}$ allows her a second input **REFUSE**, leading to output **FAIL** and modeling the abort of the protocol.

² We describe the commitment scheme in this simple notation. However, if it is based on a specific scheme, e.g. [15], the precise notation has to be slightly adapted.

Coin – Flip Protocol

1. A chooses $a \in_R \{0, 1\}$ and computes $com(a, r)$. She sends $com(a, r)$ to B.
2. B chooses $b \in_R \{0, 1\}$ and sends b to A.
3. A sends $open(a, r)$ and B checks if the opening is valid.
4. Both compute $coin = a \oplus b$.

Fig. 1. The Coin-Flip Protocol**Ideal Functionality $\mathcal{F}_{\text{COIN}}$:**

Upon receiving requests `START` from Alice and Bob, $\mathcal{F}_{\text{COIN}}$ outputs a uniformly random *coin* to Alice. It then waits to receive Alice's second input `OK` or `REFUSE` and outputs *coin* or `FAIL` to Bob, respectively.

Fig. 2. The Ideal Coin-Flip Functionality**3.2 Security**

Theorem 3.1. *The Coin – Flip Protocol is unconditionally secure against any unbounded dishonest Alice according to Definition 2.2, provided that the underlying commitment scheme is unconditionally binding.*

Proof. We construct an ideal-world adversary \hat{A}^* , such that the real output of the protocol is statistically indistinguishable from the ideal output produced by \hat{A}^* , $\mathcal{F}_{\text{COIN}}$ and A^* .

First note that a, r and $com(a, r)$ are chosen and computed as in the real protocol. From the statistically binding property of the commitment scheme, it follows that A^* 's choice bit a is uniquely determined from $com(a, r)$, since for any com , there exists at most one pair (a, r) such that $com = com(a, r)$ (except with probability negligible in n). Hence in the real-world, A^* is unconditionally bound to her bit before she learns B's choice bit, which means a is independent of b . Therefore in Step 2, the simulator can correctly (but not necessarily efficiently) compute a (and r). Note that, in the case of unconditional security, we do not have to require the simulation to be efficient. We show in Section 5 how to extend the commitment in order to extract A^* 's inputs efficiently. Finally, due to the properties of XOR, A^* cannot tell the difference between the random b computed (from the ideal, random *coin*) in the simulation in Step 3 and the randomly chosen b of the real-world. It follows that the simulated output is statistically indistinguishable from the output in the real protocol. \square

To prove security against any dishonest quantum-computationally bounded B^* , we show that there exists an ideal-world simulation \hat{B}^* with output quantum-computationally indistinguishable from the output of the protocol in the

Ideal – World Simulation \hat{A}^* :

1. Upon receiving $com(a, r)$ from A^* , \hat{A}^* sends `START` and then `OK` to $\mathcal{F}_{\text{COIN}}$ as first and second input, respectively, and receives a uniformly random *coin*.
2. \hat{A}^* computes a and r from $com(a, r)$.
3. \hat{A}^* computes $b = coin \oplus a$ and sends b to A^* .
4. \hat{A}^* waits to receive A^* 's last message and outputs whatever A^* outputs.

Fig. 3. The Ideal-World Simulation \hat{A}^*

real-world. In a classical simulation, where we can simply use rewinding, a polynomial-time simulator works as follows. It inquires *coin* from $\mathcal{F}_{\text{COIN}}$, chooses random a and r , and computes $b' = coin \oplus a$ as well as $com(a, r)$. It then sends $com(a, r)$ to B^* and receives B^* 's choice bit b . If $b = b'$, the simulation was successful. Otherwise, the simulator rewinds B^* and repeats the simulation. Note that our security proof should hold also against any quantum adversary. The polynomial-time quantum simulator proceeds similarly to its classical analogue but requires quantum registers as work space and relies on the *quantum rewinding lemma* of Watrous [20] (see Lemma 11 in Appendix A).

In the paper, Watrous proves how to construct a quantum zero-knowledge proof system for graph isomorphism using his (ideal) quantum rewinding lemma. The protocol proceeds as a Σ -protocol, i.e. a protocol in three-move form, where the verifier flips a single coin in the second step and sends this challenge to the prover. Since these are the essential aspects also in our **Coin – Flip Protocol**, we can apply Watrous' quantum rewinding technique (with slight modifications) as a black-box to our protocol. We also follow his notation and line of argument here. For a more detailed description and proofs, we refer to [20].

Theorem 3.2. *The Coin – Flip Protocol is quantum-computationally secure against any polynomial-time bounded, dishonest Bob according to Definition 2.3, provided that the underlying commitment scheme is quantum-computationally hiding and the success probability of quantum rewinding achieves a non-negligible lower bound p_0 .*

Proof. Let W denote B^* 's auxiliary input register, containing an \tilde{n} -qubit state $|\psi\rangle$. Furthermore, let V and B denote B^* 's work space, where V is an arbitrary polynomial-size register and B is a single qubit register. A 's classical messages are considered in the following as being stored in quantum registers A_1 and A_2 . In addition, the quantum simulator uses registers R , containing all possible choices of a classical simulator, and G , representing its guess b' on B^* 's message b in the second step. Finally, let X denote a working register of size \tilde{k} , which is initialized to the state $|0^{\tilde{k}}\rangle$ and corresponds to the collection of all registers as described above except W .

The quantum rewinding procedure is implemented by a general quantum circuit R_{coin} with input $(W, X, \mathbf{B}^*, coin)$. As a first step, it applies a unitary (\tilde{n}, \tilde{k}) -quantum circuit Q to (W, X) to simulate the conversation, obtaining registers (G, Y) . Then, a test takes place to observe whether the simulation was successful. In that case, R_{coin} outputs the resulting quantum register. Otherwise, it *quantumly rewinds* by applying the reverse circuit Q^\dagger on (G, Y) to retrieve (W, X) and then a phase-flip transformation on X before another iteration of Q is applied. Note that R_{coin} is essentially the same circuit as R described in [20], but in our application it depends on the value of a given $coin$, i.e., we apply R_0 or R_1 for $coin = 0$ or $coin = 1$, respectively. In more detail, Q transforms (W, X) to (G, Y) by the following unitary operations:

- (1) It first constructs the superposition

$$\frac{1}{\sqrt{2^{l+1}}} \sum_{a,r} |a, r\rangle_R |com(a, r)\rangle_{A_1} |b' = coin \oplus a\rangle_G |open(a, r)\rangle_{A_2} |0\rangle_B |0^{\tilde{k}'}\rangle_V |\psi\rangle_W,$$

where $\tilde{k}' < \tilde{k}$. Note that the state of registers (A_1, G, A_2) corresponds to a uniform distribution of possible transcripts of the interaction between the players.

- (2) For each possible $com(a, r)$, it then simulates \mathbf{B}^* 's possible actions by applying a unitary operator to (W, V, B, A_1) with A_1 as control:

$$\frac{1}{\sqrt{2^{l+1}}} \sum_{a,r} |a, r\rangle_R |com(a, r)\rangle_{A_1} |b'\rangle_G |open(a, r)\rangle_{A_2} |b\rangle_B |\tilde{\phi}\rangle_V |\tilde{\psi}\rangle_W,$$

where $\tilde{\phi}$ and $\tilde{\psi}$ describe modified quantum states.

- (3) Finally, a CNOT-operation is applied to pair (B, G) with B as control to check whether the simulator's guess of \mathbf{B}^* 's choice was correct. The result of the CNOT-operation is stored in register G .

$$\frac{1}{\sqrt{2^{l+1}}} \sum_{a,r} |a, r\rangle_R |com(a, r)\rangle_{A_1} |b' \oplus b\rangle_G |open(a, r)\rangle_{A_2} |b\rangle_B |\tilde{\phi}\rangle_V |\tilde{\psi}\rangle_W.$$

If we denote with Y the register that contains the residual $\tilde{n} + \tilde{k} - 1$ -qubit state, the transformation from (W, X) to (G, Y) by applying Q can be written as

$$Q \left(|\psi\rangle_W |0^{\tilde{k}}\rangle_X \right) = \sqrt{p} |0\rangle_G |\phi_{good}(\psi)\rangle_Y + \sqrt{1-p} |1\rangle_G |\phi_{bad}(\psi)\rangle_Y,$$

where $0 < p < 1$ and $|\phi_{good}(\psi)\rangle$ denotes the state, we want the system to be in for a successful simulation. R_{coin} then measures the qubit in register G with respect to the standard basis, which indicates success or failure of the simulation. A successful execution (where $b = b'$) results in outcome 0 with probability p . In that case, R_{coin} outputs Y . A measurement outcome 1 indicates $b \neq b'$, in which case R_{coin} quantumly rewinds the system, applies a phase-flip (on register X) and repeats the simulation, i.e.

$$Q\left(2\left(\mathbb{I} \otimes |0^{\tilde{k}}\rangle\langle 0^{\tilde{k}}|\right) - \mathbb{I}\right)Q^\dagger.$$

Watrous’ ideal quantum rewinding lemma (without perturbations) then states the following: Under the condition that the probability p of a successful simulation is non-negligible and independent of any auxiliary input, the output $\rho(\psi)$ of R has square-fidelity close to 1 with state $|\phi_{good}(\psi)\rangle$ of a successful simulation, i.e.,

$$\langle \phi_{good}(\psi) | \rho(\psi) | \phi_{good}(\psi) \rangle \geq 1 - \varepsilon$$

with error bound $0 < \varepsilon < \frac{1}{2}$. Note that for the special case where p equals $1/2$ and is independent of $|\psi\rangle$, the simulation terminates after at most one rewinding.

However, we cannot apply the exact version of Watrous’ rewinding lemma in our simulation, since the commitment scheme in the protocol is only (quantum-) computationally hiding. Instead, we must allow for small perturbations in the quantum rewinding procedure as follows. Let adv denote B^* ’s advantage over a random guess on the committed value due to his computing power, i.e. $adv = |p - 1/2|$. From the hiding property, it follows that adv is negligible in the security parameter n . Thus, we can argue that the success probability p is *close* to independent of the auxiliary input and Watrous’ quantum rewinding lemma with small perturbations, as stated in the appendix (Lemma [D](#)), applies with $q = \frac{1}{2}$ and $\varepsilon = adv$. All operations in Q can be performed by polynomial-size circuits, and thus, the simulator has polynomial size (in the worst case). Furthermore, for negligible ε but non-negligible lower bound p_0 on the success probability p , it follows that the “closeness” of output $\rho(\psi)$ with good state $|\phi_{good}(\psi)\rangle$ is slightly reduced but quantum rewinding remains possible.

Finally, to proof security against quantum B^* , we construct an ideal-world quantum simulator \hat{B}^* (see Fig. [4](#)), interacting with B^* and the ideal functionality $\mathcal{F}_{\text{COIN}}$ and executing Watrous’ quantum rewinding algorithm. We then compare the output states of the real process and the ideal process. In case of indistinguishable outputs, quantum-computational security against B^* follows.

Ideal – World Simulation \hat{B}^* :

1. \hat{B}^* gets B^* ’s auxiliary quantum input W and working registers X .
2. \hat{B}^* sends **START** and then **OK** to $\mathcal{F}_{\text{COIN}}$. It receives a uniformly random *coin*.
3. Depending on the value of *coin*, \hat{B}^* applies the corresponding circuit R_{coin} with input W, X, B^* and *coin*.
4. \hat{B}^* receives output register Y with $|\phi_{good}(\psi)\rangle$ and “measures the conversation” to retrieve the corresponding $(com(a, r), b, open(a, r))$. It outputs whatever B^* outputs.

Fig. 4. The Ideal-World Simulation \hat{B}^*

First note that the superposition constructed as described above in circuit Q as Step (1) corresponds to all possible random choices of values in the real protocol. Furthermore, the circuit models any possible strategy of quantum B^* in Step (2), depending on control register $|com(a, r)\rangle_{A_1}$. The CNOT-operation on (B, G) in Step (3), followed by a standard measurement of G , indicate whether the guess b' on B^* 's choice b was correct. If that was not the case (i.e. $b \neq b'$ and measurement result 1), the system gets quantumly rewound by applying reverse transformations (3)-(1), followed by a phase-flip operation. The procedure is repeated until the measurement outcome is 0 and hence $b = b'$. Watrous' technique then guarantees that, assuming negligible ε and non-negligible p_0 , then ε' is negligible and thus, the final output $\rho(\psi)$ of the simulation is close to good state $|\phi_{good}(\psi)\rangle$. It follows that the output of the ideal simulation is indistinguishable from the output in the real-world for any quantum-computationally bounded B^* . \square

4 Applications

4.1 Interactive Quantum Zero-Knowledge

Zero-knowledge proofs are an important building block for larger cryptographic protocols. The notion of (interactive) zero-knowledge (ZK) was introduced by Goldwasser et al. [11]. Informally, ZK proofs for any NP language L yield no other knowledge to the verifier than the validity of the assertion proved, i.e. $x \in L$. Thus, only this one bit of knowledge is communicated from prover to verifier and zero additional knowledge. For a survey about zero-knowledge, see for instance [9,10].

Blum et al. [2] showed that the interaction between prover and verifier in any ZK proof can be replaced by sharing a short, random common reference string according to some distribution and available to all parties from the start of the protocol. Note that a CRS is a weaker requirement than interaction. Since all information is communicated mono-directional from prover to verifier, we do not have to require any restriction on the verifier.

As in the classical case, where ZK protocols exist if one-way functions exist, quantum zero-knowledge (QZK) is possible under the assumption that quantum one-way functions exist. In [14], Kobayashi showed that a common reference string or shared entanglement is necessary for non-interactive quantum zero-knowledge. Interactive quantum zero-knowledge protocols in restricted settings were proposed by Watrous in the honest verifier setting [19] and by Damgård et al. in the CRS model [5], where the latter introduced the first Σ -protocols for QZK withstanding even active quantum attacks. In [20], Watrous then proved that several interactive protocols are zero-knowledge against general quantum attacks.

Recently, Hallgren et al. [13] showed how to transform a Σ -protocol with stage-by-stage honest verifier zero-knowledge into a new Σ -protocol that is zero-knowledge against all classical and quantum verifiers. They propose special bit commitment schemes to limit the number of rounds, and view each round as a

IQZK $^{\mathcal{F}_{\text{COIN}}}$ Protocol:

(COIN)

1. A and B invoke $\mathcal{F}_{\text{COIN}}$ k times. If A blocks any output coin_i for $i = 1, \dots, k$ (by sending REFUSE as second input), B aborts the protocol.

(CRS)

2. A and B compute $\omega = \text{coin}_1 \dots \text{coin}_k$.

(NIZK)

3. A sends $\pi(\omega, x)$ to B. B checks the proof and accepts or rejects accordingly.

Fig. 5. Intermediate Protocol for IQZK

stage in which an honest verifier simulator is assumed. Then, by using a technique of [7], each stage can be converted to obtain zero-knowledge against any classical verifier. Finally, Watrous' quantum rewinding lemma is applied in each stage to prove zero-knowledge also against any quantum verifier.

Here, we propose a simpler transformation from non-interactive (quantum) zero-knowledge (NIZK) to interactive quantum zero-knowledge (IQZK) by combining the **Coin – Flip Protocol** with any **NIZK Protocol**. Our coin-flipping generates a truly random *coin* even in the case of a malicious quantum verifier. A sequence of such coins can then be used in any subsequent **NIZK Protocol**, which is also secure against quantum verifiers, due to its mono-direction. Here, we define a (NIZK)-subprotocol as given in [2]: Both parties A and B get common input x . A common reference string ω of size k allows the prover A, who knows a witness w , to give a non-interactive zero-knowledge proof $\pi(\omega, x)$ to a (quantum-) computationally bounded verifier B. By definition, the (NIZK)-subprotocol is complete and sound and satisfies zero-knowledge.

The **IQZK Protocol** is shown in Figure 7. To prove that it is an interactive quantum zero-knowledge protocol, we first construct an intermediate **IQZK $^{\mathcal{F}_{\text{COIN}}}$ Protocol** (see Fig. 5) that runs with the ideal functionality $\mathcal{F}_{\text{COIN}}$. Then we prove that the **IQZK $^{\mathcal{F}_{\text{COIN}}}$ Protocol** satisfies completeness, soundness and zero-knowledge according to standard definitions. Finally, by replacing the calls to $\mathcal{F}_{\text{COIN}}$ with our **Coin – Flip Protocol**, we can complete the transformation to the final **IQZK Protocol**.

Completeness: If $x \in L$, the probability that (A, B) rejects x is negligible in the length of x .

From the ideal functionality $\mathcal{F}_{\text{COIN}}$ it follows that each coin_i in Step 1 is uniformly random for all $i = 1, \dots, k$. Hence, ω in Step 2 is a uniformly random common reference string of size k . By definition of any (NIZK)-subprotocol, we have acceptance probability

$$\Pr[\omega \in_R \{0, 1\}^k, \pi(\omega, x) \leftarrow A(\omega, x, w) : B(\omega, x, \pi(\omega, x)) = 1] > 1 - \varepsilon'',$$

where ε'' is negligible in the length of x . Thus, completeness for the IQZK $^{\mathcal{F}_{\text{COIN}}}$ Protocol follows.

Soundness: If $x \notin L$, then for any unbounded prover A^* , the probability that (A^*, B) accepts x is negligible in the length of x .

Any dishonest A^* might stop the IQZK $^{\mathcal{F}_{\text{COIN}}}$ Protocol at any point during execution. For example, she can block the output in Step 1 or she can refuse to send a proof π in the (NIZK)-subprotocol. Furthermore, A^* can use an invalid ω (or x) for π . In all of these cases, B will abort without even checking the proof. Therefore, A^* 's best strategy is to “play the entire game”, i.e. to execute the entire IQZK $^{\mathcal{F}_{\text{COIN}}}$ Protocol without making obvious cheats.

A^* can only convince B in the (NIZK)-subprotocol of a π for any given (i.e. normally generated) ω with negligible probability

$$\Pr[\omega \in_R \{0, 1\}^k, \pi(\omega, x) \leftarrow A^*(\omega, x) : B(\omega, x, \pi(\omega, x)) = 1] .$$

Therefore, the probability that A^* can convince B in the entire IQZK $^{\mathcal{F}_{\text{COIN}}}$ Protocol in case of $x \notin L$ is also negligible (in the length of x) and its soundness follows.

Zero-Knowledge: An interactive proof system (A, B^*) for language L is quantum zero-knowledge, if for any quantum verifier B^* , there exists a simulator $\hat{S}_{\text{IQZK}^{\mathcal{F}_{\text{COIN}}}}$, such that $\hat{S}_{\text{IQZK}^{\mathcal{F}_{\text{COIN}}}} \stackrel{q}{\approx} (A, B^*)$ on common input $x \in L$ and arbitrary additional (quantum) input to B^* .

We construct simulator $\hat{S}_{\text{IQZK}^{\mathcal{F}_{\text{COIN}}}}$, interacting with dishonest B^* and simulator \hat{S}_{NIZK} . Under the assumption on the zero-knowledge property of any NIZK Protocol, there exists a simulator \hat{S}_{NIZK} that, on input $x \in L$, generates a randomly looking ω together with a valid proof π for x (without knowing witness w). $\hat{S}_{\text{IQZK}^{\mathcal{F}_{\text{COIN}}}}$ is described in Figure 6. It receives a random string ω from \hat{S}_{NIZK} , which now replaces the string of coins produced by the calls to $\mathcal{F}_{\text{COIN}}$ in the IQZK $^{\mathcal{F}_{\text{COIN}}}$ Protocol. The “merging” of coins into ω in Step 2 of the protocol (Fig. 5) is equivalent to the “splitting” of ω into coins in Step 3 of the simulation (Fig. 6). Thus, the simulated proof $\pi(\omega, x)$ is indistinguishable from a real proof, which shows that the IQZK $^{\mathcal{F}_{\text{COIN}}}$ Protocol is zero-knowledge.

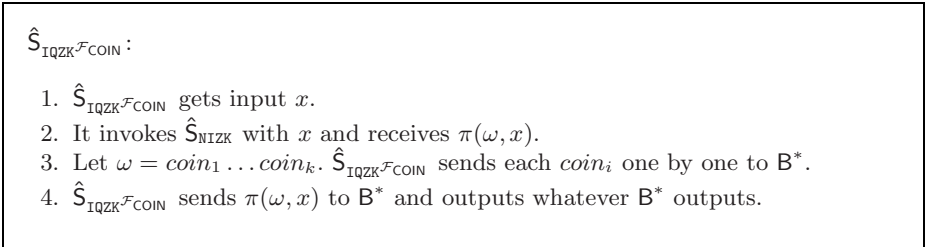


Fig. 6. The Simulation of the Intermediate Protocol for IQZK

IQZK Protocol:

(CFP) For all $i = 1, \dots, k$ repeat Steps 1. – 4.

1. A chooses $a_i \in_R \{0, 1\}$ and computes $\text{com}(a_i, r_i)$. She sends $\text{com}(a_i, r_i)$ to B.
2. B chooses $b_i \in_R \{0, 1\}$ and sends b_i to A.
3. A sends $\text{open}(a_i, r_i)$ and B checks if the opening is valid.
4. Both compute $\text{coin}_i = a_i \oplus b_i$.

(CRS)

5. A and B compute $\omega = \text{coin}_1 \dots \text{coin}_k$.

(NIZK)

6. A sends $\pi(\omega, x)$ to B. B checks the proof and accepts or rejects accordingly.

Fig. 7. Interactive Quantum Zero-Knowledge

It would be natural to think that the IQZK Protocol could be proved secure simply by showing that the $\text{IQZK}^{\mathcal{F}_{\text{COIN}}}$ Protocol implements some appropriate functionality and then use the composition theorem from [8]. Unfortunately, a zero-knowledge protocol – which is not necessarily a proof of knowledge – cannot be modeled by a functionality in a natural way. We therefore instead prove explicitly that the IQZK Protocol has the standard properties of a zero-knowledge proof as follows.

Completeness: From the analysis of the Coin – Flip Protocol and its indistinguishability from the ideal functionality $\mathcal{F}_{\text{COIN}}$, it follows that if both players honestly choose random bits, each coin_i for all $i = 1, \dots, k$ in the (CFP)-subprotocol is generated uniformly at random. Thus, ω is a random common reference string of size k and the acceptance probability of the (NIZK)-subprotocol as given above holds. Completeness for the IQZK Protocol follows.

Soundness: Again, we only consider the case where A^* executes the entire protocol without making obvious cheats, since otherwise, B immediately aborts. Assume that A^* could cheat in the IQZK Protocol, i.e., B would accept an invalid proof with non-negligible probability. Then we could combine A^* with simulator \hat{A}^* of the Coin – Flip Protocol (Fig. 3) to show that the $\text{IQZK}^{\mathcal{F}_{\text{COIN}}}$ Protocol was not sound. This, however, is inconsistent with the previously given soundness argument and thus proves by contradiction that the IQZK Protocol is sound.

Zero-Knowledge: A simulator \hat{S}_{IQZK} can be composed of simulator $\hat{S}_{\text{IQZK}^{\mathcal{F}_{\text{COIN}}}}$ (Fig. 6) and simulator \hat{B}^* for the Coin – Flip Protocol (Fig. 4). \hat{S}_{IQZK} gets classical input x as well as quantum input W and X . It then receives a valid proof π and a random string ω from \hat{S}_{NIZK} . As in $\hat{S}_{\text{IQZK}^{\mathcal{F}_{\text{COIN}}}}$, ω is split into $\text{coin}_1 \dots \text{coin}_k$. For each coin_i , it will then invoke \hat{B}^* to simulate one coin-flip execution with coin_i as result. In other words, whenever \hat{B}^* asks $\mathcal{F}_{\text{COIN}}$ to output a bit (Step 2, Fig. 4), it instead receives this coin_i . The transcript of the simulation, i.e. $\pi(\omega, x)$

as well as $(com(a_i, r_i), b_i, open(a_i, r_i)) \forall i = 1, \dots, k$ and $\omega = coin_1 \dots coin_k$, is indistinguishable from the transcript of the IQZK Protocol for any quantum-computationally bounded B^* , which concludes the zero-knowledge proof.

4.2 Generating Commitment Keys for Improved Quantum Protocols

Recently, Damgård et al. [4] proposed a general compiler for improving the security of a large class of quantum protocols. Alice starts such protocols by transmitting random BB84-qubits to Bob who measures them in random bases. Then some classical messages are exchanged to accomplish different cryptographic tasks. The original protocols are typically unconditionally secure against cheating Alice, and secure against a so-called *benignly* dishonest Bob, i.e., Bob is assumed to handle most of the received qubits as he is supposed to. Later on in the protocol, he can deviate arbitrarily. The improved protocols are then secure against an arbitrary computationally bounded (quantum) adversary. The compilation also preserves security in the bounded-quantum-storage model (BQSM) that assumes the quantum storage of the adversary to be of limited size. If the original protocol was BQSM-secure, the improved protocol achieves hybrid security, i.e., it can only be broken by an adversary who has large quantum memory *and* large computing power.

Briefly, the argument for computational security proceeds along the following lines. After the initial qubit transmission from A to B , B commits to all his measurement bases and outcomes. The (keyed) dual-mode commitment scheme that is used must have the special properties that the key can be generated by one of two possible key-generation algorithms: \mathcal{G}_H or \mathcal{G}_B . Depending of the key in use, the scheme provides both flavors of security. Namely, with key pk_H generated by \mathcal{G}_H , respectively pk_B produced by \mathcal{G}_B , the commitment scheme is unconditionally hiding respectively unconditionally binding. Furthermore, the scheme is secure against a quantum adversary and it holds that $pk_H \stackrel{v}{\approx} pk_B$. The commitment construction is described in full detail in [4].

In the real-life protocol, B uses the unconditionally hiding key pk_H to maintain unconditional security against any unbounded A^* . To argue security against a computationally bounded B^* , an information-theoretic argument involving simulator \hat{B}' (see [4]) is given to prove that B^* cannot cheat with the unconditionally binding key pk_B . Security in real life then follows from the quantum-computational indistinguishability of pk_H and pk_B .

The CRS model is assumed to achieve high efficiency and practicability. Here, we discuss integrating the generation of a common reference string from scratch based on our quantum-secure coin-flipping. Thus, we can implement the *entire process* in the quantum world, starting with the generation of a CRS without any initially shared information and using it during compilation as commitment key.³

³ Note that implementing the entire process comes at the cost of a non constant-round construction, added to otherwise very efficient protocols under the CRS-assumption.

As mentioned in [4], a dual-mode commitment scheme can be constructed from the lattice-based cryptosystem of Regev [18]. It is based on the learning with error problem, which can be reduced from worst-case (quantum) hardness of the (general) shortest vector problem. Hence, breaking Regev’s cryptosystem implies an efficient algorithm for approximating the lattice problem, which is assumed to be hard even quantumly. Briefly, the cryptosystem uses dimension k as security parameter and is parametrized by two integers m and p , where p is a prime, and a probability distribution on \mathbb{Z}_p . A regular public key for Regev’s scheme is indistinguishable from a case where a public key is chosen independently from the secret key, and in this case, the ciphertext carries essentially no information about the message. Thus, the public key of a regular key pair can be used as the unconditional binding key pkB' in the commitment scheme for the ideal-world simulation. Then for the real protocol, an unconditionally hiding commitment key pkH' can simply be constructed by uniformly choosing numbers in $\mathbb{Z}_p^k \times \mathbb{Z}_p$. Both public keys will be of size $O(mk \log p)$, and the encryption process involves only modular additions, which makes its use simple and efficient.

The idea is now the following. We add (at least) k executions of our `Coin – Flip Protocol` as a first step to the construction of [4] to generate a uniformly random sequence $\text{coin}_1 \dots \text{coin}_k$. These k random bits produce a pkH' as sampled by \mathcal{G}_H , except with negligible probability. Hence, in the real-world, Bob can use $\text{coin}_1 \dots \text{coin}_k = \text{pkH}'$ as key for committing to all his basis choices and measurement outcomes. Since an ideal-world adversary \hat{B}' is free to choose any key, it can generate $(\text{pkB}', \text{sk}')$, i.e. a regular public key together with a secret key according to Regev’s cryptosystem. For the security proof, write $\text{pkB}' = \text{coin}_1 \dots \text{coin}_k$. In the simulation, \hat{B}' first invokes \hat{B}^* for each coin_i to simulate one coin-flip execution with coin_i as result. As before, whenever \hat{B}^* asks $\mathcal{F}_{\text{COIN}}$ to output a bit, it instead receives this coin_i . Then \hat{B}' has the possibility to decrypt dishonest B^* ’s commitments during simulation, which binds B^* unconditionally to his committed measurement bases and outcomes. Finally, as we proved in the analysis of the `Coin – Flip Protocol` that pkH' is a uniformly random string, Regev’s proof of semantic security shows that $\text{pkH}' \stackrel{q}{\approx} \text{pkB}'$, and (quantum-) computational security of the real protocols in [4] follows.

5 On Efficient Simulation in the CRS Model

For our `Coin – Flip Protocol` in the plain model, we cannot claim universal composability. As already mentioned, in case of unconditional security against dishonest A^* according to Definition 2.2, we do not require the simulator to be efficient. In order to achieve efficient simulation, \hat{A}^* must be able to extract the choice bit efficiently out of A^* ’s commitment, such that A^* ’s input is defined after this step. The standard approach to do this is to give the simulator some trapdoor information related to the common reference string, that A^* does not have in real life. Therefore, we extend the commitment scheme to build in such

a trapdoor and ensure efficient extraction. To further guarantee UC-security, we circumvent the necessity of rewinding \mathbf{B}^* by extending the construction also with respect to equivocability.

We will adapt an approach to our set-up, which is based on the idea of UC-commitments [3] and already discussed in the full version of [4]. We require a Σ -protocol for a (quantumly) hard relation $R = \{(x, w)\}$, i.e. an honest verifier perfect zero-knowledge interactive proof of knowledge, where the prover shows that he knows a witness w such that the problem instance x is in the language L ($(x, w) \in R$). Conversations are of form $(a_\Sigma, c_\Sigma, z_\Sigma)$, where the prover sends a_Σ , the verifier challenges him with bit c_Σ , and the prover replies with z_Σ . For practical candidates of R , see e.g. [5]. Instead of the simple commitment scheme, we use the keyed dual-mode commitment scheme described in Section 4.2 but now based on a multi-bit version of Regev's scheme [17]. Still we construct it such that depending of the key \mathbf{pkH} or \mathbf{pkB} , the scheme provides both flavors of security and it holds that $\mathbf{pkH} \stackrel{\circ}{\approx} \mathbf{pkB}$.

In real life, the CRS consists of commitment key \mathbf{pkB} and an instance x' for which it holds that $\nexists w'$ such that $(x', w') \in R$, where we assume that $x \stackrel{\circ}{\approx} x'$. To commit to bit a , \mathbf{A} runs the honest verifier simulator to get a conversation (a_Σ, a, z_Σ) . She then sends a_Σ and two commitments c_0, c_1 to \mathbf{B} , where $c_a = \text{com}_{\mathbf{pkB}}(z_\Sigma, r)$ and $c_{1-a} = \text{com}_{\mathbf{pkB}}(0^{z'}, r')$ with randomness r, r' and $z' = |z|$. Then, a, z_Σ, r is send to open the relevant one of c_0 or c_1 , and \mathbf{B} checks that (a_Σ, a, z_Σ) is an accepting conversation. Assuming that the Σ -protocol is honest verifier zero-knowledge and \mathbf{pkB} leads to unconditionally binding commitments, the new commitment construction is again unconditionally binding.

During simulation, $\hat{\mathbf{A}}^*$ chooses a \mathbf{pkB} in the CRS such that it knows the matching decryption key \mathbf{sk} . Then, it can extract \mathbf{A}^* 's choice bit a by decrypting both c_0 and c_1 and checking which contains a valid z_Σ such that (a_Σ, a, z_Σ) is accepting. Note that not both c_0 and c_1 can contain a valid reply, since otherwise, \mathbf{A}^* would know a w' such that $(x', w') \in R$. In order to simulate in case of \mathbf{B}^* , $\hat{\mathbf{B}}^*$ chooses the CRS as \mathbf{pkH} and x . Hence, the commitment is unconditionally hiding. Furthermore, it can be equivocated, since $\exists w$ with $(x, w) \in R$ and therefore, c_0, c_1 can both be computed with valid replies, i.e. $c_0 = \text{com}_{\mathbf{pkH}}(z_{0\Sigma}, r)$ and $c_1 = \text{com}_{\mathbf{pkH}}(z_{1\Sigma}, r')$. Quantum-computational security against $\hat{\mathbf{B}}^*$ follows from the indistinguishability of the keys \mathbf{pkB} and \mathbf{pkH} and the indistinguishability of the instances x and x' , and efficiency of both simulations is ensured due to extraction and equivocability.

Acknowledgments

We thank Christian Schaffner and Serge Fehr for useful comments on an earlier version of the paper and the former also for discussing the issue of efficient simulation in earlier work. CL acknowledges financial support by the MOBISEQ research project funded by NABIIT, Denmark.

References

1. Bennett, C.H., Brassard, G., Crépeau, C., Skubiszewska, M.-H.: Practical quantum oblivious transfer. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 351–366. Springer, Heidelberg (1992)
2. Blum, M., Feldman, P., Micali, S.: Non-interactive zero-knowledge and its applications (extended abstract). In: 20th Annual ACM Symposium on Theory of Computing (STOC), pp. 103–112 (1988)
3. Canetti, R., Fischlin, M.: Universally Composable Commitments. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 19–40. Springer, Heidelberg (2001)
4. Damgård, I.B., Fehr, S., Lunemann, C., Salvail, L., Schaffner, C.: Improving the security of quantum protocols via commit-and-open. In: Halevi, S. (ed.) Advances in Cryptology—CRYPTO 2009. LNCS, vol. 5677, pp. 408–427. Springer, Heidelberg (2009), <http://arxiv.org/abs/0902.3918>
5. Damgård, I.B., Fehr, S., Salvail, L.: Zero-knowledge proofs and string commitments withstanding quantum attacks. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 254–272. Springer, Heidelberg (2004)
6. Damgård, I.B., Fehr, S., Salvail, L., Schaffner, C.: Secure identification and QKD in the bounded-quantum-storage model. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 342–359. Springer, Heidelberg (2007)
7. Damgård, I.B., Goldreich, O., Wigderson, A.: Hashing functions can simplify zero-knowledge protocol design (too). Technical Report RS-94-39, BRICS, Department of Computer Science, Aarhus University, Denmark (1994)
8. Fehr, S., Schaffner, C.: Composing quantum protocols in a classical environment. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 350–367. Springer, Heidelberg (2009)
9. Goldreich, O.: Foundations of Cryptography. Basic Tools, vol. I. Cambridge University Press, Cambridge (2001)
10. Goldreich, O.: Zero-knowledge twenty years after its invention (2002), <http://www.wisdom.weizmann.ac.il/~oded/papers.html>
11. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof-systems (extended abstract). In: 17th Annual ACM Symposium on Theory of Computing (STOC), pp. 291–304 (1985)
12. van de Graaf, J.: Towards a formal definition of security for quantum protocols. PhD thesis, Université de Montréal (1997)
13. Hallgren, S., Kolla, A., Sen, P., Zhang, S.: Making classical honest verifier zero knowledge protocols secure against quantum attacks. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfssdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 592–603. Springer, Heidelberg (2008)
14. Kobayashi, H.: Non-interactive quantum perfect and statistical zero-knowledge. In: Ibaraki, T., Katoh, N., Ono, H. (eds.) ISAAC 2003. LNCS, vol. 2906, pp. 178–188. Springer, Heidelberg (2003)
15. Naor, M.: Bit commitment using pseudorandomness. *Journal of Cryptology* 4(2), 151–158 (1991)
16. Nielsen, M.A., Chuang, I.L.: *Quantum Computation and Quantum Information*. Cambridge University Press, Cambridge (2000)
17. Peikert, C., Vaikuntanathan, V., Waters, B.: A framework for efficient and composable oblivious transfer. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 554–571. Springer, Heidelberg (2008)

18. 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 (2005)
19. Watrous, J.: Limits on the power of quantum statistical zero-knowledge. In: 43rd Annual IEEE Symposium on Foundations of Computer Science (FOCS), pp. 459–468 (2002)
20. Watrous, J.: Zero-knowledge against quantum attacks. *SIAM Journal on Computing* 39.1, 25–58 (2009); Preliminary version in 38th Annual ACM Symposium on Theory of Computing (STOC), pp. 296–305 (2006)
21. Wootters, W.K., Zurek, W.H.: A single quantum cannot be cloned. *Nature* 299, 802–803 (1982)

A Watrous’ Quantum Rewinding Lemma

Lemma 1 (Quantum Rewinding Lemma with small perturbations [20]).

Let Q be the unitary (\tilde{n}, \tilde{k}) -quantum circuit as given in [20]. Furthermore, let $p_0, q \in (0, 1)$ and $\varepsilon \in (0, \frac{1}{2})$ be real numbers such that

1. $|p - q| < \varepsilon$
2. $p_0(1 - p_0) \leq q(1 - q)$, and
3. $p_0 \leq p$

for all \tilde{n} -qubit states $|\psi\rangle$. Then there exists a general quantum circuit R of size

$$O\left(\frac{\log(1/\varepsilon)\text{size}(Q)}{p_0(1 - p_0)}\right)$$

such that, for every \tilde{n} -qubit state $|\psi\rangle$, the output $\rho(\psi)$ of R satisfies

$$\langle \phi_{\text{good}}(\psi) | \rho(\psi) | \phi_{\text{good}}(\psi) \rangle \geq 1 - \varepsilon'$$

where $\varepsilon' = 16\varepsilon \frac{\log^2(1/\varepsilon)}{p_0^2(1 - p_0)^2}$.

Note that p_0 denotes the lower bound on the success probability p , for which the procedure guarantees correctness. Furthermore, for negligible ε but non-negligible p_0 , it follows that ε' is negligible. For a more detailed description of the lemma and the corresponding proofs, we refer to [20].

On the Power of Two-Party Quantum Cryptography

Louis Salvail^{1,*}, Christian Schaffner^{2,**}, and Miroslava Sotáková³

¹ Université de Montréal (DIRO), QC, Canada
salvail@iro.umontreal.ca

² Centrum Wiskunde & Informatica (CWI) Amsterdam, The Netherlands
c.schaffner@cwi.nl

³ SUNY Stony Brook (Dept. of Computer Science), NY, USA
mirka@cs.au.dk

Abstract. We study quantum protocols among two distrustful parties. Under the sole assumption of correctness—guaranteeing that honest players obtain their correct outcomes—we show that every protocol implementing a non-trivial primitive necessarily leaks information to a dishonest player. This extends known impossibility results to all non-trivial primitives. We provide a framework for quantifying this leakage and argue that leakage is a good measure for the privacy provided to the players by a given protocol. Our framework also covers the case where the two players are helped by a trusted third party. We show that despite the help of a trusted third party, the players cannot amplify the cryptographic power of any primitive. All our results hold even against quantum honest-but-curious adversaries who honestly follow the protocol but purify their actions and apply a different measurement at the end of the protocol. As concrete examples, we establish lower bounds on the leakage of standard universal two-party primitives such as oblivious transfer.

Keywords: two-party primitives, quantum protocols, quantum information theory, oblivious transfer.

1 Introduction

Quantum communication allows to implement tasks which are classically impossible. The most prominent example is quantum key distribution [4] where two honest players establish a secure key against an eavesdropper. In the two-party setting however, quantum and classical cryptography often show similar limits. Oblivious transfer [22], bit commitment [24,23], and even fair coin tossing [18] are impossible to realize securely both classically and quantumly. On the other

* Supported by QUSEP (funded by the Danish Natural Science Research Council), Canada's NSERC, and the QuantumWorks network.

** Supported by EU fifth framework project QAP IST 015848 and the NWO VICI project 2004-2009.

hand, quantum cryptography allows for some weaker primitives impossible in the classical world. For example, quantum coin-flipping protocols with maximum bias of $\frac{1}{\sqrt{2}} - \frac{1}{2}$ exist¹ against any adversary [8] while remaining impossible based solely on classical communication. A few other weak primitives are known to be possible with quantum communication. For example, the generation of an additive secret-sharing for the product xy of two bits, where Alice holds bit x and Bob bit y , has been introduced by Popescu and Rohrlich as machines modeling non-signaling non-locality (also called NL-boxes) [29]. If Alice and Bob share an EPR pair, they can simulate an NL-box with symmetric error probability $\sin^2 \frac{\pi}{8}$ [29,3]. Equivalently, Alice and Bob can implement *1-out-of-2 oblivious transfer* (1-2-OT) privately provided the receiver Bob gets the bit of his choice only with probability of error $\sin^2 \frac{\pi}{8}$ [1]. It is easy to verify that even with such imperfection these two primitives are impossible to realize in the classical world. This discussion naturally leads to the following question:

- Which two-party cryptographic primitives are possible to achieve using quantum communication?

Most standard classical two-party primitives have been shown impossible to implement securely against weak quantum adversaries reminiscent to the classical honest-but-curious (HBC) behavior [22]. The idea behind these impossibility proofs is to consider parties that *purify* their actions throughout the protocol execution. This behavior is indistinguishable from the one specified by the protocol but guarantees that the joint quantum state held by Alice and Bob at any point during the protocol remains pure. The possibility for players to behave that way in any two-party protocol has important consequences. For instance, the impossibility of quantum bit commitment follows from this fact [24,23]: After the commit phase, Alice and Bob share the pure state $|\psi^x\rangle \in \mathcal{H}_A \otimes \mathcal{H}_B$ corresponding to the commitment of bit x . Since a proper commitment scheme provides no information about x to the receiver Bob, it follows that $\text{tr}_A |\psi^0\rangle\langle\psi^0| = \text{tr}_A |\psi^1\rangle\langle\psi^1|$. In this case, the Schmidt decomposition guarantees that there exists a unitary $U_{0,1}$ acting only on Alice's side such that $|\psi^1\rangle = (U_{0,1} \otimes \mathbb{I}_B)|\psi^0\rangle$. In other words, if the commitment is concealing then Alice can open the bit of her choice by applying a suitable unitary transform only to her part. A similar argument allows to conclude that 1-2-OT is impossible [22]: Suppose Alice is sending the pair of bits (b_0, b_1) to Bob through 1-2-OT. Since Alice does not learn Bob's selection bit, it follows that Bob can get bit b_0 before undoing the reception of b_0 and transforming it into the reception of b_1 using a local unitary transform similar to $U_{0,1}$ for bit commitment. For both these primitives, privacy for one player implies that local actions by the other player can transform the honest execution with one input into the honest execution with another input.

In this paper, we investigate the cryptographic power of two-party quantum protocols against players that purify their actions. This *quantum honest-but-curious* (QHBC) behavior is the natural quantum version of classical HBC

¹ In fact, protocols with better bias are known for weak quantum coin flipping [25,26,27].

behavior. We consider the setting where Alice obtains random variable X and Bob random variable Y according to the joint probability distribution $P_{X,Y}$. Any $P_{X,Y}$ models a two-party cryptographic primitive where neither Alice nor Bob provide input. For the purpose of this paper, this model is general enough since any two-party primitive with inputs can be randomized (Alice and Bob pick their input at random) so that its behavior can be described by a suitable joint probability distribution $P_{X,Y}$. If the randomized version $P_{X,Y}$ is shown to be impossible to implement securely by any quantum protocol then also the original primitive with inputs is impossible.

Any quantum protocol implementing $P_{X,Y}$ must produce, when both parties purify their actions, a joint pure state $|\psi\rangle \in \mathcal{H}_{AA'} \otimes \mathcal{H}_{BB'}$ that, when subsystems of A and B are measured in the computational basis, leads to outcomes X and Y according to the distribution $P_{X,Y}$. Notice that the registers A' and B' only provide the players with extra working space and, as such, do not contribute to the output of the functionality (so parties are free to measure them the way they want). In this paper, we adopt a somewhat strict point of view and define a quantum protocol π for $P_{X,Y}$ to be *correct* if and only if the correct outcomes X, Y are obtained *and* the registers A' and B' do not provide any additional information about Y and X respectively since otherwise π would be implementing a different primitive $P_{XX',YY'}$ rather than $P_{X,Y}$.

The state $|\psi\rangle$ produced by any correct protocol for $P_{X,Y}$ is called a *quantum embedding* of $P_{X,Y}$. An embedding is called *regular* if the registers A' and B' are empty. Any embedding $|\psi\rangle \in \mathcal{H}_{AA'} \otimes \mathcal{H}_{BB'}$ can be produced in the QHBC model by the trivial protocol asking Alice to generate $|\psi\rangle$ before sending the quantum state in $\mathcal{H}_{BB'}$ to Bob. Therefore, it is sufficient to investigate the cryptographic power of embeddings in order to understand the power of two-party quantum cryptography in the QHBC model.

Notice that if X and Y were provided privately to Alice and Bob—through a trusted third party for instance—then the expected amount of information one party gets about the other party’s output is minimal and can be quantified by the Shannon mutual information $I(X; Y)$ between X and Y . Assume that $|\psi\rangle \in \mathcal{H}_{AA'} \otimes \mathcal{H}_{BB'}$ is the embedding of $P_{X,Y}$ produced by a correct quantum protocol. We define the leakage of $|\psi\rangle$ as

$$\Delta_\psi := \max \{ S(X; BB') - I(X; Y), S(Y; AA') - I(Y; X) \}, \quad (1)$$

where $S(X; BB')$ (resp. $S(Y; AA')$) is the information the quantum registers BB' (resp. AA') provide about the output X (resp. Y). That is, the leakage is the maximum amount of extra information about the other party’s output given the quantum state held by one party. It turns out that $S(X; BB') = S(Y; AA')$ holds for all embeddings, exhibiting a symmetry similar to its classical counterpart $I(X; Y) = I(Y; X)$ and therefore, the two quantities we are taking the maximum of (in the definition of leakage above) coincide.

CONTRIBUTIONS. Our first contribution establishes that the notion of leakage is well behaved. We show that the leakage of any embedding for $P_{X,Y}$ is lower bounded by the leakage of some regular embedding of the same primitive. Thus,

in order to lower bound the leakage of any correct implementation of a given primitive, it suffices to minimize the leakage over all its regular embeddings. We also show that the only non-leaking embeddings are the ones for trivial primitives, where a primitive $P_{X,Y}$ is said to be (*cryptographically*) *trivial* if it can be generated by a classical protocol against HBC adversaries². It follows that any quantum protocol implementing a non-trivial primitive $P_{X,Y}$ must leak information under the sole assumption that it produces (X,Y) with the right joint distribution. This extends known impossibility results for two-party primitives to all non-trivial primitives.

Embeddings of primitives arise from protocols where Alice and Bob have full control over the environment. Having in mind that any embedding of a non-trivial primitive leaks information, it is natural to investigate what tasks can be implemented without leakage with the help of a trusted third party. The notion of leakage can easily be adapted to this scenario. We show that no cryptographic two-party primitive can be implemented without leakage with just one call to the ideal functionality of a weaker primitive³. This new impossibility result does not follow from the ones known since they all assume that the state shared between Alice and Bob is pure.

We then turn our attention to the leakage of correct protocols for a few concrete universal primitives. From the results described above, the leakage of any correct implementation of a primitive can be determined by finding the (regular) embedding that minimizes the leakage. In general, this is not an easy task since it requires to find the eigenvalues of the reduced density matrix $\rho_A = \text{tr}_B |\psi\rangle\langle\psi|$ (or equivalently $\rho_B = \text{tr}_A |\psi\rangle\langle\psi|$). As far as we know, no known results allow us to obtain a non-trivial lower bound on the leakage (which is the difference between the mutual information and accessible information) of non-trivial primitives. One reason being that in our setting we need to lower bound this difference with respect to a measurement in one particular basis. However, when $P_{X,Y}$ is such that the bit-length of either X or Y is short, the leakage can be computed precisely. We show that any correct implementation of 1-2-OT necessarily leaks $\frac{1}{2}$ bit. Since NL-boxes and 1-2-OT are locally equivalent, the same minimal leakage applies to NL-boxes [38]. This is a stronger impossibility result than the one by Lo [22] since he assumes perfect/statistical privacy against one party while our approach only assumes correctness (while both approaches apply even against QHBC adversaries). We finally show that for Rabin-OT and 1-2-OT of r -bit strings (i.e. ROT^r and $1\text{-}2\text{-OT}^r$ respectively), the leakage approaches 1 exponentially in r . In other words, correct implementations of these two primitives trivialize as r increases since the sender gets almost all information about Bob's

² We are aware of the fact that our definition of triviality encompasses cryptographically interesting primitives like coin-tossing and generalizations thereof for which highly non-trivial protocols exist [27,48]. However, the important fact (for the purpose of this paper) is that all these primitives can be implemented by *trivial* classical protocols against HBC adversaries.

³ The weakness of a primitive will be formally defined in terms of entropic monotones for classical two-party computation introduced by Wolf and Wullschlegel [36], see Section 4.2.

reception of the string (in case of ROT^r) and Bob’s choice bit (in case of $1\text{-}2\text{-OT}^r$). These are the first quantitative impossibility results for these primitives and certainly the first time the hardness of implementing different flavors of string OTs is shown to increase as the strings to be transmitted get longer.

Finally, we note that our lower bounds on the leakage of the randomized primitives also lower-bound the minimum leakage for the standard versions of these primitives⁴ where the players choose their inputs uniformly at random. While we focus on the typical case where the primitives are run with uniform inputs, the same reasoning can be applied to primitives with arbitrary distributions of inputs.

RELATED WORK. Our framework allows to quantify the minimum amount of leakage whereas standard impossibility proofs as the ones of [23,24,22,2,7] do not in general provide such quantification since they usually assume privacy for one player in order to show that the protocol must be totally insecure for the other player⁵. By contrast, we derive lower bounds for the leakage of any correct implementation. At first glance, our approach seems contradictory with standard impossibility proofs since embeddings leak the same amount towards both parties. To resolve this apparent paradox it suffices to observe that in previous approaches only the adversary purified its actions whereas in our case both parties do. If a honest player does not purify his actions then some leakage may be lost by the act of irreversibly and unnecessarily measuring some of his quantum registers.

Our results complement the ones obtained by Colbeck in [10] for the setting where Alice and Bob have inputs and obtain identical outcomes (called single-function computations). [10] shows that in any correct implementation of primitives of a certain form, an honest-but-curious player can access more information about the other party’s input than it is available through the ideal functionality. Unlike [10], we deal in our work with the case where Alice and Bob do not have inputs but might receive different outputs according to a joint probability distributions. We show that only trivial distributions can be implemented securely in the QHBC model. Furthermore, we introduce a quantitative measure of protocol-insecurity that lets us answer which embedding allow the least effective cheating.

Another notion of privacy in quantum protocols, generalizing its classical counterpart from [9,21], is proposed by Klauck in [19]. Therein, two-party quantum protocols with inputs for computing a function $f : \mathcal{X} \times \mathcal{Y} \rightarrow \mathcal{Z}$, where \mathcal{X} and \mathcal{Y} denote Alice’s and Bob’s respective input spaces, and privacy against QHBC

⁴ The definition of leakage of an embedding can be generalized to protocols with inputs, where it is defined as $\max\{\sup_{V_B} S(X; V_B) - I(X; Y), \sup_{V_A} S(V_A; Y) - I(X; Y)\}$, where X and Y involve both inputs and outputs of Alice and Bob, respectively. The supremum is taken over all possible (quantum) views V_A and V_B of Alice and Bob obtained by their (QHBC-consistent) actions (and containing their inputs).

⁵ Trade-offs between the security for one and the security for the other player have been considered before, but either the relaxation of security has to be very small [22] or the trade-offs are restricted to particular primitives such as commitments [34,6].

adversaries are considered. Privacy of a protocol is measured in terms of *privacy loss*, defined for each round of the protocol and fixed distribution of inputs $P_{X',Y'}$ by $S(B; X|Y) = H(X|Y) - S(X|B, Y)$, where B denotes Bob's private working register, and $X := (X', f(X', Y'))$, $Y := (Y', f(X', Y'))$ represent the complete views of Alice and Bob, respectively. Privacy loss of the entire protocol is then defined as the supremum over all joint input distributions, protocol rounds, and states of working registers. In our framework, privacy loss corresponds to $S(X; YB) - I(X; Y)$ from Alice point's of view and $S(Y; XA) - I(X; Y)$ from Bob's point of view. Privacy loss is therefore very similar to our definition of leakage except that it requires the players to get their respective honest outputs. As a consequence, the protocol implementing $P_{X,Y}$ by asking one party to prepare a regular embedding of $P_{X,Y}$ before sending her register to the other party would have no privacy loss. Moreover, the scenario analyzed in [19] is restricted to primitives which provide the same output $f(X, Y)$ to both players. Another difference is that since privacy loss is computed over all rounds of a protocol, a party is allowed to abort which is not considered QHBC in our setting. In conclusion, the model of [19] is different from ours even though the measures of privacy loss and leakage are similar. [19] provides interesting results concerning trade-offs between privacy loss and communication complexity of quantum protocols, building upon similar results of [9,21] in the classical scenario. It would be interesting to know whether a similar operational meaning can also be assigned to the new measure of privacy, introduced in this paper.

A recent result by Künzler et al. [20] shows that two-party functions that are securely computable against active quantum adversaries form a strict subset of the set of functions which are securely computable in the classical HBC model. This complements our result that the sets of securely computable functions in both HBC and QHBC models are the same.

ROADMAP. In Section 2, we introduce the cryptographic and information-theoretic notions and concepts used throughout the paper. We define, motivate, and analyze the generality of modeling two-party quantum protocols by embeddings in Section 3 and define trivality of primitives and embeddings. In Section 4, we define the notion of leakage of embeddings, show basic properties and argue that it is a reasonable measure of privacy. In Section 5, we explicitly lower bound the leakage of some universal two-party primitives. Finally, in Section 6 we discuss possible directions for future research and open questions.

2 Preliminaries

QUANTUM INFORMATION THEORY. Let $|\psi\rangle_{AB} \in \mathcal{H}_{AB}$ be an arbitrary pure state of the joint systems A and B . The states of these subsystems are $\rho_A = \text{tr}_B |\psi\rangle\langle\psi|$ and $\rho_B = \text{tr}_A |\psi\rangle\langle\psi|$, respectively. We denote by $S(A) := S(\rho_A)$ and $S(B) := S(\rho_B)$ the von Neumann entropy (defined as the Shannon entropy of the eigenvalues of the density matrix) of subsystem A and B respectively. Since the joint system is in a pure state, it follows from the Schmidt decomposition that $S(A) = S(B)$ (see e.g. [28]). Analogously to their classical counterparts, we

can define quantum conditional entropy $S(A|B) := S(AB) - S(B)$, and quantum mutual information $S(A; B) := S(A) + S(B) - S(AB) = S(A) - S(A|B)$. Even though in general, $S(A|B)$ can be negative, $S(A|B) \geq 0$ is always true if A is a classical register. Let $R = \{(P_X(x), \rho_R^x)\}_{x \in \mathcal{X}}$ be an ensemble of states ρ_R^x with prior probability $P_X(x)$. The average quantum state is $\rho_R = \sum_{x \in \mathcal{X}} P_X(x) \rho_R^x$. The famous result by Holevo upper-bounds the amount of classical information about X that can be obtained by measuring ρ_R :

Theorem 2.1 (Holevo bound [14,32]). *Let Y be the random variable describing the outcome of some measurement applied to ρ_R for $R = \{P_X(x), \rho_R^x\}_{x \in \mathcal{X}}$. Then, $I(X; Y) \leq S(\rho_R) - \sum_x P_X(x) S(\rho_R^x)$, where equality can be achieved if and only if $\{\rho_R^x\}_{x \in \mathcal{X}}$ are simultaneously diagonalizable.*

Note that if all states in the ensemble are pure and all different then in order to achieve equality in the theorem above, they have to form an orthonormal basis of the space they span. In this case, the variable Y achieving equality is the measurement outcome in this orthonormal basis.

DEPENDENT PART. The following definition introduces a random variable describing the correlation between two random variables X and Y , obtained by collapsing all values x_1 and x_2 for which Y has the same conditional distribution, to a single value.

Definition 2.2 (Dependent part [36]). *For two random variables X, Y , let $f_X(x) := P_{Y|X=x}$. Then the dependent part of X with respect to Y is defined as $X \searrow Y := f_X(X)$.*

The dependent part $X \searrow Y$ is the minimum random variable among the random variables computable from X for which $X \leftrightarrow X \searrow Y \leftrightarrow Y$ forms a Markov chain [36]. In other words, for any random variable $K = f(X)$ such that $X \leftrightarrow K \leftrightarrow Y$ is a Markov chain, there exists a function g such that $g(K) = X \searrow Y$. Immediately from the definition we get several other properties of $X \searrow Y$ [36]: $H(Y|X \searrow Y) = H(Y|X)$, $I(X; Y) = I(X \searrow Y; Y)$, and $X \searrow Y = X \searrow (Y \searrow X)$. The second and the third formula yield $I(X; Y) = I(X \searrow Y; Y \searrow X)$.

The notion of dependent part has been further investigated in [13,15,37]. Wullschleger and Wolf have shown that quantities $H(X \searrow Y|Y)$ and $H(Y \searrow X|X)$ are monotones for two-party computation [37]. That is, none of these values can increase during classical two-party protocols. In particular, if Alice and Bob start a protocol from scratch then classical two-party protocols can only produce (X, Y) such that: $H(X \searrow Y|Y) = H(Y \searrow X|X) = 0$, since $H(X \searrow Y|Y) > 0$ if and only if $H(Y \searrow X|X) > 0$ [37]. Conversely, any primitive satisfying $H(X \searrow Y|Y) = H(Y \searrow X|X) = 0$ can be implemented securely in the honest-but-curious (HBC) model. We call such primitives *trivial*⁶.

⁶ See Footnote 2 for a caveat about this terminology.

PURIFICATION. All security questions we ask are with respect to (*quantum*) *honest-but-curious* adversaries. In the classical honest-but-curious adversary model (HBC), the parties follow the instructions of a protocol but store all information available to them. Quantum honest-but-curious adversaries (QHBC), on the other hand, are allowed to behave in an arbitrary way that cannot be distinguished from their honest behavior by the other player.

Almost all impossibility results in quantum cryptography rely upon a quantum honest-but-curious behavior of the adversary. This behavior consists in *purifying* all actions of the honest players. Purifying means that instead of invoking classical randomness from a random tape, for instance, the adversary relies upon quantum registers holding all random bits needed. The operations to be executed from the random outcome are then performed quantumly without fixing the random outcomes. For example, suppose a protocol instructs a party to pick with probability p state $|\phi^0\rangle_C$ and with probability $1 - p$ state $|\phi^1\rangle_C$ before sending it to the other party through the quantum channel C . The purified version of this instruction looks as follows: Prepare a quantum register in state $\sqrt{p}|0\rangle_R + \sqrt{1-p}|1\rangle_R$ holding the random process. Add a new register initially in state $|0\rangle_C$ before applying the unitary transform $U : |r\rangle_R|0\rangle_C \mapsto |r\rangle_R|\phi^r\rangle_C$ for $r \in \{0, 1\}$, send register C through the quantum channel and keep register R .

From the receiver's point of view, the purified behavior is indistinguishable from the one relying upon a classical source of randomness because in both cases, the state of register C is $\rho = p|\phi^0\rangle\langle\phi^0| + (1-p)|\phi^1\rangle\langle\phi^1|$. All operations invoking classical randomness can be purified similarly [23,24,22,17]. The result is that measurements are postponed as much as possible and only extract information required to run the protocol in the sense that only when both players need to know a random outcome, the corresponding quantum register holding the random coin will be measured. If both players purify their actions then the joint state at any point during the execution will remain pure, until the very last step of the protocol when the outcomes are measured.

SECURE TWO-PARTY COMPUTATION. In Section 5, we investigate the leakage of several universal cryptographic two-party primitives. By universality we mean that any two-party secure function evaluation can be reduced to them. We investigate the completely randomized versions where players do not have inputs but receive randomized outputs instead. Throughout this paper, the term *primitive* usually refers to the joint probability distribution defining its randomized version. Any protocol implementing the standard version of a primitive (with inputs) can also be used to implement a randomized version of the same primitive, with the “inputs” chosen according to an arbitrary fixed probability distribution.

3 Two-Party Protocols and Their Embeddings

3.1 Correctness

In this work, we consider *cryptographic primitives* providing X to honest player Alice and Y to honest player Bob according to a joint probability distribution

$P_{X,Y}$. The goal of this section is to define when a protocol π *correctly implements* the primitive $P_{X,Y}$. The first natural requirement is that once the actions of π are purified by both players, measurements of registers A and B in the computational basis⁷ provide joint outcome $(X, Y) = (x, y)$ with probability $P_{X,Y}(x, y)$.

Protocol π can use extra registers A' on Alice's and B' on Bob's side providing them with (quantum) working space. The purification of all actions of π therefore generates a pure state $|\psi\rangle \in \mathcal{H}_{AB} \otimes \mathcal{H}_{A'B'}$. A second requirement for the correctness of the protocol π is that these extra registers are only used as working space, i.e. the final state $|\psi\rangle_{ABA'B'}$ is such that the content of Alice's working register A' does not give her any further information about Bob's output Y than what she can infer from her honest output X and vice versa for B' . Formally, we require that $S(XA'; Y) = I(X; Y)$ and $S(X; YB') = I(X; Y)$ or equivalently, that $A' \leftrightarrow X \leftrightarrow Y$ and $X \leftrightarrow Y \leftrightarrow B'$ form Markov chains⁸.

Definition 3.1. *A protocol π for $P_{X,Y}$ is correct if measuring registers A and B of its final state in the computational basis yields outcomes X and Y with distribution $P_{X,Y}$ and the final state satisfies $S(X; YB') = S(XA'; Y) = I(X; Y)$ where A' and B' denote the extra working registers of Alice and Bob. The state $|\psi\rangle \in \mathcal{H}_{AB} \otimes \mathcal{H}_{A'B'}$ is called an embedding of $P_{X,Y}$ if it can be produced by the purification of a correct protocol for $P_{X,Y}$.*

We would like to point out that our definition of correctness is stronger than the usual classical notion which only requires the correct distribution of the output of the honest players. For example, the trivial classical protocol for the primitive $P_{X,Y}$ in which Alice samples both player's outputs XY , sends Y to Bob, but keeps a copy of Y for herself, is *not correct* according to our definition, because it implements a fundamentally different primitive, namely $P_{XY,Y}$.

3.2 Regular Embeddings

We call an embedding $|\psi\rangle_{ABA'B'}$ *regular* if the working registers A', B' are empty. Formally, let $\Theta_{n,m} := \{\theta : \{0, 1\}^n \times \{0, 1\}^m \rightarrow [0 \dots 2\pi]\}$ be the set of functions mapping bit-strings of length $m + n$ to real numbers between 0 and 2π .

Definition 3.2. *For a joint probability distribution $P_{X,Y}$ where $X \in \{0, 1\}^n$ and $Y \in \{0, 1\}^m$, we define the set*

$$\mathcal{E}(P_{X,Y}) := \left\{ |\psi\rangle \in \mathcal{H}_{AB} : |\psi\rangle = \sum_{x \in \{0,1\}^n, y \in \{0,1\}^m} e^{i\theta(x,y)} \sqrt{P_{X,Y}(x,y)} |x, y\rangle_{AB}, \theta \in \Theta_{n,m} \right\},$$

⁷ It is clear that every quantum protocol for which the final measurement (providing (x, y) with distribution $P_{X,Y}$ to the players) is not in the computational basis can be transformed into a protocol of the described form by two additional local unitary transformations.

⁸ Markov chains with quantum ends have been defined in [11] and used in subsequent works such as [12]. It is straightforward to verify that the entropic condition $S(XA'; Y) = I(X; Y)$ is equivalent to $A' \leftrightarrow X \leftrightarrow Y$ being a Markov chain and similarly for the other condition.

and call any state $|\psi\rangle \in \mathcal{E}(P_{X,Y})$ a regular embedding of the joint probability distribution $P_{X,Y}$.

Clearly, any $|\psi\rangle \in \mathcal{E}(P_{X,Y})$ produces (X,Y) with distribution $P_{X,Y}$ since the probability that Alice measures x and Bob measures y in the computational basis is $|\langle\psi|x,y\rangle|^2 = P_{X,Y}(x,y)$. In order to specify a particular regular embedding one only needs to give the description of the *phase function* $\theta(x,y)$. We denote by $|\psi_\theta\rangle \in \mathcal{E}(P_{X,Y})$ the quantum embedding of $P_{X,Y}$ with phase function θ . The constant function $\theta(x,y) := 0$ for all $x \in \{0,1\}^n, y \in \{0,1\}^m$ corresponds to what we call *canonical embedding* $|\psi_0\rangle := \sum_{x,y} \sqrt{P_{X,Y}(x,y)}|x,y\rangle_{AB}$.

In Lemma 4.3 below we show that every primitive $P_{X,Y}$ has a regular embedding which is in some sense the most secure among all embeddings of $P_{X,Y}$.

3.3 Trivial Classical Primitives and Trivial Embeddings

In this section, we define *triviality* of classical primitives and (bipartite) embeddings. We show that for any non-trivial classical primitive, its canonical quantum embedding is also non-trivial. Intuitively, a primitive $P_{X,Y}$ is *trivial* if X and Y can be generated by Alice and Bob from scratch in the classical honest-but-curious (HBC) mode⁹. Formally, we define triviality via an entropic quantity based on the notion of *dependent part* (see Section 2).

Definition 3.3. A primitive $P_{X,Y}$ is called *trivial* if it satisfies $H(X \searrow Y|Y) = 0$, or equivalently, $H(Y \searrow X|X) = 0$. Otherwise, the primitive is called non-trivial.

Definition 3.4. A regular embedding $|\psi\rangle_{AB} \in \mathcal{E}(P_{X,Y})$ is called *trivial* if either $S(X \searrow Y|B) = 0$ or $S(Y \searrow X|A) = 0$. Otherwise, we say that $|\psi\rangle_{AB}$ is non-trivial.

Notice that unlike in the classical case, $S(X \searrow Y|B) = 0 \Leftrightarrow S(Y \searrow X|A) = 0$ does not hold in general. As an example, consider a shared quantum state where the computational basis corresponds to the Schmidt basis for only one of its subsystems, say for A . Let $|\psi\rangle = \alpha|0\rangle_A|\xi_0\rangle_B + \beta|1\rangle_A|\xi_1\rangle_B$ be such that both subsystems are two-dimensional, $\{|\xi_0\rangle, |\xi_1\rangle\} \neq \{|0\rangle, |1\rangle\}$, $\langle\xi_0|\xi_1\rangle = 0$, and $|\langle\xi_0|0\rangle| \neq |\langle\xi_1|0\rangle|$. We then have $S(X|B) = 0$ and $S(Y|A) > 0$ while $X = X \searrow Y$ and $Y = Y \searrow X$.

To illustrate this definition of triviality, we argue in the following that if a primitive $P_{X,Y}$ has a trivial regular embedding, there exists a classical protocol which generates X,Y securely in the HBC model. Let $|\psi\rangle \in \mathcal{E}(P_{X,Y})$ be trivial and assume without loss of generality that $S(Y \searrow X|A) = 0$. Intuitively, this means that Alice can learn everything possible about Bob’s outcome Y (Y could include some private coin-flips on Bob’s side, but that is “filtered out” by the dependent part). More precisely, Alice holding register A can measure her part of

⁹ See Footnote 2 for a caveat about this terminology.

the shared state to completely learn a realization of $Y \setminus X$, specifying $P_{X|Y=y}$. She then chooses X according to the distribution $P_{X|Y=y}$. An equivalent way of trivially generating (X, Y) classically is the following classical protocol:

1. Alice samples $P_{X|Y=y'}$ from distribution $P_{Y \setminus X}$ and announces its outcome to Bob. She samples x from the distribution $P_{X|Y=y'}$.
2. Bob picks y with probability $P_{Y|Y \setminus X = P_{X|Y=y'}}$.

Of course, the same reasoning applies in case $S(X \setminus Y|B) = 0$ with the roles of Alice and Bob reversed.

In fact, the following lemma (whose proof can be found in the full version [33]) shows that any non-trivial primitive $P_{X,Y}$ has a non-trivial embedding, i.e. there exists a quantum protocol correctly implementing $P_{X,Y}$ while leaking less information to QHBC adversaries than any classical protocol for $P_{X,Y}$ in the HBC model.

Lemma 3.5. *If $P_{X,Y}$ is a non-trivial primitive then the canonical embedding $|\psi_0\rangle \in \mathcal{E}(P_{X,Y})$ is also non-trivial.*

4 The Leakage of Quantum Embeddings

We formally define the leakage of embeddings and establish properties of the leakage. The proofs of all statements in this section can be found in the full version [33].

4.1 Definition and Basic Properties of Leakage

A perfect implementation of $P_{X,Y}$ simply provides X to Alice and Y to Bob and does nothing else. The expected amount of information that one random variable gives about the other is $I(X; Y) = H(X) - H(X|Y) = H(Y) - H(Y|X) = I(Y; X)$. Intuitively, we define the *leakage of a quantum embedding* $|\psi\rangle_{ABA'B'}$ of $P_{X,Y}$ as the larger of the two following quantities: the extra amount of information Bob's quantum registers BB' provide about X and the extra amount Alice's quantum state in AA' provides about Y respectively in comparison to “the minimum amount” $I(X; Y)$.¹⁰

Definition 4.1. *Let $|\psi\rangle \in \mathcal{H}_{ABA'B'}$ be an embedding of $P_{X,Y}$. We define the leakage $|\psi\rangle$ as*

$$\Delta_\psi(P_{X,Y}) := \max\{S(X; BB') - I(X; Y), S(AA'; Y) - I(X; Y)\}.$$

Furthermore, we say that $|\psi\rangle$ is δ -leaking if $\Delta_\psi(P_{X,Y}) \geq \delta$.

¹⁰ There are other natural candidates for the notion of leakage such as the difference in difficulty between guessing Alice's output X by measuring Bob's final quantum state B and based on the output of the ideal functionality Y . While such definitions do make sense, they turn out not to be as easy to work with and it is an open question whether the natural properties described later in this section can be established for these notions of leakage as well.

It is easy to see that the leakage is non-negative since $S(X; BB') \geq S(X; \tilde{B})$ for \tilde{B} the result of a quantum operation applied to BB' . Such an operation could be the trace over the extra working register B' and a measurement in the computational basis of each qubit of the part encoding Y , yielding $S(X; \tilde{B}) = I(X; Y)$.

We want to argue that our notion of leakage is a good measure for the privacy of the player's outputs. In the same spirit, we will argue that the minimum achievable leakage for a primitive is related to the "hardness" of implementing it. We start off by proving several basic properties about leakage.

For a general state in $\mathcal{H}_{ABA'B'}$ the quantities $S(X; BB') - I(X; Y)$ and $S(AA'; Y) - I(X; Y)$ are not necessarily equal. Note though that they coincide for regular embeddings $|\psi\rangle \in \mathcal{E}(P_{X,Y})$ produced by a correct protocol (where the work spaces A' and B' are empty): Notice that $S(X; B) = S(X) + S(B) - S(X, B) = H(X) + S(B) - H(X) = S(B)$ and because $|\psi\rangle$ is pure, $S(A) = S(B)$. Therefore, $S(X; B) = S(A; Y)$ and the two quantities coincide. The following lemma states that this actually happens for *all* embeddings and hence, the definition of leakage is symmetric with respect to both players.

Lemma 4.2 (Symmetry). *Let $|\psi\rangle \in \mathcal{H}_{ABA'B'}$ be an embedding of $P_{X,Y}$. Then,*

$$\Delta_\psi(P_{X,Y}) = S(X; BB') - I(X; Y) = S(AA'; Y) - I(X; Y).$$

The next lemma shows that the leakage of an embedding of a given primitive is lower-bounded by the leakage of some regular embedding of the same primitive, which simplifies the calculation of lower bounds for the leakage of embeddings.

Lemma 4.3. *For every embedding $|\psi\rangle$ of a primitive $P_{X,Y}$, there is a regular embedding $|\psi'\rangle$ of $P_{X,Y}$ such that $\Delta_\psi(P_{X,Y}) \geq \Delta_{\psi'}(P_{X,Y})$.*

So far, we have defined the leakage of an embedding of a primitive. The natural definition of the leakage of a primitive is the following.

Definition 4.4. *We define the leakage of a primitive $P_{X,Y}$ as the minimal leakage among all protocols correctly implementing $P_{X,Y}$. Formally,*

$$\Delta_{P_{X,Y}} := \min_{|\psi\rangle} \Delta_\psi(P_{X,Y}),$$

where the minimization is over all embeddings $|\psi\rangle$ of $P_{X,Y}$.

Notice that the minimum in the previous definition is well-defined, because by Lemma 4.3, it is sufficient to minimize over regular embeddings $|\psi\rangle \in \mathcal{E}(P_{X,Y})$. Furthermore, the function $\Delta_\psi(P_{X,Y})$ is continuous on the compact (i.e. closed and bounded) set $[0, 2\pi]^{|\mathcal{X} \times \mathcal{Y}|}$ of complex phases corresponding to elements $|x, y\rangle_{AB}$ in the formula for $|\psi\rangle_{AB} \in \mathcal{E}(P_{X,Y})$ and therefore it achieves its minimum.

The following theorem shows that the leakage of any embedding of a primitive $P_{X,Y}$ is lower-bounded by the minimal leakage achievable for primitive $P_{X \setminus Y, Y \setminus X}$ (which due to Lemma 4.3 is achieved by a regular embedding).

Theorem 4.5. *For any primitive $P_{X,Y}$, $\Delta_{P_{X,Y}} \geq \Delta_{P_{X \setminus Y, Y \setminus X}}$.*

Proof (Sketch). The proof idea is to pre-process the registers storing X and Y in a way allowing Alice and Bob to convert a regular embedding of $P_{X,Y}$ (for which the minimum leakage is achieved) into a regular embedding of $P_{X \setminus Y, Y \setminus X}$ by measuring parts of these registers. It follows that on average, the leakage of the resulting regular embedding of $P_{X \setminus Y, Y \setminus X}$ is at most the leakage of the embedding of $P_{X,Y}$ the players started with. Hence, there must be a regular embedding of $P_{X \setminus Y, Y \setminus X}$ leaking at most as much as the best embedding of $P_{X,Y}$. See [33] for the complete proof. \square

4.2 Leakage as Measure of Privacy and Hardness of Implementation

The main results of this section are consequences of the Holevo bound (Theorem 2.1).

Theorem 4.6. *If a two-party quantum protocol provides the correct outcomes of $P_{X,Y}$ to the players without leaking extra information, then $P_{X,Y}$ must be a trivial primitive.*

Proof. Theorem 4.5 implies that if there is a 0-leaking embedding of $P_{X,Y}$ than there is also a 0-leaking embedding of $P_{X \setminus Y, Y \setminus X}$. Let us therefore assume that $|\psi\rangle$ is a non-leaking embedding of $P_{X,Y}$ such that $X = X \setminus Y$ and $Y = Y \setminus X$. We can write $|\psi\rangle$ in the form $|\psi\rangle = \sum_x \sqrt{P_X(x)}|x\rangle|\varphi_x\rangle$ and get $\rho_B = \sum_x P_X(x)|\varphi_x\rangle\langle\varphi_x|$. For the leakage of $|\psi\rangle$ we have: $\Delta_\psi(P_{X,Y}) = S(X;B) - I(X;Y) = S(\rho_B) - I(X;Y) = 0$. From the Holevo bound (Theorem 2.1) follows that the states $\{|\varphi_x\rangle\}_x$ form an orthonormal basis of their span (since $X = X \setminus Y$, they are all different) and that Y captures the result of a measurement in this basis, which therefore is the computational basis. Since $Y = Y \setminus X$, we get that for each x , there is a single $y_x \in \mathcal{Y}$ such that $|\varphi_x\rangle = |y_x\rangle$. The primitives $P_{X \setminus Y, Y \setminus X}$ and $P_{X,Y}$ are therefore trivial. \square

In other words, the only primitives that two-party quantum protocols can implement correctly (without the help of a trusted third party) and without leakage are the trivial ones! We note that it is not necessary to use the strict notion of correctness from Definition 3.1 in this theorem, but a more complicated proof can be done solely based on the correct distribution of the values. This result can be seen as a quantum extension of the corresponding characterization for the cryptographic power of classical protocols in the HBC model. Whereas classical two-party protocols cannot achieve anything non-trivial, their quantum counterparts necessarily leak information when they implement non-trivial primitives.

The notion of leakage can be extended to protocols involving a trusted third party (see [33]). A special case of such protocols are the ones where the players are allowed one call to a black box for a certain non-trivial primitive. It is natural to ask which primitives can be implemented without leakage in this case. As it turns out, the monotones $H(X \setminus Y|Y)$ and $H(Y \setminus X|X)$, introduced in [36], are also monotones for quantum computation, in the sense that all joint

random variables X', Y' that can be generated by quantum players without leakage using one black-box call to $P_{X,Y}$ satisfy $H(X' \searrow Y'|Y') \leq H(X \searrow Y|Y)$ and $H(Y' \searrow X'|X') \leq H(Y \searrow X|X)$.

Theorem 4.7. *Suppose that primitives $P_{X,Y}$ and $P_{X',Y'}$ satisfy $H(X' \searrow Y'|Y') > H(X \searrow Y|Y)$ or $H(Y' \searrow X'|X') > H(Y \searrow X|X)$. Then any implementation of $P_{X',Y'}$ using just one call to the ideal functionality for $P_{X,Y}$ leaks information.*

4.3 Reducibility of Primitives and Their Leakage

This section is concerned with the following question: Given two primitives $P_{X,Y}$ and $P_{X',Y'}$ such that $P_{X,Y}$ is reducible to $P_{X',Y'}$, what is the relationship between the leakage of $P_{X,Y}$ and the leakage of $P_{X',Y'}$? We use the notion of reducibility in the following sense: We say that a primitive $P_{X,Y}$ is *reducible in the HBC model* to a primitive $P_{X',Y'}$ if $P_{X,Y}$ can be securely implemented in the HBC model from (one call to) a secure implementation of $P_{X',Y'}$. The above question can also be generalized to the case where $P_{X,Y}$ can be computed from $P_{X',Y'}$ only with certain probability. Notice that the answer, even if we assume perfect reducibility, is not captured in our previous result from Lemma 4.3, since an embedding of $P_{X',Y'}$ is not necessarily an embedding of $P_{X,Y}$ (it might violate the correctness condition). However, under certain circumstances, we can show that $\Delta_{P_{X',Y'}} \geq \Delta_{P_{X,Y}}$.

Theorem 4.8. *Assume that primitives $P_{X,Y}$ and $P_{X',Y'} = P_{X'_0 X'_1, Y'_0 Y'_1}$ satisfy the condition:*

$$\sum_{x,y: P_{X'_0, Y'_0 | X'_1=x, Y'_1=y} \simeq P_{X,Y}} P_{X'_1, Y'_1}(x, y) \geq 1 - \delta,$$

where the relation \simeq means that the two distributions are equal up to relabeling of the alphabet. Then, $\Delta_{P_{X',Y'}} \geq (1 - \delta)\Delta_{P_{X,Y}}$.

This theorem allows us to derive a lower bound on the leakage of 1-out-of-2 Oblivious Transfer of r -bit strings in Section 5.

5 The Leakage of Universal Cryptographic Primitives

In this section, we exhibit lower bounds on the leakage of some universal two-party primitives. In the following table, ROT^r denotes the r -bit string version of randomized Rabin OT, where Alice receives a random r -bit string and Bob receives the same string or an erasure symbol, each with probability 1/2. Similarly, $1\text{-}2\text{-OT}^r$ denotes the string version of 1-2-OT, where Alice receives two r -bit strings and Bob receives one of them. By $1\text{-}2\text{-OT}_p$ we denote the noisy version of 1-2-OT, where the 1-2-OT functionality is implemented correctly only with probability $1 - p$. Table 1 summarizes the lower bounds on the leakage of these primitives (the derivations can be found in the full version [33]). We note that Wolf and Wullschlegel [38] have shown that a randomized 1-2-OT can be

Table 1. Lower bounds on the leakage for universal two-party primitives

primitive	leaking at least	comments
ROT^1	$(h(\frac{1}{4}) - \frac{1}{2}) \approx 0.311$	same leakage for all regular embeddings
ROT^r	$(1 - O(r2^{-r}))$	same leakage for all regular embeddings
1-2-OT, SAND	$\frac{1}{2}$	minimized by canonical embedding
1-2-OT ^r	$(1 - O(r2^{-r}))$	(suboptimal) lower bound
1-2-OT _p	$\frac{(1/2-p-\sqrt{p(1-p)})^2}{8 \ln 2}$	if $p < \sin^2(\pi/8) \approx 0.15$, (suboptimal) lower bound

transformed by local operations into an additive sharing of an AND (here called SAND). Therefore, our results for 1-2-OT below also apply to SAND.

1-2-OT^r and 1-2-OT_p are primitives where the direct evaluation of the leakage for a general embedding $|\psi_\theta\rangle$ is hard, because the number of possible phases increases exponentially in the number of qubits. Instead of computing $S(A)$ directly, we derive (suboptimal) lower bounds on the leakage.

Based on the examples of ROT^r and 1-2-OT, it is tempting to conjecture that the leakage is always minimized for the canonical embedding, which agrees with the geometric intuition that the minimal pairwise distinguishability of quantum states in a mixture minimizes the von Neumann entropy of the mixture. However, Jozsa and Schlienz have shown that this intuition is sometimes incorrect [16]. In a quantum system of dimension at least three, we can have the following situation: For two sets of pure states $\{|u_i\rangle\}_{i=1}^n$ and $\{|v_i\rangle\}_{i=1}^n$ satisfying $|\langle u_i|u_j\rangle| \leq |\langle v_i|v_j\rangle|$ for all i, j , there exist probabilities p_i such that for $\rho_u := \sum_{i=1}^n p_i |u_i\rangle\langle u_i|$, $\rho_v := \sum_{i=1}^n p_i |v_i\rangle\langle v_i|$, it holds that $S(\rho_u) < S(\rho_v)$. As we can see, although each pair $|u_i\rangle, |u_j\rangle$ is more distinguishable than the corresponding pair $|v_i\rangle, |v_j\rangle$, the overall ρ_u provides us with less uncertainty than ρ_v . It follows that although for the canonical embedding $|\psi_0\rangle = \sum_y |\varphi_y\rangle|y\rangle$ of $P_{X,Y}$ the mutual overlaps $|\langle \varphi_y|\varphi_{y'}\rangle|$ are clearly maximized, it does not necessarily imply that $S(A)$ in this case is minimal over $\mathcal{E}(P_{X,Y})$. It is an interesting open question to find a primitive whose canonical embedding does not minimize the leakage or to prove that no such primitive exists.

For the primitive $P_{X,Y}^{\text{OT}_p}$, our lower bound on the leakage only holds for $p < \sin^2(\pi/8) \approx 0.15$. Notice that in reality, the leakage is strictly positive for any embedding of $P_{X,Y}^{\text{OT}_p}$ with $p < 1/4$, since for $p < 1/4$, $P_{X,Y}^{\text{OT}_p}$ is a non-trivial primitive. On the other hand, $P_{X,Y}^{\text{OT}_{1/4}}$ is a trivial primitive implemented securely by the following protocol in the classical HBC model:

1. Alice chooses randomly between her input bits x_0 and x_1 and sends the chosen value x_a to Bob.
2. Bob chooses his selection bit c uniformly at random and sets $y := x_a$.

Equality $x_c = y$ is satisfied if either $a = c$, which happens with probability $1/2$, or if $a \neq c$ and $x_a = x_{1-a}$, which happens with probability $1/4$. Since the

two events are disjoint, it follows that $x_c = y$ with probability $3/4$ and that the protocol implements $P_{X,Y}^{\text{ot}_{1/4}}$. The implementation is clearly secure against honest-but-curious Alice, since she does not receive any message from Bob. It is also secure against Bob, since he receives only one bit from Alice. By letting Alice randomize the value of the bit she is sending, the players can implement $P_{X,Y}^{\text{ot}_p}$ securely for any value $1/4 < p \leq 1/2$.

6 Conclusion and Open Problems

We have provided a quantitative extension of qualitative impossibility results for two-party quantum cryptography. All non-trivial primitives leak information when implemented by quantum protocols. Notice that demanding a protocol to be non-leaking does in general not imply the privacy of the players' outputs. For instance, consider a protocol implementing 1-2-OT but allowing a curious receiver with probability $\frac{1}{2}$ to learn both bits simultaneously or with probability $\frac{1}{2}$ to learn nothing about them. Such a protocol for 1-2-OT would be non-leaking but nevertheless insecure. Consequently, Theorem 4.6 not only tells us that any quantum protocol implementing a non-trivial primitive must be insecure, but also that a privacy breach will reveal itself as leakage. Our framework allows to quantify the leakage of any two-party quantum protocol correctly implementing a primitive. The impossibility results obtained here are stronger than standard ones since they only rely on the cryptographic correctness of the protocol. Furthermore, we present lower bounds on the leakage of some universal two-party primitives.

A natural open question is to find a way to identify good embeddings for a given primitive. In particular, how far can the leakage of the canonical embedding be from the best one? Such a characterization, even if only applicable to special primitives, would allow to lower bound their leakage and would also help to understand the power of two-party quantum cryptography in a more concise way.

It would also be interesting to find a measure of cryptographic non-triviality for two-party primitives and to see how it relates to the minimum leakage of any implementation by quantum protocols. For instance, is it true that quantum protocols for primitive $P_{X,Y}$ leak more if the minimum (total variation) distance between $P_{X,Y}$ and any trivial primitive increases?

Another question we leave for future research is to define and investigate other notions of leakage, e.g. in the one-shot setting instead of in the asymptotic regime (as outlined in Footnote 10). Results in the one-shot setting have already been established for data compression [30], channel capacities [31], state-merging [35,5] and other (quantum-) information-theoretic tasks.

Furthermore, it would be interesting to find more applications for the concept of leakage, considered also for protocols using an environment as a trusted third party. In this direction, we have shown in Theorem 4.7 that any two-party quantum protocol for a given primitive, using a black box for an “easier” primitive, leaks information. Lower-bounding this leakage is an interesting open question.

We might also ask how many copies of the “easier” primitive are needed to implement the “harder” primitive by a quantum protocol, which would give us an alternative measure of non-triviality of two-party primitives.

References

1. Ambainis, A.: personal communication (2005)
2. Ariano, G.M.D., Kretschmann, D., Schlingemann, D., Werner, R.F.: Reexamination of quantum bit commitment: The possible and the impossible. *Physical Review A (Atomic, Molecular, and Optical Physics)* 76(3), 032328 (2007)
3. Barrett, J., Linden, N., Massar, S., Pironio, S., Popescu, S., Roberts, D.: Nonlocal correlations as an information-theoretic resource. *Physical Review A* 71, 022101 (2005)
4. Bennett, C.H., Brassard, G.: Quantum cryptography: Public key distribution and coin tossing. In: *Proceedings of IEEE International Conference on Computers, Systems, and Signal Processing*, pp. 175–179 (1984)
5. Berta, M.: Single-shot quantum state merging. Master’s thesis, ETH Zurich (2008)
6. Buhrman, H., Christandl, M., Hayden, P., Lo, H.-K., Wehner, S.: Possibility, impossibility and cheat-sensitivity of quantum bit string commitments. *Physical Review A* 78, 022316 (2008)
7. Buhrman, H., Christandl, M., Schaffner, C.: Impossibility of two-party secure function evaluation (in preparation, 2009)
8. Chailloux, A., Kerenidis, I.: Optimal quantum strong coin flipping (2009), <http://arxiv.org/abs/0904.1511>
9. Chor, B., Kushilevitz, E.: A zero-one law for boolean privacy. *SIAM J. Discrete Math.* 4(1), 36–47 (1991)
10. Colbeck, R.: The impossibility of secure two-party classical computation (August 2007), <http://arxiv.org/abs/0708.2843>
11. Damgård, I.B., Fehr, S., Salvail, L., Schaffner, C.: Secure identification and QKD in the bounded-quantum-storage model. In: Menezes, A. (ed.) *CRYPTO 2007*. LNCS, vol. 4622, pp. 342–359. Springer, Heidelberg (2007)
12. Fehr, S., Schaffner, C.: Composing quantum protocols in a classical environment. In: Reingold, O. (ed.) *TCC 2009*. LNCS, vol. 5444, pp. 350–367. Springer, Heidelberg (2009)
13. Fitz, M., Wolf, S., Wullschlegel, J.: Pseudo-signatures, broadcast, and multi-party computation from correlated randomness. In: Franklin, M. (ed.) *CRYPTO 2004*. LNCS, vol. 3152, pp. 562–579. Springer, Heidelberg (2004)
14. Holevo, A.S.: Information-theoretical aspects of quantum measurement. *Problemy Peredači Informacii* 9(2), 31–42 (1973)
15. Imai, H., Müller-Quade, J., Nascimento, A., Winter, A.: Rates for bit commitment and coin tossing from noisy correlation. In: *Proceedings of 2004 IEEE International Symposium on Information Theory*, p. 47 (June 2004)
16. Jozsa, R., Schlienz, J.: Distinguishability of states and von neumann entropy. *Phys. Rev. A* 62(1), 012301 (2000)
17. Kent, A.: Promising the impossible: Classical certification in a quantum world (2004), <http://arxiv.org/abs/quant-ph/0409029>
18. Kitaev, A.: Quantum coin-flipping. presented at QIP 2003. A review of this technique can be found in (2003), <http://lightlike.com/~carlosm/publ>

19. Klauck, H.: On quantum and approximate privacy. *Theory of Computing Systems* 37(1), 221–246 (2004); <http://arxiv.org/abs/quant-ph/0110038>, also in the Proceedings of STACS (2002)
20. Künzler, R., Müller-Quade, J., Raub, D.: Secure computability of functions in the it setting with dishonest majority and applications to long-term security. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 238–255. Springer, Heidelberg (2009)
21. Kushilevitz, E.: Privacy and communication complexity. *SIAM J. Discrete Math.* 5(2), 273–284 (1992)
22. Lo, H.-K.: Insecurity of quantum secure computations. *Physical Review A* 56(2), 1154–1162 (1997)
23. Lo, H.-K., Chau, H.F.: Is quantum bit commitment really possible? *Physical Review Letters* 78(17), 3410–3413 (1997)
24. Mayers, D.: Unconditionally secure quantum bit commitment is impossible. *Physical Review Letters* 78(17), 3414–3417 (1997)
25. Mochon, C.: Quantum weak coin-flipping with bias of 0.192. In: 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS), pp. 2–11 (2004)
26. Mochon, C.: A large family of quantum weak coin-flipping protocols. *Phys. Rev. A* 72, 022341 (2005)
27. Mochon, C.: Quantum weak coin flipping with arbitrarily small bias (2007), <http://arxiv.org/abs/0711.4114>
28. Nielsen, M.A., Chuang, I.L.: *Quantum Computation and Quantum Information*. Cambridge University Press, Cambridge (2000)
29. Popescu, S., Rohrlich, D.: Quantum nonlocality as an axiom. *Foundations of Physics* 24(3), 379–385 (1994)
30. Renner, R., Wolf, S.: Simple and tight bounds for information reconciliation and privacy amplification. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 199–216. Springer, Heidelberg (2005)
31. Renner, R., Wolf, S., Wullschleger, J.: The single-serving channel capacity. In: Proceedings of the International Symposium on Information Theory (ISIT). IEEE, Los Alamitos (July 2006), <http://arxiv.org/abs/cs.IT/0608018>
32. Ruskai, M.B.: Inequalities for quantum entropy: A review with conditions for equality. *Journal of Mathematical Physics* 43(9), 4358–4375 (2002)
33. Salvail, L., Sotáková, M., Schaffner, C.: On the power of two-party quantum cryptography (2009), <http://arxiv.org/abs/0902.4036>
34. Spekkens, R.W., Rudolph, T.: Degrees of concealment and bindingness in quantum bit commitment protocols. *Phys. Rev. A* 65(1), 012310 (2001)
35. Winter, A., Renner, R.: Single-shot state merging (2007) (unpublished note)
36. Wolf, S., Wullschleger, J.: Zero-error information and applications in cryptography. In: IEEE Information Theory Workshop (ITW), San Antonio, Texas (October 2004)
37. Wolf, S., Wullschleger, J.: New monotones and lower bounds in unconditional two-party computation. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 467–477. Springer, Heidelberg (2005)
38. Wolf, S., Wullschleger, J.: Oblivious transfer and quantum non-locality. In: International Symposium on Information Theory (ISIT 2005), pp. 1745–1748 (2005)

Security Bounds for the Design of Code-Based Cryptosystems

Matthieu Finiasz¹ and Nicolas Sendrier²

¹ ENSTA

² INRIA, team-project SECRET

Abstract. Code-based cryptography is often viewed as an interesting “Post-Quantum” alternative to the classical number theory cryptography. Unlike many other such alternatives, it has the convenient advantage of having only a few, well identified, attack algorithms. However, improvements to these algorithms have made their effective complexity quite complex to compute. We give here some lower bounds on the work factor of idealized versions of these algorithms, taking into account all possible tweaks which could improve their practical complexity. The aim of this article is to help designers select durably secure parameters.

Keywords: computational syndrome decoding, information set decoding, generalized birthday algorithm.

Introduction

Code-based cryptography has received renewed attention with the recent interest for “Post-Quantum Cryptography” (see for instance [5]). Several new interesting proposals have been published in the last few months [3,20,18]. For those new constructions as well as for previously known code-based cryptosystems, precise parameters selection is always a sensitive issue. Most of the time the most threatening attacks are based on decoding algorithms for generic linear codes. There are two main families of algorithms, Information Set Decoding (ISD), and Generalized Birthday Algorithm (GBA). Each family being suited for some different parameter ranges.

ISD is part of the folklore of algorithmic coding theory and is among the most efficient techniques for decoding errors in an arbitrary linear code. One major step in the development of ISD for the cryptanalysis of the McEliece encryption scheme is Stern’s variant [22] which mixes birthday attack with the traditional approach. A first implementation description [10], with several improvements, led to an attack of $2^{64.2}$ binary operations for the original McEliece parameters, that is decoding 50 errors in a code of length 1024 and dimension 524. More recently [6], a new implementation was proposed with several new improvements, with a binary workfactor of $2^{60.5}$. Furthermore, the authors report a real attack (with the original parameters) with a computational effort of about 2^{58} CPU cycles. The above numbers are accurate estimates of the real cost of a decoding

attack. They involve several parameters that have to be optimized and furthermore, no close formula exists, making a precise evaluation rather difficult.

GBA was introduced by Wagner in 2002 [25] but was not specifically designed for decoding. Less generic version of this algorithm had already been used in the past for various cryptanalytic applications [9][11]. Its first successful use to cryptanalyse a code-based system is due to Coron and Joux [12]. In particular, this work had a significant impact for selecting the parameters of the FSB hash function [1].

Most previous papers on decoding attacks were written from the point of view of the attacker and were looking for upper bounds on the work factor of some specific implementation. One exception is the asymptotic analysis for ISD that has been recently presented in [8]. Here we propose a designer approach and we aim at providing tools to easily select secure parameters.

For both families, we present new idealized version of the algorithms, which encompass all variants and improvements known in cryptology as well as some new optimizations. This allows us to give easy to compute lower bounds for decoding attacks up to the state of the art.

We successively study three families of algorithms, first the “standard” birthday attack, then two evolutions of this technique, namely Stern’s variant of information set decoding and Wagner’s generalized birthday algorithm. In each case we propose very generic lower bounds on their complexity. Finally, we illustrate our work with case studies of some of the main code-based cryptosystems.

1 The Decoding Problem in Cryptology

Problem 1 (Computational Syndrome Decoding - CSD). *Given a matrix $H \in \{0, 1\}^{r \times n}$, a word $s \in \{0, 1\}^r$ and an integer $w > 0$, find a word $e \in \{0, 1\}^n$ of Hamming weight $\leq w$ such that $eH^T = s$.*

We will denote $\text{CSD}(H, s, w)$ an instance of that problem. It is equivalent to decoding w errors in a code with parity check matrix H . The decision problem associated with computational syndrome decoding, namely, Syndrome Decoding, is NP-complete [4].

This problem appears in code-based cryptography and for most systems it is the most threatening known attack (sometimes the security can be reduced to CSD alone [1][23]). Throughout the paper we will denote

$$W_{n,w} = \{e \in \{0, 1\}^n \mid \text{wt}(e) = w\}$$

the set of all binary words of length n and Hamming weight w . The instances of CSD coming from cryptology usually have solutions. Most of the time, this solution is unique. This is the case for public-key encryption schemes [17][21] or for identification schemes [23][24]. However, if the number w of errors is larger than the Gilbert-Varshamov distance¹ we may have a few, or even a large number, of solutions. Obtaining one of them is enough. This is the case for digital signatures [13] or for hashing [1][2].

¹ The Gilbert-Varshamov distance is the smallest integer d_0 such that $\binom{n}{d_0} \geq 2^r$.

2 The Birthday Attack for Decoding

We consider an instance $\text{CSD}(H, s, w)$ of the computational syndrome decoding. If the weight w is even, we partition the columns of H in two subsets (a priori of equal size). For instance, let $H = (H_1 \mid H_2)$ and let us consider the sets $\mathcal{L}_1 = \{e_1 H_1^T \mid e_1 \in W_{n/2, w/2}\}$ and $\mathcal{L}_2 = \{s + e_2 H_2^T \mid e_2 \in W_{n/2, w/2}\}$. Any element of $\mathcal{L}_1 \cap \mathcal{L}_2$ provides a pair (e_1, e_2) such that $e_1 H_1 = s + e_2 H_2$ and $e_1 + e_2$ is a solution to $\text{CSD}(H, s, w)$. This collision search has to be repeated $1/\text{Pr}_{n,w}$ times on average where $\text{Pr}_{n,w}$ is the probability that one of the solutions splits evenly between the left and right parts of H . Let $C_{n,r,w}$ denote the total number of columns sums we have to compute. If the solution is unique, we have²

$$\text{Pr}_{n,w} = \frac{\binom{n/2}{w/2}^2}{\binom{n}{w}} \text{ and } C_{n,r,w} = \frac{|\mathcal{L}_1| + |\mathcal{L}_2|}{\text{Pr}_{n,w}} = \frac{2\binom{n}{w}}{\binom{n/2}{w/2}} \approx 2\sqrt{\binom{n}{w}} \sqrt[4]{\frac{\pi w}{2}}$$

This number is close to the improvement expected when the birthday paradox can be applied (*i.e.* replacing an enumeration of N elements by an enumeration of $2\sqrt{N}$ elements). In this section, we will show that the factor $\sqrt[4]{\pi w/2}$ can be removed and that the formula often applies when w is odd. We will also provide cost estimations and bounds.

2.1 A Decoding Algorithm Using the Birthday Paradox

The algorithm presented in Table 1 generalizes the birthday attack for decoding presented above. For any fixed values of n , r and w this algorithm uses three parameters (to be optimized): an integer ℓ and two sets of constant weight words W_1 and W_2 .

The idea is to operate as much as possible with partial syndromes of size $\ell < r$ and to make the full comparison on r bits only when we have a partial match. Increasing the size of W_1 (and W_2) will lead to a better trade-off, ideally with a single execution of (MAIN LOOP).

Definition 1. For any fixed value of n , r and w , we denote $\text{WF}_{\text{BA}}(n, r, w)$ the minimal binary work factor (average cost in binary operations) of the algorithm of Table 1 to produce a solution to CSD , for any choices of parameters W_1 , W_2 and ℓ .

An Estimation of the Cost. We will use the following assumptions (discussed in appendix):

(B1) For all pairs (e_1, e_2) examined in the algorithm, the sums $e_1 + e_2$ are uniformly and independently distributed in $W_{n,w}$.

² We use Stirling's formula to approximate factorials. The approximation we give is valid because $w \ll n$.

Table 1. Birthday decoding algorithm

For any fixed values of n , r and w , the following algorithm uses three parameters: an integer $\ell > 0$, $W_1 \subset W_{n, \lfloor w/2 \rfloor}$ and $W_2 \subset W_{n, \lceil w/2 \rceil}$. We denote by $h_\ell(x)$ the first ℓ bits of any $x \in \{0, 1\}^r$.

```

procedure BirthdayDecoding
input:  $H_0 \in \{0, 1\}^{r \times n}$ ,  $s \in \{0, 1\}^r$ 
  repeat (MAIN LOOP)
     $P \leftarrow$  random  $n \times n$  permutation matrix
     $H \leftarrow H_0 P$ 
    for all  $e \in W_1$ 
       $i \leftarrow h_\ell(eH^T)$  (BA 1)
      write( $e, i$ ) // store  $e$  in some data structure at index  $i$ 
    for all  $e_2 \in W_2$ 
       $i \leftarrow h_\ell(s + e_2 H^T)$  (BA 2)
       $S \leftarrow$  read( $i$ ) // extract the elements stored at index  $i$ 
      for all  $e_1 \in S$ 
        if  $e_1 H^T = s + e_2 H^T$  (BA 3)
          return  $(e_1 + e_2)P^T$  (SUCCESS)

```

(B2) The cost of the execution of the algorithm is approximatively equal to

$$\ell \cdot \#(\text{BA 1}) + \ell \cdot \#(\text{BA 2}) + K_0 \cdot \#(\text{BA 3}), \quad (1)$$

where K_0 is the cost for testing $e_1 H^T = s + e_2 H^T$ given that $h_\ell(e_1 H^T) = h_\ell(s + e_2 H^T)$ and $\#(\text{BA } i)$ is the expected number of execution of the instruction (BA i) before we meet the (SUCCESS) condition.

Proposition 1. Under assumptions **(B1)** and **(B2)**. We have³

$$\text{WF}_{\text{BA}}(n, r, w) \approx 2L \log(K_0 L) \text{ with } L = \min\left(\sqrt{\binom{n}{w}}, 2^{r/2}\right)$$

and K_0 is the cost for executing the instruction (BA 3) (i.e. testing $eH^T = s$).

Remarks

1. When $\binom{n}{w} > 2^r$, the cost will depend of the number of syndromes 2^r instead of the number of words of weight w . This corresponds to the case where w is larger than the Gilbert-Varshamov distance and we have multiple solutions. We only need one of those solutions and thus the size of the search space is reduced.

³ Here and after, “log” denotes the base 2 logarithm (and “ln” the Neperian logarithm).

2. It is interesting to note the relatively low impact of K_0 , the cost of the test in (BA 3). Between an extremely conservative lower bound of $K_0 = 2$, an extremely conservative upper bound of $K_0 = wr$ and a more realistic $K_0 = 2w$ the differences are very small.
3. In the case where w is odd and $\binom{n}{\lfloor w/2 \rfloor} < L$, the formula of Proposition 1 is only a lower bound. A better estimate would be

$$\text{WF}_{\text{BA}}(n, r, w) \approx 2L' \log \left(K_0 \frac{L^2}{L'} \right) \text{ with } L' = \frac{\binom{n}{\lfloor w/2 \rfloor}^2 + L^2}{2 \binom{n}{\lfloor w/2 \rfloor}}. \quad (2)$$

4. Increasing the size of $|W_1|$ (and $|W_2|$) can be easily and efficiently achieved by “overlapping” H_1 and H_2 (see the introduction of this section). More precisely, we take for W_1 all words of weight $w/2$ using only the n' first coordinates (with $n/2 < n' < n$). Similarly, W_2 will use the n' (or more) last coordinates.

2.2 Lower Bounds

As the attacker can make a clever choice of W_1 and W_2 which may contradict assumption (B1), we do not want to use it for the lower bound. The result remains very close to the estimate of the previous sections except for the multiplicative constant which is $\sqrt{2}$ instead of 2.

Theorem 1. *For any fixed value of n , r and w , we have*

$$\text{WF}_{\text{BA}}(n, r, w) \geq \sqrt{2}L \log(K_0L) \text{ with } L = \min \left(\sqrt{\binom{n}{w}}, 2^{r/2} \right).$$

where K_0 is the cost for executing the instruction (BA 3).

3 Information Set Decoding (ISD)

We will consider here Stern’s algorithm [22], which is the best known decoder for cryptographic purposes, and some of its implemented variants by Canteaut-Chabaud [10] and Bernstein-Lange-Peters [6]. Our purpose is to present a lower bound which takes all known improvements into account.

3.1 A New Variant of Stern’s Algorithm

Following other works [15, 16], J. Stern describes in [22] an algorithm to find a word of weight w in a binary linear code of length n and dimension k (and codimension $r = n - k$). The algorithm uses two additional parameters p and ℓ (both positive integers). We present here a generalized version which acts on the parity check matrix H_0 of the code (instead of the generator matrix). Table 2

Table 2. Generalized ISD algorithm

For any fixed values of n , r and w , the following algorithm uses four parameters: two integers $p > 0$ and $\ell > 0$ and two sets $W_1 \subset W_{k+\ell, \lfloor p/2 \rfloor}$ and $W_2 \subset W_{k+\ell, \lfloor p/2 \rfloor}$. We denote by $h_\ell(x)$ the last ℓ bits of any $x \in \{0, 1\}^r$.	
procedure ISDecoding	
input: $H_0 \in \{0, 1\}^{r \times n}$, $s_0 \in \{0, 1\}^r$	
repeat	(MAIN LOOP)
$P \leftarrow$ random $n \times n$ permutation matrix	
$(H', U) \leftarrow$ PGElim($H_0 P$) // partial elimination as in (3)	
$s \leftarrow s_0 U^T$	
for all $e \in W_1$	
$i \leftarrow h_\ell(e H'^T)$ (ISD 1)	
write(e, i) // store e in some data structure at index i	
for all $e_2 \in W_2$	
$i \leftarrow h_\ell(s + e_2 H'^T)$ (ISD 2)	
$S \leftarrow$ read(i) // extract the elements stored at index i	
for all $e_1 \in S$	
if wt($s + (e_1 + e_2) H'^T$) = $w - p$ (ISD 3)	
return ($P, e_1 + e_2$) (SUCCESS)	

describes the algorithm. The partial Gaussian elimination of $H_0 P$ consists in finding U ($r \times r$ and non-singular) and H (and H') such that⁴

$$U H_0 P = H = \begin{array}{c} \begin{array}{|cc|} \hline r - \ell & k + \ell \\ \hline 1 & \\ \vdots & \\ & 1 \\ \hline \ell & 0 \\ \hline \end{array} & \begin{array}{|c|} \hline H' \\ \hline \end{array} \\ \hline \end{array} \quad (3)$$

where U is a non-singular $r \times r$ matrix. Let $s = s_0 U^T$. If e is a solution of $\text{CSD}(H, s, w)$ then $e P^T$ is a solution of $\text{CSD}(H_0, s_0, w)$. Let (P, e') be the output of the algorithm, *i.e.*, $\text{wt}(s + e' H'^T) = w - p$, and let e'' be the first $r - \ell$ bits of $s + e' H'^T$, the word $e = (e'' \mid e')$ is a solution of $\text{CSD}(H, s, w)$.

Definition 2. For any fixed value of n , r and w , we denote $\text{WF}_{\text{ISD}}(n, r, w)$ the minimal binary work factor (average cost in binary operations) of the algorithm of Table 2 to produce a solution to CSD , for any choices of parameters ℓ, p, W_1 and W_2 .

3.2 Estimation of the Cost of the New Variant

To evaluate the cost of the algorithm we will assume that only the instructions (ISD i) are significant. This assumption is stronger than for the birthday attack,

⁴ In the very unlikely event that the first $r - \ell$ columns are linearly dependent, we can change P .

because it means that the Gaussian elimination at the beginning of every (MAIN LOOP) costs nothing. It is a valid assumption as we only want a lower bound. Moreover, most of the improvements introduced in [10,6] are meant to reduce the relative cost of the Gaussian elimination. We claim that within this “free Gaussian elimination” assumption any lower bound on the algorithm of Table 2 will apply on all the variants of [6,10]. Our estimations will use the following assumptions:

- (I1) For all pairs (e_1, e_2) examined in the algorithm, the sums $e_1 + e_2$ are uniformly and independently distributed in $W_{k+\ell,p}$.
- (I2) The cost of the execution of the algorithm is approximatively equal to

$$\ell \cdot \sharp(\text{ISD } 1) + \ell \cdot \sharp(\text{ISD } 2) + K_{w-p} \cdot \sharp(\text{ISD } 3), \tag{4}$$

where K_{w-p} is the average cost for checking $\text{wt}(s + (e_1 + e_2)H^T) = w - p$ and $\sharp(\text{ISD } i)$ is the expected number of executions of the instruction (ISD i) before we meet the (SUCCESS) condition.

Proposition 2. *Under assumptions (I1) and (I2). If $\binom{n}{w} < 2^r$ (single solution) or if $\binom{n}{w} > 2^r$ (multiple solutions) and $\binom{r}{w-p} \binom{k}{p} \ll 2^r$, we have (we recall that $k = n - r$)*

$$\text{WF}_{\text{ISD}}(n, r, w) \approx \min_p \frac{2\ell \min(\binom{n}{w}, 2^r)}{\lambda \binom{r-\ell}{w-p} \sqrt{\binom{k+\ell}{p}}} \text{ with } \ell = \log\left(K_{w-p} \sqrt{\binom{k}{p}}\right)$$

with $\lambda = 1 - e^{-1} \approx 0.63$. If $\binom{n}{w} > 2^r$ (multiple solutions) and $\binom{r}{w-p} \binom{k}{p} \geq 2^r$, we have

$$\text{WF}_{\text{ISD}}(n, r, w) \approx \min_p \frac{2\ell 2^{r/2}}{\sqrt{\binom{r-\ell}{w-p}}} \text{ with } \ell = \log\left(K_{w-p} \frac{2^{r/2}}{\sqrt{\binom{r}{w-p}}}\right).$$

Remarks

1. For a given set of parameters the expected number of execution of (MAIN LOOP) is $N = 1/(1 - \exp(-X))$ where $X = \binom{r+\ell}{w-p} \binom{k+\ell}{p} / \min(2^r, \binom{n}{w})$.
2. The second formula applies when $X > 1$, that is when the expected number of execution of (MAIN LOOP) is (not much more than) one. In that case, as for the birthday attack, the best strategy is to use $W_2 = W_{k+\ell, \lceil p/2 \rceil}$ (i.e. as large as possible) and W_1 is as small as possible but large enough to have only one execution of (MAIN LOOP) with probability close to 1.
3. When $X \ll 1$, we have $N = 1/(1 - \exp(-X)) \approx 1/X$ and the first formula applies.
4. When $X < 1$, the first formula still gives a good lower bound. But it is less tight when X gets closer to 1.
5. When p is small and odd the above estimates for WF_{ISD} are not always accurate. The adjustment is similar to what we have in (2) (see the remarks following the birthday decoder estimation). In practice, if $\binom{k+\ell}{\lceil p/2 \rceil} < \sqrt{\binom{k+\ell}{p}}$ it is probably advisable to discard this odd value of p .

6. We use the expression $\ell = \log(K_{w-p}L_p(0))$ for the optimal value of ℓ (where $L_p(\ell) = \sqrt{\binom{k+\ell}{p}}$ or $L_p(\ell) = 2^{r/2}/\sqrt{\binom{r-\ell}{w-p}}$ respectively in the first case or in the second case of the Proposition). In fact a better value would be a fixpoint of the mapping $\ell \mapsto L_p(\ell)$. In practice $L_p(0)$ is a very good approximation.

3.3 Gain Compared with Stern’s Algorithm

Stern’s algorithm corresponds to a complete Gaussian elimination and to a particular choice of W_1 and W_2 in the algorithm of Table 2. A full Gaussian elimination is applied to the permuted matrix H_0P and we get U and H' such that:

$$UH_0P = H = \begin{array}{c} \begin{array}{|c|} \hline 1 \\ \hline \end{array} \begin{array}{|c|} \hline \vdots \\ \hline \end{array} \begin{array}{|c|} \hline \vdots \\ \hline \end{array} \begin{array}{|c|} \hline 1 \\ \hline \end{array} \begin{array}{|c|} \hline H_1 \\ \hline \end{array} \begin{array}{|c|} \hline H_2 \\ \hline \end{array} \end{array} \quad (5)$$

The ℓ -bit collision search is performed on k columns, moreover p is always even and W_1 and W_2 will use $p/2$ columns of H_1 and H_2 . The variants presented in [6,10] consist in reducing the cost of the Gaussian elimination, or, for the same H' , to use different “slices” ($H_1 \mid H_2$) of ℓ rows. All other improvements lead to an operation count which is close to what we have in (4). The following formula, obtained with the techniques of the previous section, gives a tight lower bound all those variants.

$$\text{WF}_{\text{Stern}}(n, r, w) \approx \min_p \frac{2\ell \binom{n}{w}}{\binom{r-\ell}{w-p} \binom{k/2}{p/2}} \text{ with } \ell = \log \left(K_{w-p} \binom{k/2}{p/2} \right).$$

The gain of the new version of ISD is $\approx \lambda^4 \sqrt{\pi p/2}$ which is rather small in practice and correspond to the improvement of the “birthday paradox” part of the algorithm.

4 Generalized Birthday Algorithm (GBA)

4.1 General Principle

The generalized birthday technique is particularly efficient for solving Syndrome Decoding-like problems with a large number of solutions. Suppose one has to solve the following problem:

Problem 2. Given a function $f : \mathbb{N} \mapsto \{0, 1\}^r$ and an integer a , find a set of 2^a indexes x_i such that:

$$\bigoplus_{i=0}^{2^a-1} f(x_i) = 0.$$

In this problem, f will typically return the x_i -th column of a binary matrix H . Note that, here, f is defined upon an infinite set, meaning that there are an infinity of solutions. To solve this problem, the Generalized Birthday Algorithm (GBA) does the following:

- build 2^a lists L_0, \dots, L_{2^a-1} , each containing $2^{\frac{r}{a+1}}$ different vectors $f(x_i)$
- pairwise merge lists L_{2j} and L_{2j+1} to obtain 2^{a-1} lists L'_j of XORs of 2 vectors $f(x_i)$. Only keep XORs of 2 vectors starting with $\frac{r}{a+1}$ zeros. On average, the lists L'_j will contain $2^{\frac{r}{a+1}}$ elements.
- pairwise merge the new lists L'_{2j} and L'_{2j+1} to obtain 2^{a-2} lists L''_j of XORs of 4 vectors $f(x_i)$. Only keep XORs of 4 vectors starting with $2\frac{r}{a+1}$ zeros. On average, the lists L''_j will still contain $2^{\frac{r}{a+1}}$ elements.
- continue these merges until only 2 lists remain. These 2 lists will be composed of $2^{\frac{r}{a+1}}$ XORs of 2^{a-1} vectors $f(x_i)$ starting with $(a-1)\frac{r}{a+1}$ zeros.
- as only $2\frac{r}{a+1}$ bits of the previous vectors are non-zero, a simple application of the standard birthday technique is enough to obtain 1 solution (on average).

As all the lists manipulated in this algorithm are of the same size, the complexity of the algorithm is easy to compute: $2^a - 1$ merge operations have to be performed, each of them requiring to sort a list of size $2^{\frac{r}{a+1}}$. The complexity is thus $O(2^a \frac{r}{a} 2^{\frac{r}{a+1}})$. For simplicity we will only consider a lower bound of the effective complexity of the algorithm: if we denote by L the size of the largest list in the algorithm, the complexity is lower-bounded by $O(L \log L)$. this gives a complexity of $O(\frac{r}{a+1} 2^{\frac{r}{a+1}})$.

Minimal Memory Requirements. The minimal memory requirements for this algorithm are not as easy to compute. If all the lists are chosen to be of the same size (as in the description of the algorithm we give), then it is possible to compute the solution by storing at most a lists at a time in memory. This gives us a memory complexity of $O(a 2^{\frac{r}{a+1}})$. However, the starting lists can also be chosen of different sizes so as to store only smaller lists.

In practice, for each merge operation, only one of the two lists has to be stored in memory, the second one can always be computed on the fly. As a consequence, looking at the tree of all merge operations (see Fig. [1](#)), half the lists of the tree can be computed on the fly (the lists in dashed line circles). Let $L = 2^{\frac{r}{a+1}}$ and suppose one wants to use the Generalized Birthday Algorithm storing only lists of size $\frac{L}{\lambda}$ for a given λ . Then, in order to get, on average, a single solution in the end, the lists computed on the fly should be larger. For instance, in the example of Fig. [1](#) one should have:

- $|L''_1| = \lambda L$, $|L'_3| = \lambda^2 L$, and $|L_7| = \lambda^3 L$,
- $|L'_1| = L$ and $|L_3| = \lambda L$,
- $|L_1| = L$ and $|L_5| = L$.

In the general case this gives us a time/memory tradeoff when using GBA: one can divide the memory complexity by λ at the cost of an increase in time complexity by a factor λ^a . However, many other combinations are also possible depending on the particular problem one has to deal with.

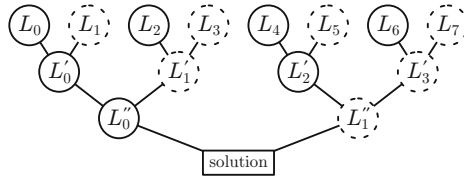


Fig. 1. Merge operations in the Generalized Birthday Algorithm. All lists in dashed line circles can be computed on the fly.

4.2 GBA under Constraints

In the previous section, we presented a version of GBA where the number of vectors available was unbounded and where the number of vectors to XOR was a power of 2. In practice, when using GBA to solve instances of the CSD problem only n different r -bit vectors are available and w can be any number. We thus consider an idealized version of GBA so as to bound the complexity of “real world” GBA. The bounds we give are not always very tight. See for instance [7] for the analysis of a running implementation of GBA under realistic constraints.

If w is not a power of 2, some of the starting lists L_j should contain vectors $f(x_i)$ and others XORs of 2 or more vectors $f(x_i)$. We consider that the starting lists all contain XORs of $\frac{w}{2^a}$ vectors $f(x_i)$, even if this is not an integer. This will give the most time efficient algorithm, but will of course not be usable in practice.

The length of the matrix n limits the size of the starting lists. For GBA to find one solution on average, one needs lists L_j of size $2^{\frac{r}{a+1}}$. As the starting lists contain XORs of $\frac{w}{2^a}$ vectors, we need $\binom{n}{\frac{w}{2^a}} \geq 2^{\frac{r}{a+1}}$. However, this constraint on a is not sufficient: if all the starting lists contain the same vectors, all XORs will be found many times and the probability of success will drop. To avoid this, we need lists containing different vectors and this can be done by isolating the first level of merges.

- first we select 2^{a-1} distinct vectors s_j of a bits such that $\bigoplus s_j = 0$.
- then we pairwise merge lists L_{2j} and L_{2j+1} to obtain lists L'_j containing elements having their a first bits equal to s_j .

After this first round, we have 2^{a-1} lists of XORs of $\frac{2w}{2^a}$ vectors such that, if we XOR the a first bits of one element from each list we obtain 0. Also, all the lists contain only distinct elements, which means we are back in the general case of GBA, except we now have 2^{a-1} lists of vectors of length $r - a$. These lists all have a maximum size $L = \frac{1}{2^a} \binom{n}{\frac{2w}{2^a}}$ and can be obtained from starting lists L_j of size $\sqrt{\binom{n}{\frac{2w}{2^a}}}$ (see Sect. 2). We get the following constraint on a :

$$\frac{1}{2^a} \binom{n}{\frac{2w}{2^a}} \geq 2^{\frac{r-a}{a}}. \quad (6)$$

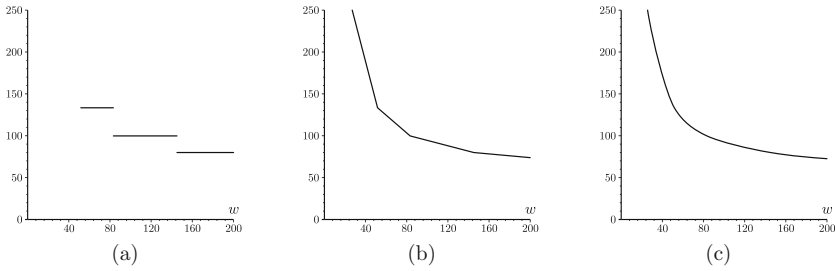


Fig. 2. Logarithm of the complexity of the Generalized Birthday Algorithm for given n and r when w varies. (a) with no optimization, (b) when the lists are initialized with shortened vectors, and (c) when a is not an integer.

In practice, after the first level of merges we are not exactly in the general case of GBA: if, for example, $s_0 \oplus s_1 = s_2 \oplus s_3$, after the second merges, lists L''_0 and L''_1 would contain exactly the same elements. This can be avoided by using another set of target values s'_j such that $\bigoplus s'_j = 0$ for the second level of merges (as for the first level) and so on for the subsequent levels of merges (except the last two levels).

Using Non-Integer Values for a . Equation (6) determines the largest possible value of a that can be used with GBA. For given n and r , if w varies, the complexity of the algorithm will thus have a stair-like shape (see Fig. 2(a)). The left-most point of each step corresponds to the case where Equation (6) is an equality. However, when it is not an equality, it is possible to gain a little: instead of choosing values s_j of a bits one can use slightly larger values and thus start the second level of merge with shorter vectors. This gives a broken-line complexity curve (see Fig. 2(b)). This is somehow similar to what Minder and Sinclair denote by “extended k -tree algorithm” [19]. In practice, this is almost equivalent to using non-integer values for a (see Fig. 2(c)). We will thus assume that in GBA, a is a real number, chosen such that Equation (6) is an equality.

Proposition 3. *We can lower bound the binary work factor $WF_{\text{GBA}}(n, r, w)$ of GBA applied to solving an instance of CSD with parameters (n, r, w) by:*

$$WF_{\text{GBA}}(n, r, w) \geq \frac{r - a}{a} 2^{\frac{r-a}{a}}, \text{ with } a \text{ such that } \frac{1}{2^a} \binom{n}{\frac{2w}{2^a}} = 2^{\frac{r-a}{a}}.$$

Note that this gives us a bound on the minimal time complexity of GBA but does not give any bound on the memory complexity of the algorithm. Also, this bound is computed using an idealized version of the algorithm: one should not expect to achieve such a complexity in practice, except in some cases where a is an integer and w a power of 2.

5 Case Studies

Now that we have given some bounds on the complexities of the best algorithms to solve CSD problems, we propose to study what happens when using them to attack existing constructions.

Note that in this section, as in the whole paper, we only consider the resistance to decoding attacks. Code-based cryptosystems may also be vulnerable to structural attacks. However, no efficient structural attack is known for binary Goppa codes (McEliece encryption and CFS signature) or for prime order random quasi-cyclic codes (FSB hash function).

5.1 Attacking the McEliece Cryptosystem

In the McEliece [17] and Niederreiter [21] cryptosystems the security relies on two different problems: recovering the private key from the public key and decrypting an encrypted message. Decrypting consists in finding an error pattern e of weight w , such that $e \times H^T = c$ where H is a binary matrix derived from the public key and c is a syndrome derived from the encrypted message one wants to decrypt. Here, we suppose that the structural attack consisting in recovering the private key is infeasible and can assume that H is a random binary matrix. Decryption thus consists in solving an instance of the CSD problem where one knows that one and only one solution exists.

Having a single solution rules out any attempt to use GBA, or at least, any attempt to use GBA would consist in using the classical birthday attack. For this reason the best attacks against the McEliece and Niederreiter cryptosystems are all based on ISD. Table 3 gives the work factors we obtain using our bound from Sect. 3. For the classical McEliece parameters (10, 50) this bound can be compared to the work factors computed by non-idealized algorithms. Canteaut and Chabaud [10] obtained a work factor of $2^{64.2}$ and Bernstein, Lange and Peters [6] a work factor of $2^{60.5}$. As one can see, the gap between our bound and their complexities is very small indicating two things:

- our bound on ISD is tight when evaluating the practical security of some McEliece parameters,
- the best ISD-based algorithms are sufficiently advanced to make our assumption that Gaussian elimination is free almost realistic. Almost no margin is left for these techniques to improve and better attacks will need to introduce new methods.

5.2 Attacking the CFS Signature Scheme

The attack we present here is due to Daniel Bleichenbacher, but was never published. We present what he explained through private communication including a few additional details.

The CFS signature scheme [13] is based on the Niederreiter cryptosystem: signing a document requires to hash it into a syndrome and then try to decode

Table 3. Work factors for the ISD lower-bound we computed for some typical McEliece/Niederreiter parameters. The code has length $n = 2^m$ and codimension $r = mw$ and corrects w errors.

(m, w)	optimal p	optimal ℓ	binary work factor
(10, 50)	4	22	$2^{59.9}$
(11, 32)	6	33	$2^{86.8}$
(12, 41)	10	54	$2^{128.5}$

this syndrome. However, for a Goppa code correcting w errors, only a fraction $\frac{1}{w!}$ of the syndromes are decodable. Thus, a counter is appended to the message and the signer tries successive counter values until one hash is decodable. The signature consists of both the error pattern of weight w corresponding to the syndrome and the value of the counter giving this syndrome.

Attacking this construction consists in forging a valid signature for a chosen message. One must find a matching counter and error pattern for a given document. This looks a lot like a standard CSD problem instance. However, here there is one major difference with the case of McEliece or Niederreiter: instead of having one instance to solve, one now needs to solve one instance among many instances. One chooses a document and hashes it with many different counters to obtain many syndromes: each syndrome corresponds to a different instance. It has no importance which instance is solved, each of them can give a valid “forged” signature.

For ISD algorithms, having multiple instances available is of little help, however, for GBA, this gives us one additional list. Even though Goppa code parameters are used and an instance has less than a solution on average, this additional list makes the application of GBA with $a = 2$ possible. This will always be an “unbalanced” GBA working as follows:

- first, build 3 lists L_0 , L_1 , and L_2 of XORs of respectively w_0 , w_1 and w_2 columns of H (with $w = w_0 + w_1 + w_2$). These lists can have a size up to $\binom{n}{w_i}$ but smaller sizes can be used,
- merge the two lists L_0 and L_1 into a list L'_0 of XORs of $w_0 + w_1$ columns of H , keeping only those starting with λ zeros (we will determine the optimal choice for λ later). L'_0 contains $\frac{1}{2^\lambda} \binom{n}{w_0+w_1}$ elements on average.
- All the following computations are done on the fly and additional lists do not have to be stored. Repeat the following steps:
 - choose a counter and compute the corresponding document hash (an element of the virtual list L_3),
 - XOR this hash with all elements of L_2 matching on the first λ bits (to obtain elements of the virtual list L'_1),
 - look up each of these XORs in L'_0 : any complete match gives a valid signature.

The number L of hashes one will have to compute on average is such that:

$$\frac{1}{2^\lambda} \binom{n}{w_0 + w_1} \times \frac{L}{2^\lambda} \binom{n}{w_2} = 2^{r-\lambda} \Leftrightarrow L = \frac{2^{r+\lambda}}{\binom{n}{w_0+w_1} \binom{n}{w_2}}.$$

The memory requirements for this algorithm correspond to the size of the largest list stored. In practice, the first level lists L_i can be chosen so that L'_0 is always the largest, and the memory complexity is $\frac{1}{2^\lambda} \binom{n}{w_0+w_1}$. The time complexity corresponds to the size of the largest list manipulated: $\max(\frac{1}{2^\lambda} \binom{n}{w_0+w_1}, L, \frac{L}{2^\lambda} \binom{n}{w_2})$. The optimal choice is always to choose $w_0 = \lceil \frac{w}{3} \rceil$, $w_2 = \lfloor \frac{w}{3} \rfloor$, and $w_1 = w - w_0 - w_2$. Then, two different cases can occur: either L'_1 is the largest list, or one of L'_0 and L_3 is. If L'_1 is the largest, we choose λ so as to have a smaller list L'_0 and so a smaller memory complexity. Otherwise, we choose λ so that L'_0 and L_3 are of the same size to optimize the time complexity. Let \mathcal{T} be the size of the largest list we manipulate and \mathcal{M} the size of the largest list we store. The algorithm has time complexity $O(\mathcal{T} \log \mathcal{T})$ and memory complexity $O(\mathcal{M} \log \mathcal{M})$ with:

$$\left\{ \begin{array}{l} \text{if } \frac{2^r}{\binom{n}{w-\lfloor w/3 \rfloor}} \geq \sqrt{\frac{2^r}{\binom{n}{\lfloor w/3 \rfloor}}} \text{ then } \mathcal{T} = \frac{2^r}{\binom{n}{w-\lfloor w/3 \rfloor}} \text{ and } \mathcal{M} = \frac{\binom{n}{w-\lfloor w/3 \rfloor}}{\binom{n}{\lfloor w/3 \rfloor}}, \\ \text{else} \quad \mathcal{T} = \mathcal{M} = \sqrt{\frac{2^r}{\binom{n}{\lfloor w/3 \rfloor}}}. \end{array} \right.$$

This algorithm is realistic in the sense that only integer values are used, meaning that effective attacks should have time/memory complexities close to those we present in Table 4. Of course, for a real attack, other time/memory tradeoffs might be more advantageous, resulting in other choices for λ and the w_i .

5.3 Attacking the FSB Hash Function

FSB [1] is a candidate for the SHA-3 hash competition. The compression function of this hash function consists in converting the input into a low weight word and then multiplying it by a binary matrix H . This is exactly a syndrome computation and inverting this compression function requires to solve an instance

Table 4. Time/memory complexities of Bleichenbacher's attack against the CFS signature scheme. The parameters are Goppa code parameters so $r = mw$ and $n = 2^m$.

	$w = 8$	$w = 9$	$w = 10$	$w = 11$	$w = 12$
$m = 15$	$2^{51.0} / 2^{51.0}$	$2^{60.2} / 2^{43.3}$	$2^{63.1} / 2^{55.9}$	$2^{67.2} / 2^{67.2}$	$2^{81.5} / 2^{54.9}$
$m = 16$	$2^{54.1} / 2^{54.1}$	$2^{63.3} / 2^{46.5}$	$2^{66.2} / 2^{60.0}$	$2^{71.3} / 2^{71.3}$	$2^{85.6} / 2^{59.0}$
$m = 17$	$2^{57.2} / 2^{57.2}$	$2^{66.4} / 2^{49.6}$	$2^{69.3} / 2^{64.2}$	$2^{75.4} / 2^{75.4}$	$2^{89.7} / 2^{63.1}$
$m = 18$	$2^{60.3} / 2^{60.3}$	$2^{69.5} / 2^{52.7}$	$2^{72.4} / 2^{68.2}$	$2^{79.5} / 2^{79.5}$	$2^{93.7} / 2^{67.2}$
$m = 19$	$2^{63.3} / 2^{63.3}$	$2^{72.5} / 2^{55.7}$	$2^{75.4} / 2^{72.3}$	$2^{83.6} / 2^{83.6}$	$2^{97.8} / 2^{71.3}$
$m = 20$	$2^{66.4} / 2^{66.4}$	$2^{75.6} / 2^{58.8}$	$2^{78.5} / 2^{76.4}$	$2^{87.6} / 2^{87.6}$	$2^{101.9} / 2^{75.4}$
$m = 21$	$2^{69.5} / 2^{69.5}$	$2^{78.7} / 2^{61.9}$	$2^{81.5} / 2^{80.5}$	$2^{91.7} / 2^{91.7}$	$2^{105.9} / 2^{79.5}$
$m = 22$	$2^{72.6} / 2^{72.6}$	$2^{81.7} / 2^{65.0}$	$2^{84.6} / 2^{84.6}$	$2^{95.8} / 2^{95.8}$	$2^{110.0} / 2^{83.6}$

Table 5. Complexities of the ISD and GBA bounds we propose for the official FSB parameters

	n	r	w	inversion		collision	
				ISD	GBA	ISD	GBA
FSB ₁₆₀	5×2^{18}	640	80	$2^{211.1}$	$2^{156.6}$	$2^{100.3}$	$2^{118.7}$
FSB ₂₂₄	7×2^{18}	896	112	$2^{292.0}$	$2^{216.0}$	$2^{135.3}$	$2^{163.4}$
FSB ₂₅₆	2^{21}	1 024	128	$2^{330.8}$	$2^{245.6}$	$2^{153.0}$	$2^{185.7}$
FSB ₃₈₄	23×2^{16}	1 472	184	$2^{476.7}$	$2^{360.2}$	$2^{215.5}$	$2^{268.8}$
FSB ₅₁₂	31×2^{16}	1 984	248	$2^{687.8}$	$2^{482.1}$	$2^{285.6}$	$2^{359.3}$

of the CSD problem. Similarly, finding a collision on the compression function requires to find two low weight words having the same syndrome, that is, a word of twice the Hamming weight with a null syndrome. In both cases, the security of the compression function (and thus of the whole hash function) can be reduced to the hardness of solving some instances of the CSD problem. For inversion (or second preimage), the instances are of the form $\text{CSD}(H, w, s)$ and, for collision, of the form $\text{CSD}(H, 2w, 0)$.

Compared to the other code-based cryptosystems we presented, here, the number of solutions to these instances is always very large: we are studying a compression function, so there are a lot of collisions, and each syndrome has a lot of inverses. For this reason, both ISD and GBA based attacks can be used. Which of the two approaches is the most efficient depends on the parameters. However, for the parameters proposed in [1], ISD is always the best choice for collision search and GBA the best choice for inversion (or second preimage). Table 5 contains the attack complexities given by our bounds for the proposed FSB parameters. As you can see, the complexities obtained with GBA for inversion are lower than the standard security claim. Unfortunately this does not give an attack on FSB for many reasons: the version of GBA we consider is idealized and using non-integer values of a is not practical, but most importantly, the input of the compression of FSB is not any word of weight w , but only regular words, meaning that the starting lists for GBA will be much smaller in practice, yielding a smaller a and higher complexities.

Conclusion

In this article we have reviewed the two main families of algorithms for solving instances of the CSD problem. For each of these we have discussed possible tweaks and described idealized versions of the algorithms covering those tweaks. The work factors we computed for these idealized versions are lower bounds on the effective work factor of existing real algorithms, but also on the future improvements that could be implemented. Solving CSD more efficiently than these bounds would require to introduce new techniques, never applied to code-based cryptosystems.

For these reasons, the bounds we give can be seen as a tool one can use to select parameters for code-based cryptosystems. We hope they can help other designers choose durable parameters with more ease.

References

1. Augot, D., Finiasz, M., Gaborit, Ph., Manuel, S., Sendrier, N.: SHA-3 proposal: FSB. Submission to the SHA-3 NIST competition (2008)
2. Augot, D., Finiasz, M., Sendrier, N.: A family of fast syndrome based cryptographic hash function. In: Dawson, E., Vaudenay, S. (eds.) *Mycrypt 2005*. LNCS, vol. 3715, pp. 64–83. Springer, Heidelberg (2005)
3. Berger, T., Cayrel, P.-L., Gaborit, P., Otmani, A.: Reducing key length of the mceliece cryptosystem. In: Preneel, B. (ed.) *AFRICACRYPT 2009*. LNCS, vol. 5580, pp. 60–76. Springer, Heidelberg (to appear, 2009)
4. Berlekamp, E.R., McEliece, R.J., van Tilborg, H.C.: On the inherent intractability of certain coding problems. *IEEE Transactions on Information Theory* 24(3) (1978)
5. Bernstein, D., Buchmann, J., Ding, J. (eds.): *Post-Quantum Cryptography*. Springer, Heidelberg (2008)
6. Bernstein, D., Lange, T., Peters, C.: Attacking and defending the McEliece cryptosystem. In: Buchmann, J., Ding, J. (eds.) *PQCrypto 2008*. LNCS, vol. 5299, pp. 31–46. Springer, Heidelberg (2008)
7. Bernstein, D.J., Lange, T., Peters, C., Niederhagen, R., Schwabe, P.: Implementing wagner’s generalized birthday attack against the sha-3 candidate fsb. *Cryptology ePrint Archive*, Report 2009/292 (2009), <http://eprint.iacr.org/>
8. Bernstein, D.J., Lange, T., Peters, C., van Tilborg, H.: Explicit bounds for generic decoding algorithms for code-based cryptography. In: *Pre-proceedings of WCC 2009*, pp. 168–180 (2009)
9. Camion, P., Patarin, J.: The knapsack hash function proposed at crypto 1989 can be broken. In: Davies, D.W. (ed.) *EUROCRYPT 1991*. LNCS, vol. 547, pp. 39–53. Springer, Heidelberg (1991)
10. Canteaut, A., Chabaud, F.: A new algorithm for finding minimum-weight words in a linear code: Application to McEliece’s cryptosystem and to narrow-sense BCH codes of length 511. *IEEE Transactions on Information Theory* 44(1), 367–378 (1998)
11. 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)
12. Coron, J.-S., Joux, A.: Cryptanalysis of a provably secure cryptographic hash function. *Cryptology ePrint Archive* (2004), <http://eprint.iacr.org/2004/013/>
13. Courtois, N., Finiasz, M., Sendrier, N.: How to achieve a McEliece-based digital signature scheme. In: Boyd, C. (ed.) *ASIACRYPT 2001*. LNCS, vol. 2248, pp. 157–174. Springer, Heidelberg (2001)
14. Finiasz, M., Sendrier, N.: Security bounds for the design of code-based cryptosystems. *Cryptology ePrint Archive*, Report 2009/414 (2009), <http://eprint.iacr.org/>
15. Lee, P.J., Brickell, E.F.: An observation on the security of McEliece’s public-key cryptosystem. In: Günther, C.G. (ed.) *EUROCRYPT 1988*. LNCS, vol. 330, pp. 275–280. Springer, Heidelberg (1988)

16. Leon, J.S.: A probabilistic algorithm for computing minimum weights of large error-correcting codes. *IEEE Transactions on Information Theory* 34(5), 1354–1359 (1988)
17. McEliece, R.J.: A public-key cryptosystem based on algebraic coding theory. In: *DSN Prog. Rep., Jet Prop. Lab., California Inst. Technol., Pasadena, CA*, pp. 114–116 (January 1978)
18. Aguilar Melchor, C., Cayrel, P.-L., Gaborit, P.: A new efficient threshold ring signature scheme based on coding theory. In: Buchmann, J., Ding, J. (eds.) *PQCrypto 2008*. LNCS, vol. 5299, pp. 1–16. Springer, Heidelberg (2008)
19. Minder, L., Sinclair, A.: The extended k -tree algorithm. In: Mathieu, C. (ed.) *Proceedings of SODA 2009*, pp. 586–595. SIAM, Philadelphia (2009)
20. Misoczki, R., Barreto, P.S.L.M.: Compact McEliece keys from Goppa codes. *Cryptography ePrint Archive*, Report 2009/187 (2009), <http://eprint.iacr.org/>
21. Niederreiter, H.: Knapsack-type cryptosystems and algebraic coding theory. *Prob. Contr. Inform. Theory* 15(2), 157–166 (1986)
22. Stern, J.: A method for finding codewords of small weight. In: Wolfmann, J., Cohen, G. (eds.) *Coding Theory 1988*. LNCS, vol. 388, pp. 106–113. Springer, Heidelberg (1989)
23. Stern, J.: A new identification scheme based on syndrome decoding. In: Stinson, D.R. (ed.) *CRYPTO 1993*. LNCS, vol. 773, pp. 13–21. Springer, Heidelberg (1994)
24. Véron, P.: A fast identification scheme. In: *IEEE Conference, ISIT 1995*, Whistler, BC, Canada, September 1995, p. 359 (1995)
25. Wagner, D.: A generalized birthday problem. In: Yung, M. (ed.) *CRYPTO 2002*. LNCS, vol. 2442, pp. 288–303. Springer, Heidelberg (2002)

A Comments on the Assumptions

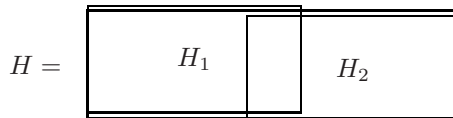
We have assumed the following in Sect. [2](#):

- (B1) For all pairs (e_1, e_2) examined in the algorithm, the sums $e_1 + e_2$ are uniformly and independently distributed in $W_{n,w}$.
- (B2) The cost of the execution of the algorithm is approximatively equal to

$$\ell \cdot \#(\text{BA 1}) + \ell \cdot \#(\text{BA 2}) + K_0 \cdot \#(\text{BA 3}),$$

where K_0 is the cost for testing $e_1 H^T = s + e_2 H^T$ given that $h_\ell(e_1 H^T) = h_\ell(s + e_2 H^T)$ and $\#(\text{BA } i)$ is the expected number of execution of the instruction (BA i) before we meet the (SUCCESS) condition.

The first assumption has to do with the way the attacker chooses the sets W_1 and W_2 . In the version presented at the beginning of Sect. [2](#), they use different sets of columns and thus all pairs (e_1, e_2) lead to different words $e = e_1 + e_2$. When W_1 and W_2 increase, there is some waste, that is some words $e = e_1 + e_2$ are obtained several times. A clever choice of W_1 and W_2 may decrease this waste, but this seems exceedingly difficult. The “overlapping” approach



is easy to implement and behaves (almost) as if W_1 and W_2 were random (it is even sometimes slightly better). The second assumption counts only ℓ binary operations to perform the sum of $w/2$ columns of ℓ bits. This can be achieved by a proper scheduling of the loops and by keeping partial sums. This was described and implemented in [6]. We also neglect the cost of control and memory handling instructions. This is certainly optimistic but on modern processors most of those costs can be hidden in practice. The present work is meant to give security levels rather than a cryptanalysis costs. So we want our estimates to be implementation independent as much as possible.

Similar comments apply to the assumptions (I1) and (I2) of Sect. 3.

B A Sketch of the Proof of Proposition 2

We provide here some clues for the proof of Proposition 2. More details on this proof and on the proofs of the other results of this paper can be found in the extended version [14].

Proof. (of Proposition 2 - Sketch) In one execution of (MAIN LOOP) we examine $\lambda(z) \binom{k+\ell}{p}$ distinct value of $e_1 + e_2$, where $z = |W_1||W_2|/\binom{k+\ell}{p}$ and $\lambda(z) = 1 - \exp(-z)$. The probability for one particular element of $W_{k+\ell,p}$ to lead to a solution is

$$P = \frac{\binom{r-\ell}{w-p}}{\min\left(\binom{n}{w}, 2^r\right)}.$$

Thus the probability for one execution of (MAIN LOOP) to lead to (SUCCESS) is

$$P_p(\ell) = 1 - (1 - P)^{\lambda(z) \binom{k+\ell}{p}} \approx 1 - \exp\left(-\frac{\lambda(z)}{N_p(\ell)}\right) \quad \text{where } N_p(\ell) = \frac{\min\left(\binom{n}{w}, 2^r\right)}{\binom{r-\ell}{w-p} \binom{k+\ell}{p}}$$

When $N_p(\ell)$ is large (much larger than 1), we have $P_p(\ell) \approx \lambda(z)/N_p(\ell)$ and a good estimate for the cost is

$$\frac{N_p(\ell)}{\lambda(z)} \left(\ell|W_1| + \ell|W_2| + K_{w-p} \frac{\lambda(z) \binom{k+\ell}{p}}{2^\ell} \right).$$

Choosing $|W_1|$, $|W_2|$, ℓ and z which minimize this formula leads to the first formula of the statement.

Else we have $N_p(\ell) < 1$ and the expected number of execution of (MAIN LOOP) is not much higher than one (obviously it cannot be less). In that case we are in a situation very similar to a birthday attack in which the list size is $L = \sqrt{1/P} = 2^{r/2}/\sqrt{\binom{r-\ell}{w-p}}$. This gives a cost of $2L \log(K_{w-p}L)$ which has to be minimized in ℓ , leading to the second formula of the statement.

Rebound Attack on the Full LANE Compression Function*

Krystian Matusiewicz¹, María Naya-Plasencia², Ivica Nikolić³,
Yu Sasaki⁴, and Martin Schläffer⁵

¹ Department of Mathematics, Technical University of Denmark, Denmark

`k.matusiewicz@mat.dtu.dk`

² INRIA project-team SECRET, France

`maria.naya.plasencia@inria.fr`

³ University of Luxembourg, Luxembourg

`ivica.nikolic@uni.lu`

⁴ NTT Corporation, Japan

`sasaki.yu@lab.ntt.co.jp`

⁵ IAIK, Graz University of Technology, Austria

`martin.schlaeffer@iaik.tugraz.at`

Abstract. In this work, we apply the rebound attack to the AES based SHA-3 candidate LANE. The hash function LANE uses a permutation based compression function, consisting of a linear message expansion and 6 parallel lanes. In the rebound attack on LANE, we apply several new techniques to construct a collision for the full compression function of LANE-256 and LANE-512. Using a relatively sparse truncated differential path, we are able to solve for a valid message expansion and colliding lanes independently. Additionally, we are able to apply the inbound phase more than once by exploiting the degrees of freedom in the parallel AES states. This allows us to construct semi-free-start collisions for full LANE-256 with 2^{96} compression function evaluations and 2^{88} memory, and for full LANE-512 with 2^{224} compression function evaluations and 2^{128} memory.

Keywords: SHA-3, LANE, hash function, cryptanalysis, rebound attack, semi-free-start collision.

1 Introduction

In the last few years the cryptanalysis of hash functions has become an important topic within the cryptographic community. The attacks on the MD4 family of hash functions (MD5, SHA-1) have especially weakened the confidence in the security of this design strategy [13, 14]. Many new and interesting hash function designs have been proposed as part of the NIST SHA-3 competition [11]. The

* This work was supported in part by the European Commission through the ICT programme under contract ICT-2007-216676 ECRYPT II. The authors would like to thank Janusz Szmidt and Florian Mendel for useful discussions.

large number of submissions and different design strategies require different and improved cryptanalytic techniques as well.

At FSE 2009, Mendel *et al.* published the rebound attack [9] - a new technique for analysis of hash functions which has been applied first to reduced versions of the Whirlpool [2] and Grøstl [4] compression functions. Recently, the rebound attack on Whirlpool has been extended in [8], which in some parts is similar to our attack. The main idea of the rebound attack is to use the available degrees of freedom in the internal state to efficiently fulfill the low probability parts in the middle of a differential trail. The straight-forward application of the rebound attack to AES based constructions allows a quick and thorough analysis of these hash functions.

In this work, we improve the rebound attack and apply it to the SHA-3 candidate LANE. The hash function LANE [5] uses an iterative construction based on the Merkle-Damgård design principle [3,10] and has been first analyzed in [15]. The permutation based compression function consists of a linear message expansion and 6 parallel lanes. The permutations of each lane are based on the round transformations of the AES. In the rebound attack on LANE, we first search for differences and values, according to a specific truncated differential path. This truncated differential path is constructed such that a collision and a valid expanded message can be found with a relatively high probability. By using the degrees of freedom in the chaining values, we are able to construct a semi-free-start collision for the full versions of LANE-256 with 2^{96} compression function evaluations and memory of 2^{88} , and for LANE-512 with 2^{224} compression function evaluations and memory of 2^{128} . Although these collisions on the compression function do not imply an attack on the hash functions, they violate the reduction proofs of Merkle and Damgård, and Andreeva [1].

2 Description of LANE

The cryptographic hash function LANE [5] is one of the submissions to the NIST SHA-3 competition [11]. It is an iterated hash function that supports four digest sizes (224, 256, 384 and 512 bits) and the use of a salt. Since LANE-224 and LANE-256 are rather similar except for truncation, we write LANE-256 whenever we refer to both of them. The same holds for LANE-384 and LANE-512.

The hashing of a message proceeds as follows. First, the initial chaining value H_{-1} , of size 256 bits for LANE-256, and 512 bits for LANE-512, is set to an initial value that depends on the digest size n and the optional salt value S . At the same time, the message is padded and split into message blocks M_i of length 512 bits for LANE-256, and 1024 bits for LANE-512. Then, a compression function f is applied iteratively to process message blocks one by one as $H_i = f(H_{i-1}, M_i, C_i)$, where C_i is a counter that indicates the number of message bits processed so far. Finally, after all the message blocks are processed, the final digest is derived from the last chaining value, the message length and the salt by an additional call to the compression function.

2.1 The Compression Function

The compression function of LANE-256 transforms 256 bits (512 in the case of LANE-512) of the chaining value and 512 bits (resp. 1024 bits) of the message block into a new chaining value of 256 bits (512 bits). It uses a 64-bit counter value C_i . For the detailed structure of the compression function we refer to the specification of LANE [5]. First, the chaining value and the message block are processed by a message expansion that produces an expanded state with doubled size. Then, this expanded state is processed in two layers. The first layer is composed of six permutation lanes P_0, \dots, P_5 in parallel, and the second layer of two parallel lanes Q_0, Q_1 .

2.2 The Message Expansion

The message expansion of LANE takes a message block M_i and a chaining value H_{i-1} and produces the input to six permutations P_0, \dots, P_5 . In LANE-256, the 512-bit message block M_i is split into four 128-bit blocks m_0, m_1, m_2, m_3 and the 256-bit chaining value H_{i-1} is split into two 128-bit words h_0, h_1 as follows $m_0 || m_1 || m_2 || m_3 \leftarrow M_i, h_0 || h_1 \leftarrow H_{i-1}$. Then, six more 128-bit words $a_0, a_1, b_0, b_1, c_0, c_1$ are computed

$$\begin{aligned} a_0 &= h_0 \oplus m_0 \oplus m_1 \oplus m_2 \oplus m_3, & a_1 &= h_1 \oplus m_0 \oplus m_2, \\ b_0 &= h_0 \oplus h_1 \oplus m_0 \oplus m_2 \oplus m_3, & b_1 &= h_0 \oplus m_1 \oplus m_2, \\ c_0 &= h_0 \oplus h_1 \oplus m_0 \oplus m_1 \oplus m_2, & c_1 &= h_0 \oplus m_0 \oplus m_3. \end{aligned} \quad (1)$$

Each of these 128-bit values, as in AES, can be seen as 4×4 matrix of bytes. In the following, we will use the notion $x[i, j]$ when we refer to the byte of the matrix x with row index i and column index j , starting from 0.

The values $a_0 || a_1, b_0 || b_1, c_0 || c_1, h_0 || h_1, m_0 || m_1, m_2 || m_3$ become inputs to the six permutations P_0, \dots, P_5 described below. The message expansion for larger variants of LANE is identical but all the values are doubled in size.

2.3 The Permutations

Each permutation lane P_i operates on a state that can be seen as a double AES state (2×128 -bits) in the case of LANE-256 or quadruple AES state (4×128 -bits) for LANE-512. The permutation reuses the transformations **SubBytes** (SB), **ShiftRows** (SR) and **MixColumns** (MC) of the AES with the only exception, that due to the larger state size, they are applied twice or four times in parallel.

Additionally, there are three new round transformations introduced in LANE. **AddConstant** adds a different value to each column of the lane state and **AddCounter** adds part of the counter C_i to the state. Since our attacks do not depend on these functions, we skip their details here. The third transformation is **SwapColumns** (SC) - used for mixing parallel AES states. Let x_i be a column of a lane state. In LANE-256, **SwapColumns** swaps the two right columns of the left half-state with the

two left columns of the right half-state, and in LANE-512, `SwapColumns` ensures that each column of an AES state gets swapped to a different AES state:

$$\begin{aligned} SC_{256}(x_0||x_1||\dots||x_7) &= x_0||x_1||x_4||x_5||x_2||x_3||x_6||x_7 \\ SC_{512}(x_0||x_1||\dots||x_{15}) &= x_0||x_4||x_8||x_{12}||x_1||x_5||x_9||x_{13}|| \\ &\quad x_2||x_6||x_{10}||x_{14}||x_3||x_7||x_{11}||x_{15} . \end{aligned}$$

The complete round transformation consists of the sequential application of all these transformations in the given order. The last round omits `AddConstant` and `AddCounter`. Each of the permutations P_j consists of six rounds in the case of LANE-256 and eight rounds for LANE-512.

The permutations Q_0 and Q_1 are irrelevant to our attack because we will get collisions before these permutations. An interested reader can find a detailed description of Q_0 and Q_1 in [5].

3 The Rebound Attack on LANE

In this section first we give a short overview of the rebound attack in general and then, describe the different phases of the rebound attack on LANE in detail.

3.1 The Rebound Attack

The rebound attack was published by Mendel *et al.* in [9] and is a new tool for the cryptanalysis of hash functions. The rebound attack uses truncated differences [6] and is related to the attack by Peyrin [12] on the hash function Grindahl [7]. The main idea of the rebound attack is to use the available degrees of freedom in the internal state to fulfill the low probability parts in the middle of a differential path. It consists of an inbound and subsequent outbound phase. The inbound phase is an efficient meet-in-the-middle phase, which exploits the available degrees of freedom in the middle of a differential path. In the mostly probabilistic outbound phase, the matches of the inbound phase are computed backwards and forwards to obtain an attack on the hash or compression function. Usually, the inbound phase is repeated many times to generate enough starting points for the outbound phase. In the following, we describe the inbound and outbound phase of the rebound attack on LANE.

3.2 Outline of the Rebound Attack on LANE

Due to the message expansion of LANE, at least 4 lanes are active in a differential attack. We will launch a semi-free-start collision attack, and therefore we assume the differences in (h_0, h_1) to be zero. Hence, lane P_3 is not active and we choose P_1 and thus, (b_0, b_1) to be not active as well. The active lanes in our attack on LANE are P_0, P_2, P_4 and P_5 . The corresponding truncated differential path for the P-lanes of LANE-256 is shown in Fig. 2. This path is very similar to the truncated differential path for LANE-256 shown in the LANE specification

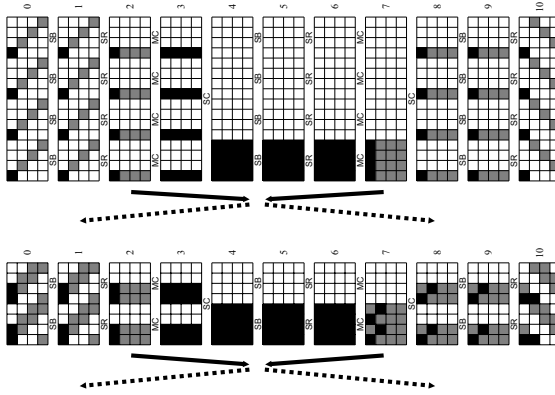


Fig. 1. The inbound phase for LANE-256 (left) and LANE-512 (right). Black bytes are active, gray bytes fixed by solutions of the inbound phase.

[Fig. 4.2, page 33], but turned upside-down. The truncated differential path used in the attack on LANE-512 is the same as in the LANE specification [Fig. 4.3, page 34] and shown in Fig. 3. The main idea of these paths is to use differences in only one of the parallel AES states for the inbound phases. This allows us to use the freedom in the other states to satisfy the outpound phases. Since we search for a collision after the P-lanes, we do not need to consider the Q-lanes.

The main idea of the attack on LANE is that we can apply more than one efficient inbound phase by using the degrees of freedom and the relatively slow diffusion due to the 2 (or 4) parallel AES states of LANE-256 (or LANE-512). The positions of the active bytes of two consecutive inbound phases are chosen such that when merging them, the number of the common active bytes of these phases is as small as possible. Since we can find many independent solutions for these inbound phases, we store them in some lists to be merged. In the outbound phase of the attack we merge the results of the inbound phases and further, merge the results of all active P-lanes. Note that the merging of two lists can be done efficiently. In each merging step, a number of conditions need to be fulfilled for the elements of the new list. We merge the lists in a clever order, such that we find one colliding pair for the compression function at the end.

In more detail, we first filter the results of each inbound phase for those solutions, which can connect both inbound phases (see Fig. 2). Then, we merge the resulting lists of two lanes such that we get a collision after the P-lanes, and parts of the message expansion are fulfilled. Finally, we filter the results of the left P-lanes (P_0, P_2) and the right P-lanes (P_4, P_5), such that the conditions on the whole message expansion are fulfilled. In the attack, we try to keep the size of the intermediate results at a reasonable size. We need to ensure, that the complexity of generating the lists is below $2^{n/2}$, but still get enough solutions in each phase to continue with the attack.

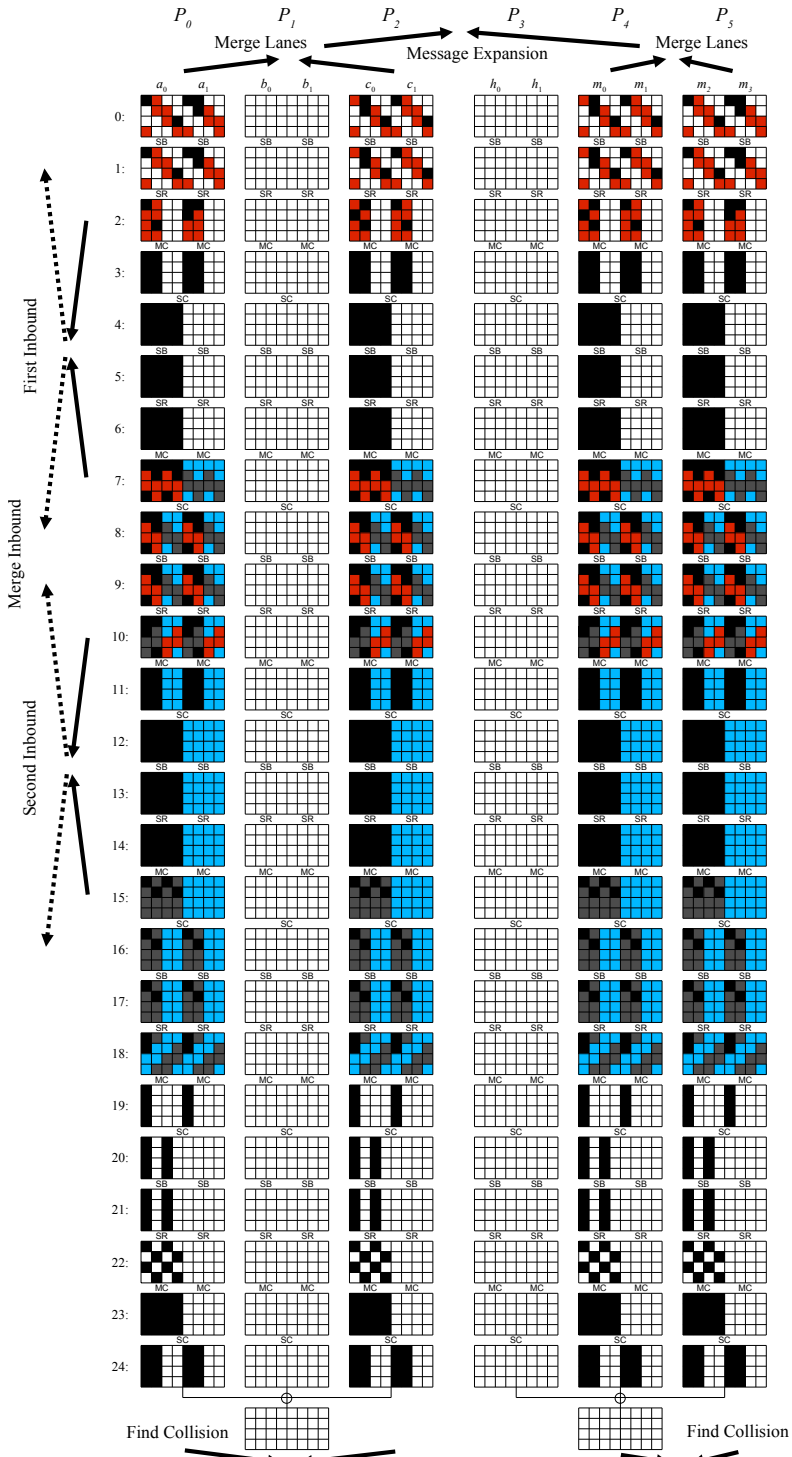


Fig. 2. The truncated differential path for 6 rounds of LANE-256. Black bytes are active, red (gray) bytes correspond to the first inbound phase, gray (dark gray) bytes to the second inbound phase and blue (light gray) bytes are used to find collisions in the P-lanes (colors in brackets correspond to grayscale printing).

3.3 The Inbound Phase

In the rebound attack on LANE, we first apply the inbound phase for a number of times. Therefore, we will explain this phase and the corresponding probabilities in detail here. In the inbound phase, we search for differences and values conforming to the truncated differential path for LANE-256 or LANE-512 shown in Fig. [11](#), with active bytes marked by black bytes. We only describe the application of one inbound phase here. In the example of Fig. [11](#), we have 16 active S-boxes between state #4 and state #5. It follows from the MDS property of MixColumns, that this path has at least one active byte in each of the 4 corresponding columns prior to the first, and after the second MixColumns transformation (state #2 and state #7). Note that the active bytes in state #2 and state #7 can also be at any position marked by gray bytes.

In the inbound phase, we first choose random differences for the 4 active bytes after the second MixColumns transformation (state #7). These differences are linearly propagated backward to 16 active bytes at the output of the previous SubBytes layer (state #5). Next, we take random differences for the 4 active bytes prior to the first MixColumns transformation (state #2) and linearly propagate forward to 16 active bytes at the input of SubBytes (state #4). Then, we need to find a match for the input and output differences of all 16 active S-boxes. For a single S-box, the probability that a random S-box differential exists is about one half, which can be verified easily by computing the differential distribution table of the AES S-box (see [9](#) for more details).

For each matching S-box, we get at least two (in some cases 4) possible byte values such that the S-box differential holds. Hence, we get at least 2^{16} possible values for one full AES state, such that the differential path for the chosen differences in state #2 and state #7 holds. In other words, after trying 2^{16} non-zero differences of state #2 and state #7, we get at least 2^{16} solutions for the truncated differential path between state #2 and state #7. Hence, the average complexity to find one solution for the inbound phase (differences and values) is about 1. Note that this holds for both, LANE-256 and LANE-512.

3.4 The Outbound Phase

After we have found differences and values for each inbound phase of the active lanes, we need to connect these results and propagate them outwards in the outbound phase. In backward direction, we need to match the message expansion at the input of each lane. In forward direction, we need to match the differences of two P-lanes on each side to get a collision. We describe the conditions for these two parts according to our truncated differential path in the following.

The Message Expansion. After the inbound phases, we get values and differences at the input and output of the 4 active lanes P_0 , P_2 , P_4 and P_5 . Since we have zero differences in (h_0, h_1) and (b_0, b_1) , we get using the message expansion for lane P_1 (see Equation [\(11\)](#)):

$$\Delta b_0 = 0 = \Delta m_0 \oplus \Delta m_2 \oplus \Delta m_3, \quad \Delta b_1 = 0 = \Delta m_1 \oplus \Delta m_2$$

Hence, we get the following relation for the message differences in m_0 , m_1 , m_2 , and m_3 :

$$\Delta m_1 = \Delta m_2 = \Delta m_0 \oplus \Delta m_3 \quad (2)$$

Using (II) we get for the differences in the expanded message words (a_0, a_1) and (c_0, c_1) :

$$\Delta a_0 = \Delta m_1, \Delta a_1 = \Delta m_3, \Delta c_0 = \Delta m_0, \Delta c_1 = \Delta m_2 \quad (3)$$

and thus, the following relations between a_0 , a_1 , c_0 , and c_1 :

$$\Delta a_0 = \Delta c_1 = \Delta a_1 \oplus \Delta c_0 \quad (4)$$

Beside the differences, we also need to match the values of the message expansion. Since we aim for a semi-free-start collision, we can freely choose the chaining value (h_0, h_1) such that the conditions on (a_0, a_1) are satisfied:

$$h_0 = a_0 \oplus m_0 \oplus m_1 \oplus m_2 \oplus m_3, \quad h_1 = a_1 \oplus m_0 \oplus m_2$$

That means we have conditions on the input (c_0, c_1) left, which we need to match with the message words m_0 , m_1 , m_2 and m_3 . Since we can vary lanes P_0, P_2 and P_4, P_5 independently in the following attacks, we can satisfy these conditions by merging the results of both sides. Using the equations of the message expansion, we get for (c_0, c_1) using the values of (a_0, a_1) :

$$c_0 = a_0 \oplus a_1 \oplus m_0 \oplus m_2 \oplus m_3, \quad c_1 = a_0 \oplus m_1 \oplus m_2$$

We can rearrange these equations in order to have all terms corresponding to P_0, P_2 on the left side and all terms of P_4, P_5 on the right side:

$$m_0 \oplus m_2 \oplus m_3 = c_0 \oplus a_0 \oplus a_1, \quad m_1 \oplus m_2 = c_1 \oplus a_0 \quad (5)$$

For merging the two sides, we will compute, store and compare the following values of each list:

$$v_1 = c_0 \oplus a_0 \oplus a_1, \quad v_2 = c_1 \oplus a_0, \quad v_3 = m_0 \oplus m_2 \oplus m_3, \quad v_4 = m_1 \oplus m_2$$

Colliding P-Lanes. In the forward direction, we need to find a collision for the differences in P_0 and P_2 , such that $\Delta P_0 \oplus \Delta P_2 = 0$ and for the differences in P_4 and P_5 , such that $\Delta P_4 \oplus \Delta P_5 = 0$. Note that we can swap the order of the last MixColumns with the XOR operation of the P-lanes since both transformations are linear. Hence, we only need to match the differences after the last SubBytes layer in each of the two active lanes. The blue bytes in Fig. 2 of LANE-256, or the red, blue and yellow bytes in Fig. 3 of LANE-512 are independent of the inbound phase. Hence, we can use the freedom in these bytes to find a collision after the P-lanes.

4 Semi-Free-Start Collision for LANE-256

In the rebound attack on LANE-256, we construct a semi-free-start collision for the full compression function using 2^{96} compression function evaluations and memory requirements of 2^{88} . We will use the 6-round truncated differential path given in Fig. 2 which is very similar to the one shown in the LANE specification [Fig. 4.2, page 33]. We search for a collision after the P-lanes of LANE and use the same truncated differential path in the 4 active lanes P_0, P_2, P_4 and P_5 . Since we do not consider differences in h_0 and h_1 , but we fix their values, the result will be a semi-free-start collision. The attack on LANE-256 consists basically of the following parts:

1. **First Inbound Phase:** Apply the inbound phase at the beginning of the truncated differential path (state #2 to state #7) for each lane P_0, P_2, P_4, P_5 independently.
2. **Second Inbound Phase:** Apply the inbound phase in the middle of each lane again (state #10 to state #15).
3. **Merge Inbound Phases:** Merge the results of the two inbound phases (state #7 to state #10).
4. **Merge Lanes:** Merge the two neighboring lanes P_0, P_2 and P_4, P_5 and satisfy according differences of the message expansion.
5. **Message Expansion:** Merge the two sides (P_0, P_2) and (P_4, P_5) and satisfy the remaining conditions on the message expansion (differences and values).
6. **Find Collisions:** Choose remaining free values (neutral bytes) to find a collision for each side (P_0, P_2) and (P_4, P_5) independently.
7. **Message Expansion:** Merge the two sides (P_0, P_2) and (P_4, P_5) and satisfy the conditions on the message expansion of the remaining bytes.

4.1 First Inbound Phase

We start the attack on LANE-256 by applying the first inbound phase to each of the 4 active lanes P_0, P_2, P_4, P_5 independently. In each lane, we start with 5 active bytes in state #2 and 8 active bytes in state #7 and choose 2^{96} random non-zero differences for these 13 bytes (note that we could choose up to 2^{104} differences). We propagate backward and forward to 16 active bytes at the input (state #4) and output (state #5) of the SubBytes layer in between. We get at least 2^{96} solutions for the inbound phase with a complexity of 2^{96} (see Sect. 3.3). For each result, only the red and black bytes in Fig. 2 are determined, i.e. the differences as well as the actual values of the bytes are found. Note that we have chosen the position of active bytes in state #0, such that at least one term of Equation (2) or (4) is zero for each byte. At this point, we can compute backwards to state #0 and independently verify the condition on one byte of the input differences:

$$\begin{aligned}
 P_0 : \Delta a_0[0, 0] &= \Delta a_1[0, 0], & P_4 : \Delta m_0[2, 3] &= \Delta m_1[2, 3] \\
 P_2 : \Delta c_0[2, 3] &= \Delta c_1[2, 3], & P_5 : \Delta m_2[0, 0] &= \Delta m_3[0, 0]
 \end{aligned}$$

The condition on each of these bytes is fulfilled with a probability of 2^{-8} and we store the 2^{88} valid results of each lane P_0 , P_2 , P_4 and P_5 in the corresponding lists L_0 , L_2 , L_4 and L_5 . Note that we store the values and differences of state #10 (red and black bytes) in these lists, since we need to merge these bytes with the second inbound phase in the following. For an efficient merging step, the lists are stored in hash tables (or sorted) according to the bytes to be merged (differences and values of active bytes in state #10).

4.2 Second Inbound Phase

Next, we apply the inbound phase again to match the differences at SubBytes between state #12 and state #13. We start with 2^{64} differences in the 8 active bytes of state #10 and 2^{32} differences in the 4 active bytes of state #15. Hence, we get about 2^{96} solutions for the second inbound phase with a complexity of 2^{96} . For each result, the gray and black values in Fig. 2 between state #7 and state #18 are determined. Again, this means we fix the actual values of these bytes. The results of the second inbound phase for each lane are stored in lists L'_0 , L'_2 , L'_4 and L'_5 . A node of each lists holds the values and differences of state #10 (gray and black bytes). Again, the lists are stored in hash tables (or sorted) according to the bytes (black bytes) to be merged.

4.3 Merge Inbound Phases

The two previous inbound phases overlap in 8 active bytes (state #7 to state #10). We connect the two inbound phases by checking the conditions on the overlapping bytes of state #10. Since both values and differences need to match, we get a condition on 128 bits. We merge the 2^{88} results of the first inbound phase and 2^{96} results of the second inbound phase to get $2^{88} \times 2^{96} \times 2^{-128} = 2^{56}$ differential paths for each lane. A pair connecting both inbound phases is found trivially. For each node of the first list (for example L_0), we check the overlapping bytes against the values of the second list (L'_0). Since the second list is a hash table, the effort for producing all 2^{56} valid pairs is 2^{88} hash table lookups.

Note that for each pair which satisfies and connects both inbound phases, the differences and values between state #0 and state #18 (black, red and gray bytes) are determined. We compute and store the 2^{56} input values and differences of state #0 in lists L_0 , L_2 , L_4 and L_5 . Although we still do not know half of the state, each of these input pairs conforms to the whole truncated differential path from state #0 to state #24 with a probability of 1. In other words, we know that in state #24, there are at most the given bytes active.

4.4 Merge Lanes

Next, we continue with merging the solutions of each lane by considering the message expansion. We first combine the inputs of lane P_0 and P_2 by merging lists L_0 and L_2 . When merging these lists, we need to satisfy the conditions on

the differences of the message expansion. We have conditions on 5 active bytes of state #0 in lane P_0 and P_2 (see Fig. 2). Remember that we have chosen the position of these active bytes, such that at least one term of Equation (2) or (4) is zero. Hence, we only need to check if *two* corresponding byte differences are equal. Since we have already verified one byte difference (see Sect. 4.1), we have 4 byte condition left:

$$\Delta a_0[0, 0] = \Delta c_1[0, 0], \quad \Delta a_1[0, 1] = \Delta c_0[0, 1] \quad (6)$$

$$\Delta a_1[1, 1] = \Delta c_0[1, 1], \quad \Delta a_0[2, 3] = \Delta c_0[2, 3] \quad (7)$$

These conditions are fulfilled with a probability of 2^{-32} and by merging two lists (L_0 and L_2) of size 2^{56} , we get $2^{56} \times 2^{56} \times 2^{-32} = 2^{80}$ valid matches which we store in list L_{02} . We repeat the same for lane P_4 and P_5 by merging lists L_4 and L_5 . We get 2^{80} matches for list L_{45} as well, since we need to fulfill the 32-bit conditions on the differences of the following 4 bytes:

$$\Delta m_1[0, 0] = \Delta m_2[0, 0], \quad \Delta m_0[0, 1] = \Delta m_3[0, 1] \quad (8)$$

$$\Delta m_0[1, 1] = \Delta m_3[1, 1], \quad \Delta m_0[2, 3] = \Delta m_2[2, 3] \quad (9)$$

Again, if we use hash tables or the previous lists are sorted according to the bytes to match, the merge operation can be performed very efficiently. Hence, the total complexity to produce the lists L_{02} and L_{45} is determined by their final size and requires an effort of around 2^{80} computations.

4.5 Message Expansion

For all entries of the lists L_{02} and L_{45} , the values in 32 bytes and differences in 10 bytes of each of (a_0, a_1, c_0, c_1) and (m_0, m_1, m_2, m_3) have been fixed (red and black bytes in state #0 of Fig. 2). Note that the conditions on the differences of each side on its own have already been fulfilled ($P_0 \leftrightarrow P_2$ and $P_4 \leftrightarrow P_5$). Hence, if we just fulfill the conditions on the remaining differences between $P_0 \leftrightarrow P_4$, then the conditions on $P_2 \leftrightarrow P_5$ are satisfied as well. Using Equations (2)-(4), the position of active bytes in Fig. 2 and the already matched differences of Sect. 4.1 and Sect. 4.4, we only have the following 4 byte conditions left:

$$\Delta a_0[0, 0] = \Delta m_1[0, 0], \quad \Delta a_1[0, 1] = \Delta m_0[0, 1]$$

$$\Delta a_1[1, 1] = \Delta m_0[1, 1], \quad \Delta a_0[2, 3] = \Delta m_0[2, 3]$$

Note that we also need to fulfill the conditions on the values of the states. Remember that we can freely choose the chaining values (h_0, h_1) to satisfy the values in the first 16 bytes of the message expansion (a_0, a_1) . To fulfill the conditions on the 16 bytes of (c_0, c_1) we need to satisfy Equation (5) using the corresponding values v_1, v_2, v_3 and v_4 . Hence, we need to find a match for the following values and differences by merging lists L_{02} and L_{45} :

- 8 bytes of v_1 from L_{02} with v_3 from L_{45} ,
- 8 bytes of v_2 from L_{02} with v_4 from L_{45} ,
- 4 bytes of differences in L_{02} and in L_{45} .

Since we have 2^{80} elements in each list and conditions on 160 bits, we expect to find $2^{80} \times 2^{80} \times 2^{-160} = 1$ result. This result satisfies the message expansion for all lanes and is a solution for the truncated differential path of each active lane between state #0 and state #24. However, we do not get a collision at the end of the P-lanes yet, since we do not know the differences of state #24.

4.6 Find Collisions

In this phase of the attack, we search for a collision at the end of the P-lanes (P_0, P_2) and (P_4, P_5) using the remaining freedom in the second half of the state. Note that the 16-byte difference in state #24 is obtained from 8-byte difference in state #22 with the linear transforms `MixColumns` and `SwapColumns`. Hence, the collision space (the 16 bytes where the two lanes differ) has only 2^{64} distinct elements. If we take a look at Fig. 2, we get for the values in state #7:

- The black, red and gray bytes represent values which have already been determined by the previous parts of the attack.
- The blue bytes represent values not yet determined and can be used to vary the differences in state #22.

To find a collision between two lanes, we can still choose 2^{64} values for the blue bytes in state #7 of each lane and store these results in lists L_0, L_2, L_4 and L_5 . Note that for these 2^{64} values, we get only 2^{32} different values for the two free bytes in the first and fifth column of state #18. Hence, we can only iterate through 2^{32} differences in state #22 for each lane. However, this is enough to find one colliding difference for each side, since $2^{32} \times 2^{32} \times 2^{-64} = 1$. By repeating this step 2^{32} times for each side, we expect $2^{64} \times 2^{64} \times 2^{-64} = 2^{64}$ results for each merged list L_{02} and L_{45} .

4.7 Message Expansion

Finally, we need to match the message expansion for the remaining 32 bytes of each side. Hence, we just repeat the same procedure as we did for the first half of state #0, except that we only need to match the values of 32 bytes but no differences. Again, we can use the remaining bytes of (h_0, h_1) to fulfill the conditions on 16 bytes of (a_0, a_1) . Since, we have 2^{64} solutions in each list L_{02} and L_{45} , we expect to find $2^{64} \times 2^{64} \times 2^{-128} = 1$ colliding pair for (c_0, c_1) and thus, a collision for the full compression function of LANE-256.

4.8 Complexity

Let us find the complexity of the whole attack. The first inbound phase requires 2^{96} computations and 2^{88} memory, the second inbound requires 2^{96} computations

and 2^{96} memory, and the merging of the inbound phases requires 2^{88} hash table lookups and 2^{56} memory. Obviously, the second inbound phase and the merge inbound phases can be united to lower the memory requirement of these three steps. Namely, we create the lists L_0 , L_2 , L_4 and L_5 in the first inbound phase. Then, for each differential path of the second inbound phase, instead of storing it in a list, we immediately check if it can be merged with some differential from the lists. Only if it can be merged, we do the outbound phase and compute state #0. Hence, the first three steps of our attack require around 2^{96} computations and 2^{88} memory. The merge lanes step requires 2^{80} computations and memory. The message expansion steps require 2^{80} computations, while the find collisions steps require 2^{32} computations. Hence, the total attack complexity is around 2^{96} computations and 2^{88} memory. Note that the cost of each computation is never greater than the cost of one compression function evaluation. Therefore, the complexity to find a semi-free-start collision for all 6 rounds of LANE-256 is about 2^{96} compression function evaluations and 2^{88} memory.

5 Semi-Free-Start Collision for LANE-512

In the rebound attack on LANE-512, we construct a semi-free-start collision for the full, 8-round compression function using 2^{224} compression function evaluations and memory requirements of 2^{128} . We use the same iterative truncated differential path as shown in the specification of LANE-512 [Fig. 4.3, page 34], which is given in Fig. 3. Similar to the attack on LANE-256, we search for a collision after the P-lanes and use the same truncated differential path in the 4 active lanes P_0 , P_2 , P_4 and P_5 . The attack on LANE-512 consists basically of the following parts:

1. **First Inbound Phase:** Apply the inbound phase at the beginning of the truncated differential path (state #2 to state #7) for each lane P_0 , P_2 , P_4 , P_5 independently.
2. **Merge Lanes:** Merge the two neighboring lanes P_0, P_2 and P_4, P_5 and satisfy according differences of the message expansion.
3. **Message Expansion:** Merge the two sides (P_0, P_2) and (P_4, P_5) and satisfy the remaining conditions on the message expansion (differences and values).
4. **Second Inbound Phase:** Apply the inbound phase in the middle of each lane again (state #10 to state #15).
5. **Merge Inbound Phases:** Merge the results of the two inbound phases.
6. **Starting Points:** Choose random values for the brown bytes in state #7 to get enough starting points for the subsequent phases.
7. **Merge Lanes:** Merge the values of the starting points for the two neighboring lanes P_0, P_2 and P_4, P_5 and satisfy the according differences of the message expansion.
8. **Message Expansion:** Merge the two sides (P_0, P_2) and (P_4, P_5) and satisfy the remaining conditions on the message expansion (differences and values) for the starting points.

9. **Third Inbound Phase:** Apply the inbound phase at the end of each lane for a third time (state #18 to state #23).
10. **Merge Inbound Phases:** Merge the results of the three inbound phases and use the remaining freedom in between.
11. **Find Collisions:** Merge the corresponding two lanes to find a collision for each side (P_0, P_2) and (P_4, P_5) independently.
12. **Message Expansion:** Merge the two sides (P_0, P_2) and (P_4, P_5) and satisfy the conditions on the message expansion of the remaining bytes.

5.1 First Inbound Phase

We start the attack on LANE-512 by applying the first inbound phase to each of the 4 active lanes P_0, P_2, P_4, P_5 independently. In each lane, we start with 8 active bytes in state #2 and 4 active bytes in state #7 and choose 2^{84} random non-zero differences for these 12 bytes (note that we could choose up to 2^{96} differences). We propagate backward and forward to 16 active bytes at the input (state #4) and output (state #5) of the SubBytes layer in between. We get at least 2^{84} matches for the inbound phase with a complexity of 2^{84} (see Sect. 3.3). For each result, the gray and black bytes in Fig. 3 are determined. Hence, we can already verify the condition on one byte of the input differences for each lane by computing backwards to state #0:

$$\begin{aligned}
 P_0 : \Delta a_0[2, 2] &= \Delta a_1[2, 2], & P_0 : \Delta a_0[2, 6] &= \Delta a_1[2, 6] \\
 P_2 : \Delta c_0[1, 1] &= \Delta c_1[1, 1], & P_2 : \Delta c_0[1, 5] &= \Delta c_1[1, 5] \\
 P_4 : \Delta m_0[1, 1] &= \Delta m_1[1, 1], & P_4 : \Delta m_0[1, 5] &= \Delta m_1[1, 5] \\
 P_5 : \Delta m_2[2, 2] &= \Delta m_3[2, 2], & P_5 : \Delta m_2[2, 6] &= \Delta m_3[2, 6]
 \end{aligned}$$

The conditions on each of the lanes are fulfilled with a probability of 2^{-16} and we store the 2^{68} valid matches of each lane P_0, P_2, P_4 and P_5 in the corresponding lists L_0, L_2, L_4 and L_5 .

5.2 Merge Lanes

Next, we continue with merging the solutions of each lane by considering the message expansion. We first combine the results of lane P_0 and P_2 by merging lists L_0 and L_2 . When merging these lists, we need to satisfy the conditions on the differences of the message expansion for the following 6 bytes:

$$\begin{aligned}
 \Delta a_1[0, 0] &= \Delta c_0[0, 0], & \Delta a_1[0, 4] &= \Delta c_0[0, 4] \\
 \Delta a_0[1, 1] &= \Delta c_0[1, 1], & \Delta a_0[1, 5] &= \Delta c_0[1, 5] \\
 \Delta a_0[2, 2] &= \Delta c_1[2, 2], & \Delta a_0[2, 6] &= \Delta c_1[2, 6]
 \end{aligned}$$

Since this match is fulfilled with a probability of 2^{-48} and we merge two lists of size 2^{68} , we get $2^{68} \times 2^{68} \times 2^{-48} = 2^{88}$ valid matches which we store in L_{02} .

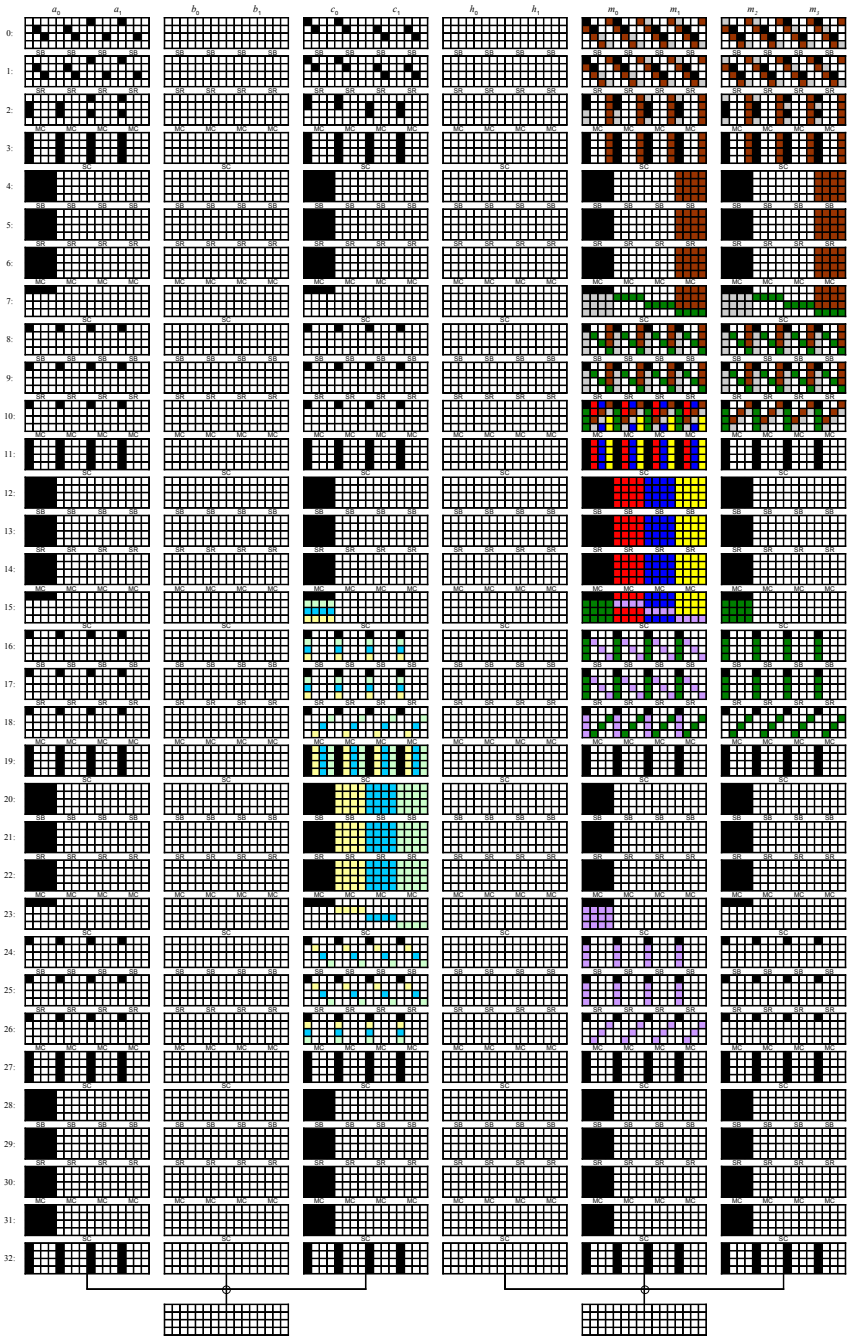


Fig. 3. The truncated differential path for 8 rounds of LANE-512. Lane P_0 shows the plain truncated differential path, lane P_2 other possible truncated differential paths and lane P_4 and P_5 are used to describe the attack.

We repeat the same for lane P_4 and P_5 merge lists L_4 and L_5 . We get 2^{88} matches for list L_{45} , since we need to fulfill conditions on differences of 6 bytes as well:

$$\begin{aligned}\Delta m_0[0, 0] &= \Delta m_3[0, 0], & \Delta m_0[0, 4] &= \Delta m_3[0, 4] \\ \Delta m_0[1, 1] &= \Delta m_2[1, 1], & \Delta m_0[1, 5] &= \Delta m_2[1, 5] \\ \Delta m_1[2, 2] &= \Delta m_2[2, 2], & \Delta m_1[2, 6] &= \Delta m_2[2, 6]\end{aligned}$$

5.3 Message Expansion

For all entries of lists L_{02} and L_{45} , the values in 32 bytes and differences in 16 bytes of each of (a_0, a_1, c_0, c_1) and (m_0, m_1, m_2, m_3) have been fixed (gray and black bytes in state #0 of Fig. 3). Since the conditions on the differences of each side on its own have already been fulfilled, we just need to match the conditions on the remaining 6-byte differences between each side (P_0, P_2) and (P_4, P_5) :

$$\begin{aligned}\Delta a_1[0, 0] &= \Delta m_0[0, 0], & \Delta a_1[0, 4] &= \Delta m_0[0, 4] \\ \Delta a_0[1, 1] &= \Delta m_0[1, 1], & \Delta a_0[1, 5] &= \Delta m_0[1, 5] \\ \Delta a_0[2, 2] &= \Delta m_1[2, 2], & \Delta a_0[2, 6] &= \Delta m_1[2, 6]\end{aligned}$$

Remember that we can freely choose the chaining values (h_0, h_1) to satisfy the values in the first 16 bytes of the message expansion (a_0, a_1) . To fulfill the conditions on the 16 bytes of (c_0, c_1) we need to find matches for the following values and differences using lists L_{02} and L_{45} :

- 8 bytes of v_1 from L_{02} with v_3 from L_{45} ,
- 8 bytes of v_2 from L_{02} with v_4 from L_{45} ,
- 6 bytes of differences in L_{02} and in L_{45} .

Since we have 2^{88} elements in each list and conditions on 176 bits, we expect to find $2^{88} \times 2^{88} \times 2^{-176} = 1$ result. This result satisfies the message expansion for all lanes and is a solution for the truncated differential path of each active lane between state #0 and state #10.

5.4 Second Inbound Phase

Next, we apply the inbound phase again to match the differences at **SubBytes** between state #12 and state #13. After the first inbound phase, the values of 16 bytes in state #10 (black and gray bytes), and the difference in 16 bytes (1st AES-block) of state #12 (black bytes) have already been fixed. Hence we can start with 2^{32} possible 4-byte differences in state #15, compute backwards to state #13 and need to match the differences in the **SubBytes** layer. We expect to find at least 2^{32} solutions for the second inbound phase (see Sect. 3.3).

5.5 Merge Inbound Phases

The result of the second inbound phase are 2^{32} values for the 16 bytes in state #10 (green and black bytes). From the first inbound phase, we have obtained

one solution for 16 bytes in state #10 (gray and black bytes) as well. In these 16 bytes, the values of the 4 active bytes (black) overlap between both inbound phases and the probability for a successful match is 2^{-32} . Among the 2^{32} results of the second inbound phase, we expect to find one solution to match the values of state #10. Once we have found a match, we can compute the values of the newly determined 12 bytes in state #7, marked by green bytes in Fig. 3.

5.6 Starting Points

In this phase of the attack, we will compute a number of starting points which we will need for the subsequent steps. For each lane, we choose random values for the 12 bytes in state #7 (marked by brown bytes in Fig. 3) and compute the corresponding 16-byte values in state #0. We repeat this step 2^{64} times and store the results in the corresponding lists L'_0 , L'_2 , L'_4 or L'_5 .

5.7 Merge Lanes

Next, we merge lists L'_0 and L'_2 to get the list L'_{02} , consisting of 2^{128} values for the 32 newly determined bytes of (m_0, m_1, m_2, m_3) (brown bytes of state #0 in lane P_0 and P_2). Further, we merge lists L'_4 and L'_5 to get the list L'_{45} of size 2^{128} containing the 32 byte values of (a_0, a_1, c_0, c_1) .

5.8 Message Expansion

Finally, we satisfy the conditions of the message expansion on (a_0, a_1) using the values of (h_0, h_1) , and use the two lists L'_{02} and L'_{45} to satisfy the conditions on (c_0, c_1) . Since we need to match 16 bytes of (c_0, c_1) and have 2^{128} elements in both lists, we expect $2^{128} \times 2^{128} \times 2^{-128} = 2^{128}$ matching pairs which we store in list L_s . We will use these values in a later phase of the attack.

5.9 Third Inbound Phase

Now, we extend the truncated differential path by applying a third inbound phase between state #18 and state #23 for each active lane. Note that the values in 16 bytes of state #18 (black and green bytes), and the differences in 16 bytes (1st AES-block) of state #20 (black bytes) have already been fixed due to the second inbound phase. Similar to the second inbound phase, we start with 2^{32} 4-byte differences in state #23 and compute backwards to state #21 to get a match for the SubBytes layer. Since we have 2^{32} starting differences, we expect to find 2^{32} results for the third inbound phase, with fixed values and differences for the 16 bytes in state #15 (purple and black bytes).

5.10 Merge Inbound Phases

The values of the second and the third inbound phase overlap in 4 active bytes (black) of state #18. Since we have 2^{32} results of the third inbound phase, we

expect to find one solution after merging the two phases. Once we have found a match, we can compute the values of the newly determined 12 bytes in state #15, marked by purple bytes in Fig. 3. Next, we need to connect all three inbound phases. For all possible 8-byte values of state #10 marked by red bytes, we compute the 16 corresponding bytes in state #15 (2nd AES-block). If the computed values satisfy the 4 bytes in state #15 marked by purple, we store the result of each lane in the corresponding lists L_0^a , L_2^a , L_4^a and L_5^a . In total, we obtain $2^{64} \cdot 2^{-32} = 2^{32}$ entries in each list. We repeat the same for the bytes marked by blue and yellow, and generate the lists L_i^b and L_i^c for each of the active lanes with index $i \in \{0, 2, 4, 5\}$. For each lane, we merge the three lists L_i^a , L_i^b and L_i^c and store the 2^{96} results in lists L_i^* . Note that for each entry in these lists, we can determine all values and differences of the corresponding lane.

5.11 Find Collisions

In this phase of the attack, we finally search for a collision at the end of the P-lanes (P_0, P_2) and (P_4, P_5) using the elements of lists L_i^* . To find a collision at the end of the P-lanes, we need to match the 16 byte differences in state #32 of the two corresponding active lanes such that $\Delta(P_0 \oplus P_2) = 0$ and $\Delta(P_4 \oplus P_5) = 0$. Note that we can satisfy these conditions independently for each side (P_0, P_2) and (P_4, P_5). Since we need to match 128 bits and we have 2^{96} elements in each list L_i^* , we expect to find $2^{96} \cdot 2^{96} \cdot 2^{-128} = 2^{64}$ collisions for each side. We store the corresponding inputs (a_0, a_1, c_0, c_1) for the collisions between lane P_0 and P_2 in list L_{02}^* and the inputs (m_0, m_1, m_2, m_3) for the collisions between lane P_4 and P_5 in list L_{45}^* .

5.12 Message Expansion

Finally, we need to match the message expansion for the remaining 32 bytes of each side. Hence, we just repeat the same procedure as we did for the first part of state #0, except that we only need to match the values of 32 bytes but no differences. Again, we use the values of (h_0, h_1) to satisfy the conditions on (a_0, a_1) first. Then, we match the values of the 32 bytes in (c_0, c_1) . Since we only have 2^{64} entries in both of L_{02}^* and L_{45}^* , the success probability for a match is $2^{64} \cdot 2^{64} \cdot 2^{-256} = 2^{-128}$. However, we can still repeat from Sect. 5.6 using a different starting point stored in list L_s . Since we have 2^{128} elements in list L_s , we can repeat the previous steps up to 2^{128} times. Hence, we expect to find one valid match for the message expansion and thus, a collision for the full compression function of LANE-512.

5.13 Complexity

The total complexity of the rebound attack on LANE-512 is determined by the merging step after the third inbound phase. This step has a complexity of 2^{96} compression function evaluations and is repeated 2^{128} times. The memory

requirements are determined by the largest lists, which are L'_{02} and L'_{45} (or L_s) with a size of 2^{128} . Hence, the total complexity to find a semi-free-start collision for LANE-512 is about $2^{128} \cdot 2^{96} = 2^{224}$ compression function evaluations and 2^{128} in memory.

6 Conclusion

In this work, we have applied the rebound attack to the hash function LANE. In the attack we use a truncated differential path with differences concentrating mostly in one part of the lanes. Due to the relatively slow diffusion of parallel AES rounds, we are therefore able to solve parts of the lanes independently. First, we search for differences and values (for parts of the state) according to the truncated differential path and also satisfy the message expansion. Then, we choose values which can be changed such that the truncated differential path and according message expansion still holds. The freedom in these values is then used to search for a collision at the end of the lanes without violating the differential path or message expansion.

In the rebound attack on LANE, we are able to construct semi-free-start collisions for full round LANE-224 and LANE-256 with 2^{96} compression function evaluations and memory of 2^{80} , and for full round LANE-512 with complexity of 2^{224} compression function evaluations and memory of 2^{128} . Although these collisions on the compression function do not imply an attack on the hash functions, they violate the reduction proofs of Merkle and Damgård, or Andreeva in the case of LANE. However, due to the limited degrees of freedom, a collision attack on the hash function seems to be difficult for full round LANE.

References

1. Andreeva, E.: On LANE modes of Operation. Technical Report, COSIC (2008)
2. Barreto, P.S.L.M., Rijmen, V.: The Whirlpool Hashing Function. Submitted to NESSIE (September 2000), <http://www.larc.usp.br/~pbarreto/WhirlpoolPage.html> (revised May 2003)
3. Damgård, I.: A Design Principle for Hash Functions. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 416–427. Springer, Heidelberg (1990)
4. Gauravaram, P., Knudsen, L.R., Matusiewicz, K., Mendel, F., Rechberger, C., Schl affer, M., Thomsen, S.S.: Gr ostl – a SHA-3 candidate (2008), <http://www.groestl.info>
5. Indestege, S.: The LANE hash function. Submission to NIST (2008), <http://www.cosic.esat.kuleuven.be/publications/article-1181.pdf>
6. Knudsen, L.R.: Truncated and Higher Order Differentials. In: Preneel, B. (ed.) FSE 1994. LNCS, vol. 1008, pp. 196–211. Springer, Heidelberg (1995)
7. Knudsen, L.R., Rechberger, C., Thomsen, S.S.: The Grindahl Hash Functions. In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 39–57. Springer, Heidelberg (2007)

8. Lamberger, M., Mendel, F., Rechberger, C., Rijmen, V., Schl affer, 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)
9. Mendel, F., Rechberger, C., Schl affer, M., Thomsen, S.S.: The Rebound Attack: Cryptanalysis of Reduced Whirlpool and Gr ostl. In: Dunkelman, O. (ed.) FSE. LNCS, vol. 5665, pp. 260–276. Springer, Heidelberg (2009)
10. Merkle, R.C.: One Way Hash Functions and DES. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 428–446. Springer, Heidelberg (1990)
11. National Institute of Standards and Technology: Announcing Request for Candidate Algorithm Nominations for a New Cryptographic Hash Algorithm (SHA-3) Family. Federal Register Notice (November 2007), <http://csrc.nist.gov>
12. Peyrin, T.: Cryptanalysis of Grindahl. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 551–567. Springer, Heidelberg (2007)
13. 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)
14. 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)
15. Wu, S., Feng, D., Wu, W.: Cryptanalysis of the LANE Hash Function. In: Jacobson, M.J., Rijmen, V., Safavi-Naini, R. (eds.) SAC 2009. LNCS, vol. 5867, pp. 126–140. Springer, Heidelberg (2009)

Rebound Distinguishers: Results on the Full Whirlpool Compression Function

Mario Lamberger¹, Florian Mendel¹, Christian Rechberger¹,
Vincent Rijmen^{1,2,3}, and Martin Schl affer¹

¹ Institute for Applied Information Processing and Communications
Graz University of Technology, Inffeldgasse 16a, A-8010 Graz, Austria

² Department of Electrical Engineering ESAT/COSIC, Katholieke Universiteit
Leuven. Kasteelpark Arenberg 10, B-3001 Heverlee, Belgium

³ Interdisciplinary Institute for BroadBand Technology (IBBT), Belgium
mario.lamberger@iaik.tugraz.at

Abstract. Whirlpool is a hash function based on a block cipher that can be seen as a scaled up variant of the AES. The main difference is the (compared to AES) extremely conservative key schedule. In this work, we present a distinguishing attack on the full compression function of Whirlpool. We obtain this result by improving the rebound attack on reduced Whirlpool with two new techniques. First, the inbound phase of the rebound attack is extended by up to two rounds using the available degrees of freedom of the key schedule. This results in a near-collision attack on 9.5 rounds of the compression function of Whirlpool with a complexity of 2^{176} and negligible memory requirements. Second, we show how to turn this near-collision attack into a distinguishing attack for the full 10 round compression function of Whirlpool. This is the first result on the full Whirlpool compression function.

Keywords: hash functions, cryptanalysis, near-collision, distinguisher.

1 Introduction

In the last few years the cryptanalysis of hash functions has become an important topic within the cryptographic community. Especially the collision attacks on the MD4 family of hash functions (MD4, MD5, SHA-1) have weakened the security assumptions of these commonly used hash functions [6,7,17,24,25,26]. Still, most of the existing cryptanalytic work has been published for this particular family of hash functions. Therefore, the analysis of alternative hash functions is of great interest. In this article, we will present a security analysis of the Whirlpool hash function with respect to collision resistance.

Whirlpool is the only hash function standardized by ISO/IEC 10118-3:2004 (since 2000) that does not follow the MD4 design strategy. Furthermore, it has been evaluated and approved by NESSIE [20]. Whirlpool is commonly considered to be a conservative block-cipher based design with an extremely conservative key schedule and follows the wide-trail design strategy [4,5]. Since its proposal in 2000, only a few results have been published.

Table 1. Summary of results for Whirlpool. Complexities are given in compression function evaluations, a memory unit refers to a state (512 bits). The complexities in brackets refer to modified attacks using a precomputed table taking 2^{128} time/memory to set up.

target	rounds	complexity runtime/memory	type	source
block cipher W	6	$2^{120}/2^{120}$	distinguisher	Knudsen [11]
hash function	4.5	$2^{120}/2^7$	collision	Mendel <i>et al.</i> FSE 2009 [16]
hash function	6.5	$2^{128}/2^7$	near-collision	
compression function	5.5	$2^{120}/2^7$	collision	
compression function	7.5	$2^{128}/2^7$	near-collision	
hash function	5.5	$2^{120+s}/2^{64-s}$	collision	Appendix A
hash function	7.5	$2^{128+s}/2^{64-s}$	near-collision	Appendix A
compression function	7.5	$2^{184}/2^8$ ($2^{120}/2^{128}$)	collision	Sect. 4
compression function	9.5	$2^{176}/2^8$ ($2^{112}/2^{128}$)	near-collision	Sect. 4
compression function	10	$2^{188}/2^8$ ($2^{121}/2^{128}$)	distinguisher	Sect. 5

Related Work. At FSE 2009, Mendel *et al.* proposed a new technique for the analysis of hash functions: the *rebound attack* [16]. It can be applied to both block cipher based and permutation based constructions. The idea of the rebound attack is to divide an attack into two phases, an inbound and an outbound phase. In the inbound phase, degrees of freedom are used, such that in the outbound phase several rounds can be bypassed in both forward- and backwards direction. This led to successful attacks on round-reduced Whirlpool for up to 7.5 (out of 10) rounds. The results are summarized in Table 1.

For the block cipher W that is implicitly used in the Whirlpool compression function, Knudsen described an integral distinguisher for 6 out of 10 rounds [11]. Furthermore, it is assumed that this property may extend also to 7 rounds. Note that in [12] similar techniques were used to obtain known-key distinguishers for 7-rounds of the AES.

Our Contribution. The main contribution of this paper is a distinguishing attack on the full compression function of Whirlpool which is achieved by improving upon the work of Mendel *et al.* in [16] in several ways.

We start with a description of the hash function Whirlpool. Then, in Sect. 3, we give an overview of the rebound attack and show how it is applied to reduced versions of Whirlpool. In Sect. 4, we describe our improvement of the rebound attack on Whirlpool in detail. This technique enables us to add two rounds in the inbound phase of the attack and thus gives a collision and near-collision attack on the Whirlpool compression function reduced to 7.5 and 9.5 rounds, respectively. Based on this, we describe in Sect. 5 a new generic attack and show how to distinguish the full (all 10 rounds) compression function of Whirlpool from a random function by turning the near-collision attack for 9.5 rounds into a distinguishing attack for 10 rounds. To the best of our knowledge this is the first result on the full Whirlpool compression function. Table 1 summarizes the previous results on Whirlpool as well as the contributions of this paper.

2 Description of Whirlpool

Whirlpool is a cryptographic hash function designed by Barreto and Rijmen in 2000 [1]. It is an iterative hash function based on the Merkle-Damgård design principle (cf. [18]). It processes 512-bit message blocks and produces a 512-bit hash value. If the message length is not a multiple of 512, an unambiguous padding method is applied. For the description of the padding method we refer to [1]. Let $M = M_1 \| M_2 \| \dots \| M_t$ be a t -block message (after padding). The hash value $h = H(M)$ is computed as follows:

$$H_0 = IV \tag{1}$$

$$H_j = W(H_{j-1}, M_j) \oplus H_{j-1} \oplus M_j \quad \text{for } 0 < j \leq t \tag{2}$$

$$h = H_t \tag{3}$$

where IV is a predefined initial value and W is a 512 bit block cipher used in the Miyaguchi-Preneel mode [18]. The block cipher W used by Whirlpool is very similar to the Advanced Encryption Standard (AES) [19].

The state update transformation and the key schedule update an 8×8 state S and K of 64 bytes in 10 rounds. In one round, the state is updated by the round transformation r_i as follows:

$$r_i \equiv \text{AK} \circ \text{MR} \circ \text{SC} \circ \text{SB}.$$

The round transformations are briefly described here:

- the non-linear layer **SubBytes** (SB) applies an S-Box to each byte of the state independently.
- the cyclical permutation **ShiftColumns** (SC) rotates the bytes of column j downwards by j positions.
- the linear diffusion layer **MixRows** (MR) is a right-multiplication by the 8×8 circulant MDS matrix $\text{cir}(1, 1, 4, 1, 8, 5, 2, 9)$.
- the key addition **AddRoundKey** (AK) adds the round key K_i to the 8×8 state, and **AddConstant** (AC) adds the round constant C_i to the 8×8 state of the key schedule.

After the last round of the state update transformation, the initial value or previous chaining value H_{j-1} , the message block M_j , and the output value of the last round are combined (xored), resulting in the output of one iteration. A detailed description of the hash function is given in [1].

We denote the resulting state of round transformation r_i by S_i and the intermediate states after **SubBytes** by S_i^{SB} , after **ShiftColumns** by S_i^{SC} and after **MixRows** by S_i^{MR} . The initial state prior to the first round is denoted by $S_0 = M_j \oplus K_0$. The same notation is used for the key schedule with round keys K_i with $K_0 = H_{j-1}$.

3 The Rebound Attack

The rebound attack is a new tool for the cryptanalysis of hash functions and was published by Mendel *et al.* in [16]. It is a differential attack. The main

idea is to use the available degrees of freedom in a collision attack to efficiently fulfill the low probability parts in the middle of a differential trail. The rebound attack consists of an inbound phase with a meet-in-the-middle part in order to exploit the available degrees of freedom, and a subsequent probabilistic outbound phase. AES based hash functions are a natural target for this attack, since their construction principle allows a simple application of the idea.

3.1 Basic Attack Strategy

In the rebound attack, the compression function, internal block cipher or permutation of a hash function is split into three sub-parts. Let W be a block cipher, then $W = W_{fw} \circ W_{in} \circ W_{bw}$.

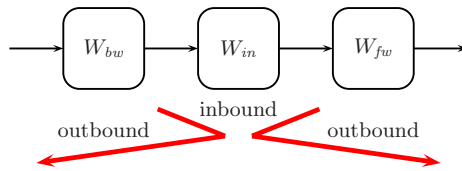


Fig. 1. A schematic view of the rebound attack. The attack consists of an inbound and two outbound phases.

The rebound attack can be described by two phases (see Fig. 1):

- **Inbound phase:** Is a meet-in-the-middle phase in W_{in} , which is aided by the degrees of freedom that are available to a hash function cryptanalyst. This very efficient combination of meet-in-the-middle techniques with the exploitation of available degrees of freedom is called the **match-in-the-middle approach**.
- **Outbound phase:** In the second phase, the matches of the inbound phase are computed in both forward- and backward direction through W_{fw} and W_{bw} to obtain desired collisions or near-collisions. If the differential trail through W_{fw} and W_{bw} has a low probability, one has to repeat the inbound phase to obtain more starting points for the outbound phase.

3.2 Preliminaries for the Rebound Attack on Whirlpool

In the following, we want to briefly summarize some well known facts that will be frequently used in the subsequent sections.

- *Truncated differentials:* Knudsen [10] proposed truncated differentials as a tool in block cipher cryptanalysis. In a *standard* differential attack (cf. [2]), the full difference between two inputs/outputs is considered whereas in the case of truncated differentials, the differences is only partially determined, *i.e.* for every byte, we only check if there is a difference or not. A byte having a non-zero difference is called *active*.

- *Difference Propagation in MixRows*: Since the MixRows operation is a linear transformation, standard differences propagate through MixRows in a deterministic way whereas truncated differences behave in a probabilistic way. The MDS property of the MixRows transformation ensures that the sum of the number of active input and output bytes is at least 9 (cf. [1]). In general, the probability of any $x \rightarrow y$ transition with $1 \leq x, y \leq 8$ satisfying $x + y \geq 9$ is approximately $2^{(y-8) \cdot 8}$. For a detailed description of the propagation of truncated differences in MixRows we refer to [16], see also [21].
- *Differential Properties of SubBytes*: Let $a, b \in \{0, 1\}^8$. For the Whirlpool S-box, we are interested in the number of solutions to the equation

$$S(x) \oplus S(x \oplus a) = b. \quad (4)$$

Exhaustively counting over all 2^{16} differentials shows that the number of solutions to (4) can only be 0, 2, 4, 6, 8 and 256, which occur with frequency 39655, 20018, 5043, 740, 79 and 1, respectively. The task to return all solutions x to (4) for a given differential (a, b) is best solved by setting up a precomputed table of size 256×256 which stores the solutions (if there are any) for each (a, b) .

However, it is easy to see that for any permutation S (to be more precise, for any injective map) the expected number of solutions to (4) is always 1. We get that $2^{-16} \sum_a \sum_b \#\{x \mid S(x \oplus a) \oplus S(x) = b\} = 2^{-16} \sum_a 2^8 = 1$, because for a fixed a , every solution x belongs to a unique b . Since the inputs to all the S-boxes are independent, the same reasoning is valid for the full SubBytes transformation.

3.3 Application to Round-Reduced Whirlpool

In this section, we will briefly describe the application of the rebound attack to the hash function Whirlpool. A detailed description of the attack is given in [16]. For a good understanding of our results, it is recommended to study these previous results on Whirlpool very carefully.

The rebound attack on Whirlpool is a differential attack which uses a differential trail with the minimum number of active S-boxes according to the wide trail design strategy. The core of the rebound attack on Whirlpool is a 4 round differential trail, where the fully active state is placed in the middle:

$$1 \xrightarrow{r_1} 8 \xrightarrow{r_2} 64 \xrightarrow{r_3} 8 \xrightarrow{r_4} 1$$

In the rebound attack, one first splits the block cipher W into three sub-ciphers $W = W_{fw} \circ W_{in} \circ W_{bw}$, such that the most expensive part of the differential trail is covered by the inbound phase W_{in} . In the inbound phase, the available degrees of freedom (in terms of actual values of the state) are used to guarantee that the differential trail in W_{in} holds. The differential trail in the outbound phase (W_{fw}, W_{bw}) is supposed to have a relatively high probability. While standard XOR differences are used in the inbound phase, truncated differentials are used in the outbound phase of the attack.

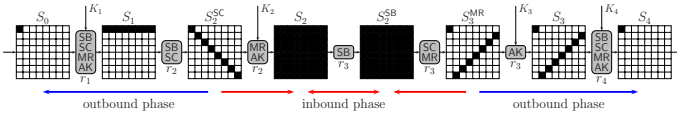


Fig. 2. A schematic view of the rebound attack on 4 rounds of Whirlpool with round key inputs. Black state bytes are active.

In the following, we briefly describe the inbound and outbound phase of the rebound attack on 4 rounds of Whirlpool. For a more detailed description, we refer to the original paper [16].

Inbound Phase. In the first step of the inbound phase, we choose a random difference with 8 active bytes at the input of MixRows of round r_2 (S_2^{SC}). Note that we need an active byte in each row of the state (see Fig. 2) to get a fully active state after the MixRows transformation. Since AddRoundKey does not change the difference, we get a fully active state at the input of SubBytes of round r_3 (S_2). Then, we start with another difference in 8 active bytes at the output of MixRows of round r_3 (S_3^{MR}) and propagate backwards. Again, since we have an active byte in each row, we get a fully active state at the output of SubBytes of round r_3 .

In the second step of the inbound phase, the match-in-the-middle step, we look for a matching input/output difference of the SubBytes layer of round r_3 . This is done as described in Sect. 3.2 with a precomputed 256×256 lookup table. Note that we can repeat the inbound phase at most about 2^{128} times. As indicated in Sect. 3.2, we expect one solution per trial, that is, we can produce at most 2^{128} actual values that follow the differential trail in the inbound phase.

Outbound Phase. In contrast to the inbound phase, we use truncated differentials in the outbound phase of the attack. By propagating the matching differences and state values through the next SubBytes layer outwards, we get a truncated differential in 8 active bytes in both backward and forward direction. These truncated differentials need to propagate from 8 to 1 active byte through the MixRows transformation, both in the backward and forward direction (see Fig. 2). The propagation of truncated differentials through the MixRows transformation can be modelled in a probabilistic way, see Sect. 3.2. Since we need to fulfill one $8 \rightarrow 1$ transitions in the backward and forward direction, the probability of the outbound phase is $2^{-2 \cdot 56} = 2^{-112}$. In other words, we have to repeat the inbound phase about 2^{112} times to generate 2^{112} starting points for the outbound phase of the attack.

3.4 Previous Results on Round-Reduced Whirlpool

Extending the 4 round trail in both, the inbound and outbound phase, leads to attacks on round reduced Whirlpool for up to 7.5 (out of 10) rounds (where 0.5 rounds consist only of SubBytes and ShiftColumns). To be more precise, by

extending the outbound phase of the attack by 0.5 and 2.5 rounds, one can construct a collision and near-collision for the Whirlpool hash function reduced to 4.5 and 6.5 rounds, respectively. The collision attack has a complexity of about 2^{120} and the near-collision attack has a complexity of about 2^{128} . Furthermore, by additionally extending the inbound phase of the attack by 1 round, one can find a collision and a near-collision for the compression function of Whirlpool reduced to 5.5 and 7.5 rounds with a complexity of 2^{120} and 2^{128} , respectively. Note that adding this round in the inbound phase is possible, since in a compression function attack, one can use the degrees of freedom of the key schedule (chaining value) to guarantee that the trail in the inbound phase holds. All results are summarized in Table 1 and for more details on these results we refer to [16].

4 Improved Rebound Attack on the Whirlpool Compression Function

In this section, we improve the inbound phase of the original rebound attack on Whirlpool. By using a new differential trail and extensively using the available degrees of freedom of the key schedule, we can add 2 additional rounds to the inbound phase of the attacks. The basic idea is to have two instead of one inbound phase (match-in-the-middle step) and connect them using the available degrees of freedom from the key schedule. The outbound phase of the attacks is identical as in the previous attacks on 5.5 and 7.5 rounds for the compression function of Whirlpool. As a result, we obtain a collision and a near-collision attack for the compression function of Whirlpool reduced 7.5 and 9.5 rounds, respectively.

4.1 Inbound Phase

In this section, we describe the improved inbound phase of the attack in detail. We use the following sequence of active bytes:

$$8 \xrightarrow{r_1} 64 \xrightarrow{r_2} 8 \xrightarrow{r_3} 8 \xrightarrow{r_4} 64 \xrightarrow{r_5} 8$$

In order to find inputs following the differential of the inbound phase, we split it into two parts. In the first part, we apply the match-in-the-middle step with active bytes $8 \rightarrow 64 \rightarrow 8$ twice in rounds 1-2 and 4-5. In the second part, we need to connect the resulting 8 active bytes and 64 (byte) values of the state between round 2 and 4 using the degrees of freedom we have in the choice of the round key values (see Fig. 3).

Inbound Part 1. In this part of the inbound phase, we apply the match-in-the-middle step twice for rounds 1-2 and 4-5 (see Fig. 3), which can be summarized as follows:

1. Precomputation: For the S-box, compute a 256×256 lookup table as described in Sect. 3.2.

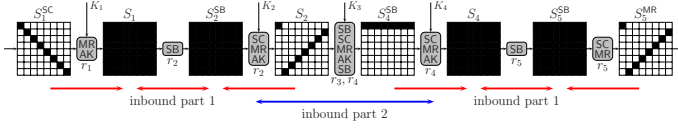


Fig. 3. The inbound phase of the attack

2. Match-in-the-middle (rounds 1-2):

- (a) Start with 8 active bytes at the output of **AddRoundKey** in round r_2 (S_2) and propagate backward to the output of **SubBytes** in round r_2 (S_2^{SB}).
- (b) Start with 8 active bytes at the input of **MixRows** in round r_1 (S_1^{SC}) and propagate forward to the input of **SubBytes** in round r_2 (S_1). Note that we can compute forward and solve the following step for each row independently.
- (c) Connect the input and output of the S-boxes of round r_2 by choosing the actual values of the state S_1 , respectively S_2^{SB} , using the lookup table generated in the precomputation step. After repeating step (b) for each row about 2^8 times we expect to find a match for the 8 S-boxes and thus 2^8 actual values (see Sect. 3.2). Since we do this for all rows independently, we get about 2^{64} actual values for the full state S_1 , respectively S_2^{SB} , such that the trail holds.

3. Match-in-the-middle (rounds 4-5): Do the same as in Step 2.

Hence, we get 2^{64} candidates for S_2^{SB} and 2^{64} candidates for S_4 after the first part of the inbound phase of the attack with a complexity of about 2^9 round transformations.

Inbound Part 2. In the second part of the inbound phase, we have to connect the 8 active bytes (64 (bit) conditions) as well as the actual values (512 (bit) conditions) of S_2^{SB} and S_4 by choosing the subkeys K_2 , K_3 and K_4 accordingly. Therefore, we have to solve the following equation:

$$\text{MR}(\text{SC}(\text{SB}(\text{MR}(\text{SC}(\text{SB}(\text{MR}(\text{SC}(S_2^{SB})) \oplus K_2))) \oplus K_3))) \oplus K_4 = S_4 \quad (5)$$

with

$$\begin{aligned} K_3 &= \text{MR}(\text{SC}(\text{SB}(K_2))) \oplus C_3 \\ K_4 &= \text{MR}(\text{SC}(\text{SB}(K_3))) \oplus C_4. \end{aligned} \quad (6)$$

Since we have 2^{64} candidates for S_2^{SB} , 2^{64} candidates for S_4 and 2^{512} candidates for the 3 subkeys K_2 , K_3 , K_4 (because of (6)), we expect to find 2^{64} solutions.

Since $S_2^{MR} = \text{MR}(\text{SC}(S_2^{SB}))$, we can rewrite the above equation as follows:

$$\text{MR}(\text{SC}(\text{SB}(\text{MR}(\text{SC}(\text{SB}(S_2^{MR} \oplus K_2))) \oplus K_3))) \oplus K_4 = S_4 \quad (7)$$

Note that one can always change the order of **SC** and **SB** in the Whirlpool block cipher without affecting the output of one round. In order to make the

subsequent description of the attack easier, we do this here and get the following equation.

$$\text{MR}(\text{SC}(\text{SB}(\text{MR}(\text{SB}(\text{SC}(S_2^{\text{MR}} \oplus K_2)))) \oplus K_3)) \oplus K_4 = S_4 \tag{8}$$

Furthermore, MR and SC are linear transformations and hence we can rewrite the above equation as follows:

$$\text{SB}(\text{MR}(\text{SB}(S_2^* \oplus K_2^*)) \oplus K_3) \oplus K_4^{\text{SB}} = X \tag{9}$$

with $S_2^* = \text{SC}(S_2^{\text{MR}})$, $K_2^* = \text{SC}(K_2)$, $K_4^{\text{SB}} = \text{SB}(K_4)$, $X = \text{SC}^{-1}(\text{MR}^{-1}(S_4 \oplus C_4))$.

In the following, this equivalent description is used to connect the values and differences of the two states S_2^{MR} and S_4 .

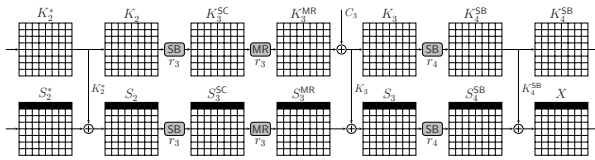


Fig. 4. The second part of the inbound phase. Black state bytes are active.

Remember that the two 8-byte differences of S_2^* and X have already been fixed due to the previous steps. Furthermore, we can choose from 2^{64} values for each of the states S_2^* and X . Now, we use equation (9) to determine the subkey K_2^* such that we get a solution for the inbound phase of the attack. Note that we can solve (9) for each row of the equation independently (see Fig. 4). It can be summarized as follows.

1. Compute the 8-byte difference and the 2^{64} values of the state S_2^* from S_2^{SB} , and compute the 8-byte difference and the 2^{64} values of the state X from S_4 . Note that we can compute and store the values of S_2^* and X row-by-row and independently. Hence, both the complexity and memory requirements for this step are 2^8 instead of 2^{64} .
2. Repeat the following steps for all 2^{64} values of the first row of S_2 to get 2^{64} matches for S_2^* to S_4 :
 - (a) For the chosen value of the first row of S_2 , forward compute the differences and values to the first row of S_3 .
 - (b) Choose the first row of the key K_3 such that the differential of the S-box between S_3 and S_4^{SB} holds.
 - (c) Compute the first row of K_2^* , S_2^* , K_4^{SB} and X . Since we have 2^{64} values for the first row of S_2^* and 2^{64} values for the first row of X , we expect to find a match on both sides. In other words, we have now connected the values and differences of the first row.
 - (d) Next, we connect the values of rows 2-8 independently by a simple brute-force search over all 2^{64} corresponding key values of K_2^* . Since we have to connect 64 bit values and we test 2^{64} key values we expect to always find a solution.

In total, we get 2^{64} matches connecting state S_2^* to state X with a complexity of 2^{128} and memory requirements of 2^8 . In other words, with the values of S_2^* , X and the corresponding key K_2^* , we get 2^{64} starting points for the outbound phase of the attack. Hence, the average complexity to find one starting point for the outbound phase is 2^{64} . It is important to note that one can construct a total of 2^{192} starting points in the inbound phase to be used in the outbound phase of the attack.

Note that step 2 (d) can be implemented using a precomputed lookup table of size 2^{128} . In this lookup table each row of the key K_2 (64 bits) is saved for the corresponding two rows of S_2^* and X (64 bits each). Using this lookup table, we can find one starting point for the outbound phase with an average complexity of 1. However, the complexity to generate this lookup table is 2^{128} .

4.2 Outbound Phase

In the outbound phase of the attack, we further extend the differential path backward and forward. By propagating the matching differences and state values through the next SubBytes layer, we get a truncated differential in 8 active bytes for each direction. These truncated differentials need to follow a specific active byte pattern to result in a collision on 7.5 rounds and a near-collision on 9.5 rounds, respectively. In the following, we will describe the outbound phase for the collision and near-collision attack in detail.

Collision for 7.5 Rounds. By adding 1 round in the beginning and 1.5 rounds at the end of the trail, we get a collision for 7.5 rounds for the compression function of Whirlpool. In the attack, we use the following sequence of active bytes:

$$1 \xrightarrow{r_1} 8 \xrightarrow{r_2} 64 \xrightarrow{r_3} 8 \xrightarrow{r_4} 8 \xrightarrow{r_5} 64 \xrightarrow{r_6} 8 \xrightarrow{r_7} 1 \xrightarrow{r_{7.5}} 1$$

As described in Sect. 3.2, the propagation of truncated differentials through the MixRows transformation is modelled in a probabilistic way. For the differential trail to hold, we need that the truncated differentials in the outbound phase propagate from 8 to 1 active byte through the MixRows transformation, both in the backward and forward direction (see Fig. 5). Since the transition from 8 active bytes to 1 active byte through the MixRows transformation has a probability of about 2^{-56} , the probability of this part of the outbound phase is $2^{-2 \cdot 56} = 2^{-112}$. Furthermore, to construct a collision at the output (after the feed-forward), the exact value of the input and output difference has to match. Since only one byte is active (see Fig. 5), this can be fulfilled with a probability

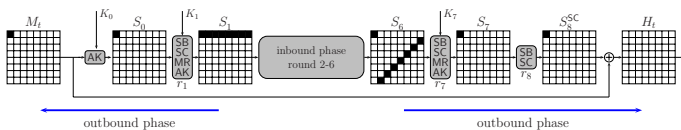


Fig. 5. Differential trail for collision attack on 7.5 rounds

of 2^{-8} . Hence, the probability of the outbound phase is $2^{-112} \cdot 2^{-8} = 2^{-120}$. In other words, we have to generate 2^{120} starting points (for the outbound phase) in the inbound phase of the attack to find a collision for the compression function of Whirlpool reduced to 7.5 rounds.

Since we can find one starting point with an average complexity of about 2^{64} and memory requirements of 2^8 , we can find a collision with a complexity of about $2^{120+64} = 2^{184}$. The complexity of the attack can be further improved on the cost of higher memory requirements. By using a lookup table with 2^{128} entries (generated in a precomputation step), we can find one starting point for the inbound phase with an average complexity of 1. In other words, we can find a collision for the compression function reduced to 7.5 rounds with a complexity of about 2^{120} . However, the precomputation step (constructing the lookup table) has a complexity of about 2^{128} .

Near-Collision for 9.5 Rounds. The collision attack on 7.5 rounds for the compression function can be further extended by adding one round at the beginning and one round at the end of the trail in the outbound phase. The result is a near-collision attack on 9.5 rounds for the compression function of Whirlpool with the following sequence of active bytes:

$$8 \xrightarrow{r_1} 1 \xrightarrow{r_2} 8 \xrightarrow{r_3} 64 \xrightarrow{r_4} 8 \xrightarrow{r_5} 8 \xrightarrow{r_6} 64 \xrightarrow{r_7} 8 \xrightarrow{r_8} 1 \xrightarrow{r_9} 8 \xrightarrow{r_{9.5}} 8$$

Since the 1-byte difference at the beginning and end of the 7.5 round trail will always result in 8 active bytes after one MixRows transformation (see Sect. 3.2), we can go backward 1 round and forward 1 round with no additional cost. Using the feed-forward, the position of two active S-boxes match and cancel each other with a probability of 2^{-16} . Hence, we get a collision in 50 and 52 bytes for the compression function of Whirlpool with a complexity of about 2^{176} and $2^{176+16} = 2^{192}$, respectively. With a precomputation step with complexity of 2^{128} and similar memory requirement, one can find a near-collision for the compression function of Whirlpool with a complexity of about 2^{112} (collision in 50 bytes) and 2^{128} (collision in 52 bytes), respectively.

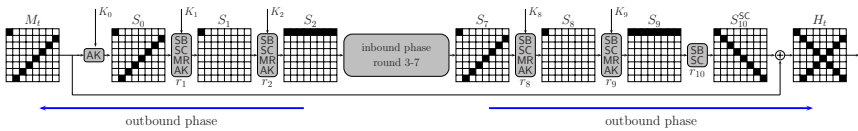


Fig. 6. In the attack on 9.5 rounds we extend the trail one more round at the beginning and at the end of the outbound phase to get a near-collision of Whirlpool

5 A Subspace Distinguisher for 10 Rounds

In this section, we present the first cryptanalytic result on the full Whirlpool compression function. The method for extending the previous result on 9.5 rounds is extended to full 10 rounds of the compression function by defining

a different attack scenario. Instead of aiming for a near-collision, we are interested in distinguishing the Whirlpool compression function from a random function. For this, we will introduce a new kind of distinguishing attack, a so called *subspace distinguisher*. In the following, $\mathbb{F}_2 = GF(2)$ always denotes the finite field of order 2.

For the subspace distinguishing attack, we consider the following problem:

Problem 1. *Given a function f mapping to \mathbb{F}_2^N , try to find t input pairs such that the corresponding output differences belong to a vector space of dimension at most n for some $n \leq N$.*

Remark. We define Problem [1](#) in this generic way in order to make it more generally applicable. This will be shown in the extended version of this paper.

5.1 Solving Problem [1](#) for the Whirlpool Compression Function

In this section, we show how the compression function attack described in Sect. [4](#) can be used to distinguish the full Whirlpool compression function from a random function.

Obviously, the difference between two Whirlpool states can be seen as a vector in the vector space of dimension $N = 512$ over \mathbb{F}_2 . The crucial observation is that the attack of Sect. [4](#) can be interpreted as an algorithm that can find t difference vectors in \mathbb{F}_2^{512} (output differences of the compression function) that form a vector space of dimension $n \leq 128$.

To see this, observe that by extending the differential trail from 9.5 to 10 rounds, the 8 active bytes in S_{10}^{SC} will always result in a fully active state S_{10} due to the properties of the MixRows transformation. Thus the near-collision is destroyed. However, if we look again at Fig. [6](#), the differences in M_t and the differences in S_{10}^{SC} can be seen as (difference) vectors belonging to subspaces of \mathbb{F}_2^{512} of dimension at most 64.

Even though after the application of MixRows and AddRoundKey the state S_{10} is fully active in terms of truncated differentials, the differences in S_{10} still belong to a subspace of \mathbb{F}_2^{512} of dimension at most 64 due to the properties of MixRows. Therefore, after the feed-forward, we can conclude that the differences at the output of the compression function form a subspace of \mathbb{F}_2^{512} of dimension $n \leq 128$.

Hence, we can use the attack of Sect. [4](#) to find t difference vectors forming a vector space of dimension $n \leq 128$ with a complexity of $t \cdot 2^{176}$ or $t \cdot 2^{112}$ using a precomputation step with complexity 2^{128} . Note that $t \leq 2^{192-112} = 2^{80}$ due to the remaining degrees of freedom in the inbound phase of the attack.

Now the main question is for which values of t our attack is more efficient than a generic attack. In other words, how do we have to choose t such that we can distinguish the compression function of Whirlpool from a random function. Therefore, we first have to bound the complexity of the generic attack. This is described in the next section.

5.2 Solving Problem [1](#) for a Random Function

Remarks on the Security Model. In order to discuss generic attack scenarios, we will have to choose a security model. We will adopt the black box

model introduced by Shannon [23]. In this model, a block cipher can be seen as a family of functions parameterized by the secret key $k \in \mathcal{K}$, that is, $E : \{0, 1\}^{|k|} \times \{0, 1\}^N \mapsto \{0, 1\}^N$, where for each $k \in \mathcal{K}$, E_k is seen as a uniformly chosen random permutation on $\{0, 1\}^N$.

In [3] it was shown, that an ideal block cipher based hash function in the Miyaguchi-Preneel mode is collision resistant and non-invertible. Based on this, we model our compression function f as black box oracle to which only forward queries are admissible. We also want to note that in all of the following, when we are talking about complexity, we are talking about *query complexity*. Note that the practical complexity is always greater or equal to the query complexity.

The Generic Approach. In this generic approach the only property used about f is the fact that the outputs of f are contained in the vector space \mathbb{F}_2^N .

Let us now assume that an adversary is making Q queries to the function f . Assuming that $Q \ll 2^{N/2}$, we thus get $K = \binom{Q}{2}$ differences ($\in \mathbb{F}_2^N$) coming from these Q queries. For given n and $t \gg n$, we now want to calculate the probability that among these K difference vectors, we have t vectors that span a space of dimension less or equal to n .

We will need the following fact about matrices over finite fields. Let $E(t, N, d)$ denote the number of $t \times N$ matrices over \mathbb{F}_2 that have rank equal to d . Then, it is well known (cf. [9] or [13]) that

$$E(t, N, d) = \prod_{i=0}^{d-1} (2^N - 2^i) \cdot \binom{t}{d}_2 = \prod_{i=0}^{d-1} \frac{(2^N - 2^i) \cdot (2^t - 2^i)}{2^d - 2^i}, \tag{10}$$

where $\binom{t}{d}_2$ denotes the q -binomial coefficient with $q = 2$.

Proposition 1. *Let $n, t, N \in \mathbb{N}$ be given such that $t \gg N > n$. We assume a set of K vectors chosen uniformly at random from \mathbb{F}_2^N . Let $\Pr(K, t, N, n)$ denote the probability that t of these K vectors span a space of dimension not larger than n . Then, we have*

$$\Pr(K, t, N, n) \leq \binom{K}{t} 2^{-t \cdot N} \sum_{d=1}^n E(t, N, d) \tag{11}$$

$$\leq \frac{1}{\sqrt{2\pi t}} \left(\frac{Ke}{t} \right)^t 2^{-(N-n)(t-n)-(n-1)}. \tag{12}$$

Proof. Based on the definition of $E(t, N, d)$, it is easy to see that (11) is an upper bound for $\Pr(K, t, N, n)$.

Computing the second bound consists of two steps. Bounding the binomial coefficient and bounding the rest. We get

$$2^{-t \cdot N} \sum_{d=1}^n E(t, N, d) \leq 2^{-t \cdot N} \cdot 2 \cdot E(t, N, n) \tag{13}$$

$$\leq 2^{-t \cdot N + 1} \left(\frac{(2^t - 2^{n-1}) \cdot (2^N - 2^{n-1})}{2^n - 2^{n-1}} \right)^n \tag{14}$$

$$\leq 2^{-t \cdot N+1} \left(2^{n-1} \cdot 2^{t-(n-1)} \cdot 2^{N-(n-1)} \right)^n \quad (15)$$

$$= 2^{-(t-n)(N-n)-(n-1)}. \quad (16)$$

These inequalities are based on two facts. First, it is easy to show that for $t \gg N > n$, we have $E(t, N, n) \leq \sum_{d=1}^n E(t, N, d) \leq 2 \cdot E(t, N, n)$. This can be proven by using induction over n and elementary properties of the q -binomial coefficient. Second, (14) follows from the fact that the function defined by $f(x) = (2^t - x)(2^N - x)/(2^n - x)$ is strictly increasing on the interval $x \in [0, 2^{n-1}]$.

For the binomial coefficient $\binom{K}{t}$ we combine the simple estimate $\binom{K}{t} \leq K^t/t!$ with the following inequality based on Stirling's formula [22]:

$$\sqrt{2\pi t} t^{t+\frac{1}{2}} e^{-t+\frac{1}{12t+1}} < t! < \sqrt{2\pi t} t^{t+\frac{1}{2}} e^{-t+\frac{1}{12t}} \quad (17)$$

From this we get $\binom{K}{t} \leq \frac{1}{\sqrt{2\pi t}} \left(\frac{K \cdot e}{t}\right)^t$ and with (16), this proves the proposition. ■

As a corollary, we can give a lower bound for the number of random vectors needed to fulfill the conditions of the proposition with a certain probability.

Corollary 1. *For a given probability p and given N, n, t as in Proposition 1, the number K of random vectors needed to contain t vectors spanning a space of dimension not larger than n with a probability p is lower bounded by*

$$K \geq \frac{1}{e} \left(p\sqrt{2\pi t} \right)^{\frac{1}{t}} \cdot t \cdot 2^{\frac{(N-n)(t-n)+(n-1)}{t}}. \quad (18)$$

and the number of queries Q to f needed to produce t vectors spanning a space of dimension not larger than n with a probability p is lower bounded by

$$Q \geq \sqrt{\frac{2}{e}} \left(p\sqrt{2\pi t} \right)^{\frac{1}{2t}} \cdot \sqrt{t} \cdot 2^{\frac{(N-n)(t-n)+(n-1)}{2t}}. \quad (19)$$

Proof. Equation (18) follows immediately from (12) and (19) follows from setting $K = \binom{Q}{2} = Q(Q-1)/2$ in (18). ■

5.3 Complexity of the Distinguishing Attack

Table 2 shows the complexities of the generic approach and our dedicated approach for several values of t . As can be seen in the table, one can distinguish the full Whirlpool compression function from random with a complexity of about 2^{188} with $t = 2^{12}$ (or 2^{121} with $t = 2^9$ using a precomputation table). In other words, when performing 2^{188} queries to a random function (19) shows that the probability for solving Problem 1 for $t = 2^{12}$ is $\ll 1$. To the best of our knowledge this is the first result on the full Whirlpool compression function.

Table 2. Values for t , Q (query complexity), C (complexity of our attack), and C_p (complexity of our attack with precomputation) for $p = 1, N = 512, n = 128$

$\log_2(t)$	$\log_2(Q)$	$\log_2(C)$	$\log_2(C_p)$	$\log_2(t)$	$\log_2(Q)$	$\log_2(C)$	$\log_2(C_p)$
9	148.41	185	121	13	195.29	189	125
10	172.84	186	122	14	197.28	190	126
11	185.31	187	123	15	198.53	191	127
12	191.80	188	124	16	199.40	192	128

6 Conclusion

In this paper, we have proposed a new kind of distinguishing attack for cryptanalysis of hash functions. We have successfully attacked the Whirlpool compression function. To the best of our knowledge this is the first attack on full Whirlpool.

We have obtained this result by improving the rebound attack on reduced Whirlpool. First, the inbound phase of the rebound attack was extended by up to two rounds using the available degrees of freedom from the key schedule. This resulted in a near-collision attack on 9.5 rounds of the compression function of Whirlpool. Second, we have shown how to turn this rebound near-collision attack into a distinguishing attack for the full 10 round compression function of Whirlpool.

The idea seems applicable to a wider range of hash function constructions. In particular, the attacks described in this paper can be applied to the hash function Maelstrom [8] in a straight forward manner because of the similarity to Whirlpool (see also [16]). Several SHA-3 candidates are a natural target for this new kind of attack, see for instance [14,15]. Furthermore, subspace distinguishers can be applied to block ciphers as well. This will be discussed in an extended version of this paper.

Acknowledgements

The authors wish to thank the anonymous referees for useful comments and discussions. The work in this paper has been supported in part by the European Commission under contract ICT-2007-216646 (ECRYPT II) and in part by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy).

References

1. Barreto, P.S.L.M., Rijmen, V.: The WHIRLPOOL Hashing Function. Submitted to NESSIE (September 2000), <http://www.larc.usp.br/~pbarreto/WhirlpoolPage.html> (2008/12/11) (revised May 2003)
2. 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)

3. Black, J., Rogaway, P., Shrimpton, T.: Black-Box Analysis of the Block-Cipher-Based Hash-Function Constructions from PGV. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 320–335. Springer, Heidelberg (2002)
4. Daemen, J., Rijmen, V.: The Wide Trail Design Strategy. In: Honary, B. (ed.) Cryptography and Coding 2001. LNCS, vol. 2260, pp. 222–238. Springer, Heidelberg (2001)
5. Daemen, J., Rijmen, V.: The Design of Rijndael. Information Security and Cryptography. Springer, Heidelberg (2002), ISBN 3-540-42580-2
6. De Cannière, C., Mendel, F., Rechberger, C.: Collisions for 70-Step SHA-1: On the Full Cost of Collision Search. In: Adams, C.M., Miri, A., Wiener, M.J. (eds.) SAC 2007. LNCS, vol. 4876, pp. 56–73. Springer, Heidelberg (2007)
7. 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)
8. Filho, D.G., Barreto, P.S., Rijmen, V.: The Maelstrom-0 hash function. In: SBSeg 2006 (2006)
9. Fisher, S.D.: Classroom Notes: Matrices over a Finite Field. Amer. Math. Monthly 73(6), 639–641 (1966)
10. Knudsen, L.R.: Truncated and Higher Order Differentials. In: Preneel, B. (ed.) FSE 1994. LNCS, vol. 1008, pp. 196–211. Springer, Heidelberg (1995)
11. Knudsen, L.R.: Non-random properties of reduced-round Whirlpool. NESSIE public report, NES/DOC/UIB/WP5/017/1 (2002)
12. Knudsen, L.R., Rijmen, V.: Known-key distinguishers for some block ciphers. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 315–324. Springer, Heidelberg (2007)
13. Lidl, R., Niederreiter, H.: Finite Fields, Encyclopedia of Mathematics and its Applications, 2nd edn., vol. 20. Cambridge University Press, Cambridge (1997); with a foreword by P. M. Cohn
14. 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)
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.) FSE 2009. LNCS, vol. 5665, pp. 260–276. Springer, Heidelberg (2009)
17. 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)
18. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press, Boca Raton (1997), <http://www.cacr.math.uwaterloo.ca/hac/>
19. National Institute of Standards and Technology: FIPS PUB 197, Advanced Encryption Standard (AES). Federal Information Processing Standards Publication 197, U.S. Department of Commerce (November 2001)
20. NESSIE: New European Schemes for Signatures, Integrity, and Encryption. IST-1999-12324, <http://cryptonessie.org/>
21. Peyrin, T.: Cryptanalysis of Grindahl. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 551–567. Springer, Heidelberg (2007)

22. Robbins, H.: A remark on Stirling’s formula. *Amer. Math. Monthly* 62, 26–29 (1955)

23. Shannon, C.E.: *Communication Theory of Secrecy Systems*. Bell Systems Technical Journal 28, 656–715 (1949)

24. 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)

25. 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)

26. Wang, X., Yu, H.: How to Break MD5 and Other Hash Functions. In: Cramer, R. (ed.) *EUROCRYPT 2005*. LNCS, vol. 3494, pp. 19–35. Springer, Heidelberg (2005)

A Attacks on the Hash Function

In this section, we present a collision and near-collision for the Whirlpool hash function. The attacks are a straight forward extension of the collision and near-collision attack on 4.5 and 6.5 rounds of Whirlpool presented in [16]. By adding one round in the inbound phase we can find a collision and a near-collision for Whirlpool reduced to 5.5 and 7.5 rounds, respectively. The core of the attack is a 5 round differential trail, where two fully active states are placed in the middle:

$$1 \xrightarrow{r_1} 8 \xrightarrow{r_2} 64 \xrightarrow{r_3} 64 \xrightarrow{r_4} 8 \xrightarrow{r_5} 1$$

Since the outbound phase of the attacks is identical to the previous attacks (see Sect. 4), we only discuss the inbound phase of the attack here (see Fig. 7).

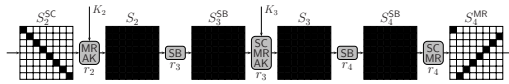


Fig. 7. The inbound phase of the collision attack and near-collision attack on the hash function

It can be summarized as follows.

1. Precomputation: For the S-box, compute a 256×256 lookup table as described in Sect. 3.2
2. Start with 8 active bytes (differences) at the input of MixRows in round r_2 (S_2^{SC}) and propagate forward to the input of SubBytes in round r_3 (S_2).
3. Start with 8 active bytes at the output of MixRows in round r_4 (S_4^{MR}) and propagate backward to the output of SubBytes in round r_4 (S_4^{SB}).
4. Next we have to connect the states S_2 and S_4^{SB} such that the differential trail holds. In other words, we have to find the actual values for S_2 such that:

$$SB(MR(SC(SB(S_2))) \oplus K_3) \oplus SB(MR(SC(SB(S_2 \oplus \Delta_1))) \oplus K_3) = \Delta_2$$

where Δ_1 denotes the active bytes (differences) in S_2 and Δ_2 denotes the active bytes (differences) in S_4^{SB} . In the following, we will show how this equation can be solved with a complexity of about 2^{64} by solving the equation for sets of 8 bytes independently. It can be summarized as follows.

- (a) For all 2^{64} values of $S_2[0, 0], S_2[1, 7], \dots, S_2[7, 1]$ compute the first row of S_4^{SB} and check if the above equation holds. Note that due to `ShiftColumns`, these bytes are shifted to the first row of S_3^{SC} and `MixRows` works on each row independently. In other words, we get 2^{64} candidates for each row of S_4^{SB} . Hence, after testing all 2^{64} candidates for the first row of S_4^{SB} we expect to find a match for the first row of Δ_2 .
- (b) Do the same for the corresponding 8 bytes for row 2-8 of S_4^{SB} . After testing each set of 8 bytes independently, we will find a state S_2 such that the differential trail is connected. Finishing this step of the attack has a complexity of about $8 \cdot 2^{64} \text{ MixRows}$ ($\approx 2^{64}$ round computations).

Hence, we can compute one starting point for the outbound phase with a complexity of about 2^{64} . Note that the complexity of the inbound phase can be significantly reduced at the cost of higher memory requirements. By saving 2^{64-s} candidates for S_4^{SB} in a list, we can do a standard time/memory tradeoff with a complexity of about 2^{120+s} and memory requirements of 2^{64-s} . By setting $s = 0$ we can find 2^{64} starting points with a complexity of 2^{64} and similar memory requirements of 2^{64} .

Hence, we can find a collision for Whirlpool reduced to 5.5 rounds with a complexity of about 2^{120} and a near-collision for 7.5 rounds in 50 (respectively 52) bytes with a complexity of about 2^{120} and 2^{112} (respectively 2^{128}). All attacks have memory requirements of 2^{64} .

MD5 Is Weaker Than Weak: Attacks on Concatenated Combiners

Florian Mendel, Christian Rechberger, and Martin Schl affer

Institute for Applied Information Processing and Communications (IAIK)
Graz University of Technology, Inffeldgasse 16a, A-8010 Graz, Austria
`christian.rechberger@iaik.tugraz.at`

Abstract. We consider a long standing problem in cryptanalysis: attacks on hash function combiners. In this paper, we propose the first attack that allows collision attacks on combiners with a runtime below the birthday-bound of the *smaller* compression function. This answers an open question by Joux posed in 2004.

As a concrete example we give such an attack on combiners with the widely used hash function MD5. The cryptanalytic technique we use combines a partial birthday phase with a differential inside-out technique, and may be of independent interest. This potentially reduces the effort for a collision attack on a combiner like MD5||SHA-1 for the first time.

Keywords: hash functions, cryptanalysis, MD5, combiner, differential.

1 Introduction

The recent spur of cryptanalytic results on popular hash functions like MD5 and SHA-1 [28,30,31] suggests that they are (much) weaker than originally anticipated, especially with respect to collision resistance. It seems non-trivial to propose a concrete hash function which inspires long term confidence. Even more so as we seem unable to construct collision resistant primitives from potentially simpler primitives [27]. Hence constructions that allow to hedge bets, like concatenated combiners, are of great interest. Before we give a preview of our results in the following, we will first review work on combiners.

Review of work on combiners. The goal of combiners is to have at least some bound on the expected security even if (some of the) hash functions get broken, for various definitions of “security” and “broken”. Joux [12] showed (by using multi-collisions) that the collision resistance of a combiner can not be expected to be much higher than the birthday bound of the component (=hash function) with the largest output size.

On the other hand, combiners seem to be very robust when it comes to collision security up to the birthday bound (of the component with the smallest output size): By using techniques similar to Coron *et al.* [3], Hoch and Shamir [11] showed that only very mild assumptions on a compression function are needed to

achieve a collision resistance of at least $O(2^{n/2})$. In fact, using a model proposed by Liskov [15], they show that none of the compression functions need to be collision, nor preimage resistant in the usual sense.

Motivation: cryptanalysis of combiners. Concatenating the output of hash function is often used by implementors to “hedge bets” on hash functions. A combiner of the form MD5||SHA-1 as used in SSL 3.0/TLS 1.0 and TLS 1.1 [78] is an example of such a strategy. Let’s assume we are given a combiner of the form MD5||SHA-1. Let’s further assume that a breakthrough in cryptanalysis of SHA-1 brings down the complexity of a collision search attack to 2^{52} . We know that the best collision search attacks on MD5 are as fast as 2^{15} [29]. So what is the best collision attack on the combiner? The best known method due to Joux is only as good as a birthday attack on the smaller of the two hash functions in the combiner. There is no known method which would allow to reduce the total effort below this bound, *i.e.* 2^{64} :

Currently, the best solution at our disposal is to combine the (hypothetic) SHA-1 attack with Joux’s multicollision approach. Find a 2^{64} -multicollision for SHA-1 with effort $2^{52} \cdot 64 = 2^{58}$, and then perform a birthday-type search in this 2^{64} collision to single out a collision which also collides for MD5. The total effort will be 2^{64} . In fact, reductions of the effort for SHA-1 collision search will only marginally improve the attack on the combiner. How to improve upon this? Analyzing the combiner as a whole may be prohibitively complicated. The resistance of two-pipe designs with sufficiently different pipes like RIPEMD-160 [10] against recent collision search attacks also gives hints in this direction.

Preview of our results: We propose a new method that allows a cryptanalyst to focus on the hash functions individually while still potentially allowing attacks on combiners with a runtime below the birthday-bound of the smaller compression function. This also answers an open question by Joux posed in 2004 [12]. For this, we start with definitions in Section 2. In Section 3 we give a high-level description of our attack strategy on a concatenation combiner without going into the details of a particular compression function. Next, we consider as a concrete cryptanalytic example combiners that use MD5. We first give an alternative description of MD5 in Section 4 which will turn out to be beneficial (and in fact as our experiments suggest necessary) in Section 5, where we describe the cryptanalytic techniques we need, to be able to use the high-level attack description.

For the cryptanalysis, we employ a combination of a birthday-style attack and a differential inside-out technique that uses different parts of a collision characteristic at different stages of an attack, both before and after a birthday phase. The differential technique may be of independent interest, also for improving known types of collision attacks on MD5, or for finding one-block collisions. In Section 6 we give practical results which allow us to estimate the actual security MD5 is able to give in a combiner. Finally, we conclude and discuss open problem in Section 7.

2 Definitions

In the remainder of the paper we give a few definitions. We give a classification of collision attacks on compression functions and hash functions. Let an iterated hash function F be built by iterating a compression function $f : \{0, 1\}^m \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ as follows:

- Split the message M of arbitrary length into k blocks x_i of size m .
- Set h_0 to a pre-specified IV
- Compute $\forall x_i : h_i = f(h_{i-1}, x_i)$
- Output $F(M) = h_k$

Classification for compression function collision attacks. Higher numbers mean less degrees of freedom for an attacker and are hence more difficult to obtain cryptanalytically.

- **Compression collision attacks of type 0**
Compute h_{i-1} , h_{i-1}^* , m_i and m_i^* s. t. $f(h_{i-1}, m_i) = f(h_{i-1}^*, m_i^*)$. Note that early attacks by den Boer and Bosselaers [11], and Dobbertin [9] on MD5 are of this type.
- **Compression collision attacks of type 1**
Given h_{i-1} , compute m_i and m_i^* s. t. $f(h_{i-1}, m_i) = f(h_{i-1}^*, m_i^*)$.
- **Compression collision attacks of type 2**
Given h_{i-1} and h_{i-1}^* , compute m_i and m_i^* s. t. $f(h_{i-1}, m_i) = f(h_{i-1}^*, m_i^*)$
- **Compression collision attacks of type 3**
Given h_{i-1} and h_{i-1}^* , compute m_i s. t. $f(h_{i-1}, m_i) = f(h_{i-1}^*, m_i)$

Later in the paper, it will be useful to have a weakened version of the collision attack on the compression function of type 3.

- **Compression collision attacks of type 3w**
Given h_{i-1} and h_{i-1}^* from an efficiently enumerable subset s (of size $|s| = 2^{n-z}$) of all 2^{2n} possible pairs (h_{i-1}, h_{i-1}^*) , compute m_i s. t. $f(h_{i-1}, m_i) = f(h_{i-1}^*, m_i)$.

Complementing types 1-3 of the compression function attacks, one may define similar attack settings for the hash function as well. For sake of concreteness, we also give examples related to MD5.

- **Hash collision attacks of type 1:** Given m_0 , compute m_1 and m_1^* such that $F(m_0||m_1) = F(m_0||m_1^*)$. This is the most simple way to violate the collision resistance of a hash function. For MD5, see Wang *et al.* [31]. The prefix m_0 may be the string of length 0, or any other message block.
- **Hash collision attacks of type 2:** Given m_0 and m_0^* , compute m_1 and m_1^* such that $F(m_0||m_1) = F(m_0^*||m_1^*)$. This type of attack is much more demanding from a cryptanalytic view as it needs to cope with arbitrary prefixes and hence arbitrary chaining input differences (Stevens *et al.* [28]). In turn it allows much more powerful attacks, as can be seen by the recent attacks on certificate authorities using MD5 [29].

- **Hash collision attacks of type 3 (new, in this paper):** Given m_0 and m_0^* , compute m_1 such that $F(m_0||m_1) = F(m_0^*||m_1)$. This type of attack is in turn much more difficult than type 2, as it halves the degrees of freedom available to an attacker. The message difference is fixed (to zero), this means that for each MD5 compression function, instead of 1024 degrees of freedom, only 512 degrees of freedom via the message input are available to an attacker.

This leads us to the informal definition of a weak hash function, complementing the concept of a weak compression function from [15]. A weak hash function may be modeled as a random oracle, but offers additionally oracles that allow collision attacks on the hash function of type 1 and type 2, but not of type 3. The purpose of this introduction of a weak hash function is to show that MD5 can not even meet the requirements of a weak hash function, even though no type 3 collision attack on the MD5 compression function are known.

We may define the security of a hash function as a component in a concatenated combiner against collision attacks (concatenated combiner collision security, or simply C^3 security) of an n -bit hash function as the effort to find a collision attack of type 3. For MD5, despite all cryptanalytic advances in recent years, this is 2^{64} . In this paper, we show an attack suggesting that the C^3 security of MD5 is less.

3 Outline of Attack Strategies

In the following we assume it is possible to devise collision attacks of type 3w on the compression function below the birthday bound. These collision attacks will need a suitable differential path, and a method to find message pair which conforms to such a differential path. We will discuss this problem for the case of MD5 in Section 5. This alone is not enough for our attack to work, but based on such a result we propose to continue as follows. We first show how to devise a collision attack of type 3 on a hash function using a combination of birthday techniques and differential shortcut techniques. Then we continue and apply such an attack on a combiner.

3.1 Collision Attack of Type 3

The attack we propose (see Fig. 1 for an illustration) consists of three phases. A preparation phase that computes target differences (1), a birthday phase (using M_1) (2) and a differential phase (using M_2) that performs a type 3w collision attack (3), and is executed in this order.

Before the birthday phase (2), the differential phase needs to be “prepared” as follows (1). We generate a number of 2^x distinct characteristics (also called paths) through the compression function on a heap with the following property: no message difference, an arbitrary input difference (δ_2), and no output difference ($\delta_2 \boxplus \delta_3 = 0$). Let’s assume each of them, when given a suitable chaining input pair, results in an effort of 2^w (or less) to find a conforming message pair. Let 2^y

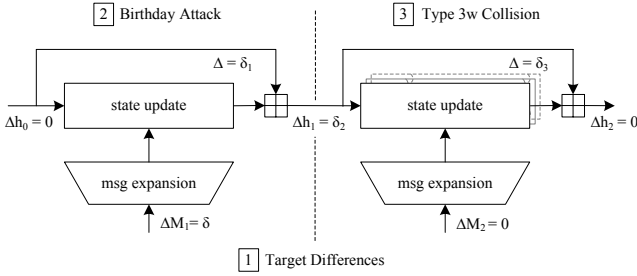


Fig. 1. Outline of attack strategy

be the cost of this path generation in terms of equivalent compression function computations. Let’s further assume that each of these paths has an average number of z independent conditions on the chaining input (CI).

A single path with z conditions on the CI in fact can be used for 2^{n-z} possible pairs of CIs. Since there exist 2^{2n} pairs, 2^{n+z} randomly generated pairs would be needed before one matches the CI described by the path (δ_1 matches δ_2 , and the conditions are fulfilled). Using birthday techniques, this is expected to take $2^{(n+z)/2}$ time. Given all 2^x paths, only 2^{n+x-z} randomly generated pairs are needed, which in turn is expected to take $2^{(n-x+z)/2}$ time. Hence, if $x > z$, the runtime is expected to be below the birthday bound.

For obtaining a single hash collision of type 3, the overall method may be seen as a successful cryptanalytic attack, if the sum of the runtimes for the path generation, the birthday phase, and the work to find a conforming message pair using a particular path is below the birthday bound, *i.e.* if $2^y + 2^{(n-x+z)/2} + 2^w < 2^{n/2}$. For obtaining many hash collisions of type 3, the effort to generate the heap of paths (1) may be negligible, hence to goal would be reduced to $2^{(n-x+z)/2} + 2^w < 2^{n/2}$.

3.2 Attack on the Combiner $F_1(M)||F_2(M)$

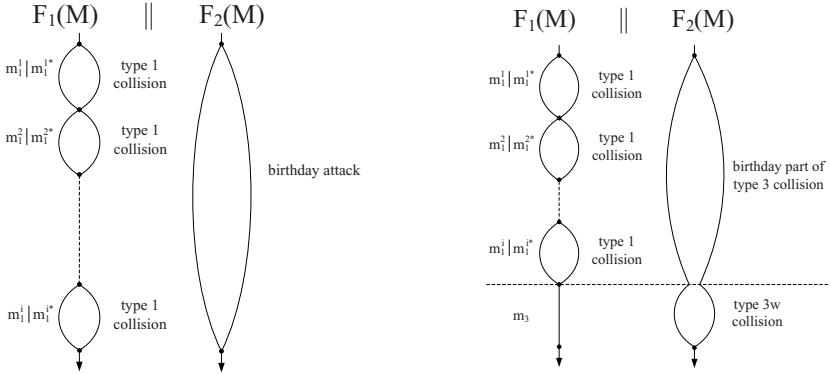
We now discuss how to use a type 3 collision attack on a hash to devise an attack on a combiner of two hash functions using it, where the first of two hash functions suffers from a type 1 collision attack.

The setting: Let $F_1(\cdot)$ and $F_2(\cdot)$ be two hash functions with output size n_1 and n_2 . For the sake of simplicity we assume in the following that $n_1 = n_2 = n$. Let’s further assume that F_1 suffers from a type 1 collision attack, *i.e.* given m_0 , let the effort to find a m_1 and m_1^* such that $F_1(m_0||m_1) = F_1(m_0||m_1^*)$ be $2^{c_1} < 2^{n/2}$. Furthermore, assume that F_2 suffers from a type 3 collision attack, *i.e.* given m_2 and m_2^* , compute m_3 such that $F_2(m_2||m_3) = F_2(m_2^*||m_3)$ be $2^{c_2} < 2^{n/2}$. In more detail, as noted above, 2^{n+z-x} randomly generated pairs (m_2, m_2^*) are needed. The introduced symbols are summarized in Table 1.

We are now ready to formulate the new collision attack on the combiner $F_1(M)||F_2(M)$ that combines both attacks. It is also illustrated in Fig. 2.

Table 1. Symbols used in the description of the attack

symbol	description
n	output size
w	\log_2 of the cost of finding a conforming message pair
x	\log_2 of the number of distinct characteristics
y	\log_2 of the cost of the preparatory path generation
z	number of conditions on the chaining input
c_1	\log_2 of the cost of type 1 collision attack on the first hash function
c_2	\log_2 of the cost of type 3 collision attack on the second hash function



(a) The known approach due to Joux does not allow to exploit shortcut collision attacks on both hash functions. The lower bound is hence a birthday attack on the “smaller” hash function.

(b) New collision construction using type 3 collisions allows to exploit shortcuts attacks in both hash functions without considering the interaction in the cryptanalysis.

Fig. 2. Comparison of collision attack on a combiner

1. Let m_0 be the string of size zero and perform the type 1 collision attack on F_1 and obtain a (m_1^1, m_1^{1*}) such that $F_1(m_1^1) = F_1(m_1^{1*})$. Note that $F_2(m_1^1)$ does not collide with $F_2(m_1^{1*})$.
2. Repeat the step above while replacing m_0 with the concatenation of all previously found messages $(n+z-x)/2 - 1$ times. This means, for the i -th step (for $i = 2 \dots (n+z-x)/2$), let $m_0 = m_1^1 || \dots || m_1^i$ and obtain a (m_1^i, m_1^{i*}) such that $F_1(m_1^i) = F_1(m_1^{i*})$.
3. Note that by using Joux’s multicollision method, we have produced a $2^{(n+z-x)/2}$ -collision for F_1 .
4. Perform the type 3 attack of F_2 as follows. For the birthday-part of the type 3 attacks, use the $(n+z-x)/2$ collisions in F_1 to obtain the required 2^{n+z-x} pairs of prefixes m_2 and m_2^* .
5. Continue with the differential shortcut part of the type 3 attack as outlined in the previous subsection, *i.e.* find a suffix m_3 such that there is a collision between

$$F_2(m_1^1 || m_1^2 || \dots || m_1^{(n+z-x)/2} || m_3)$$

and

$$F_2(m_1^{1*} || m_1^{2*} || \dots || m_1^{((n+z-x)/2)*} || m_3).$$

6. Also, the collision in F_1 remains.

$$F_1(m_1^1 || m_1^2 || \dots || m_1^{(n+z-x)/2} || m_3)$$

collides with

$$F_1(m_1^{1*} || m_1^{2*} || \dots || m_1^{((n+z-x)/2)*} || m_3),$$

as after the multicollision the message block m_3 without a difference is added.

7. As the same message constitutes a collision for both F_1 and F_2 , this in turn results in a collision for the combiner.

The computational complexity of this procedure is as follows. The type 1 collision search on F_1 in step 1 is repeated $(n+z-x)/2$ times, which sums up to an effort of $(n+z-x)/2 \cdot 2^{c_1}$. Afterwards the type 3 collision search in F_2 is performed using the obtained multicollision. This consists of a birthday part and a type 3w compression function attack, in total costing 2^{c_2} computations. Hence, the total complexity is $(n+z-x) \cdot 2^{c_1-1} + 2^{c_2}$, and reusing the calculation for c_2 from Section 3 we arrive at

$$(n+z-x) \cdot 2^{c_1-1} + 2^y + 2^{(n-x+z)/2} + 2^w. \quad (1)$$

4 Alternative Description of MD5

MD5 is an iterative hash function based on the Merkle-Damg ard design principle [4,19]. It processes 512-bit input message blocks and produces a 128-bit hash value. If the message length is not a multiple of 512, an unambiguous padding method is applied. For the description of the padding method we refer to [24]. The design of MD5 is similar to the design principles of MD4 [23]. In the following, we briefly describe the compression function of MD5. It basically consists of two parts: message expansion and state update transformation. A detailed description of the MD5 hash function is given in [24].

4.1 Message Expansion

The message expansion of MD5 is a permutation of the 16 message words m_i in each round. For each of the four rounds, a permutation of these 16 message words is used, resulting in 64 32-bit words, denoted by W_i , with $0 \leq i \leq 63$. For the permutation defining the ordering of message words we refer to [24].

4.2 State Update Transformation

The state update transformation of MD5 starts from a (fixed) initial value IV ($A_{-4}, A_{-3}, A_{-2}, A_{-1}$) of four 32-bit registers and updates them in 4 rounds of 16 steps each. The state update transformation of MD5 works on four state

variables. The state update transformation can be written to update one variable only:

$$A_i = A_{i-1} + (A_{i-4} + f(A_{i-1}, A_{i-2}, A_{i-3}) + W_i + K_i) \lll s_i.$$

However, in our case it turned out that a description which updates 2 state variables A_i and B_i is beneficial. In this case, one step is computed as follows (see also Fig. 3):

$$\begin{aligned} B_i &= (A_{i-4} + f(A_{i-1}, A_{i-2}, A_{i-3}) + W_i + K_i) \lll s_i \\ A_i &= A_{i-1} + B_i. \end{aligned}$$

In each step of MD5, different step constants K_i , rotation values s_i and Boolean functions f are used. For the definition of the constants and the rotation values we refer to [24]. The Boolean function f differs for each round of MD5: IF is used in the first round, IF3 is used in round 2, and XOR is used in round 3 and ONX is used in the last round:

$$\begin{aligned} \text{IF}(x, y, z) &= xy \oplus \neg xz \\ \text{IF3}(x, y, z) &= zx \oplus \neg zy \\ \text{XOR}(x, y, z) &= x \oplus y \oplus z \\ \text{ONX}(x, y, z) &= y \oplus (x \vee \neg z) \end{aligned}$$

After the last step of the state update transformation, the initial value and the output values of the last four step are combined, resulting in the final value of one iteration known as Davies-Meyer hash construction (feed forward). The result is the final hash value or the initial value for the next message block.

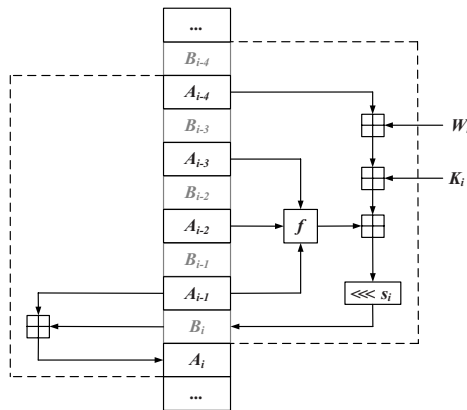


Fig. 3. Alternative description of the step update transformation of MD5 using two state variables A_i and B_i

5 Path Search Technique for MD5 Type 3 Collisions

We now tackle the problem of finding collision attacks on the compression function of MD5 of type 3w. Various automated path search techniques for MD4-like hash functions have been proposed in the past. In this section, we describe the new path search technique we developed to solve the problem. In fact it can be seen as a variation of the fine grained condition propagation originally proposed in [6].

5.1 Overview

As illustrated in Fig. 4, the MSB-path of [1] is a building block of our technique. Starting from this MSB-path in the middle of the compression function we will study and search for many characteristics which propagate through the ONX round in the forward direction, and through the IF round in the backward direction in a non-linear way. The constraint is that, despite different rotation values and Boolean functions, resulting differences in both ends of the state update will cancel out after the feed-forward operation.

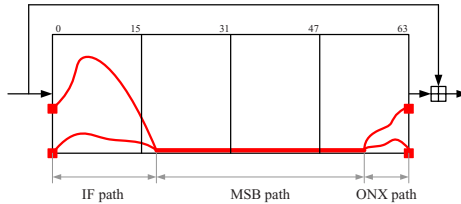


Fig. 4. The outline of the type 3w collision search with IF-path, MSB-path and ONX path

5.2 Reviewing the Path Search of De Canni ere/Rechberger

In 2006, De Canni ere/Rechberger [6] propose the concept of generalized conditions. The generalized conditions on a particular pair of words will be denoted by ∇X . ∇X represents as a set the values for which the conditions are satisfied. In order to write this in a compact way, we will reuse the notation listed in Table 2.

In [6], the authors describe a heuristic method to find complex nonlinear characteristics for SHA-1 in an efficient way. Follow-up work directly applied this method in various settings in the context of SHA-0 and SHA-1 [5][13][16][32]. The approach may be described as follows.

1. The starting point is a number of constraints (on the message difference and some target differences in the state) for the characteristic.
2. The basic idea of the algorithm is to randomly pick a bit position which is not restricted yet (*i.e.*, a ‘?’-bit), impose a zero-difference at this position (a ‘-’-bit), and calculate how the condition propagates. This is repeated until all unrestricted bits have been eliminated, or until it runs into an inconsistency, in which case it starts again.

Table 2. Notation for generalized conditions, possible conditions on a pair of bits. The right half is for completeness only, and will not be used in the paper.

(x_i, x_i^*)	(0, 0)	(1, 0)	(0, 1)	(1, 1)	(x_i, x_i^*)	(0, 0)	(1, 0)	(0, 1)	(1, 1)
?	✓	✓	✓	✓	3	✓	✓	-	-
-	✓	-	-	✓	5	✓	-	✓	-
x	-	✓	✓	-	7	✓	✓	✓	-
0	✓	-	-	-	A	-	✓	-	✓
u	-	✓	-	-	B	✓	✓	-	✓
n	-	-	✓	-	C	-	-	✓	✓
1	-	-	-	✓	D	✓	-	✓	✓
#	-	-	-	-	E	-	✓	✓	✓

3. The basic idea was improved by also sometimes picking ‘x’-bits once they start to appear, guessing the sign of their differences (‘u’ or ‘n’), and doing a backtracking if this does not lead to a solution.

5.3 The Path Search for MD5

We found that a direct mapping of this strategy to the case of MD5 did not lead to satisfactory results. It was not possible, with significant computational resources, to find a non-linear characteristic for the given setting. There are two main reasons for this difficulty. The first problem is caused by having two modular additions (separated by a rotation operation) within one state update. Fig. 3 shows the iterative step function of MD5 with variables A_i and B_i . Hence, two different carry expansions may occur and by guessing only bits of the state A_i , conditions propagate slowly and contradictions are detected at a very late stage. Table 3 shows an example with many free (‘?’) bits in B_i due to guessing bits only on A_i .

The second problem are the reduced starting constraints with only a few bit differences set in the chaining input. In the case of the type 3w collision search, there are no input difference in the message and only very few differences in the chaining input and at the chaining output. By guessing even more zero-differences (‘-’-bits), the found characteristics tend to get very sparse. In fact, these sparse characteristics are impossible, which is not detected early enough by the path search algorithm. Hence, most of the time is spent with paths whose impossibility should be detected earlier. An example for a sparse (in state variable A_i), but impossible characteristic is given in Table 3.

To avoid these problems, the new MD5 path search strategy works as follows:

1. The starting point are only a small number of constraints (the chaining input difference, no message difference and the MSB path) for the characteristic.
2. Instead of just picking bits of A_i , randomly pick non-restricted bits of the state B_i as well.
3. *Immediately* guess the sign of any unrestricted difference (‘x’-bits), as soon as it occurs and do a backtracking if the guess leads to a contradiction.
4. If all ‘x’-bits have been determined, continue with randomly guessing zero-differences until the next ‘x’-bit occurs.

Table 3. A sparse but impossible characteristic due to guessing too many zero-differences in A_i . Further, conditions do not quickly propagate into B_i and contradictions are detected at a very late stage.

i	$\nabla B_i, \nabla A_i$	∇W_i
-4	A: n-----u-----	
-3	A: n0-----B?-----n-----	
-2	A: n1-----n-?-?-?-?n-----	
-1	A: un-----0?x-----?x-----	
	B: ??????????????????-----?x-----	
0	A: -n-----0-----Du-----	W: -----
	B: ????????	
1	A: 7n-x-?0-----Du-----0-----	W: -----
	B: ???????x-----????x-----	
2	A: -B-??-n-----	W: -----
	B: ???x-----	
3	A: --D?E?#u-----0-----	W: -----
	B: -----	
4	A: -----n-----	W: -----
	B: ???????x-----	
5	A: -----	W: -----
	B: -----	
6	A: 0-----0-----	W: -----
	B: -----#-----	
7	A: 0-----	W: -----#-----
	B: x-----	
8	A: n-----	W: -----

Whenever a contradiction occurs, a simple backtracking strategy (depth first search) is applied. Using this improved strategy, global contradictions (impossible characteristics) are found at an earlier stage and impossible paths are less likely. The disadvantage of this strategy is that long carry expansions are more likely to occur and the resulting characteristic are less sparse. However, since we apply the path search mostly in the first round of MD5, even a high number of conditions can be fulfilled using simple message modification techniques [31].

6 Practical Realization and Results

We now describe implementations of several parts of the attack. This illustrates and details the method, and also serves as a validity check of the attack. To recapitulate our earlier description, the practical implementation of a type 3 collision is divided into three steps:

- **Preparatory phase.** Many special paths are searched and put on a heap.
- **Birthday phase.** Looking through possible pairs of prefixes, a pair needs to be found that matches one of the paths on the heap.
- **Differential attack phase.** Search for a conforming message pair using one of the characteristics generated earlier.

An optimization that is important in practice, is as follows. Starting from the MSB path in the middle of the MD5 compression function, it suffices to compute many paths through the last round (ONX part). The last steps of this path will impose conditions (of type 'n' and 'u') on the chaining input. This information is enough for the birthday phase. The result of the birthday phase is a prefix

pair that is compatible with a particular path on the heap. It remains to finish the characteristic, the IF part, to connect to the MSB part in the middle (see Fig. 4 for an illustration of the different parts). Having to deal with an actual chaining input pair in this phase of the attack imposes more constraints on the path search. However, as we detail in Section 6.1 and also illustrate with the characteristic in the table in Appendix A, these constraints can be dealt with in practice and do not impose any limitation on the attack.

6.1 Runtime for IF Path Search

In experiments involving the equivalent of about 2000 hours on a single core, we have verified the average runtime to find a single IF path is about 36 hours on a single core, which is about 2^{17} seconds in which about 2^{38} MD5 computations¹ could be done. For these experiments, we not only generated paths for a particular starting point, as the choice of a particular starting point has unpredictable consequences for a particular heuristic (this was also observed in [6]). Instead we generated many (about 30) starting points (*i.e.* different sets of conditions on the chaining input) in a random way to derive meaningful average runtime estimates. This suggests that, using the proposed strategy, we can expect to find a path for every set of constraints, albeit with somewhat varying runtime. In turn, this allows us to estimate the workfactor for a type 3w collision attack on MD5.

We found that the runtime for the search for IF-path does not depend on the number of differences in the CV². The generation of the corresponding IF-paths can be delayed until after the birthday phase, contributes to the final search complexity only in an additive way, and is hence negligible.

6.2 A Type 3 Collision Attack Based on Actually Generated Paths

For the practical generation of type 3w collision attacks on the compression function of MD5, that in turn lead to a type 3 collision attack on the MD5 hash, we constrain ourselves to differential paths which result in runtimes for finding a colliding message pair below 2^{58} . For the preparatory step, it suffices to generate useful ONX paths. An ONX path is useful if it has a high probability, as the probability of a collision characteristic in the last round affects the resulting effort for finding a conforming message pair in a direct way. In order to give a bound on the allowable probability for the ONX path, we argue as follows. Among the four rounds (consisting of 16 steps each) the first round can easily be dealt with via simple message modification. The second round is an MSB-path and contains 16 conditions (the Boolean function needs to behave as expected at every step once, see also [1]), the third round contains no conditions as the Boolean function is an XOR, and the fourth round contains the more complex ONX-path. Improvements upon the original type 1 collision attack on MD5 by

¹ Each of our 2.0 GHz AMD Opteron(tm) cores performs about 2^{21} MD5 computations per second using OpenSSL 0.9.8g.

² We tested a range between 1 and 20.

Wang *et al.* concentrated on fulfilling more conditions in round 2. In a work from 2005 [25], 14 conditions could already be fulfilled. Subsequent work by Klima [14] and Stevens *et al.* [29] significantly improved upon this. Conservatively assuming to be able to only fulfill 14 conditions suggests that round number four should not have more than $58 - 16 + 14 = 56$ conditions. In Section 6.4 we give several reasons why this is a very conservative assumption.

Another important parameter of ONX paths is the number and position of differences it has in the last four steps, as this determines (except for carries via the feed-forward operation) the uniqueness of the set of allowed pairs of chaining inputs that can be canceled.

Inhere, we report on empirical findings using an actual implementation of parts of the attack. In total we spent an equivalent of about 15000 hours on a single core. The number of distinct paths for type 3w compression function attacks on MD5 we found together with their number of conditions on the IV is as follows:

number of conditions on IV	1	2	3	4	5	6	7	8	9	10
number of paths	0	0	10	130	1216	6556	21523	49293	87116	127018

Not all found paths may be of use. Let p_i be the number of distinct paths with i conditions on the IV, we want to find a j such that $(\sum_{i=1}^j p_i) - 2^j$ is maximal. Using the actually generated paths as described above, we found about $2^{17.34}$ paths with distinct constraints (with at most 9 relevant conditions) on the chaining input. Including also all found paths with 10 conditions would only improve the attack only if more than $2^{17.34}$ paths would be added, which is not the case.

Using the notation of Section 3, this means $x=17.3$, $w < 58$, and $z \leq 8$. Based on this, a type 3 collision has a runtime of $2^{(128-17.34+9)/2} (+2^{58}) = 2^{60.19}$, which is faster than the expected 2^{64} for an ideal hash function of this size. Hence, MD5 offers a C^3 security of no more than 60 bits.

Note however, that in this calculation, there is a gross imbalance between time spent on generating paths (15000 CPU hours are about 2^{47} MD5 computations) and the total runtime of the attack. Assuming to spend *e.g.* 2^7 times more computational resources in the path generation might well lead to an increase. from $x = 17$ to 24, which in turn would decrease the runtime of the overall type 3 collision attack on MD5 to 2^{57} , and would lead to an attack on the combiner MD5||SHA-1 with complexity less than 2^{59} (assuming the type 1 collision attack on SHA-1 is fast enough).

6.3 On Memory Requirements

Both, the generic method due to Joux and the new approach using a type 3 collision attack, can be implemented without requiring access to large memory. For both cases, this results in a runtime loss of about a factor $n/2$, hence the relative advantage of the new approach over the generic method remains. Memory requirements of the attack (birthday phase and differential shortcut phase) are as follows.

Birthday phase. A naive implementation of the birthday phase would require a table of size $2^{(n-x+z)/2}$ in order to generate enough pairs to find a match with one of the 2^x paths. However, distinguished point methods may be used on a truncated version of the output of the compression function.³

Let t be the size of the subset of bits that is needed to represent all 2^x paths. A lower bound for t is $2x/3$, since every bit that is truncated leaves three possibilities for a path ('n', 'u', or '-'). In practice, t is higher. A memory-less method will find a partially suitable pair in time $2^{(n-t)/2}$, which would need to be repeated $2^{(t-x+z)}$ times if done independently (and hence impose the additional condition $x - z > t/2$ on the attack to be more efficient than a generic attack).

However, as described in [21,22], the distinguished points method can be used to take advantage of the birthday effect also for generating more collisions (or suitable pairs), by keeping the entries in the list of each of the distinguished points. A parallelizable version with linear speed gain is described in [20]. Hence the search needs to be repeated only $2^{(t-x+z)/2}$ times. As a result, a “memory-less” version of the birthday phase for the dedicated combiner attack behaves to a large extent as a “memoryless” version of a generic birthday attack. What is needed is memory to store 2^z candidate pairs which are the outcome of the birthday phase. In all practical settings, z is small.

Differential shortcut phase. Storing the precomputed paths for the shortcut attacks: in the order of a kilobyte per path. For practical values of x between 10 and 20, storage costs are negligible and access to this memory is only needed once.

6.4 On Conservative Estimates

There are several reasons our estimates can be considered to be very conservative:

- Basing assumption on speed-up methods (message modification, tunnels) is very conservative for the following reason. The lack of message differences, and the very simple MSB path in round 2 gives more freedom to apply speed-up methods as is the case in type 1 collision search attacks in earlier work.
- Also, early stop methods which further speed-up collision search are not considered.
- Runtime of various path search scenarios are measurements of actual implementations, whose runtime may be optimized by some constant factor.
- For our calculations, we use the highest possible allowed value for w (worst case). The expected value is in fact lower.

7 Conclusions and Open Problems

We proposed a new attack that allows collision attacks on combiners with a runtime below the birthday-bound of the smaller compression function when

³ We will use the term “memoryless” to refer to these techniques, although they do in fact require some memory, albeit much less than a naive table-based approach.

the smaller compression function is MD5, potentially reducing a collision attack on a combiner like MD5||SHA-1 for the first time. This also answers an open question by Joux posed in 2004. The cryptanalytic technique we proposed for this is a combination of a birthday-style attack and a differential inside-out technique that uses different parts of a collision characteristic at different stages of an attack, both before and after a birthday phase. This technique may be of independent interest. Based on only the characteristics we generated in practical experiments with limited computational resources, a collision attack on the combiner with MD5 would already be around 2^{60} (if the “normal” collision attack on the other hash functions is fast enough), however we argued that such an estimate is *very conservative* for various reasons.

This illustrates that the MD5 hash function can not meet the requirements of a “weak hash function” as informally defined in this paper. Various open questions arise from this work: In a vein similar to concatenated combiners, or the Zipper construction [15], is it possible to come up with other collision resistant constructions that can use MD5, even though our results can be interpreted as showing that MD5 is “weaker than weak”? Another open problem is related to the application of our new cryptanalytic method to hash function constructions that use two or more parallel streams, like RIPEMD-160 [10], as well as several SHA-3 candidates⁴. So far it proved difficult to obtain results on RIPEMD-160, even for interesting reduced variants [17].

Acknowledgements. The authors wish to thank the anonymous referees for useful comments and discussions. The work in this paper has been supported in part by the European Commission under contract ICT-2007-216646 (ECRYPT II) and in part by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy).

References

1. den Boer, B., Bosselaers, A.: Collisions for the Compression Function of MD5. In: Helleseht, T. (ed.) EUROCRYPT 1993, vol. 765, pp. 293–304. Springer, Heidelberg (1994)
2. Brassard, G. (ed.): CRYPTO 1989. LNCS, vol. 435. Springer, Heidelberg (1990)
3. Coron, J.S., Dodis, Y., Malinaud, C., Puniya, P.: Merkle-Damg ard Revisited: How to Construct a Hash Function. In: Shoup [26], pp. 430–448
4. Damg ard, I.: A Design Principle for Hash Functions. In: Brassard [2], pp. 416–427
5. De Canni ere, C., Mendel, F., Rechberger, C.: Collisions for 70-Step SHA-1: On the Full Cost of Collision Search. In: Adams, C.M., Miri, A., Wiener, M.J. (eds.) SAC 2007. LNCS, vol. 4876, pp. 56–73. Springer, Heidelberg (2007)
6. De Canni ere, 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)
7. Dierks, T., Allen, C.: The TLS Protocol Version 1.0. IETF Request for Comments: 2246 (1999)

⁴ In fact we can already refer to a round-1 candidate of the SHA-3 competition which was broken [18] by techniques related to the new ideas presented in this paper.

8. Dierks, T., Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.1. IETF Request for Comments: 4346 (2006)
9. Dobbertin, H.: Cryptanalysis of MD5 Compress (1996)
10. Dobbertin, H., Bosselaers, A., Preneel, B.: RIPEMD-160: A Strengthened Version of RIPEMD. In: Gollmann, D. (ed.) FSE 1996. LNCS, vol. 1039, pp. 71–82. Springer, Heidelberg (1996)
11. Hoch, J.J., Shamir, A.: On the Strength of the Concatenated Hash Combiner When All the Hash Functions Are Weak. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfssdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 616–630. Springer, Heidelberg (2008)
12. Joux, A.: Multicollisions in Iterated Hash Functions. Application to Cascaded Constructions. In: Franklin, M. K. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 306–316. Springer, Heidelberg (2004)
13. Joux, A., Peyrin, T.: Hash Functions and the (Amplified) Boomerang Attack. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 244–263. Springer, Heidelberg (2007)
14. Klima, V.: Tunnels in hash functions: Md5 collisions within a minute. Cryptology ePrint Archive, Report 2006/105 (2006), <http://eprint.iacr.org/>
15. Liskov, M.: Constructing an Ideal Hash Function from Weak Ideal Compression Functions. In: Biham, E., Youssef, A.M. (eds.) SAC 2006. LNCS, vol. 4356, pp. 358–375. Springer, Heidelberg (2007)
16. Manuel, S., Peyrin, T.: Collisions on SHA-0 in One Hour. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 16–35. Springer, Heidelberg (2008)
17. Mendel, F., Pramstaller, N., Rechberger, C., Rijmen, V.: On the Collision Resistance of RIPEMD-160. In: Katsikas, S.K., López, J., Backes, M., Gritzalis, S., Preneel, B. (eds.) ISC 2006. LNCS, vol. 4176, pp. 101–116. Springer, Heidelberg (2006)
18. Mendel, F., Schl affer, M.: On Free-start Collisions and Collisions for TIB3. In: Proceedings of ISC, Springer, Heidelberg (2009)
19. Merkle, R.C.: One Way Hash Functions and DES. In: Brassard [2], pp. 428–446
20. van Oorschot, P.C., Wiener, M.J.: Parallel Collision Search with Cryptanalytic Applications. *J. Cryptology* 12(1), 1–28 (1999)
21. Quisquater, J.-J., Delescaille, J.-P.: How Easy Is Collision Search? Application to DES. In: Quisquater, J.-J., Vandewalle, J. (eds.) EUROCRYPT 1989. LNCS, vol. 434, pp. 429–434. Springer, Heidelberg (1990)
22. Quisquater, J.J., Delescaille, J.P.: How Easy is Collision Search. New Results and Applications to DES. In: Brassard [2], pp. 408–413
23. Rivest, R.L.: The MD4 Message Digest Algorithm. In: Menezes, A., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 303–311. Springer, Heidelberg (1991)
24. Rivest, R.L.: The MD5 Message-Digest Algorithm. IETF Request for Comments: 1321 (1992)
25. Sasaki, Y., Naito, Y., Kunihiko, N., Ohta, K.: Improved Collision Attack on MD5. Cryptology ePrint Archive, Report 2005/400 (2005), <http://eprint.iacr.org/>
26. Shoup, V. (ed.): CRYPTO 2005. LNCS, vol. 3621. Springer, Heidelberg (2005)
27. Simon, D.R.: Findings Collisions on a One-Way Street: Can Secure Hash Functions Be Based on General Assumptions? In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 334–345. Springer, Heidelberg (1998)
28. 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)

29. 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)
30. Wang, X., Yin, Y.L., Yu, H.: Finding Collisions in the Full SHA-1. In: Shoup [26], pp. 17–36
31. 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)
32. Yajima, J., Iwasaki, T., Naito, Y., Sasaki, Y., Shimoyama, T., Peyrin, T., Kunihiro, N., Ohta, K.: A Strict Evaluation on the Number of Conditions for SHA-1 Collision Search. IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences E92-A(1), 87–95 (2009)

A Supplementary Material for Obtained Results

A particular low-weight input chaining difference becomes the MSB-path in the course of 10 steps. The following table contains the full characteristics illustrating a candidates for a type 3w compression function attack. As a proof-of-concept, we provide a representative example of a conforming message pair in Table 4.

i	∇A_i	∇W_i
-4	A: u-----u-----	
-3	A: u-----	
-2	A: n-----0-----	
-1	A: 1-----0-----	
0	A: 0-----0u111110000000010-	W: -----
1	A: -----1u000000000000000-	W: -----
2	A: -----unnnnnnnnnnnnnnnn1	W: -----
3	A: --unnnnn-----011000101110000000	W: -----
4	A: --000000--n-u11u0000000110111-	W: -----1-----00001000111--
5	A: --011111--0-n11n0-----	W: -----0000-----1100----
6	A: -----nuuuuuu-----	W: -----
7	A: -----0110000-----	W: -----
8	A: 0-----1101101-----	W: -----
9	A: 0-----	W: -----
10	A: u-----	W: -----
11	A: n-----	W: -----
12	A: n-----	W: -----
13	A: n-----	W: -----

54	A: n-----	W: -----
55	A: n-----0-----	W: -----
56	A: u-----1-----	W: -----
57	A: n-----On-----	W: -----
58	A: u-----n-----	W: -----
59	A: n-----nu-----	W: -----
60	A: n-----n-----	W: -----
61	A: n-----11-----	W: -----
62	A: n-----0-----	W: -----
63	A: -----	W: -----
	FF: -----	
	FF: -----	
	FF: -----	
	FF: -----	

Table 4. A conforming message pair for the first 16 steps

H_1	C4F12702 D25873C9 5B88CE47 9A8EBB1D
H_2	44F12502 525873C9 DB88CE47 9A8EBB1D
ΔH_1	80000200 80000000 80000000 00000000
M_1	D830883A AA2456AA 24B9260C D2F17AE9 F893211E 08F4298C 8A0C7756 3492552F C7CB7D9D 7FB6804C 9336A183 44256E0D 6D095FCF 08D8D9EA 5D79C0BA 0F2CD7C5
M_2	D830883A AA2456AA 24B9260C D2F17AE9 F893211E 08F4298C 8A0C7756 3492552F C7CB7D9D 7FB6804C 9336A183 44256E0D 6D095FCF 08D8D9EA 5D79C0BA 0F2CD7C5
ΔM_1	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
H_3	156733C4 4A05644B 20E6A26E 7718EBA4
H_4	956733C4 CA05644B A0E6A26E F718EBA4
ΔH_2	80000000 80000000 80000000 80000000

The Intel AES Instructions Set and the SHA-3 Candidates

Ryad Benadjila¹, Olivier Billet¹, Shay Gueron^{2,3}, and Matt J.B. Robshaw¹

¹ Orange Labs, Issy les Moulineaux, France

{ryad.benadjila,olivier.billet,matt.robshaw}@orange-ftgroup.com

² University of Haifa, Israel

³ Intel Corporation, Haifa, Israel

shay.gueron@intel.com, shay@math.haifa.ac.il

Abstract. The search for SHA-3 is now well-underway and the 51 submissions accepted for the first round reflected a wide variety of design approaches. A significant number were built around Rijndael/AES-based operations and, in some cases, the AES round function itself. Many of the design teams pointed to the forthcoming Intel AES instructions set, to appear on Westmere chips during 2010, when making a variety of performance claims. In this paper we study, for the first time, the likely impact of the new AES instructions set on all the SHA-3 candidates that might benefit. As well as distinguishing between those algorithms that are AES-based and those that might be described as AES-inspired, we have developed optimised code for all the former. Since Westmere processors are not yet available, we have developed a novel software technique based on publicly available information that allows us to accurately emulate the performance of these algorithms on the currently available Nehalem processor. This gives us the most accurate insight to-date of the potential performance of SHA-3 candidates using the Intel AES instructions set.

1 Introduction

Intel has announced that a new AES instructions set¹ will be introduced in new processors such as Westmere and available early in 2010. These instructions will provide resistance to a range of software side-channel attacks [3,30] and offer significant performance benefits for encryption and decryption using AES [24]. Simultaneously the NIST SHA-3 effort [25] to establish a new cryptographic hash algorithm is well-underway and several teams of submitters have used AES-like transformations as a cryptographic building block. Several of these teams have explicitly expressed the assumption that their hashing algorithms could take advantage of AES-NI and thereby enjoy significant performance benefits. Since the Westmere processor is still unavailable, there have been no substantive efforts to assess the possible implications of this important issue. In this paper,

¹ Denoted AES-NI in this paper for “new instructions”.

we provide the first quantitative analysis that estimates the likely impact of the Intel AES instructions set on SHA-3 candidates.

The first step is to identify which SHA-3 candidates should be considered, and this is not as straightforward as it might appear. AES-NI can be used in different combinations to carry out different transformations, and so AES-NI might be used in many more ways than would naïvely be expected. As a result, there are submissions for which the variant that provides (say) 256-bit digests gains from AES-NI, while the same algorithm providing a 512-bit digest cannot.

The second step is to develop a sound methodology for implementing the different algorithms, optimising them, and measuring their performance. Clearly this is a challenge when Westmere processors are unavailable. So we developed new techniques from publicly available information—in effect, uncovering the behavior of AES-NI—and this allowed us to emulate Westmere behavior on the publicly-available Nehalem chips. While this might appear to detract from the value of the performance figures we derive, the level of validation and confirmation that took place during this work makes us confident that our results are close to the Westmere reality.

Our sole goal in this paper has been to compare the performance of SHA-3 candidates when using AES-NI. To this end, we have set aside cryptanalytic discussions [10] and we have implemented and optimised all the algorithms that we believe might benefit from AES-NI. While the authors of this paper are independent (co-)submitters of two SHA-3 proposals, we have strived to be fair and consistent. In addition, all the code is publicly available via [29] and we welcome interested parties to download and improve upon it. When Westmere processors appear, the same samples can be used for real silicon running AES-NI.

2 The Intel AES Instructions

To start we provide a brief description of the Intel AES instructions, and complete details can be found in [13,14]. Intel’s AES instructions set consists of six instructions, four of which `aesenc`, `aesenclast`, `aesdec`, and `aesdeclast` are designed to support data encryption and decryption. The names of these instructions are short for AES encryption (inner and last) round and AES decryption (inner and last) round, see Table 4 from Appendix A. These instructions have register/register and register/memory variants.

There are two other instructions for the AES key expansion but they seem to be of little use to the SHA-3 submissions and are omitted from this paper.

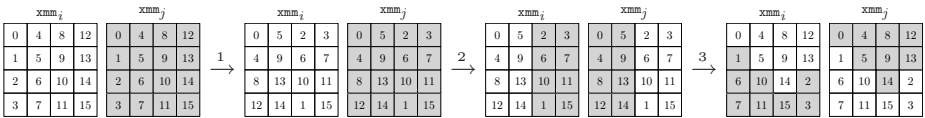
2.1 What Operations Can We Use AES-NI for?

Clearly, AES instructions can be used whenever a SHA-3 proposal uses one of the internal or final AES encryption (or decryption) rounds. But they can be used more widely than this. For instance, calling `aesdeclast` and `aesenc` back-to-back, both with a zeroed second operand, is functionally equivalent to performing AES MixColumns on the first operand, see Appendix A.

In fact if we use the `pshufb` instruction which shuffles bytes in a 128-bit word, see Appendix A, then we can isolate all of the AES-constituents using AES-NI [14], namely:

SubBytes , ShiftRows , MixColumns ,
 InvSubBytes , InvShiftRows , InvMixColumns .

To illustrate the versatility this gives us, we combine standard `xmm` instructions with AES-NI to perform encryption with Rijndael [8] operating on 256-bit blocks. The plaintext is stored in `xmmi` and `xmmj`, but AES-NI cannot be used directly since half the bytes of `xmmi` must be swapped with half the bytes of `xmmj`. However, this swap can be efficiently implemented using two `pshufb` (1) to pack the bytes to-be-swapped into two 32-bit words, two `pblendw` (2) to swap the 32-bit words, and two `pshufb` (3) to re-order the bytes giving, in total, the following state permutation:



After this, `aesenc` can be applied in parallel to `xmmi` and `xmmj`, thereby giving the appropriate `ShiftRows` for the large state, and Rijndael encryption on a larger state has been emulated. Techniques like these are important to us since it is possible that several SHA-3 candidates that do not use the complete AES round, or that use a larger state, might still benefit from AES-NI.

2.2 The “In-Scope” SHA-3 Candidates

Obviously SHA-3 candidates that use the AES round as a building block can benefit from using AES-NI. In addition, algorithms that use the AES S-box along with some byte shuffling with or without the AES MDS mixing matrix can benefit. One can also apply these operations to larger states, as we have seen for Rijndael with 256-bit blocks. The main problems in using AES-NI tend to arise when designs move away from the AES MDS matrix. Generally speaking, this dramatically limits any potential performance gain from AES-NI, particularly since most optimised assembly implementations would incorporate the MDS matrix operation into table look-ups, potentially combined with other operations. AES-NI might however still be of interest to these designs, especially in thwarting some side-channel attacks.

There are four submissions that directly, and transparently, use AES rounds for all hash output lengths. These are ECHO [2], LANE [18], SHAVITE-3 [4], and VORTEX [23]. For these algorithms it is clear that we can directly use AES-NI. There are others that are clearly inspired by Rijndael-like techniques in their construction. These include CHEETAH [22], FUGUE [15], GRØSTL [12], LESAMNTA [16], LUX [27], and TWISTER [11]. The submission SHAMATA [1] has already been withdrawn, and while some other surveys [5] describe SARMAL [31] as being AES-inspired, a non-AES S-box and MDS mixing layer take it out-of-scope.

Table 1. The SHA-3 submissions with substantial Rijndael-based components. Checkmarks indicate those that might benefit from AES-NI, for different hash output lengths.

<i>Algorithm</i>	<i>224-bit</i>	<i>256-bit</i>	<i>384-bit</i>	<i>512-bit</i>
ARIRANG	✓	✓	no	no
CHEETAH	✓	✓	no	no
ECHO	✓	✓	✓	✓
FUGUE	no	no	no	no
GRØSTL	no	no	no	no
LANE	✓	✓	✓	✓
LESAMNTA	✓	✓	✓	✓
LUX	✓	✓	no	no
SHAVITE-3	✓	✓	✓	✓
VORTEX	✓	✓	✓	✓

While LESAMNTA offers advantages for 256- and 512-bit hash outputs, it is interesting that only the 256-bit versions of CHEETAH and LUX benefit from AES-NI. By contrast, it appears that no variant of FUGUE, GRØSTL, or TWISTER are likely to benefit. These algorithms use a very different MDS mixing matrix to the AES and, as a result, end-up being too distant to use AES-NI in any efficient way. So even though a combination of AES-NI instructions could be used to isolate the S-box operations for FUGUE and GRØSTL, say, the table look-ups typically used for the MDS operations in current optimised implementations mean that there is no easy way for these algorithms to benefit from AES-NI.

Finally, even though the submission ARIRANG [6] is quite different from the Rijndael-based constructions, it might potentially benefit from AES-NI. We have therefore included it in our considerations and Table 1 summarizes the (alphabetically ordered) list of algorithms and hash output lengths that we consider.

3 Implementation and Measurements

Obviously the best way to get performance timings is to write the appropriate code, run it on a Westmere processor (the first with AES-NI), and measure the performance. However, since this processor is not yet available, we propose a new methodology that can be used to get an accurate emulation of AES-NI. We rely on the fact that Westmere (formerly Nehalem-C) and Nehalem processors share the same micro-architecture. This means that if we can find suitable instructions patterns that behave exactly as AES-NI instructions, we will get very good estimates for the future performance of AES-based SHA-3 candidates on a Westmere processor, but using today’s Nehalem processor.

Previously, a substitution instruction was proposed [23] for future processors. However this substitution does not exhibit the correct behaviour for Westmere and can give misleading results, see Section 3.1 and Appendix B. Here we provide a particularly accurate replacement instructions pattern for `aesenc` and we explain how to derive it from publicly available information only.

3.1 Replacement Instructions Pattern

The first step is to understand the exact behavior of the AES-NI instructions at the micro-operation (μop) level² in particular that of `aesenc` and `aesenclast`.

An Intel code analyzer tool (IACA [21]) is publicly available and gives the following information about `aesenc` (`aesdec` yields the same output):

Total Latency:		6 Cycles;		Total number of Uops:		3	
Num of Uops	0 - DV	1 2 - D	3 - D	4 5			
3	2				1	CP	aesenc xmm1, xmm0

(In this trace, ‘DV’ stands for the divider pipe of port 0, ‘D’ for the data fetch pipe of ports 2 and 3. Additionally, an ‘X’ in the trace will be used to denote the possible ports a μop can be dispatched to.)

This shows that `aesenc` consists of three μops , two of which are dispatched to a unit on port 0 and one which is dispatched to a unit on port 5, and that the instruction’s latency is 6 cycles. However, this information is too coarse to provide hints for the right instructions pattern replacement: we need to derive the exact scheduling of these μops . In what follows, we represent μops by bars for which the length varies according to their latency. The gray bars denote the μops on port 5 while the white ones denote the μops on port 0. Hence `▬` is a 2 cycle μop on port 5 and `▬▬` is a 3 cycle μop on port 0.

From Intel’s white paper [13] we know that AES-NI are highly parallelizable. This discards the sequential μop patterns on port 0. Moreover, the white paper explains (see Fig. 9 and 15) that `aesdec` is structured using the equivalent inverse cipher (described in Appendix B), which is confirmed by an IACA trace identical to that of `aesenc` displayed above (see Appendix B). This leads us to assume that the μop on port 5 is the exclusive-or with the key, which is corroborated by the purpose of unit 5, see [19]. Therefore, the μop on port 5 runs in cycle 6 and requires that μops from port 0 are finished.

Intel’s optimization reference manual [19] gives additional information on the possible μop latencies and throughput for each port on the Nehalem micro-architecture. In particular, we see that μops dispatched on port 0 can only have latencies 1, 4, or 5 cycles, and that μops on port 5 all have a 1 cycle latency. Since `aesenc` has a total latency of 6 cycles, this only leaves the following possible patterns: `▬▬▬▬▬`, `▬▬▬▬▬▬`, and `▬▬▬▬▬▬`. (Two μops cannot start at the same cycle in the same unit but a μop is started as soon as possible to maximize the overall throughput). It is impossible that a μop on port 0 performs the `SubBytes` and/or `ShiftRows` step while it runs in parallel with the other μop performing the `MixColumns` step which would then need the output of the first μop . So both μops on port 0 perform at least one of the four `MixColumn` multiplications of the `MixColumns` step. The most natural way of doing this is to symmetrically split the computation on two independent halves of the state. In this case, the two μops on port 0 have the same latency, which only leaves the `▬▬▬▬▬▬` pattern. This is again supported by the IACA trace of `aesimc` instruction, as well as the choice of inverse equivalent cipher for `aesdec`.

² Instructions are split into micro-operations and dispatched to specialized CPU units.

Now we turn to the replacement instructions set which would give exactly the same μop -behavior as the instruction `aesenc reg, reg`. A previously proposed replacement [23,17] is not appropriate for Westmere (see Appendix B). Instead, a sequence that closely simulates the μop behavior of `aesenc xmmi, xmmj` is:

```
movdqu xmmk, xmmi
mulps  xmmi, xmmj
mulps  xmmk, xmmj
xorps  xmmi, xmmk
```

For now, let us ignore the `movdqu` instruction. The IACA trace displayed below shows that the last three instructions of the replacement behave exactly as the `aesenc xmm0, xmm1` instruction with a latency of 6 cycles. It yields two identical and independent μops (they both come from `mulps`) on port 0, a 1 cycle μop on port 5 which is forced to start after the two μops on port 0 since `xorps` has a 1 cycle μop on port 0 together with a dependency on register `xmm2`:

Total Latency:		6 Cycles;		Total number of Uops: 4					
Num of Uops	0 - DV	Ports pressure in cycles							
		1	2 - D	3 - D	4	5			
1	X	1					X	CP	movdqu xmm2, xmm0
1	1							CP	mulps xmm0, xmm1
1	1							CP	mulps xmm2, xmm1
1							1	CP	xorps xmm0, xmm2

The reader might wonder why we added the `movdqu` instruction to the beginning of the replacement: by introducing a dependency on `xmm0`, we try to prevent the processor from re-ordering the instructions at the prefetch and re-order step. Hence, `movdqu` acts as a fence and ensures that the replacement fragment exhibits a similar atomic behavior as `aesenc`. Since `movdqu` only has a latency of one cycle and can be dispatched on port 0, 1, or 5, it will in most cases execute on port 1 in parallel of the other μops —and does not interfere with the replacement, and rarely on port 5 or 0 which would add one cycle to the replacement latency.

Note however, that though the replacement allows for a very good simulation of `aesenc` in terms of latency, throughput, and port behavior, it does introduce a significant issue: the use of a third register `xmmk` ($k = 2$ in IACA’s trace) might interfere with code surrounding the replacement by introducing false dependencies. We took extra care in our implementations to avoid these when using the replacement. This was not an easy task, especially for those SHA-3 candidates that make heavy use of AES-NI parallelism such as ECHO and LANE.

Another potential issue is that the `aesenc` instruction is 5 to 10 bytes long depending on the variant whereas our replacement is 13 to 22 bytes. This can lead to an efficiency penalty as the prefetch buffer of the Nehalem micro-architecture has a size of 16 bytes. However an experiment (see Appendix B) shows that the size of replacement is unlikely to be a significant factor.

Finally, we refer the reader to Appendix B for a justification of our choice of the following replacement for memory-based variants like `aesenc xmmi, [mem]`:

```
movdqu xmmk, xmmi
mulps  xmmi, [mem]
mulps  xmmk, xmmj
xorps  xmmi, xmmk
```

as well as for a discussion regarding replacements for other AES-NI instructions.

3.2 Timing Methodology

For each in-scope candidate and for each hash output length, we implemented two versions of the submission. These were identical in every way, except one had AES-NI instructions and was used to ensure the correctness of our AES-NI optimized implementation against the NIST-submitted test vectors with Intel’s Emulator [20]; the other had AES-NI instructions substituted with their replacements allowing it to run on a Nehalem to derive performance estimates.

To get consistent results over the candidates, we measured the number of cycles (using `rdtsc` instructions and averaging over more than 10^8 samples to get stable results) taken by the compression function of each algorithm on the same Nehalem machine running Linux. However NIST’s API was fully implemented to check correctness and, in many cases, these were taken from the reference code sent to NIST by the submitters. To eliminate as much noise as possible from the OS, high priority scheduling was allocated to the measured code. All algorithms were implemented by the same programmers, providing a somewhat uniform level of optimization.

4 Candidate Descriptions and AES-NI Implementations

In this section we consider the design and discuss the implementation of the in-scope candidates. Full details of the algorithms can be found in the respective algorithm descriptions, so we only give a brief overview of their functionality along with insights into their design with regards to AES-NI. Our implementation proposals will be available from our website [29].

ARIRANG is a single-pipe compression function-based proposal. The bulk of the computation in the compression function consists of the 40-step expansion of a 512-bit message block, which is highly efficient in general purpose registers and can be pre-computed, and a `StepFunction` that is repeated 40 times. `StepFunction` requires eight exclusive-ors, four fixed rotations, and two calls to a function G^{256} that uses elements of the AES. For longer hash outputs, the equivalent function G^{512} uses a larger MDS matrix that cannot be emulated using AES-NI, and so any potential gain is restricted to 256-bit outputs.

However, the extent of this gain is very limited since ARIRANG uses $\frac{1}{4}$ of an AES round as a building block, but the latency cost of `aesenc` while only performing $\frac{1}{4}$ of an AES round means that the performance of AES-NI, when compared to the use of lookup tables, is not competitive. Attempts to parallelize two of the $\frac{1}{4}$ AES rounds introduced too many overheads. We conclude that AES-NI is unlikely to offer any substantial benefits to ARIRANG.

CHEETAH is a single-pipe compression function-based proposal. The compression function consists of two strands of computation: a message-dependent EXPANDED BLOCK is generated which provides a key-like input to encrypt the INTERNAL STATE. While the computations on EXPANDED BLOCK and INTERNAL STATE are both Rijndael-inspired, the former uses a different non-AES MDS

matrix that is hard to emulate. Thus this key derivation is unlikely to benefit from AES-NI and the use of look-up tables seems better suited.

For operations on the INTERNAL STATE, the 224- and 256-bit versions of CHEETAH use an operation `InternalRound` that can be emulated using AES-NI. However, the inherent sequential nature of the rounds and the fact that AES-NI cannot be used in the most straightforward way means that while there are gains, they are not as significant as they might be for some other submissions.

For the 384- and 512-bit versions, the operation `InternalRound` is modified to use a larger MDS matrix that, once again, cannot exploit AES-NI. So for these larger outputs, there is unlikely to be any gain with AES-NI.

ECHO is a double-pipe compression-based hash function. The 224- and 256-bit (resp. 384- and 512-bit) versions encrypt a sixteen 128-bit words state in eight (resp. ten) rounds of a compression function calculation. The encryption round applies two AES rounds to each word of the state with a counter or salt as a key, followed by a `BIG.MixColumns` MDS and row shift operation that provides mixing across the entire state. For all hash output lengths, ECHO can benefit from AES-NI and, while ECHO is primarily a double-pipe compression-based hash function, a simple single-pipe variant was announced at the first NIST workshop. We therefore include it in our considerations.

The AES encryption rounds are directly performed with `aesenc` with pre-computed keys in memory. This allows the algorithm to take full advantage of the AES-NI parallelism. The `BIG.MixColumns` operation however cannot further benefit from AES-NI, though it is based on `MixColumns`. As an ECHO encryption round does not vary with the output length, the same optimizations apply.

LANE is a single-pipe compression function-based hash function. `COMPRESS` consists of a message expansion, a set of six P-PERMUTATIONS, and then a set of two Q-PERMUTATIONS. As both sets of permutations are based on the AES round, LANE benefits from AES-NI at all hash function output lengths.

Both PERMUTATIONS are made of $L = 2$ (resp. $L = 4$) lines of AES rounds for hash outputs of 256 (resp. 512) bits and after each round of AES in each line, an operation `SwapColumns` mixes the L computation strands. LANE therefore offers two levels of parallelism: the P- and Q-PERMUTATIONS and the lines inside the permutations. The latter does not allow to take full advantage of AES-NI parallelism as `SwapColumns` breaks the instructions flow so we use the two levels of parallelism simultaneously: we compute an AES round for each of the $6L$ lines of the P-PERMUTATIONS in parallel before applying `SwapColumns` in each P-PERMUTATION, and do the same for the Q-PERMUTATIONS. (The code is completely unrolled and all keys are precomputed.)

For 256-bit outputs, the state nicely fits the available `xmm` registers. But for 512-bit outputs, the state does not fit anymore and only three P-PERMUTATIONS are computed in parallel instead of all six as before. This, in itself, does not change the AES-NI throughput as the number of lines is doubled in each PERMUTATION and thus the same number of AES rounds as before is performed in parallel. However, the 512-bit version of `SwapColumns` imposes an additional overhead.

LESAMNTA is a single-pipe compression function-based hash function. The underlying block cipher has the general topology of an unbalanced Feistel cipher; at each round two strands of the eight that comprise the cipher state are updated using a message dependent “subkey” and the round function f_{256} (resp. f_{512}) for the 256-bit (resp. 512-bit) hash output. The subkey generation and the f_{256} and f_{512} functions in the encryption path all involve AES-like operations and LESAMNTA can potentially benefit from AES-NI.

For the 256-bit version, the key schedule poses few problems. However, one difficulty for encryption path is that the AES-like transformations operate on 64-bit values and the MDS matrix is distinct from that of AES. The MDS matrix $\begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$ that is used is however a submatrix of MixColumn and so inserting zero bytes at the entry of the appropriate MixColumns entries will allow to perform the AES-like transformation using AES-NI. This can be achieved with the sequence: `pshufb`, `pxor` with a particular constant, `aesenc`, and `pshufb`. Note that in this case, `aesenc` is used at $\frac{1}{2}$ of its normal efficiency.

In the case of 512-bit hash outputs, the AES-like transformation in the key schedule involves an MDS that is too different from MixColumns, and so AES-NI is not really of any use there: the keys are therefore precomputed in a classical way. However, on the encryption side the round functions now use the full AES round, which gives nice advantages.

For both sets of outputs, it is possible to use the unbalanced nature of the Feistel construction to perform four f functions in parallel for both output sizes. In the 256-bit version, this carries a greater benefit: the four instances of the sequence preparing the data mentioned above can also be grouped to increase the overall throughput.

LUX is a stream-cipher based hash function that uses two banks of cipher state; the `BUFFER` and the `CORE`. At each iteration a block of message is input to both the `BUFFER` and `CORE`, both of which are then updated with information being passed between them. Sixteen blank rounds of computation seal the hashing process after the last block of message has been processed. While the `BUFFER` transformation is very simple, the `CORE` transformation is built on Rijndael-like operations. And it is the Rijndael-like operations in the `CORE` that are the most time-consuming parts of LUX, with mixing of the `CORE` and `BUFFER` requiring only a few, simple `xmm` instructions.

For all hash output lengths, the `CORE` transformation operates on a larger state than we find in the AES. However for 256-bit hash outputs it is equivalent to Rijndael operating on 256-bit blocks and techniques described in Section 2.1 can be used. Thus LUX with 256-bit outputs will benefit from AES-NI.

When used to generate longer hash outputs, however, LUX changes the form of the MDS transformation in such a way that it cannot easily be emulated using AES-NI. It appears for these longer outputs that AES-NI will not offer any advantage. In fairness, the optimised implementations of LUX for 512-bit outputs are already extremely competitive.

As an aside on the timing methodology, it is worth observing that we implemented sixteen iterations of the classical compression function found in LUX as a

single COMPRESS operation. This avoided buffer rotations and helped treat LUX in a way that was more consistent with the other algorithms.

SHAVITE-3 is a single-pipe compression function-based design, with the compression function being built closely on a Feistel cipher. The round function for this Feistel cipher is built directly from an AES round, and the accompanying message expansion also uses the AES round function. As a result, all hash output sizes can expect to benefit from AES-NI.

For the 256-bit hash output, the round function for the 12-round Feistel cipher consists of three rounds of the AES and we can therefore use AES-NI directly. To avoid any interaction with the memory, it is much more efficient to perform key derivation inside the `xmm` registers. Key derivation produces 36 subkeys of 128 bits using a combination of a non-linear layer based on four `aesenc` operations and a linear layer. It is possible to interleave key derivation with encryption since there are sufficient registers. The linear part of the key derivation only requires a few `xmm` manipulations (if handled properly) while the four AES rounds in the key schedule can be performed in parallel. The Feistel round function involves three AES rounds, but this time they are chained. SHAVITE-3 derives a significant benefit from avoiding memory access.

For the 512-bit hash output, the underlying 14 rounds block cipher is a generalised Feistel network. At each round there are two parallel invocations of four AES rounds. Now, however, key derivation produces new 128-bit words in sets of eight, rather than four, and so this needs to be performed in place while keeping the rest of the state in registers. The linear part of key derivation can still be implemented efficiently and the eight AES rounds can be parallelized. Within the encryption operation, there are now two Feistel round functions, each with four dependent AES rounds but these can be interleaved, increasing the throughput slightly. SHAVITE-3 is very closely built around the AES round operation and gains substantially from AES-NI.

VORTEX is a single-pipe compression function-based design that uses the enveloped Merkle-Damgård construction and builds upon MDC-2 [7]. The building blocks of VORTEX are Rijndael rounds on 128-bit blocks for VORTEX-256 and Rijndael rounds on 256-bit blocks for VORTEX-512. Cross-mixing between the 128-bit strands (resp. 256-bit strands for VORTEX 512) is multiplication-based. The parameter M_T determines whether integer multiplication ($M_T = 1$) or carry-less multiplication ($M_T = 0$) is used. A motivation behind VORTEX was to directly exploit AES-NI and the carry-less multiplication instructions on future Intel processors. In this paper we consider the case of $M_T = 1$. For VORTEX with 256-bit outputs we can directly exploit the `aesenc` operation. The key schedule calls upon the AES S-box but this can be easily emulated. For the 512-bit outputs, the underlying cipher operates on 256-bit states and, using similar techniques to those described in Section 2.1, it is straightforward to operate on this larger state. In contrast to some other algorithms, e.g. ECHO and LANE, VORTEX fits into the registers. On the other hand, it turns out that there is a bit less room to exploit AES-NI parallelism.

5 Implementation Results

Performance estimates for all SHA-3 candidates considered in this paper are given in Table 5. The Nehalem measurements were made on a Core i7 920 processor³ clocked at 2.67 GHz with GNU/Linux Debian running a 2.6.26-1-amd64 kernel. The compiler was `icc for amd64, Version 11.0, Build 20081105`. As explained in Section 3, we believe that these results will be very close to the real performance of the algorithms when run on the Westmere processor. For reference, some performance figures using assembly code from OpenSSL [28] for SHA-256 and SHA-512 timed under the same methodology on the same processor are 18.6 and 12.0 cycles/Byte respectively. While our results are preliminary, we feel they are sound enough to make some general observations.

Table 2. The predicted Westmere performance in cycles/Byte for those algorithms that can benefit from the Intel AES instructions set. For illustration, we provide the optimised performance figures given by submitters at the first NIST SHA-3 workshop. Other performance data can be found at [9]. Since in all cases 224- and 384-bit outputs are obtained by truncating 256- and 512-bit outputs, we only give figures for the latter.

<i>Algorithm</i>	<i>256-bit</i>		<i>512-bit</i>	
	AES-NI	previous	AES-NI	previous
ARIRANG	14.9	14.9	–	11.3
CHEETAH	7.6	9.3	–	13.6
ECHO (<i>double-pipe</i>)	6.6	28.5	12.3	53.5
ECHO-SP (<i>single-pipe</i>)	5.7	24.4	8.1	35.7
LANE	5.5	25.7	13.9	145.0
LESAMNTA	30.8	52.7	19.9	51.2
LUX	6.6	10.2	–	9.5
SHAVITE-3	5.6	26.7	5.5	38.2
VORTEX ($M_T = 1$)	4.4	46.3	5.2	56.1

While it is tempting to group all AES/Rijndael-based SHA-3 submissions together [5], one significant point of difference is that some will not be able to take advantage of AES-NI. Further, there are some algorithms, e.g. CHEETAH and LUX, for which the shorter hash outputs are likely to gain from AES-NI while the longer hash outputs, *i.e.* 384 and 512-bit, won't. Interestingly, CHEETAH is one of the fastest AES-inspired SHA-3 submissions on the NIST reference platform. But its performance when used with AES-NI is somewhat constrained by other non-AES components and CHEETAH may be slightly less competitive than the other algorithms when using AES-NI. That said, currently optimised code for this algorithm is reasonably efficient anyway. Our results for LESAMNTA differ from those at [17] which unfortunately use a different, inappropriate replacement instruction (see Section 3.1 and Appendix B).

³ Note that to ensure stable and clean results, we disabled two features of the processor: Hyperthreading and Turbo Boost.

Table 3. For those algorithms that solely use the AES round in its entirety, we give the number of AES rounds/Byte as a crude measure of how much the AES is used during the hashing process. We also give the *cost*, which is computed as the number of cycles/AES round. In general terms, the lower the cost, the more efficiently the AES round is being used with respect to AES-NI.

<i>Algorithm</i>	<i>256-bit</i>			<i>512-bit</i>		
	AES-NI	#AES/Byte	<i>cost</i>	AES-NI	#AES/Byte	<i>cost</i>
ECHO (<i>double-pipe</i>)	6.6	1.33	4.96	12.3	2.50	4.92
ECHO-SP (<i>single-pipe</i>)	5.7	1.14	5.00	8.1	1.67	4.85
LANE	5.5	1.31	4.20	14.3	1.75	8.17
SHAVITE-3	5.6	0.81	6.91	5.5	1.31	4.20
VORTEX ($M_T = 1$)	4.4	0.72	6.11	5.2	0.72	7.22

As would be expected, algorithms that are specifically designed around the AES round operation—ECHO, LANE, SHAVITE-3, and VORTEX—have the most to gain by appealing to AES-NI. If we consider the figures for 256-bit hash outputs then, for single-pipe variants, the throughput performance of these four algorithms is similar. However there is a much greater contrast in performance when we turn to 512-bit hash outputs, and this is due to differences in design. For instance, SHAVITE-3 for 512-bit outputs gains substantially from AES-NI since the modified round function for 512-bit outputs offers many opportunities for parallelism. This is something that is especially suited to AES-NI. On the other hand, when we move from 256- to 512-bit outputs with LANE, while the number of AES operations per byte increases in roughly the same proportion as was the case for SHAVITE-3, there is a performance impact that comes from doubling the size of the lanes in the P- and Q-PERMUTATIONS. Of course, when compared to existing optimised implementations LANE will still gain considerably when using AES-NI. But it does demonstrate how different design decisions can lead to very different performance profiles.

6 Conclusions

In this paper we have provided the first in-depth analysis of the likely impact of Intel’s AES instructions set on the first round SHA-3 candidates. To do this we designed a new methodology to replicate and anticipate the likely behavior of AES-NI in Westmere and we feel that this, in itself, will be of considerable interest. We have also provided the first performance estimates for those submissions that are likely to gain from AES-NI. Throughout we have tried to make a consistent and comprehensive comparison, and we have used the best currently-available information. We believe that our predictions are accurate and, in fact, may even be conservative. All the code we have developed is public [29] and this will allow others to develop their own optimized versions and to obtain improved performance projections.

Finally this paper sheds light on what has, until now, been a somewhat hidden issue. It is clear that the new Intel AES instructions set will have a profound effect on the performance of some of the SHA-3 submissions. At the same time, this low-level support for AES will become very widespread within a few years. Certainly this is only one factor among many for the SHA-3 candidates; but it may well be one of the important ones.

References

1. Atalay, A., Kara, O., Karakoc, F., Manap, C.: Shamata Hash Function Algorithm Specifications, [26]
2. Benadjila, R., Billet, O., Gilbert, H., Macario-Rat, G., Peyrin, T., Robshaw, M., Seurin, Y.: SHA-3 Proposal: ECHO. Available from [26]
3. Bernstein, D.: Cache-timing attacks on AES, preprint (2005), <http://cr.yp.to/papers.html#cachetiming>
4. Biham, E., Dunkelman, O.: The SHAvite-3 Hash Function. Available from [26]
5. Bjørstad, T.: A Short Note on AES-inspired Hashes. Posting to NIST SHA-3 mailing list, 25 May (2009)
6. Chang, D., Hong, S., Kang, C., Kang, J., Kim, J., Lee, C., Lee, J., Lee, J., Lee, S., Lee, Y., Lim, J., Sung, J.: Arirang. Available from [26]
7. Coppersmith, D., Pilepel, S., Meyer, C.H., Matyas, S.M., Hyden, M.M., Oseas, J., Brachtel, B., Schilling, M.: Data authentication using modification detection codes based on a public one way encryption function. U.S. Patent No. 4,908,861, March 13 (1990)
8. Daemen, J., Rijmen, V.: The Design of Rijndael. Springer, Heidelberg ISBN 3-540-42580-2
9. ECRYPT. eBASH: ECRYPT Benchmarking of All Submitted Hashes, <http://bench.cr.yp.to/ebash.html>
10. ECRYPT. The SHA-3 Zoo, http://ehash.iaik.tugraz.at/wiki/The_SHA-3_Zoo
11. Fleischmann, E., Forler, C., Gorski, M.: The Twister Hash Function Family. Available from [26]
12. Gauravaram, P., Knudsen, L., Matusiewicz, K., Mendel, F., Rechberger, C., Schläffer, M., Thomsen, S.: Grøstl—a SHA-3 Candidate. Available from [26]
13. Gueron, S.: Intel's Advanced Encryption Standard (AES) Instructions Set. Intel Corporation White Paper (March 2009), <http://software.intel.com>
14. Gueron, S.: Intel's New AES Instructions for Enhanced Performance and Security. In: Dunkelman, O. (ed.) FSE 2009. LNCS, vol. 5665, pp. 51–66. Springer, Heidelberg (2009)
15. Halevi, S., Hall, W., Jutla, C.: The Hash Function Fugue. Available from [26]
16. Hirose, S., Kuwakado, H., Yoshida, H.: SHA-3 Proposal: Lesamnta. Available from [26]
17. Hirose, S., Kuwakado, H., Yoshida, H.: The Hash Function Family Lesamnta, <http://www.sdl.hitachi.co.jp/crypto/lesamnta>
18. Indestege, S.: The LANE Hash Function. Available from [26]
19. Intel Corporation. Intel 64 and IA-32 Architectures Optimization Reference Manual, Table 2-6 of, <http://www.intel.com/Assets/PDF/manual/248966.pdf>
20. Intel Corporation. Intel Software Development Emulator (SDE), <http://software.intel.com/en-us/avx/>

21. Intel Corporation. Intel IACA tool: A Static Code Analyser, <http://software.intel.com/en-us/avx/>
22. Khovratovich, D., Biryukov, A., Nikolić, I.: The Hash Function Cheetah. Available from [26]
23. Kounavis, M., Gueron, S.: Vortex: A New Family of One Way Hash Functions based on Rijndael Rounds and Carry-less Multiplication. Available from [26]
24. National Institute of Standards and Technology. FIPS 197: Advanced Encryption Standard, <http://csrc.nist.gov/publications/fips/>
25. National Institute of Standards and Technology. The SHA-3 Hash Function Competition. Available from [26]
26. National Institute of Standards and Technology. First Round Candidates of the SHA-3 Hash Function Competition, http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/submissions_md1.html
27. Nikolić, I., Biryukov, A., Khovratovich, D.: Hash Family LUX. Available from [26]
28. OpenSSL 1.0.0, <http://www.openssl.org/source/>
29. Optimised implementations of SHA-3 submissions using AES-NI, <http://crypto.rd.francetelecom.com/sha3/AES/>
30. Osvik, D., Shamir, A., Tromer, E.: Cache Attacks and Countermeasures: The Case of AES. In: Pointcheval, D. (ed.) CT-RSA 2006. LNCS, vol. 3860, pp. 1–20. Springer, Heidelberg (2006)
31. Varıcı, K., Özen, O., Kocair, Ç.: Sarmal: SHA-3 Proposal. Available from [26]

Appendix A: Instructions

Table 4. The instructions that provide AES encryption

<pre> aesenc xmm1, xmm2/m128 Tmp := xmm1; Round Key := xmm2/m128; Tmp := ShiftRows (Tmp); Tmp := SubBytes (Tmp); Tmp := MixColumns (Tmp); xmm1 := Tmp xor Round Key; </pre>	<pre> aesenclast xmm1, xmm2/m128 Tmp := xmm1; Round Key := xmm2/m128; Tmp := ShiftRows (Tmp); Tmp := SubBytes (Tmp); xmm1 := Tmp xor Round Key </pre>
---	---

Table 5. How to derive the MixColumns operation from AES-NI

```

aesdeclast xmm1, 0x0 ... 0
aesenc  xmm1, 0x0 ... 0
-----
Tmp := xmm1
Tmp := InvShiftRows (Tmp);
Tmp := InvSubBytes (Tmp);
xmm1 := Tmp xor 0x0;
-----
Tmp := xmm1
Tmp := ShiftRows (Tmp);
Tmp := SubBytes (Tmp);
Tmp := MixColumns (Tmp);
xmm1 := Tmp xor 0x0;
                
```

Description of Some Additional Operations Used in This Work

`pshufb xmm1, xmm2/m128` This instruction is used to generate a byte-wise permutation of the contents of the first 128-bit operand, where the permutation is

defined by the second operand (`xmm` register or a memory location). The second source operand (`xmm2/m128`) is used as a mask, as follows. For each byte of `xmm2/m128`, the least significant four bits specify from where to select the corresponding byte of the source operand (`xmm1`). In addition, if the most significant bit of a byte of `xmm2/m128` equals one, then, regardless of the values of the other bits in that byte, zero is written in the result byte.

`pblendw xmm1, xmm2/m128, imm8` This operation “blends” the contents of two 128-bit operands (two registers or a register and a memory location) at the granularity of 16-bit words. Words from the second operand are conditionally written to the destination operand, depending on the setting of bits in the byte operand `imm8`. If bit k of this byte is set, then word k of the source is copied to the destination. If bit k is zero, word k of the destination is unchanged.

Appendix B: Rationale Behind the Replacements

Additional IACA Traces

AES-NI provides the `aesimc` instruction to perform `InvMixColumns`:

Total Latency:		6 Cycles;		Total number of Uops: 3		
Num of Uops	0 - DV	Ports pressure in cycles			4	5
3	2	1	2	3	4	5

3	2				1	CP
aesimc xmm0, xmm1						

The IACA tool supports the `aesdec` instruction the trace of which is shown below but does not support the `aesdeclast` instructions. From what has been derived for `aesenc`, `aesdec`, and `aesimc`, it is reasonable to assume its trace would have been identical to that of `aesdec`.

Total Latency:		6 Cycles;		Total number of Uops: 3		
Num of Uops	0 - DV	Ports pressure in cycles			4	5
3	2	1	2	3	4	5

3	2				1	CP
aesdec xmm0, xmm1						

Instructions Replacement Size

In order to evaluate the possible impact on the prefetching step (the prefetch buffer has a size of 16 bytes) or on the instruction cache, we conducted the following experiment: we went through the same kind of analysis as we conducted on `aesenc` and we replaced `pmulld xmm15, [mem]` which has two sequential μ ops of 3 cycles on port 1 by

```

pminposuw xmm15, [mem]
pminposuw xmm15, xmm15
    
```

which have a single μ op on port 1 each, but are interdependent. While the size of `pmulld` is 7 bytes and the size of the proposed replacement is 17 bytes, they both ran on the Nehalem with identical timings. Not only does this lend support to our approach, but it also suggests that the increased size of our AES-NI instructions set replacement is unlikely to have a significant effect.

Instructions Replacement for the Memory Variant

The `aesenc reg, [mem]` replacement we propose is actually quite similar to the `aesenc reg, reg` one. The only difference lies in the simulation of the memory access: it shouldn't impact the μop flows and, to accurately simulate `aesenc reg, [mem]`, the corresponding μop should start at the same cycle as the first μop on port 0. This is why we chose to launch the memory access at the first `mulps` instruction:

```
movdqu xmmk, xmmi
mulps  xmmi, [mem]
mulps  xmmk, xmmj
xorps  xmmi, xmmk
```

The validity of this replacement is assessed by the two following IACA traces:

Total Latency: 12 Cycles; Total number of Uops: 4

Num of Uops	0 - DV	Ports pressure in cycles					4	5		
		1	2 - D	3 - D	4	5				
4	2		1	1	X	X		1	CP	aesenc xmm0, [0x6008f0]

Total Latency: 11 Cycles; Total number of Uops: 5

Num of Uops	0 - DV	Ports pressure in cycles					4	5		
		1	2 - D	3 - D	4	5				
1	X		1					X		movdqu xmm2, xmm0
2	1		1	1	X	X			CP	mulps xmm0, [0x6008f0]
1	1									mulps xmm2, xmm1
1							1		CP	xorps xmm0, xmm2

An unfortunate side-effect of this replacement is that it affects an additional `xmm` register, putting additional constraints when avoiding false dependencies. This mainly concerns the ECHO and LANE algorithms.

Equivalent Inverse Cipher

The equivalent inverse cipher [8] allows for a decryption structure that is very similar to that of encryption. This is achieved by noticing that the straightforward decryption algorithm

`InvShiftRows, InvSubBytes, AddRoundKey, InvMixColumns,`

can be replaced by the equivalent one

`InvSubBytes, InvShiftRows, InvMixColumns, AddRoundKey,`

as the two first rounds commute and the last two commute when the key expansion is tweaked accordingly; decryption is now similarly structured to encryption:

`SubBytes, ShiftRows, MixColumns, AddRoundKey.`

An Inappropriate Replacement

In this paragraph, we give the IACA trace for the `pmuludq` instruction. This shows that the replacement proposed in [23] is not appropriate as a generic `aesenc` replacement on the Nehalem architecture. In the trace below, `pmuludq`

has a latency of 3 cycles whereas the `aesenc` instruction has a latency of 6 cycles, so the two instructions behave differently. It is even worse at the μop level, as `aesenc` has 3 μops dispatched through ports 0 and 5 whereas `pmuludq` has a single μop dispatched on port 1: this will lead to very distinct behaviors, and almost certainly a different throughput.

Total Latency:		3 Cycles;			Total number of Uops:		1	
Num of		0 - DV		Ports pressure in cycles				
Uops		1		2 - D		3 - D		4 5
1		1						CP pmuludq xmm0, xmm1

This explains the differences in the performance of LESAMNTA derived in this paper and quoted at [17].

Group Encryption: Non-interactive Realization in the Standard Model

Julien Cathalo^{1,*}, Benoît Libert^{1,**}, and Moti Yung²

¹ Université catholique de Louvain, Crypto Group (Belgium)

² Google Inc. and Columbia University (USA)

Abstract. Group encryption (GE) schemes, introduced at Asiacrypt’07, are an encryption analogue of group signatures with a number of interesting applications. They allow a sender to encrypt a message (in the CCA2 security sense) for some member of a PKI group concealing that member’s identity (in a CCA2 security sense, as well); the sender is able to convince a verifier that, among other things, the ciphertext is valid and some anonymous certified group member will be able to decrypt the message. As in group signatures, an opening authority has the power of pinning down the receiver’s identity. The initial GE construction uses interactive proofs as part of the design (which can be made non-interactive using the random oracle model) and the design of a fully non-interactive group encryption system is still an open problem. In this paper, we give the first GE scheme, which is a pure encryption scheme in the standard model, *i.e.*, a scheme where the ciphertext is a single message and proofs are non-interactive (and do not employ the random oracle heuristic). As a building block, we use a new public key certification scheme which incurs the smallest amount of interaction, as well.

Keywords: Group encryption, anonymity, provable security.

1 Introduction

Group encryption (GE) schemes, introduced by Kiayias, Tsiounis and Yung [29], are the encryption analogue of group signatures [16]. The latter primitives basically allow a group member to sign messages in the name of a group without revealing his identity. In a similar spirit, GE systems aim to hide the identity of a ciphertext’s recipient and still guarantee that he belongs to a population of registered members in a group administered by a group manager (GM). A sender can generate an anonymous encryption of some plaintext m intended for a receiver holding a public key that was certified by the GM (message security and receiver anonymity being both in the CCA2 sense). The ciphertext is prepared while leaving an opening authority (OA) the ability to “open” the ciphertext

* This author’s research was supported by the Belgian Walloon Region project ALAWN (Programme Wist 2).

** This author acknowledges the Belgian National Fund for Scientific Research (F.R.S.-F.N.R.S.) for their financial support.

(analogously to the opening operation in group signatures) and uncover the receiver’s name. At the same time, the sender should be able to convince a verifier that (1) the ciphertext is a valid encryption under the public key of some group member holding a valid certificate; (2) if necessary, the opening authority will be able to find out who the receiver is; (3) (optionally) the plaintext is a witness satisfying some public relation.

MOTIVATIONS. The GE primitive was motivated by various privacy applications such as anonymous trusted third parties or oblivious retriever storage. Many cryptographic protocols such as fair exchange, fair encryption or escrow encryption, involve trusted third parties that remain offline most of the time and are only involved to resolve problems. Group encryption allows one to verifiably encrypt some message to such a trusted third party while hiding his identity among a set of possible trustees. For instance, a user can encrypt a key (e.g., in an “international key escrow system”) to his own national trusted representative without letting the ciphertext reveal the latter’s identity, which could leak information on the user’s citizenship. At the same time, everyone can be convinced that the ciphertext is heading for an authorized trustee.

Group encryption also finds applications in ubiquitous computing, where anonymous credentials must be transferred between peer devices belonging to the same group. Asynchronous transfers may require to involve an untrusted storage server to temporarily store encrypted credentials. In such a situation, GE schemes may be used to simultaneously guarantee that (1) the server retains properly encrypted valid credentials that it cannot read; (2) credentials have a legitimate anonymous retriever; (3) if necessary, an authority will be able to determine who the retriever is.

By combining cascaded group encryptions using multiple trustees and according to a sequence of identity discoveries and transfers, one can also implement group signatures where signers can flexibly specify how a set of trustees should operate to open their signatures.

PRIOR WORKS. Kiayias, Tsiounis and Yung (KTY) [29] formalized the concept of group encryption and provided a suitable security modeling. They presented a modular design of GE system and proved that, beyond zero-knowledge proofs, anonymous public key encryption schemes with CCA2 security, digital signatures, and equivocal commitments are necessary to realize the primitive. They also showed how to efficiently instantiate their general construction using Paillier’s cryptosystem [35] (or, more precisely, a modification of the Camenisch-Shoup [13] variant of Paillier). While efficient, their scheme is not a single message encryption, since it requires the sender to interact with the verifier in a Σ -protocol to convince him that the aforementioned properties are satisfied. Interaction can be removed using the Fiat-Shamir paradigm [20] (and thus the random oracle model [4]), but only heuristic arguments [22] (see also [14]) are then possible in terms of security.

Independently, Qin *et al.* [36] considered a closely related primitive with non-interactive proofs and short ciphertexts. However, they avoid interaction by

explicitly employing a random oracle and also rely on strong interactive assumptions. As we can see, none of these schemes is a truly non-interactive encryption scheme without the random oracle idealization.

OUR CONTRIBUTION. As already noted in various contexts such as anonymous credentials [2], rounds of interaction are expensive and even impossible at times as, in some applications, proofs should be verifiable by third parties that are not present when provers are available. In the setting of group encryption, this last concern is even more constraining as it requires the sender, who may be required to repeat proofs with many verifiers, to maintain a state and remember the random coins that he uses to encrypt *every* single ciphertext. In the frequent situation where many encryptions have to be generated using independent random coins, this becomes a definite bottleneck.

This paper solves the above problems and describes the first realization of group encryption which is a fully non-interactive encryption scheme with CCA2-security and anonymity in the standard model. In our scheme, senders do not need to maintain a state: thanks to the Groth-Sahai [27] non-interactive proof systems, the proof of a ciphertext can be generated once-and-for-all at the same time as the ciphertext itself. Furthermore, using suitable parameters and for a comparable security level, we can also shorten ciphertexts by a factor of 2 in comparison with the KTY scheme. As far as communication goes, the size of proofs allows decreasing by more than 75% the number of transmitted bits between the sender and the verifier.

Since our goal is to avoid interaction, we also design a joining protocol (*i.e.*, a protocol whereby the user effectively becomes a group member and gets his public key certified by the GM) which requires the smallest amount of interaction: as in the Kiayias-Yung group signature [30], only two messages have to be exchanged between the GM and the user and the latter need not to prove anything about his public key. In particular, rewinding is not necessary in security proofs and the join protocol can be safely executed in a concurrent environment, when many users want to register at the same time. The join protocol uses a non-interactive public key certification scheme where discrete-logarithm-type public keys can be signed as if they were ordinary messages (and without knowing the matching private key) while leaving the ability to efficiently prove knowledge of the certificate/public key using the Groth-Sahai techniques. To certify users without having to rewind¹ in security proofs, the KTY scheme uses groups of hidden order (and more precisely, Camenisch-Lysyanskaya signatures [12]). In public order groups, to the best of our knowledge, our construction is the first certification method that does not require any form of proof of knowledge of private keys. We believe it to be of independent interest as it can be used to construct group signatures (in the standard model) where the joining mechanism tolerates concurrency in the model of [30] without demanding more than two moves of interaction.

¹ Although the simulator does not need to rewind proofs of knowledge in [29], users still have to interactively prove the validity of their public key.

ORGANIZATION. In section 2, we describe the intractability assumptions that we need and recall the KTY model of group encryption. Section 3 explains the building blocks of our construction and notably describes our certification scheme. Our GE system is depicted in section 4.

2 Background

In the paper, when S is a set, $x \xleftarrow{\$} S$ denotes the action of choosing x at random in S . By $a \in \text{poly}(\lambda)$, we mean that a is a polynomial in λ while $b \in \text{negl}(\lambda)$ says that b is a negligible function of λ . When a and b are two binary strings, $a||b$ stands for their concatenation.

2.1 Complexity Assumptions

We use groups $(\mathbb{G}, \mathbb{G}_T)$ of prime order p with an efficiently computable map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ such that $e(g^a, h^b) = e(g, h)^{ab}$ for any $(g, h) \in \mathbb{G} \times \mathbb{G}$, $a, b \in \mathbb{Z}$ and $e(g, h) \neq 1_{\mathbb{G}_T}$ whenever $g, h \neq 1_{\mathbb{G}}$.

In this setting, we rely on an assumption introduced in [7] that allows constructing efficient non-interactive proofs as pointed out in [27].

Definition 1. *The Decision Linear Problem (DLIN) in \mathbb{G} , is to distinguish the distribution $D_1 = \{(g, g^a, g^b, g^{ac}, g^{bd}, g^{c+d}) | a, b, c, d \xleftarrow{\$} \mathbb{Z}_p^*\}$ from the distribution $D_2 = \{(g, g^a, g^b, g^{ac}, g^{bd}, g^z) | a, b, c, d, z \xleftarrow{\$} \mathbb{Z}_p^*\}$. The Decision Linear Assumption is the intractability of DLIN for any PPT algorithm \mathcal{D} .*

This problem amounts to deciding whether vectors $\vec{g}_1 = (g^a, 1, g)$, $\vec{g}_2 = (1, g^b, g)$ and \vec{g}_3 are linearly dependent or not. We also consider a related computational problem which bears similarities with simultaneous pairing problems [26,25].

Definition 2. *The Simultaneous Double Pairing problem (SDP) in \mathbb{G} is, given $(g_1, g_2, g_{1,c}, g_{2,d}) \in \mathbb{G}^4$, to find a triple $(u, v, w) \in \mathbb{G}^3 \setminus \{(1_{\mathbb{G}}, 1_{\mathbb{G}}, 1_{\mathbb{G}})\}$ such that $e(g_1, u) = e(g_{1,c}, w)$ and $e(g_2, v) = e(g_{2,d}, w)$.*

Like the simultaneous triple pairing assumption [25], the hardness of this problem is implied by the DLIN assumption: given $(g, g_1, g_2, g_1^c, g_2^d, \eta \stackrel{?}{=} g^{c+d})$ any algorithm that, on input of (g_1, g_2, g_1^c, g_2^d) , outputs a non-trivial (u, v, w) such that $e(g_1, u) = e(g_1^c, w)$, $e(g_2, v) = e(g_2^d, w)$ allows telling whether $\eta = g^{c+d}$ by testing if $e(g, u \cdot v) = e(\eta, w)$ (since $u = w^c$ and $v = w^d$).

We also use the Hidden Strong Diffie-Hellman (HSDH) assumption introduced in [10] as a strengthening of the Strong Diffie-Hellman assumption [6].

Definition 3. *The ℓ -Hidden Strong Diffie-Hellman problem (ℓ -HSDH) in \mathbb{G} is, given $(g, \Omega = g^\omega, u) \xleftarrow{\$} \mathbb{G}^3$ and triples $(g^{1/(\omega+s_i)}, g^{c_i}, u^{c_i})$ with $c_1, \dots, c_\ell \xleftarrow{\$} \mathbb{Z}_p^*$, to find another triple $(g^{1/(\omega+c)}, g^c, u^c)$ such that $c \neq c_i$ for $i = 1, \dots, \ell$.*

We finally need the following variant of the Diffie-Hellman assumption.

Definition 4. *The Flexible Diffie-Hellman problem (FlexDH) is, given $(g, g^a, g^b) \in \mathbb{G}^3$, where $a, b \xleftarrow{\$} \mathbb{Z}_p^*$, to find a triple (C, C^a, C^{ab}) such that $C \neq 1_{\mathbb{G}}$.*

A potentially easier problem considered in [33] only requires to output (C, C^{ab}) on input of the same values. The latter problem was proved generically hard in prime order groups [33]. In bilinear groups, any algorithm solving either of these two problems would make it easy to recognize g^{abc} on input of (g, g^a, g^b, g^c) , which is a problem suggested for the first time in [8, Section 8].

2.2 Model and Security Notions

Group encryption schemes involve a sender, a verifier, a group manager (GM) that manages the group of receivers and an opening authority (OA) that is able to uncover the identity of ciphertext receivers. A group encryption system is formally specified by the description of a relation \mathcal{R} as well as a collection $\text{GE} = (\text{SETUP}, \text{JOIN}, \langle \mathcal{G}_r, \mathcal{R}, \text{sample}_{\mathcal{R}} \rangle, \text{ENC}, \text{DEC}, \text{OPEN}, \langle \mathcal{P}, \mathcal{V} \rangle)$ of algorithms or protocols. Among these, **SETUP** is a set of initialization procedures that all take (explicitly or implicitly) a security parameter λ as input. They can be split into one that generates a set of public parameters **param** (a common reference string), one for the GM and another one for the OA. We call them $\text{SETUP}_{\text{init}}(\lambda)$, $\text{SETUP}_{\text{GM}}(\text{param})$ and $\text{SETUP}_{\text{OA}}(\text{param})$, respectively. The latter two procedures are used to produce key pairs $(\text{pk}_{\text{GM}}, \text{sk}_{\text{GM}})$, $(\text{pk}_{\text{OA}}, \text{sk}_{\text{OA}})$ for the GM and the OA. In the following, **param** is incorporated in the inputs of all algorithms although we sometimes omit to explicitly write it.

JOIN = $(\text{J}_{\text{user}}, \text{J}_{\text{GM}})$ is an interactive protocol between the GM and the prospective user. As in [30], we will restrict this protocol to have minimal interaction and consist of only two messages: the first one is the user's public key pk sent by J_{user} to J_{GM} and the latter's response is a certificate cert_{pk} for pk that makes the user's group membership effective. We do not require the user to prove knowledge of his private key sk or anything else about it. In our construction, valid keys will be publicly recognizable and users do not need to prove their validity. After the execution of **JOIN**, the GM stores the public key pk and its certificate cert_{pk} in a public directory database.

Algorithm **sample** allows sampling pairs $(x, w) \in \mathcal{R}$ (made of a public value x and a witness w) using keys $(\text{pk}_{\mathcal{R}}, \text{sk}_{\mathcal{R}})$ produced by \mathcal{G}_r . Depending on the relation, $\text{sk}_{\mathcal{R}}$ may be the empty string (as will be the case in our scheme). The testing procedure $\mathcal{R}(x, w)$ returns 1 whenever $(x, w) \in \mathcal{R}$. To encrypt a witness w such that $(x, w) \in \mathcal{R}$ for some public x , the sender fetches the pair $(\text{pk}, \text{cert}_{\text{pk}})$ from **database** and runs the randomized encryption algorithm. The latter takes as input w , a label L , the receiver's pair $(\text{pk}, \text{cert}_{\text{pk}})$ as well as public keys pk_{GM} and pk_{OA} . Its output is a ciphertext $\psi \leftarrow \text{ENC}(\text{pk}_{\text{GM}}, \text{pk}_{\text{OA}}, \text{pk}, \text{cert}_{\text{pk}}, w, L)$. On input of the same elements, the certificate cert_{pk} , the ciphertext ψ and the random coins coins_{ψ} that were used to produce it, the non-interactive algorithm \mathcal{P} generates a proof π_{ψ} that there exists a certified receiver whose public key was registered in **database** and that is able to decrypt ψ and obtain a witness w such that $(x, w) \in \mathcal{R}$. The verification algorithm \mathcal{V} takes as input ψ , pk_{GM} , pk_{OA} , π_{ψ} and the description of \mathcal{R} and outputs 0 or 1. Given ψ , L and the receiver's private key sk , the output of **DEC** is either a witness w such that $(x, w) \in \mathcal{R}$ or a rejection symbol \perp . Finally, **OPEN** takes as input a ciphertext/label pair (ψ, L) and the OA's secret key sk_{OA} and returns a receiver's public key pk .

The security model considers four properties termed correctness, message security, anonymity and soundness. In the following, we sometimes denote by $\langle \text{output}_A | \text{output}_B \rangle \leftarrow \langle A(\text{input}_A), B(\text{input}_B) \rangle (\text{common-input})$ the execution of a protocol between A and B obtaining their own outputs from their inputs.

CORRECTNESS. The correctness property requires that the following experiment returns 1 with overwhelming probability.

Experiment $\text{Expt}^{\text{correctness}}(\lambda)$
 $\text{param} \leftarrow \text{SETUP}_{\text{init}}(\lambda); (\text{pk}_{\mathcal{R}}, \text{sk}_{\mathcal{R}}) \leftarrow \mathcal{G}_r(\lambda); (x, w) \leftarrow \text{sample}_{\mathcal{R}}(\text{pk}_{\mathcal{R}}, \text{sk}_{\mathcal{R}});$
 $(\text{pk}_{\text{GM}}, \text{sk}_{\text{GM}}) \leftarrow \text{SETUP}_{\text{GM}}(\text{param}); (\text{pk}_{\text{OA}}, \text{sk}_{\text{OA}}) \leftarrow \text{SETUP}_{\text{OA}}(\text{param});$
 $\langle \text{pk}, \text{sk}, \text{cert}_{\text{pk}} | \text{pk}, \text{cert}_{\text{pk}} \rangle \leftarrow \langle J_{\text{user}}, J_{\text{GM}}(\text{sk}_{\text{GM}}) \rangle (\text{pk}_{\text{GM}});$
 $\psi \leftarrow \text{ENC}(\text{pk}_{\text{GM}}, \text{pk}_{\text{OA}}, \text{pk}, \text{cert}_{\text{pk}}, w, L);$
 $\pi_{\psi} \leftarrow \mathcal{P}(\text{pk}_{\text{GM}}, \text{pk}_{\text{OA}}, \text{pk}, \text{cert}, w, L, \psi, \text{coins}_{\psi});$
 If $((w \neq \text{DEC}(\text{sk}, \psi, L)) \vee (\text{pk} \neq \text{OPEN}(\text{sk}_{\text{OA}}, \psi, L))$
 $\vee (\mathcal{V}(\psi, L, \pi_{\psi}, \text{pk}_{\text{GM}}, \text{pk}_{\text{OA}}) = 0))$ return 0 else return 1;

MESSAGE SECURITY. The message secrecy property is defined by an experiment where the adversary has access to oracles that may be stateful (and maintain a state across queries) or stateless:

- $\text{DEC}(\text{sk})$: is a stateless oracle for the user decryption function DEC . When this oracle is restricted not to decrypt a ciphertext-label pair (ψ, L) , we denote it by $\text{DEC}^{\neg(\psi, L)}$.
- $\text{CH}_{\text{ror}}^b(\lambda, \text{pk}, w, L)$: is a real-or-random challenge oracle that is only queried once. It returns $(\psi, \text{coins}_{\psi})$ such that $\psi \leftarrow \text{ENC}(\text{pk}_{\text{GM}}, \text{pk}_{\text{OA}}, \text{pk}, \text{cert}_{\text{pk}}, w, L)$ if $b = 1$ whereas, if $b = 0$, $\psi \leftarrow \text{ENC}(\text{pk}_{\text{GM}}, \text{pk}_{\text{OA}}, \text{pk}, \text{cert}_{\text{pk}}, w', L)$ encrypts a random plaintext uniformly chosen in the space of plaintexts of length $O(\lambda)$. In either case, coins_{ψ} are the random coins used to generate ψ .
- $\text{PROVE}_{\mathcal{P}, \mathcal{P}'}^b(\text{pk}_{\text{GM}}, \text{pk}_{\text{OA}}, \text{pk}, \text{cert}_{\text{pk}}, \text{pk}_{\mathcal{R}}, x, w, \psi, L, \text{coins}_{\psi})$: is a stateful oracle that the adversary can query on multiple occasions. If $b = 1$, it runs the real prover \mathcal{P} on the inputs to produce an actual proof π_{ψ} . If $b = 0$, the oracle runs a simulator \mathcal{P}' that uses the same inputs as \mathcal{P} except witness w, coins_{ψ} and generates a simulated proof.

These oracles are used in an experiment where the adversary controls the GM, the OA and all members but the honest receiver. The adversary \mathcal{A} is the dishonest GM that certifies the honest receiver in an execution of JOIN . She has oracle access to the decryption function DEC of that receiver. At the challenge phase, she probes the challenge oracle for a label and a pair $(x, w) \in \mathcal{R}$ of her choice. After the challenge phase, she can also invoke the PROVE oracle on multiple occasions and eventually aims to guess the bit b chosen by the challenger.

As pointed out in [29], designing an efficient simulator \mathcal{P}' (for executing $\text{PROVE}_{\mathcal{P}, \mathcal{P}'}^b(\cdot)$ when $b = 0$) is part of the security proof and might require a simulated common reference string.

Definition 5. A GE scheme satisfies message security if, for any PPT adversary \mathcal{A} , the experiment below returns 1 with probability at most $1/2 + \text{negl}(\lambda)$.

Experiment $\mathbf{Expt}_A^{\text{sec}}(\lambda)$

param $\leftarrow \text{SETUP}_{\text{init}}(\lambda)$; (aux, pk_{GM}, pk_{OA}) $\leftarrow \mathcal{A}(\text{param})$;
 ⟨pk, sk, cert_{pk}|aux⟩ $\leftarrow \langle \text{J}_{\text{user}}, \mathcal{A}(\text{aux}) \rangle(\text{pk}_{\text{GM}})$;
 (aux, x, w, L, pk_R) $\leftarrow \mathcal{A}^{\text{DEC}(\text{sk}, \cdot)}(\text{aux})$; If (x, w) $\notin \mathcal{R}$ return 0;
 b $\stackrel{\$}{\leftarrow} \{0, 1\}$; (ψ , coins _{ψ}) $\leftarrow \text{CH}_{\text{for}}^b(\lambda, \text{pk}, w, L)$;
 b' $\leftarrow \mathcal{A}^{\text{PROVE}_{\mathcal{P}, \mathcal{P}'}}(\text{pk}_{\text{GM}}, \text{pk}_{\text{OA}}, \text{pk}, \text{cert}_{\text{pk}}, \text{pk}_{\mathcal{R}}, x, w, \psi, L, \text{coins}_{\psi}), \text{DEC}^{-\langle \psi, L \rangle}(\text{sk}, \cdot)(\text{aux}, \psi)$;
 If b = b' return 1 else return 0;

ANONYMITY. In anonymity attacks, the adversary controls the whole system but the opening authority and performs a kind of chosen-ciphertext attack on the encryption scheme of the OA. She registers two keys pk₀, pk₁ in database and, for a pair (x, w) $\in \mathcal{R}$ of her choosing, obtains an encryption of w under pk_b for some b $\in \{0, 1\}$ chosen by the challenger. She is granted access to decryption oracles w.r.t. both keys pk₀, pk₁. In addition, she may invoke the following oracles:

- CH_{anon}^b(pk_{GM}, pk_{OA}, pk₀, pk₁, w, L): is a challenge oracle that is only queried once by the adversary. It returns a pair (ψ , coins _{ψ}) consisting of a ciphertext $\psi \leftarrow \text{ENC}(\text{pk}_{\text{GM}}, \text{pk}_{\text{OA}}, \text{pk}_b, \text{cert}_{\text{pk}_b}, w, L)$ and the coin tosses coins _{ψ} that were used to generate ψ .
- USER(pk_{GM}): is a stateful oracle simulating two executions of J_{user} to introduce two honest users in the group. It uses a string keys where the outputs of the two executions are written.
- OPEN(sk_{OA}, .): is a stateless oracle that simulates the opening algorithm on behalf of the OA and, on input of a GE ciphertext, returns the receiver's public key.

Definition 6. A GE scheme satisfies anonymity if, for any PPT adversary \mathcal{A} , the experiment below returns 1 with a probability not exceeding $1/2 + \text{negl}(\lambda)$.

Experiment $\mathbf{Expt}_A^{\text{anon}}(\lambda)$

param $\leftarrow \text{SETUP}_{\text{init}}(\lambda)$; (pk_{OA}, sk_{OA}) $\leftarrow \text{SETUP}_{\text{OA}}(\text{param})$;
 (aux, pk_{GM}) $\leftarrow \mathcal{A}(\text{param}, \text{pk}_{\text{OA}})$; aux $\leftarrow \mathcal{A}^{\text{USER}(\text{pk}_{\text{GM}}), \text{OPEN}(\text{sk}_{\text{OA}}, \cdot)}(\text{aux})$;
 If keys $\neq (\text{pk}_0, \text{sk}_0, \text{cert}_{\text{pk}_0}, \text{pk}_1, \text{sk}_1, \text{cert}_{\text{pk}_1})(\text{aux})$ return 0;
 (aux, x, w, L, pk_R) $\leftarrow \mathcal{A}^{\text{OPEN}(\text{sk}_{\text{OA}}, \cdot), \text{DEC}(\text{sk}_0, \cdot), \text{DEC}(\text{sk}_1, \cdot)}(\text{aux})$;
 If (x, w) $\notin \mathcal{R}$ return 0;
 b $\stackrel{\$}{\leftarrow} \{0, 1\}$; (ψ , coins _{ψ}) $\leftarrow \text{CH}_{\text{anon}}^b(\text{pk}_{\text{GM}}, \text{pk}_{\text{OA}}, \text{pk}_0, \text{pk}_1, w, L)$;
 b' $\leftarrow \mathcal{A}^{\text{P}}(\text{pk}_{\text{GM}}, \text{pk}_{\text{OA}}, \text{pk}_b, \text{cert}_{\text{pk}_b}, x, w, \psi, L, \text{coins}_{\psi}, \text{OPEN}^{-\langle \psi, L \rangle}(\text{sk}_{\text{OA}}, \cdot), \text{DEC}^{-\langle \psi, L \rangle}(\text{sk}_0, \cdot), \text{DEC}^{-\langle \psi, L \rangle}(\text{sk}_1, \cdot))(\text{aux}, \psi)$;
 If b = b' return 1 else return 0;

As shown in [29], GE schemes satisfying the above notion necessarily subsume a key-private (a.k.a. receiver anonymous) [3,28] cryptosystem.

SOUNDNESS. In a soundness attack, the adversary creates the group of receivers by interacting with the honest GM. Her goal is to produce a ciphertext ψ and a convincing proof that ψ is valid w.r.t. a relation \mathcal{R} of her choice but either (1) the opening reveals a receiver's public key pk that does not belong to any group member; (2) the output pk of OPEN is not a valid public key (i.e., $\text{pk} \notin \mathcal{PK}$,

where \mathcal{PK} is the space of valid public keys); (3) the ciphertext C is not in the space $\mathcal{C}^{x,L,\text{pk}_{\mathcal{R}},\text{pk}_{\text{GM}},\text{pk}_{\text{OA}},\text{pk}}$ of valid ciphertexts. This notion is formalized by a game where the adversary is given access to a user registration oracle $\text{REG}(\text{sk}_{\text{GM}}, \cdot)$ that simulates J_{GM} . This oracle maintains a repository database where registered public keys and their certificates are stored.

Definition 7. *A GE scheme is sound if, for any PPT adversary \mathcal{A} , the experiment below returns 1 with negligible probability.*

Experiment $\text{Expt}_{\mathcal{A}}^{\text{soundness}}(\lambda)$
 $\text{param} \leftarrow \text{SETUP}_{\text{init}}(\lambda); (\text{pk}_{\text{OA}}, \text{sk}_{\text{OA}}) \leftarrow \text{SETUP}_{\text{OA}}(\text{param});$
 $(\text{pk}_{\text{GM}}, \text{sk}_{\text{GM}}) \leftarrow \text{SETUP}_{\text{GM}}(\text{param});$
 $(\text{pk}_{\mathcal{R}}, x, \psi, \pi_{\psi}, L, \text{aux}) \leftarrow \mathcal{A}^{\text{REG}(\text{sk}_{\text{GM}}, \cdot)}(\text{param}, \text{pk}_{\text{GM}}, \text{pk}_{\text{OA}}, \text{sk}_{\text{OA}});$
If $\mathcal{V}(\psi, L, \pi_{\psi}, \text{pk}_{\text{GM}}, \text{pk}_{\text{OA}}) = 0$ return 0;
 $\text{pk} \leftarrow \text{OPEN}(\text{sk}_{\text{OA}}, \psi, L);$
If $((\text{pk} \notin \text{database}) \vee (\text{pk} \notin \mathcal{PK}) \vee (\psi \notin \mathcal{C}^{x,L,\text{pk}_{\mathcal{R}},\text{pk}_{\text{GM}},\text{pk}_{\text{OA}},\text{pk}}))$
then return 1 else return 0;

2.3 Groth-Sahai Proof Systems

In the following notations, for equal-dimension vectors \vec{A} and \vec{B} containing group elements, $\vec{A} \odot \vec{B}$ stands for their component-wise product.

When based on the DLIN assumption, the Groth-Sahai (GS) proof systems [27] use a common reference string comprising vectors $\vec{g}_1, \vec{g}_2, \vec{g}_3 \in \mathbb{G}^3$, where $\vec{g}_1 = (g_1, 1, g)$, $\vec{g}_2 = (1, g_2, g)$ for some $g_1, g_2 \in \mathbb{G}$. To commit to $X \in \mathbb{G}$, one sets $\vec{C} = (1, 1, X) \odot \vec{g}_1^r \odot \vec{g}_2^s \odot \vec{g}_3^t$ with $r, s, t \stackrel{\$}{\leftarrow} \mathbb{Z}_p^*$. When the proof system is configured to give perfectly sound proofs, \vec{g}_3 is chosen as $\vec{g}_3 = \vec{g}_1^{\xi_1} \odot \vec{g}_2^{\xi_2}$ with $\xi_1, \xi_2 \stackrel{\$}{\leftarrow} \mathbb{Z}_p^*$. Commitments $\vec{C} = (g_1^{r+\xi_1 t}, g_2^{s+\xi_2 t}, X \cdot g^{r+s+t(\xi_1+\xi_2)})$ are then Boneh-Boyen-Shacham (BBS) ciphertexts that can be decrypted using $\alpha_1 = \log_g(g_1)$, $\alpha_2 = \log_g(g_2)$. In the witness indistinguishability (WI) setting, vectors $\vec{g}_1, \vec{g}_2, \vec{g}_3$ are linearly independent and \vec{C} is a perfectly hiding commitment. Under the DLIN assumption, the two kinds of CRS are indistinguishable.

To commit to an exponent $x \in \mathbb{Z}_p$, one computes $\vec{C} = \vec{\varphi}^x \odot \vec{g}_1^r \odot \vec{g}_2^s$, with $r, s \stackrel{\$}{\leftarrow} \mathbb{Z}_p^*$, using a CRS comprising vectors $\vec{\varphi}, \vec{g}_1, \vec{g}_2$. In the soundness setting $\vec{\varphi}, \vec{g}_1, \vec{g}_2$ are linearly independent vectors (typically $\vec{\varphi} = \vec{g}_3 \odot (1, 1, g)$ where $\vec{\varphi} = \vec{g}_1^{\xi_1} \odot \vec{g}_2^{\xi_2}$) whereas, in the WI setting, choosing $\vec{\varphi} = \vec{g}_1^{\xi_1} \odot \vec{g}_2^{\xi_2}$ gives a perfectly hiding commitment since \vec{C} is always a BBS encryption of $1_{\mathbb{G}}$.

To prove that committed variables satisfy a set of relations, the GS techniques replace variables by the corresponding commitments in each relation. The whole proof consists of one commitment per variable and one proof element (made of a constant number of group elements) per relation.

Such proofs are available for pairing-product relations, which are of the type

$$\prod_{i=1}^n e(\mathcal{A}_i, \mathcal{X}_i) \cdot \prod_{i=1}^n \prod_{j=1}^n e(\mathcal{X}_i, \mathcal{X}_j)^{a_{ij}} = t_T,$$

for variables $\mathcal{X}_1, \dots, \mathcal{X}_n \in \mathbb{G}$ and constants $t_T \in \mathbb{G}_T, \mathcal{A}_1, \dots, \mathcal{A}_n \in \mathbb{G}, a_{ij} \in \mathbb{G}$, for $i, j \in \{1, \dots, n\}$. Efficient proofs also exist for multi-exponentiation equations

$$\prod_{i=1}^m \mathcal{A}_i^{y_i} \cdot \prod_{j=1}^n \mathcal{X}_j^{b_j} \cdot \prod_{i=1}^m \prod_{j=1}^n \mathcal{X}_j^{y_i \gamma_{ij}} = T,$$

for variables $\mathcal{X}_1, \dots, \mathcal{X}_n \in \mathbb{G}, y_1, \dots, y_m \in \mathbb{Z}_p$ and constants $T, \mathcal{A}_1, \dots, \mathcal{A}_m \in \mathbb{G}, b_1, \dots, b_n \in \mathbb{Z}_p$ and $\gamma_{ij} \in \mathbb{G}$, for $i \in \{1, \dots, m\}, j \in \{1, \dots, n\}$.

Multi-exponentiation equations admit zero-knowledge proofs at no additional cost. On a simulated CRS (prepared for the WI setting), a trapdoor makes it is possible to simulate proofs without knowing witnesses and simulated proofs are perfectly indistinguishable from real proofs. As for pairing-product equations, zero-knowledge proofs are often possible but usually come at some expense. In the paper, we only resort to such NIZK simulators in one occasion.

In both cases, proofs for quadratic equations cost 9 group elements. Linear pairing-product equations (when $a_{ij} = 0$ for all i, j) take 3 group elements each. Linear multi-exponentiation equations of the type $\prod_{j=1}^n \mathcal{X}_j^{b_j} = T$ (resp. $\prod_{i=1}^m \mathcal{A}_i^{y_i} = T$) demand 3 (resp. 2) group elements.

3 Building Blocks

Our certification scheme uses a trapdoor commitment to group elements as an important ingredient to dispense with proofs of knowledge of users' private keys.

3.1 A Trapdoor Commitment to Group Elements

We need a trapdoor commitment scheme that allows committing to elements of a group \mathbb{G} where bilinear map arguments are taken. Commitments will have to be themselves elements of \mathbb{G} , which prevents us from using Groth's scheme [25] where commitments lie in the range \mathbb{G}_T of the pairing.

Such commitments can be obtained using the perfectly hiding Groth-Sahai commitment based on the linear assumption recalled in section 2.3. This commitment uses a common reference string describing a prime order group \mathbb{G} and a generator $f \in \mathbb{G}$. The commitment key consists of vectors $(\vec{f}_1, \vec{f}_2, \vec{f}_3)$ chosen as $\vec{f}_1 = (f_1, 1, f), \vec{f}_2 = (1, f_2, f)$ and $\vec{f}_3 = \vec{f}_1^{\xi_1} \odot \vec{f}_2^{\xi_2} \odot (1, 1, f)^{\xi_3}$, with $f_1, f_2 \xleftarrow{\$} \mathbb{G}, \xi_1, \xi_2, \xi_3 \xleftarrow{\$} \mathbb{Z}_p^*$. To commit to X , the sender picks $\phi_1, \phi_2, \phi_3 \xleftarrow{\$} \mathbb{Z}_p^*$ and sets $\vec{C}_X = (1, 1, X) \odot \vec{f}_1^{\phi_1} \odot \vec{f}_2^{\phi_2} \odot \vec{f}_3^{\phi_3}$, which, if \vec{f}_3 is parsed as $(f_{3,1}, f_{3,2}, f_{3,3})$, can be written $\vec{C}_X = (f_1^{\phi_1} \cdot f_{3,1}^{\phi_3}, f_2^{\phi_2} \cdot f_{3,2}^{\phi_3}, X \cdot f^{\phi_1 + \phi_2} \cdot f_{3,3}^{\phi_3})$. Due to the use of GS proofs, commitment openings need to only consist of group elements (and no scalar). To open $\vec{C}_X = (C_1, C_2, C_3)$, the sender reveals $(D_1, D_2, D_3) = (f^{\phi_1}, f^{\phi_2}, f^{\phi_3})$ and X . The receiver is convinced that the committed value was X by checking that

$$\begin{cases} e(C_1, f) = e(f_1, D_1) \cdot e(f_{3,1}, D_3) \\ e(C_2, f) = e(f_2, D_2) \cdot e(f_{3,2}, D_3) \\ e(C_3, f) = e(X \cdot D_1 \cdot D_2, f) \cdot e(f_{3,3}, D_3). \end{cases}$$

If a cheating sender can come up with distinct openings of \vec{C}_X , we can easily solve a $S2P$ instance $(g_1, g_2, g_{1,c}, g_{2,d})$. Namely, the commitment key is set as $(f_1, f_2, f_{3,1}, f_{3,2}) = (g_1, g_2, g_{1,c}, g_{2,d})$ and $f, f_{3,3}$ are chosen at random. When the adversary outputs $(X, (D_1, D_2, D_3))$ and $(X', (D'_1, D'_2, D'_3))$, we must simultaneously have $e(f_1, D_1/D'_1) = e(f_{3,1}, D'_3/D_3)$, $e(f_2, D_2/D'_2) = e(f_{3,2}, D'_3/D_3)$ and $e((XD_1D_2)/(X'D'_1D'_2), f) = e(f_{3,3}, D'_3/D_3)$. Hence, setting $u = D_1/D'_1$, $v = D_2/D'_2$ and $w = D'_3/D_3$ solves the $S2P$ problem as (u, v, w) can only be trivial if $X' = X$.

Using the trapdoor (ξ_1, ξ_2, ξ_3) , the receiver can equivocate commitments. Given a commitment \vec{C}_X and its opening $(X, (D_1, D_2, D_3))$, one can trapdoor open \vec{C}_X to any other $X' \in \mathbb{G}$ (and without knowing $\log_g(X')$) by computing

$$D'_1 = D_1 \cdot (X'/X)^{\xi_1/\xi_3}, \quad D'_2 = D_2 \cdot (X'/X)^{\xi_2/\xi_3}, \quad D'_3 = (X/X')^{1/\xi_3} \cdot D_3.$$

3.2 A Public Key Certification Scheme

We use a primitive that we call *non-interactive certification scheme*, which can be viewed as a signature scheme that only allows signing public keys from a specific public key space \mathcal{PK} . These keys should be signed while retaining algebraic properties that make it possible to prove knowledge of a public key and its corresponding certificate in an efficient way. In particular, signing hashed public keys is proscribed. In the interactive setting, several papers (e.g., [5,24]) describe efficient interactive protocols where a public key is jointly generated by a user and a certification authority in such a way that the user eventually obtains a certified public key and no one else learns the underlying private key. In this paper, we aim at minimizing the amount of interaction and let users generate their public key entirely on their own before requesting their certification. Ideally, we would like to be able to sign public keys without even requiring users to prove knowledge of their private key and, in particular, without having to first rewind a proof of knowledge so as to extract the user’s private key in the security proof.

A certification scheme consists of algorithms (**Setup**, **Certify**, **CertVerify**). The first one is run by a certification authority (CA) that, on input of global parameters cp , generates a key pair $(SK, PK) \leftarrow \text{Setup}(\text{cp})$. On input of cp , SK and a user’s public key pk , **Certify** generates a certificate cert_{pk} . The procedure **Verify** takes as input cp , PK , pk and cert_{pk} and outputs either 0 or 1.

Correctness mandates that $\text{CertVerify}(\text{cp}, PK, \text{pk}, \text{cert}_{\text{pk}}) = 1$ when $\text{cert}_{\text{pk}} \leftarrow \text{Certify}(\text{cp}, SK, \text{pk})$. The (strong) unforgeability [1] requirement is the same as in signature schemes. The adversary is supplied with a CA’s public key PK and access to a certification oracle $\text{Certify}(SK, \cdot)$ that can be queried for arbitrary public keys $\text{pk} \in \mathcal{PK}$. Her goal is to produce a new pair $(\text{pk}^*, \text{cert}_{\text{pk}^*}^*)$ (i.e., if pk^* was queried to $\text{Certify}(SK, \cdot)$, the output must have been different from $\text{cert}_{\text{pk}^*}^*$).

In the description hereafter, we assume common public parameters cp consisting of bilinear groups $(\mathbb{G}, \mathbb{G}_T)$ of prime order $p > 2^\lambda$, for a security parameter λ , and a generator $g \stackrel{\$}{\leftarrow} \mathbb{G}$. We also assume that certified public keys always consist of a fixed number n of group elements (i.e., $\mathcal{PK} = \mathbb{G}^n$).

INTUITION. The scheme borrows from the Boyen-Waters group signature [10] in the use of the HSDH assumption. A simplified version involves a CA that holds a public key $PK = (\Omega = g^\omega, A = (g, g)^\alpha, u, u_0, u_1 = g^{\beta_1}, \dots, u_n = g^{\beta_n})$, for private elements $SK = (\omega, \alpha, \beta_1, \dots, \beta_n)$, where n denotes the number of groups elements that certified public keys consist of. To certify a public key $\text{pk} = (X_1 = g^{x_1}, \dots, X_n = g^{x_n})$, the CA chooses an exponent $c_{\text{ID}} \stackrel{\$}{\leftarrow} \mathbb{Z}_p^*$ and computes $S_1 = (g^\alpha)^{1/(\omega+c_{\text{ID}})}, S_2 = g^{c_{\text{ID}}}, S_3 = u^{c_{\text{ID}}}, S_4 = (u_0 \cdot \prod_{i=1}^n X_i^{\beta_i})^{c_{\text{ID}}}$ and $S_5 = (S_{5,1}, \dots, S_{5,n}) = (X_1^{c_{\text{ID}}}, \dots, X_n^{c_{\text{ID}}})$. Verification then checks whether $e(S_1, \Omega \cdot S_2) = A$ and $e(S_2, u) = e(g, S_3)$ as in [10]. It must also be checked that $e(S_4, g) = e(u_0, S_2) \cdot \prod_{i=1}^n e(u_i, S_{5,i})$ and $e(S_{5,i}, g) = e(X_i, S_2)$ for $i = 1, \dots, n$.

The security of this simplified scheme can only be proven if, when answering certification queries, the simulator can control the private keys (x_1, \dots, x_n) and force them to be random values of its choice. To allow the simulator to sign arbitrary public keys without knowing the private keys, we modify the scheme so that the CA rather signs commitments (calculated as in the trapdoor commitment of section 3.1) to public key elements X_1, \dots, X_n . In the security proof, the simulator first generates a signature on n commitments $\vec{C}_i = (C_{i,1}, C_{i,2}, C_{i,3})$ to $1_{\mathbb{G}}$ that are all generated in such a way that it knows $\log_g(C_{i,j})$ for $i = 1, \dots, n$ and $j = 1, 2, 3$. Using the trapdoor of the commitment scheme, it can then open \vec{C}_i to any arbitrary public key element X_i without knowing $\log_g(X_i)$.

This use of the trapdoor commitment is reminiscent of a technique (notably used in [18]) to construct signature schemes in the standard model using chameleon hash functions [32]: the simulator first signs messages of its choice using a basic signature scheme and then “equivocates” the chameleon hashes to make them correspond to adversarially-chosen messages.

Setup(cp): given common public parameters $\text{cp} = \{g, \mathbb{G}, \mathbb{G}_T\}$, select $u, u_0 \stackrel{\$}{\leftarrow} \mathbb{G}, \alpha, \omega \stackrel{\$}{\leftarrow} \mathbb{Z}_p^*$ and set $A = e(g, g)^\alpha, \Omega = g^\omega$. Pick $\beta_{i,1}, \beta_{i,2}, \beta_{i,3} \stackrel{\$}{\leftarrow} \mathbb{Z}_p^*$ and set $\vec{u}_i = (u_{i,1}, u_{i,2}, u_{i,3}) = (g^{\beta_{i,1}}, g^{\beta_{i,2}}, g^{\beta_{i,3}})$ for $i = 1, \dots, n$. Choose $f, f_1, f_2, f_3, f_{3,1}, f_{3,2}, f_{3,3} \stackrel{\$}{\leftarrow} \mathbb{G}$ that define a commitment key consisting of vectors $\vec{f}_1 = (f_1, 1, f), \vec{f}_2 = (1, f_2, f)$ and $\vec{f}_3 = (f_{3,1}, f_{3,2}, f_{3,3})$. Define the private/public key pair as $SK = (\alpha, \omega, \{\vec{\beta}_i = (\beta_{i,1}, \beta_{i,2}, \beta_{i,3})\}_{i=1, \dots, n})$ and

$$PK = \left(\mathbf{f} = (\vec{f}_1, \vec{f}_2, \vec{f}_3), A = e(g, g)^\alpha, \Omega = g^\omega, u, u_0, \{\vec{u}_i\}_{i=1, \dots, n} \right).$$

Certify(cp, SK, pk): parse SK as $(\alpha, \omega, \{\vec{\beta}_i\}_{i=1, \dots, n})$, pk as (X_1, \dots, X_n) and do the following.

1. For each $i \in \{1, \dots, n\}$, pick $\phi_{i,1}, \phi_{i,2}, \phi_{i,3} \stackrel{\$}{\leftarrow} \mathbb{Z}_p^*$ and compute a commitment $C_i = (C_{i,1}, C_{i,2}, C_{i,3}) = (f_1^{\phi_{i,1}} \cdot f_{3,1}^{\phi_{i,3}}, f_2^{\phi_{i,2}} \cdot f_{3,2}^{\phi_{i,3}}, X_i \cdot f^{\phi_{i,1} + \phi_{i,2}} \cdot f_{3,3}^{\phi_{i,3}})$ and the matching de-commitment $(D_{i,1}, D_{i,2}, D_{i,3}) = (f^{\phi_{i,1}}, f^{\phi_{i,2}}, f^{\phi_{i,3}})$.
2. Choose $c_{\text{ID}} \stackrel{\$}{\leftarrow} \mathbb{Z}_p^*$, compute $S_1 = (g^\alpha)^{1/(\omega+c_{\text{ID}})}, S_2 = g^{c_{\text{ID}}}, S_3 = u^{c_{\text{ID}}}$ and

$$S_4 = \left(u_0 \cdot \prod_{i=1}^n (C_{i,1}^{\beta_{i,1}} \cdot C_{i,2}^{\beta_{i,2}} \cdot C_{i,3}^{\beta_{i,3}}) \right)^{c_{\text{ID}}}$$

$$S_5 = \{(S_{5,i,1}, S_{5,i,2}, S_{5,i,3})\}_{i=1, \dots, n} = \{(C_{i,1}^{c_{\text{ID}}}, C_{i,2}^{c_{\text{ID}}}, C_{i,3}^{c_{\text{ID}}})\}_{i=1, \dots, n}$$

Return $\text{cert}_{\text{pk}} = \left(\{(C_{i,1}, C_{i,2}, C_{i,3}), (D_{i,1}, D_{i,2}, D_{i,3})\}_{i=1, \dots, n}, S_1, S_2, S_3, S_4, S_5 \right)$.

CertVerify(cp, PK, pk, cert_{pk}): parse pk as (X_1, \dots, X_n) and cert_{pk} as above. Return 1 if, for $i = 1, \dots, n$, it holds that $X_i \in \mathbb{G}$ and

$$e(C_{i,1}, f) = e(f_1, D_{i,1}) \cdot e(f_{3,1}, D_{i,3}) \quad (1)$$

$$e(C_{i,2}, f) = e(f_2, D_{i,2}) \cdot e(f_{3,2}, D_{i,3}) \quad (2)$$

$$e(C_{i,3}, f) = e(X_i \cdot D_{i,1} \cdot D_{i,2}, f) \cdot e(f_{3,3}, D_{i,3}), \quad (3)$$

and if the following checks are also satisfied. Otherwise, return 0.

$$e(S_1, \Omega \cdot S_2) = A \quad (4)$$

$$e(S_2, u) = e(g, S_3) \quad (5)$$

$$e(S_4, g) = e(u_0, S_2) \cdot \prod_{i=1}^n (e(u_{i,1}, S_{5,i,1}) \cdot e(u_{i,2}, S_{5,i,2}) \cdot e(u_{i,3}, S_{5,i,3})), \quad (6)$$

$$e(S_{5,i,j}, g) = e(C_{i,j}, S_2) \quad \text{for } i = 1, \dots, n, j = 1, 2, 3 \quad (7)$$

A certificate comprises $9n + 4$ group elements. It would be interesting to avoid this linear dependency on n without destroying the algebraic properties that render the scheme compatible with Groth-Sahai proofs.

Regarding the security of this scheme, the idea of the proof of the following theorem is sketched in appendix [A](#). Due to space limitation, the complete proof is detailed in the full version of the paper.

Theorem 1. *The scheme is a secure non-interactive certification system if the HSDH, FlexDH and S2P problems are all hard in \mathbb{G} .*

We believe that the above certification scheme is of interest in its own right. For instance, it can be used to construct non-frameable group signatures that are secure in the concurrent join model of [30](#) without resorting to random oracles. To the best of our knowledge, the Kiayias-Yung construction [30](#) has remained the only scalable group signature where joining supports concurrency at both ends while requiring the smallest amount of interaction. In the standard model, our certification scheme thus appears to provide the first² way to achieve the same result. In this case, we have $n = 1$ (since prospective group members only need to certify one group element if non-frameability is ensured by signing messages as in Groth's group signature [24](#)) so that membership certificates comprise 13 group elements and their shape is fully compatible with GS proofs.

² Non-frameable group signatures described in [19,9](#) achieve concurrent security by having the prospective user generate an extractable commitment to some secret exponent (which the simulator can extract without rewinding using the trapdoor of the commitment) and prove that the committed value is the discrete log. of a public value. In the standard model, this technique requires interaction and the proof should be simulatable in zero-knowledge when proving security against framing attacks. Another technique [21](#) requires users to prove knowledge of their secret exponent using Groth-Sahai non-interactive proofs. It is nevertheless space-demanding as each bit of committed exponent requires its own extractable GS commitment.

3.3 Public Key Encryption Schemes Based on the Linear Problem

We need cryptosystems based on the DLIN assumption. The first one is Shacham’s variant [37] of Cramer-Shoup [17] and, since it is key-private [3], we use it to encrypt witnesses. We also use Kiltz’s tag-based encryption (TBE) scheme [31], where the validity of ciphertexts is publicly verifiable, to encrypt receivers’ public keys under the public key of the opening authority.

SHACHAM’S LINEAR CRAMER-SHOUP. If we assume public generators g_1, g_2, g that are parts of public parameters, each receiver’s public key is made of $n = 6$ group elements

$$\begin{aligned} X_1 &= g_1^{x_1} g^x & X_3 &= g_1^{x_3} g^y & X_5 &= g_1^{x_5} g^z \\ X_2 &= g_2^{x_2} g^x & X_4 &= g_2^{x_4} g^y & X_6 &= g_2^{x_6} g^z. \end{aligned}$$

To encrypt $m \in \mathbb{G}$ under the label L , the sender picks $r, s \stackrel{\$}{\leftarrow} \mathbb{Z}_p^*$ and computes $\psi_{\text{CS}} = (U_1, U_2, U_3, U_4, U_5) = (g_1^r, g_2^s, g^{r+s}, m \cdot X_5^r X_6^s, (X_1 X_3^\alpha)^r \cdot (X_2 X_4^\alpha)^s)$, where $\alpha = H(U_1, U_2, U_3, U_4, L) \in \mathbb{Z}_p^*$ is a collision-resistant hash [3]. Given (ψ_{CS}, L) , the receiver computes α . He returns \perp if $U_5 \neq U_1^{x_1 + \alpha x_3} U_2^{x_2 + \alpha x_4} U_3^{x_6 + \alpha y}$ and $m = U_4 / (U_1^{x_5} U_2^{x_6} U_3^z)$ otherwise.

KILTZ’S TAG-BASED ENCRYPTION SCHEME. In [31], Kiltz described a TBE scheme based on the same assumption. The public key is $(Y_1, Y_2, Y_3, Y_4) = (g^{y_1}, g^{y_2}, g^{y_3}, g^{y_4})$ if $g \in \mathbb{G}$ is part of public parameters. To encrypt $m \in \mathbb{G}$ under a tag $t \in \mathbb{Z}_p^*$, the sender picks $w_1, w_2 \stackrel{\$}{\leftarrow} \mathbb{Z}_p^*$ and computes

$$\psi_{\text{K}} = (V_1, V_2, V_3, V_4, V_5) = (Y_1^{w_1}, Y_2^{w_2}, (g^t Y_3)^{w_1}, (g^t Y_4)^{w_2}, m \cdot g^{w_1 + w_2})$$

To decrypt ψ_{K} , the receiver checks that $V_3 = V_1^{(t+y_3)/y_1}$, $V_4 = V_2^{(t+y_4)/y_2}$. If so, it outputs the plaintext $m = V_5 / (V_1^{1/y_1} V_2^{1/y_2})$. Unlike ψ_{CS} , the well-formedness of ψ_{K} is publicly verifiable in bilinear groups. The Canetti-Halevi-Katz [15] paradigm turns this scheme into a full-fledged CCA2 scheme by deriving the tag t from the verification key VK of a one-time signature, the private key SK of which is used to sign $(V_1, V_2, V_3, V_4, V_5)$.

4 A GE Scheme with Non-interactive Proofs

We build a non-interactive group encryption scheme for the Diffie-Hellman relation $\mathcal{R} = \{(X, Y), W\}$ where $e(g, W) = e(X, Y)$, for which the keys are $\text{pk}_{\mathcal{R}} = \{\mathbb{G}, \mathbb{G}_T, g\}$ and $\text{sk}_{\mathcal{R}} = \varepsilon$.

The construction slightly departs from the modular design of [29] in that commitments to the receiver’s public key and certificate are part of the proof (instead of the ciphertext), which simplifies the proof of message-security. The security of the scheme eventually relies on the HSDH, FlexDH and DLIN assumptions. All security proofs are available in the full version of the paper.

³ The proof of CCA2-security [17,37] only requires a universal one-way hash function (UOWHF) [34] but collision-resistance is required by the proof of key-privacy in [3].

SETUP_{init}(λ): choose bilinear groups $(\mathbb{G}, \mathbb{G}_T)$ of order $p > 2^\lambda$, $g \stackrel{\$}{\leftarrow} \mathbb{G}$ and $g_1 = g^{\alpha_1}$, $g_2 = g^{\alpha_2}$ with $\alpha_1, \alpha_2 \stackrel{\$}{\leftarrow} \mathbb{Z}_p^*$. Define $\vec{g}_1 = (g_1, 1, g)$, $\vec{g}_2 = (1, g_2, g)$ and $\vec{g}_3 = \vec{g}_1^{\xi_1} \odot \vec{g}_2^{\xi_2}$ with $\xi_1, \xi_2 \stackrel{\$}{\leftarrow} \mathbb{Z}_p^*$, which form a CRS $\mathbf{g} = (\vec{g}_1, \vec{g}_2, \vec{g}_3)$ for the perfect soundness setting. Select a strongly unforgeable (as defined in [1]) one time signature scheme $\Sigma = (\mathcal{G}, \mathcal{S}, \mathcal{V})$ and a random member $H : \{0, 1\}^* \rightarrow \mathbb{Z}_p$ of a collision-resistant hash family. Public parameters consists of $\text{param} = \{\lambda, \mathbb{G}, \mathbb{G}_T, g, \mathbf{g}, \Sigma, H\}$.

SETUP_{GM}(param): runs the setup algorithm of the certification scheme described in section 3.2 with $n = 6$. The obtained public key consists of $\text{pk}_{\text{GM}} = \left(\mathbf{f}, A = e(g, g)^\alpha, \Omega = g^\omega, u, u_0, \{\bar{u}_i\}_{i=1, \dots, 6} \right)$ and the matching private key is $\text{sk}_{\text{GM}} = (\alpha, \omega, \{\bar{\beta}_i = (\beta_{i,1}, \beta_{i,2}, \beta_{i,3})\}_{i=1, \dots, 6})$.

SETUP_{OA}(param): generates $\text{pk}_{\text{OA}} = (Y_1, Y_2, Y_3, Y_4) = (g^{y_1}, g^{y_2}, g^{y_3}, g^{y_4})$, as a public key for Kiltz's tag-based encryption scheme [31], and the corresponding private key as $\text{sk}_{\text{OA}} = (y_1, y_2, y_3, y_4)$.

JOIN: the user sends a linear Cramer-Shoup public key $\text{pk} = (X_1, \dots, X_6) \in \mathbb{G}^6$ to the GM and obtains a certificate

$$\text{cert}_{\text{pk}} = \left(\{(C_{i,1}, C_{i,2}, C_{i,3}), (D_{i,1}, D_{i,2}, D_{i,3})\}_{i=1, \dots, 6}, S_1, S_2, S_3, S_4, S_5 \right).$$

ENC($\text{pk}_{\text{GM}}, \text{pk}_{\text{OA}}, \text{pk}, \text{cert}_{\text{pk}}, W, L$): to encrypt $W \in \mathbb{G}$ such that $((X, Y), W) \in \mathcal{R}$ (for public elements $X, Y \in \mathbb{G}$), parse pk_{GM} , pk_{OA} and pk as above and do the following.

1. Generate a one-time signature key pair $(\text{SK}, \text{VK}) \leftarrow \mathcal{G}(\lambda)$.
2. Choose $r, s \stackrel{\$}{\leftarrow} \mathbb{Z}_p^*$ and compute a linear CS encryption of W , the result of which is denoted by ψ_{CS} , under the label $L_1 = L \parallel \text{VK}$ as per section 3.3 (and using the collision-resistant hash function specified by param).
3. For $i = 1, \dots, 6$, choose $w_{i,1}, w_{i,2} \stackrel{\$}{\leftarrow} \mathbb{Z}_p^*$ and encrypt X_i under pk_{OA} using Kiltz's TBE with the tag VK as described in section 3.3. Let $\psi_{\mathcal{K}_i}$ be the ciphertexts.
4. Set the ciphertext ψ as $\psi = \text{VK} \parallel \psi_{\text{CS}} \parallel \psi_{\mathcal{K}_1} \parallel \dots \parallel \psi_{\mathcal{K}_6} \parallel \sigma$ where σ is obtained as $\sigma = \mathcal{S}(\text{SK}, (\psi_{\text{CS}} \parallel \psi_{\mathcal{K}_1} \parallel \dots \parallel \psi_{\mathcal{K}_6} \parallel L))$.

Return (ψ, L) and coins_ψ consist of $\{(w_{i,1}, w_{i,2})\}_{i=1, \dots, 6}, (r, s)$. If the one-time signature of [23] is used, VK and σ take 3 and 2 group elements, respectively, so that ψ comprises 40 group elements.

$\mathcal{P}(\text{pk}_{\text{GM}}, \text{pk}_{\text{OA}}, \text{pk}, \text{cert}_{\text{pk}}, (X, Y), W, \psi, L, \text{coins}_\psi)$: parse pk_{GM} , pk_{OA} , pk and ψ as above. Conduct the following steps.

1. Generate commitments (as explained in section 2.3) to the $9n + 4 = 58$ group elements that cert_{pk} consists of. The resulting overall commitment $\text{com}_{\text{cert}_{\text{pk}}}$ contains 184 group elements.
2. Generate commitments to the public key elements $\text{pk} = (X_1, \dots, X_6)$ and obtain $\text{com}_{\text{pk}} = \{\text{com}_{X_i}\}_{i=1, \dots, 6}$, which consists of 18 group elements.
3. Generate a proof $\pi_{\text{cert}_{\text{pk}}}$ that $\text{com}_{\text{cert}_{\text{pk}}}$ is a commitment to a valid certificate for the public key contained in com_{pk} . For each $i = 1, \dots, 6$,

relations (1)-(3) cost 9 elements to prove (and thus 54 elements altogether). The quadratic equation (4) takes 9 elements and linear ones (5)-(6) both require 3 elements. Finally, (7) is a set of 18 linear equations which demand 54 elements altogether. The whole proof $\pi_{\text{cert}_{\text{pk}}}$ thus takes 123 group elements.

- For $i = 1, \dots, 6$, generate a NIZK proof $\pi_{\text{eq-key},i}$ that com_{X_i} (which is part of com_{pk}) and ψ_{K_i} are encryptions of the same X_i . If ψ_{K_i} comprises $(V_{i,1}, V_{i,2}, V_{i,5}) = (Y_1^{w_{i,1}}, Y_2^{w_{i,2}}, X_i \cdot g^{w_{i,1}+w_{i,2}})$ and com_{X_i} is parsed as $(c_{X_{i1}}, c_{X_{i2}}, c_{X_{i3}}) = (g_1^{\theta_{i1}} \cdot g_{3,1}^{\theta_{i3}}, g_2^{\theta_{i2}} \cdot g_{3,2}^{\theta_{i3}}, X_i \cdot g^{\theta_{i1}+\theta_{i2}} \cdot g_{3,3}^{\theta_{i3}})$, where $w_{i,1}, w_{i,2} \in \text{coins}_\psi$, $\theta_{i1}, \theta_{i2}, \theta_{i3} \in \mathbb{Z}_p^*$ and $\vec{g}_3 = (g_{3,1}, g_{3,2}, g_{3,3})$, this amounts to prove knowledge of values $w_{i,1}, w_{i,2}, \theta_{i1}, \theta_{i2}, \theta_{i3}$ such that

$$\left(\frac{V_{i,1}}{c_{X_{i1}}}, \frac{V_{i,2}}{c_{X_{i2}}}, \frac{V_{i,5}}{c_{X_{i3}}} \right) = (Y_1^{w_{i,1}} \cdot g_1^{-\theta_{i1}} \cdot g_{3,1}^{-\theta_{i3}}, \\ Y_2^{w_{i,2}} \cdot g_2^{-\theta_{i2}} \cdot g_{3,2}^{-\theta_{i3}}, g^{w_{i,1}+w_{i,2}-\theta_{i1}-\theta_{i2}} \cdot g_{3,3}^{-\theta_{i3}}).$$

Committing to $w_{i,1}, w_{i,2}, \theta_{i1}, \theta_{i2}, \theta_{i3}$ introduces 90 group elements whereas the above relations only require two elements each. Overall, proof elements $\pi_{\text{eq-key},1}, \dots, \pi_{\text{eq-key},6}$ incur 126 elements.

- Generate a NIZK proof $\pi_{\text{val-enc}}$ that $\psi_{\text{CS}} = (U_1, U_2, U_3, U_4, U_5)$ is a valid CS encryption. This requires to commit to underlying encryption exponents $r, s \in \text{coins}_\psi$ and prove that $U_1 = g_1^r$, $U_2 = g_2^s$, $U_3 = g^{r+s}$ (which only takes 3 times 2 elements as base elements are public) and $U_5 = (X_1 X_3^\alpha)^r (X_2 X_4^\alpha)^s$ (which takes 9 elements since base elements are themselves variables). Including commitments com_r and com_s to exponents r and s , $\pi_{\text{val-enc}}$ demands 21 group elements overall.
- Generate a NIZK proof $\pi_{\mathcal{R}}$ that ψ_{CS} encrypts a group element $W \in \mathbb{G}$ such that $((X, Y), W) \in \mathcal{R}$. To this end, generate a commitment $\text{com}_W = (c_{W,1}, c_{W,2}, c_{W,3}) = (g_1^{\theta_1} \cdot g_{3,1}^{\theta_3}, g_2^{\theta_2} \cdot g_{3,2}^{\theta_3}, W \cdot g^{\theta_1+\theta_2} \cdot g_{3,3}^{\theta_3})$ and prove that the underlying W is the same as the one for which $U_4 = W \cdot X_5^r X_6^s$ in ψ_{CS} . In other words, prove knowledge of $r, s, \theta_1, \theta_2, \theta_3$ such that

$$\left(\frac{U_1}{c_{W,1}}, \frac{U_2}{c_{W,2}}, \frac{U_4}{c_{W,3}} \right) = (g_1^{r-\theta_1} \cdot g_{3,1}^{-\theta_3}, \\ g_2^{s-\theta_2} \cdot g_{3,2}^{-\theta_3}, g^{-\theta_1-\theta_2} \cdot g_{3,3}^{-\theta_3} \cdot X_5^r \cdot X_6^s). \quad (8)$$

Commitments to r, s are already part of $\pi_{\text{val-enc}}$. Committing to $\theta_1, \theta_2, \theta_3$ takes 9 elements. Proving the first two relations of (8) requires 4 elements whereas the third one is quadratic and its proof is 9 elements. Proving the linear pairing-product relation $e(g, W) = e(X, Y)$ in NIZK⁴ demands 9 elements. Since $\pi_{\mathcal{R}}$ includes com_W , it entails a total of 34 elements.

⁴ It requires to introduce an auxiliary variable \mathcal{X} and prove that $e(g, W) = e(\mathcal{X}, Y)$ and $\mathcal{X} = X$, for variables W, \mathcal{X} and constants g, X, Y . The two proofs take 3 elements each and 3 elements are needed to commit to \mathcal{X} .

The proof $\pi_\psi = com_{cert_{pk}} || com_{pk} || \pi_{cert_{pk}} || \pi_{eq-key,1} || \dots || \pi_{eq-key,6} || \pi_{val-enc} || \pi_{\mathcal{R}}$ eventually takes 516 elements.

$\mathcal{V}(\text{param}, \psi, L, \pi_\psi, \text{pk}_{GM}, \text{pk}_{OA})$: parse pk_{GM} , pk_{OA} , pk , ψ and π_ψ as above. Return 1 if and only if $\mathcal{V}(\text{VK}, \sigma, (\psi_{CS} || \psi_{K_1} || \dots || \psi_{K_6} || L)) = 1$, all proofs verify and if $\psi_{K_1}, \dots, \psi_{K_6}$ are all valid tag-based encryptions w.r.t. the tag VK .

$\text{DEC}(\text{sk}, \psi, L)$: parse the ciphertext ψ as $\text{VK} || \psi_{CS} || \psi_{K_1} || \dots || \psi_{K_6} || \sigma$. Return \perp if $\mathcal{V}(\text{VK}, \sigma, (\psi_{CS} || \psi_{K_1} || \dots || \psi_{K_6} || L)) = 0$. Otherwise, use sk to decrypt (ψ_{CS}, L) .

$\text{OPEN}(\text{sk}_{OA}, \psi, L)$: parse the ciphertext ψ as $\text{VK} || \psi_{CS} || \psi_{K_1} || \dots || \psi_{K_6} || \sigma$. Return \perp if $\psi_{K_1}, \dots, \psi_{K_6}$ are not all valid TBE ciphertexts w.r.t. the tag VK or if $\mathcal{V}(\text{VK}, \sigma, (\psi_{CS} || \psi_{K_1} || \dots || \psi_{K_6} || L)) = 0$. Otherwise, decrypt $\psi_{K_1}, \dots, \psi_{K_6}$ using sk_{OA} and return the resulting $\text{pk} = (X_1, \dots, X_6)$.

From an efficiency standpoint, the length of ciphertexts is about 1.25 kB in an implementation using symmetric pairings with a 256-bit group order, which is more compact than in the Paillier-based scheme of [29] where ciphertexts take 2.5 kB using 1024-bit moduli. Moreover, our proofs only require 16.125 kB, which is significantly cheaper than in the original GE scheme [29], where interactive proofs reach a communication cost of 70 kB to achieve a 2^{-50} knowledge error.

References

1. An, J.H., Dodis, Y., Rabin, T.: On the security of joint signature and encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 83–107. Springer, Heidelberg (2002)
2. Belenkiy, M., Chase, M., Kohlweiss, M., Lysyanskaya, A.: P-signatures and noninteractive anonymous credentials. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 356–374. Springer, Heidelberg (2008)
3. Bellare, M., Boldyreva, A., Desai, A., Pointcheval, D.: Key-privacy in public-key encryption. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 566–582. Springer, Heidelberg (2001)
4. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: ACM CCS 1993, pp. 62–73 (1993)
5. Boldyreva, A., Fischlin, M., Palacio, A., Warinschi, B.: A closer look at PKI: Security and efficiency. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 458–475. Springer, Heidelberg (2007)
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., Shacham, H.: Short group signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)
8. Boneh, D., Franklin, M.: Identity based encryption from the Weil pairing. SIAM J. of Computing 32(3), 586–615 (2003); Extended abstract in Crypto 2001, LNCS, vol. 2139, pp. 213–229 (2001)
9. Boyen, X., Delerablée, C.: Expressive subgroup signatures. In: Ostrovsky, R., De Prisco, R., Visconti, I. (eds.) SCN 2008. LNCS, vol. 5229, pp. 185–200. Springer, Heidelberg (2008)

10. 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)
11. Camenisch, J., Chandran, N., Shoup, V.: A public key encryption scheme secure against key dependent chosen plaintext and adaptive chosen ciphertext attacks. In: Ghilardi, S. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 351–368. Springer, Heidelberg (2009)
12. Camenisch, J., Lysyanskaya, A.: A Signature Scheme with Efficient Protocols. In: Cimato, S., Galdi, C., Persiano, G. (eds.) SCN 2002. LNCS, vol. 2576, pp. 268–289. Springer, Heidelberg (2003)
13. 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)
14. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited. *Journal of the ACM* 51(4), 557–594 (2004)
15. Canetti, R., Halevi, S., Katz, J.: Chosen-ciphertext security from identity-based encryption. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 207–222. Springer, Heidelberg (2004)
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., 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)
18. Cramer, R., Shoup, V.: Signature schemes based on the strong rsa assumption. In: ACM CCS 1999, pp. 46–51 (1999)
19. Delerablée, C., Pointcheval, D.: Dynamic fully anonymous short group signatures. In: Nguyễn, P.Q. (ed.) VIETCRYPT 2006. LNCS, vol. 4341, pp. 193–210. Springer, Heidelberg (2006)
20. 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)
21. Fuchsbauer, G., Pointcheval, D.: Proofs on Encrypted Values in Bilinear Groups and an Application to Anonymity for Signatures. In: Shacham, H., Waters, B. (eds.) Pairing 2009. LNCS, vol. 5671, pp. 132–149. Springer, Heidelberg (2009)
22. Goldwasser, S., Tauman-Kalai, Y.: On the (In)security of the Fiat-Shamir Paradigm. In: FOCS 2003, pp. 102–115 (2003)
23. 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)
24. Groth, J.: Fully anonymous group signatures without random oracles. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 164–180. Springer, Heidelberg (2007)
25. Groth, J.: Homomorphic trapdoor commitments to group elements. *Cryptology ePrint Archive: Report 2009/007* (2009)
26. Groth, J., Lu, S.: A non-interactive shuffle with pairing based verifiability. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 51–67. Springer, Heidelberg (2007)
27. 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)

28. Halevi, S.: A Sufficient Condition for Key-Privacy. Cryptology ePrint Archive: Report 2005/005 (2005)
29. Kiayias, A., Tsiounis, Y., Yung, M.: Group encryption. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 181–199. Springer, Heidelberg (2007)
30. 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)
31. Kiltz, E.: Chosen-ciphertext security from tag-based encryption. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 581–600. Springer, Heidelberg (2006)
32. Krawczyk, H., Rabin, T.: Chameleon signatures. In: NDSS 2000 (2000)
33. Kunz-Jacques, S., Pointcheval, D.: About the security of MTI/C0 and MQV. In: De Prisco, R., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 156–172. Springer, Heidelberg (2006)
34. Naor, M., Yung, M.: Universal one-way hash functions and their cryptographic applications. In: STOC 1989, pp. 33–43 (1989)
35. 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)
36. Qin, B., Wu, Q., Susilo, W., Mu, Y., Wang, Y.: Publicly Verifiable Privacy-Preserving Group Decryption. In: Moti, Y., Liu, P., Lin, D. (eds.) Inscrypt 2008. LNCS, vol. 5487, pp. 72–83. Springer, Heidelberg (2008)
37. Shacham, H.: A Cramer-Shoup encryption scheme from the linear assumption and from progressively weaker linear variants. Cryptology ePrint Archive: Report 2007/074 (2007)
38. Shoup, V.: A proposal for the ISO standard for public-key encryption (version 2.1). manuscript (2001), <http://shoup.net/>

A Sketch of the Proof of Theorem 1

The security proof of the certification scheme considers three kinds of forgeries in the attack game.

- Type I forgeries: are such that the fake certificate $\text{cert}_{\text{pk}^*}^*$ contains a tuple of elements (S_1^*, S_2^*, S_3^*) that never appeared in outputs of certification queries.
- Type II forgeries: are such that $\text{cert}_{\text{pk}^*}^*$ contains a triple (S_1^*, S_2^*, S_3^*) that appeared in the output of some query but $\text{cert}_{\text{pk}^*}^*$ also contains commitments $\{(C_{i,1}^*, C_{i,2}^*, C_{i,3}^*)\}_{i=1,\dots,n}$ that do not match those in the output of that query.
- Type III forgeries: are such that (S_1^*, S_2^*, S_3^*) and $\{(C_{i,1}^*, C_{i,2}^*, C_{i,3}^*)\}_{i=1,\dots,n}$ are identical in $\text{cert}_{\text{pk}^*}^*$ and in the output of some certification query. On the other hand, the public key $\text{pk}^* = (X_1^*, \dots, X_n^*)$ is not the one that was certified in that query.

Type I forgeries are easily seen to break the HSDH assumption whereas Type II and Type III forgeries give rise to algorithms solving the FlexDH and S2P problems, respectively. Due to space limitations, the details are deferred to the full version of the paper. \square

On Black-Box Constructions of Predicate Encryption from Trapdoor Permutations

Jonathan Katz* and Arkady Yerukhimovich

Department of Computer Science, University of Maryland
{jkatz,arkady}@cs.umd.edu

Abstract. *Predicate encryption* is a recent generalization of identity-based encryption (IBE), broadcast encryption, attribute-based encryption, and more. A natural question is whether there exist *black-box* constructions of predicate encryption based on generic building blocks, e.g., trapdoor permutations. Boneh et al. (FOCS 2008) recently gave a negative answer for the specific case of IBE.

We show both negative and positive results. First, we identify a combinatorial property on the sets of predicates/attributes and show that, for any sets having this property, no black-box construction of predicate encryption from trapdoor permutations (or even CCA-secure encryption) is possible. Our framework implies the result of Boneh et al. as a special case, and also rules out, e.g., black-box constructions of forward-secure encryption and broadcast encryption (with many excluded users). On the positive side, we identify conditions under which predicate encryption schemes *can* be constructed based on any CPA-secure (standard) encryption scheme.

1 Introduction

In a *predicate encryption* scheme [6,13] an authority generates a master public key and a master secret key, and uses the master secret key to derive personal secret keys for individual users. A personal secret key corresponds to a predicate in some class \mathcal{F} , and ciphertexts are associated (by the sender) with an attribute in some set \mathbb{A} ; a ciphertext associated with the attribute $I \in \mathbb{A}$ can be decrypted by a secret key SK_f corresponding to the predicate $f \in \mathcal{F}$ if and only if $f(I) = 1$. The basic security guarantee provided by such schemes is that a ciphertext associated with an attribute I hides all information about the underlying message unless one has a personal secret key giving the explicit ability to decrypt; in other words, if an adversary \mathcal{A} holds keys $SK_{f_1}, \dots, SK_{f_\ell}$ for which $f_1(I) = \dots = f_\ell(I) = 0$, then \mathcal{A} should learn nothing about the message. (A formal definition is given later.)

By choosing \mathcal{F} and \mathbb{A} appropriately, predicate encryption yields as special cases many notions that are interesting in their own right. For example, by taking

* Work done while visiting IBM. Research supported by DARPA, and by the US Army Research Laboratory and the UK Ministry of Defence under agreement number W911NF-06-3-0001.

$\mathbb{A} = \{0, 1\}^n$ and letting $\mathcal{F} = \{f_{ID}\}_{ID \in \{0, 1\}^n}$ be the class of point functions (so that $f_{ID}(ID') = 1$ iff $ID = ID'$) we recover the notion of identity-based encryption (IBE) [19,4]. Similarly, it can be observed that predicate encryption encompasses fuzzy IBE [18], forward-secure (public-key) encryption [7], (public-key) broadcast encryption [9], attribute-based encryption [11,2,15], and more as special cases.

Most (though not all) existing constructions of predicate encryption schemes rely on bilinear maps. A natural question is: *what are the minimal assumptions on which predicate encryption can be based?* Of course, the answer will depend on the specific predicate class \mathcal{F} and attribute set \mathbb{A} of interest; in particular, Boneh and Waters [6] show that if \mathcal{F} is polynomial size then (for any \mathbb{A}) one can construct a predicate encryption scheme for $(\mathcal{F}, \mathbb{A})$ from any (standard) public-key encryption scheme. On the other hand, Boneh et al. [5] have recently shown that there is no *black-box* construction of IBE from trapdoor permutations.

1.1 Our Results

The specific question we consider is: *for which $(\mathcal{F}, \mathbb{A})$ can we construct a predicate encryption scheme over $(\mathcal{F}, \mathbb{A})$ based on CPA-secure encryption?* We show both negative and positive results. Before describing these results in more detail, we provide some background intuition.

A natural combinatorial construction of a predicate encryption scheme over some $(\mathcal{F}, \mathbb{A})$ from a CPA-secure encryption scheme $(\text{Gen}, \text{Enc}, \text{Dec})$ is as follows: The authority includes several public keys pk_1, \dots, pk_q in the master public key, and each personal secret key is some subset of the corresponding secret keys sk_1, \dots, sk_q . Encryption of a message m with respect to an attribute I requires “sharing” m in some way to yield m_1, \dots, m_q , and the resulting ciphertext is $\text{Enc}_{pk_1}(m_1), \dots, \text{Enc}_{pk_q}(m_q)$. Intuitively, this works if:

Correctness: Let $SK_f = \{sk_{i_1}, \dots, sk_{i_t}\}$ be a personal secret key for which $f(I) = 1$. Then the “shares” m_{i_1}, \dots, m_{i_t} should enable recovery of m .

Security: Let $\{sk_{i_1}, \dots, sk_{i_k}\} = \bigcup_{f \in \mathcal{F}: f(I)=0} SK_f$. Then the set of “shares” m_{i_1}, \dots, m_{i_k} should leak no information about m .¹

Roughly, our negative result can be interpreted as showing that this is essentially the *only* way to construct predicate encryption (in a black-box way) from CPA-secure encryption; our positive result shows how to implement the above for a specific class of predicate encryption schemes. We now provide further details.

Impossibility results. Our negative results are in the same model used by Boneh et al. [5], which builds on the model used in the seminal work of Impagliazzo and Rudich [12]. Specifically, as in [5] our negative results hold relative to a *random* oracle (with trapdoor) and so rule out black-box constructions from trapdoor permutations as well as from any (standard) CCA-secure public-key encryption scheme.

¹ This is stronger than what is required, but makes sense in a black-box setting where computational hardness comes only from the underlying CPA-secure scheme.

A slightly informal statement of our result follows. Fix $\{(\mathcal{F}_n, \mathbb{A}_n)\}_{n \in \mathbb{N}}$, a sequence of predicate classes and attribute sets indexed by the security parameter n . We say that $\{(\mathcal{F}_n, \mathbb{A}_n)\}_n$ can be q -covered if for every set system $\{S_f\}_{f \in \mathcal{F}_n}$ with $S_f \subseteq [q(n)]$ ($[q] \stackrel{\text{def}}{=} \{1, \dots, q\}$), there are polynomially-many predicates $f^*, f_1, \dots, f_p \in \mathcal{F}_n$ such that, with high probability:

1. $S_{f^*} \subseteq \bigcup_{i=1}^p S_{f_i}$.
2. There exists an $I \in \mathbb{A}_n$ with $f_1(I) = \dots = f_p(I) = 0$ but $f^*(I) = 1$.

$\{(\mathcal{F}_n, \mathbb{A}_n)\}_n$ is *easily covered* if it is q -covered for *every* polynomial q . We show:

Theorem. *If $\{(\mathcal{F}_n, \mathbb{A}_n)\}_n$ is easily covered, there is no black-box construction of a predicate encryption scheme over $\{(\mathcal{F}_n, \mathbb{A}_n)\}_n$ based on trapdoor permutations (or CCA-secure encryption).*

Intuitively, if $\{(\mathcal{F}_n, \mathbb{A}_n)\}_n$ is easily covered then the combinatorial approach discussed earlier cannot work: letting $q(n)$ be the (necessarily) polynomial number of keys for the underlying (standard) encryption scheme, no matter how the secret keys $\{sk_i\}_{i=1}^q$ are apportioned to the personal secret keys $\{SK_f\}_{f \in \mathcal{F}_n}$, an adversary can carry out the following attack (cf. Definition [2](#), below):

1. Request the keys $SK_{f_1}, \dots, SK_{f_p}$, where each $SK_{f_i} = \{sk_1, \dots\} \subseteq \{sk_i\}_{i=1}^q$.
2. Request the challenge ciphertext C to be encrypted using an attribute I for which $f_1(I) = \dots = f_p(I) = 0$ but $f^*(I) = 1$.
3. Compute the key $SK_{f^*} \subseteq \bigcup_i SK_{f_i}$ and use this key to decrypt C .

This constitutes a valid attack since SK_{f^*} suffices to decrypt C yet the adversary only requested $SK_{f_1}, \dots, SK_{f_p}$, none of which suffices on its own to decrypt C .

Turning this intuition into a formal proof must, in particular, implicitly show that the combinatorial approach sketched earlier is essentially the *only* black-box approach to building predicate encryption schemes from trapdoor permutations. Moreover, we actually prove a stronger *quantitative* version of the above theorem showing, roughly, that if $\{(\mathcal{F}_n, \mathbb{A}_n)\}_n$ is q -covered then any predicate encryption scheme over $\{(\mathcal{F}_n, \mathbb{A}_n)\}_n$ must use at least $q + 1$ underlying encryption keys.

One might wonder whether the “easily covered” condition is useful for determining whether there exist black-box constructions of predicate encryption schemes over $\{(\mathcal{F}_n, \mathbb{A}_n)\}_n$ of interest. We show that it is, in that the following corollary can be proven fairly easily given the above:

Corollary. *There are no black-box constructions of (1) identity-based encryption, (2) forward-secure encryption (for a super-polynomial number of time periods), or (3) broadcast encryption (where a super-polynomial number of users can be excluded) from trapdoor permutations.*

The first result was shown in [5](#); the point is that our impossibility result strictly generalizes theirs. Moreover, as indicated earlier, we prove a *quantitative* version of their result (as well as all other results stated in the above corollary).

Positive result. On the positive side, we show that the combinatorial approach suggested at the outset *can* be implemented for $\{(\mathcal{F}_n, \mathbb{A}_n)\}_n$ having the following

property: for each $I \in \mathbb{A}_n$ there are at most polynomially-many $f \in \mathcal{F}_n$ for which $f(I) = 0$; i.e., for each I there are at most polynomially-many predicates that are “excluded”. (The positive result from [6], where there are only polynomially-many predicates, is thus obtained as a corollary.) This is proved by analogy to broadcast encryption, using the combinatorial techniques from [14].

1.2 Comparison to the Results of Boneh et al.

Our proof relies heavily on the impossibility result from [5]. Our contribution lies in finding the right combinatorial generalization (specifically, the “easily covered” property described earlier) of the *specific* property used by Boneh et al. for the particular case of IBE, adapting their proof to our setting, and applying their ideas to the more general case of predicate encryption. Our generalization, in turn, allows us to show impossibility for several cryptosystems of interest besides IBE (cf. the corollary stated earlier), as well as to give quantitative versions of their earlier result. Our positive results have no analogue in [5].

2 Definitions

2.1 Predicate Encryption

We provide a functional definition of predicate encryption, followed by a weak definition of security that we use when proving impossibility and the standard definition of security [13] that we use when proving our positive result.

Definition 1. Fix $\{(\mathcal{F}_n, \mathbb{A}_n)\}_{n \in \mathbb{N}}$, where \mathcal{F}_n is a set of (efficiently computable) predicates over the set of attributes \mathbb{A}_n . A predicate encryption scheme over $\{\mathcal{F}_n, \mathbb{A}_n\}_{n \in \mathbb{N}}$ consists of four PPT algorithms (Setup, KeyGen, Enc, Dec) such that:

- Setup is a deterministic algorithm that takes as input a master secret key $MSK \in \{0, 1\}^n$ and outputs a master public key MPK .
- KeyGen is a deterministic algorithm that takes as input the master secret key MSK and a predicate $f \in \mathcal{F}_n$ and outputs a secret key $SK_f = \text{KeyGen}_{MSK}(f)$. (The assumption that KeyGen is deterministic is without loss of generality, since MSK may include a key for a pseudorandom function.)
- Enc takes as input the public key MPK , an attribute $I \in \mathbb{A}_n$, and a bit b . It outputs a ciphertext $C \leftarrow \text{Enc}_{MPK}(I, b)$.
- Dec takes as input a secret key SK_f and ciphertext C . It outputs either a bit b or the distinguished symbol \perp .

It is required that for all n , all $MSK \in \{0, 1\}^n$ and $MPK = \text{Setup}(MSK)$, all $f \in \mathcal{F}_n$ and $SK_f = \text{KeyGen}_{MSK}(f)$, all $I \in \mathbb{A}_n$, and all $b \in \{0, 1\}$, that if $f(I) = 1$ then $\text{Dec}_{SK_f}(\text{Enc}_{MPK}(I, b)) = b$.

Definition 2. A predicate encryption scheme over $(\mathcal{F}, \mathbb{A})$ is weakly payload hiding if the advantage of any PPT adversary \mathcal{A} in the following game is negligible:

1. $\mathcal{A}(1^n)$ outputs $I^* \in \mathbb{A}_n$ and $(f_1, \dots, f_p) \in \mathcal{F}_n$ such that $f_i(I^*) = 0$ for all i .
2. Choose $MSK \leftarrow \{0, 1\}^n$; let $MPK := \text{Setup}(MSK)$ and set $SK_{f_i} := \text{KeyGen}(MSK, f_i)$ for all i . Choose $b \leftarrow \{0, 1\}$, and compute the ciphertext $C^* \leftarrow \text{Enc}_{MPK}(I^*, b)$. Then \mathcal{A} is given $(MPK, SK_{f_1}, \dots, SK_{f_p}, C^*)$.
3. \mathcal{A} outputs b' and succeeds if $b' = b$.

The advantage of \mathcal{A} is defined as $|\Pr[\mathcal{A} \text{ succeeds}] - \frac{1}{2}|$.

Definition 3. A predicate encryption scheme over $(\mathcal{F}, \mathbb{A})$ is payload hiding if the advantage of any PPT adversary \mathcal{A} in the following game is negligible:

1. A random $MSK \in \{0, 1\}^n$ is chosen, and \mathcal{A} is given $MPK := \text{Setup}(MSK)$.
2. \mathcal{A} adaptively requests keys SK_{f_1}, \dots corresponding to predicates $f_1, \dots \in \mathcal{F}_n$.
3. At some point, \mathcal{A} outputs $I^* \in \mathbb{A}_n$. A random $b \in \{0, 1\}$ is chosen and \mathcal{A} is given the ciphertext $C^* \leftarrow \text{Enc}_{MPK}(I^*, b)$. \mathcal{A} may continue to request keys for predicates of its choice.
4. \mathcal{A} outputs b' and succeeds if (1) \mathcal{A} never requested a key for a predicate f with $f(I^*) = 1$, and (2) $b' = b$.

The advantage of \mathcal{A} is defined as $|\Pr[\mathcal{A} \text{ succeeds}] - \frac{1}{2}|$.

Our construction of Section 5 can be modified to achieve the even stronger notion of attribute hiding; we refer to [13] for a definition.

2.2 A Random Trapdoor Permutation Oracle

We assume the reader is familiar with the usual model in which black-box impossibility results are proved; see [12, 17, 5] for further details. We show an oracle \mathcal{O} relative to which trapdoor permutations and CCA-secure encryption exist, yet any construction of a predicate encryption scheme (for certain $(\mathcal{F}, \mathbb{A})$) relative to \mathcal{O} is insecure against a polynomial-time adversary given access to \mathcal{O} and a PSPACE oracle. Our oracle $\mathcal{O} = (g, e, d)$ is defined as follows, for each $n \in \mathbb{N}$:

- g is chosen uniformly from the space of permutations on $\{0, 1\}^n$. We view g as taking a secret key sk as input, and returning a public key pk .
- $e : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ maps a public key pk and a “message” $m \in \{0, 1\}^n$ to a “ciphertext” $c \in \{0, 1\}^n$. It is chosen uniformly subject to the constraint that $e(pk, \cdot)$ is a permutation on $\{0, 1\}^n$ for every pk .
- $d : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ maps a secret key sk and a ciphertext c to a message m . We require that $d(sk, c)$ outputs the unique m for which $e(g(sk), m) = c$.

With overwhelming probability \mathcal{O} is a trapdoor permutation [10, 5]. Moreover, since the components of \mathcal{O} are chosen at random subject to the above constraints (and not with some “defect” as in, e.g., [10]), \mathcal{O} implies CCA-secure encryption [1].

We denote a query α to \mathcal{O} as, e.g., $\alpha \stackrel{\text{def}}{=} [g(sk) = pk]$ and similarly for e and d queries. In describing our attack in the next section, we often use a partial oracle \mathcal{O}' that is defined only on some subset of the possible inputs. We always enforce that such oracles be consistent:

Definition 4. A partial oracle $\mathcal{O}' = (g', e', d')$ is consistent if:

1. For every $pk \in \{0, 1\}^n$, the (partial) function $e'(pk, \cdot)$ is one-to-one.
2. For every $sk \in \{0, 1\}^n$, the (partial) function $d'(sk, \cdot)$ is one-to-one.
3. For all $x \in \{0, 1\}^n$, and all sk such that $g'(sk) = pk$ is defined, the value $e'(pk, x) = c$ is defined if and only if $d'(sk, c) = x$ is defined.

3 An Impossibility Result for Predicate Encryption

We define a combinatorial property on $(\mathcal{F}_n, \mathbb{A}_n)$ and formally state our impossibility result. We describe in Section 3.1 an adversary \mathcal{A} attacking any black-box construction of a predicate encryption scheme satisfying the conditions of our theorem; an analysis of \mathcal{A} is given in Appendix A and the full version.

Fix a set \mathcal{F} and a positive integer q , and let $[q] \stackrel{\text{def}}{=} \{1, \dots, q\}$. An \mathcal{F} -set system over $[q]$ is a collection of sets $\{S_f\}_{f \in \mathcal{F}}$ where each $f \in \mathcal{F}$ is associated with a set $S_f \subseteq [q]$.

Definition 5. Let $\{(\mathcal{F}_n, \mathbb{A}_n)\}_{n \in \mathbb{N}}$ be a sequence of predicates and attributes. We say $\{(\mathcal{F}_n, \mathbb{A}_n)\}_{n \in \mathbb{N}}$ can be q -covered if there exist PPT algorithms (A_1, A_2, A_3) , where $A_2(1^n, f)$ is deterministic and outputs $I \in \mathbb{A}_n$ with $f(I) = 1$, such that for n sufficiently large:

For any \mathcal{F}_n -set system $\{S_f\}_{f \in \mathcal{F}_n}$ over $[q(n)]$, if we compute

$$f^* \leftarrow A_1(1^n); \quad I^* := A_2(1^n, f^*); \quad f_1, \dots, f_p \leftarrow A_3(1^n, f^*),$$

then with probability at least $4/5$,

1. $S_{f^*} \subseteq \bigcup S_{f_i}$;
2. $f_i(I^*) = 0$ for all i .

$\{(\mathcal{F}_n, \mathbb{A}_n)\}_{n \in \mathbb{N}}$ is easily covered if it can be q -covered for every polynomial q .

Although the above definition may seem rather complex and hard to use, we show in Section 4 that it can be applied quite easily to several interesting classes of predicate encryption schemes. Moreover, the definition is natural given the attack we will describe in the following section.

A black-box construction of predicate encryption is q -bounded if each of its algorithms makes at most q queries to \mathcal{O} . We now state our main result:

Theorem 1. If $\{(\mathcal{F}_n, \mathbb{A}_n)\}$ can be q -covered, then there is no q -bounded black-box construction of a weakly payload-hiding predicate encryption scheme over $\{(\mathcal{F}_n, \mathbb{A}_n)\}$ from trapdoor permutations (or CCA-secure encryption).

Since each algorithm defining the predicate encryption scheme can make at most polynomially-many queries to its oracle, we have

Corollary 1. If $\{(\mathcal{F}_n, \mathbb{A}_n)\}$ is easily covered, there is **no** black-box construction of a weakly payload-hiding predicate encryption scheme over $\{(\mathcal{F}_n, \mathbb{A}_n)\}$ from trapdoor permutations (or CCA-secure encryption).

3.1 The Attack

Fix an $\{(\mathcal{F}_n, \mathbb{A}_n)\}$ that can be q -covered, and let $\text{PE} = (\text{Setup}, \text{KeyGen}, \text{Enc}, \text{Dec})$ be a predicate encryption scheme over $\{(\mathcal{F}_n, \mathbb{A}_n)\}$ each of whose algorithms makes at most $q = \text{poly}(n)$ queries to $\mathcal{O} = (g, e, d)$. We assume, without loss of generality, that before any algorithm of PE makes a query of the form $[d(sk, \star)]$, it first makes the query $[g(sk)]$.

We begin the proof of Theorem 1 by describing an adversary \mathcal{A} attacking PE. Adversary \mathcal{A} is given access to \mathcal{O} and makes a polynomial number of calls to this oracle; as described, \mathcal{A} is not efficient but it runs in polynomial time given access to a PSPACE-complete oracle (or if $\mathcal{P} = \mathcal{NP}$) and this suffices to prove black-box impossibility as in previous work [12, 17, 5]. Our description of the attack is directly motivated by the attacker described in [5].

Let A_1, A_2 , and A_3 be as guaranteed by Definition 5, and let $p = \text{poly}(n)$ bound the number of predicates output by A_3 . Throughout \mathcal{A} 's execution, when it makes a query to \mathcal{O} it stores the query and the response in a list L . We also require that before \mathcal{A} makes any query of the form $[d(sk, \star)]$, it first makes the query $[g(sk)]$. Furthermore, once the query $[g(sk) = pk]$ has been made then $[e(pk, x) = y]$ is added to L if and only if $[d(sk, y) = x]$ is added to L .

Setup and challenge. $\mathcal{A}(1^n)$ computes $f^* \leftarrow A_1(1^n)$, $I^* := A_2(1^n, f^*)$, and $(f_1, \dots, f_p) \leftarrow A_3(1^n, f^*)$. Then:

1. If $f_i(I^*) = 0$ for all i , then \mathcal{A} outputs (I^*, f_1, \dots, f_p) and receives the values $(MPK, SK_{f_1}, \dots, SK_{f_p}, C^*)$ from the challenger (cf. Definition 2).
2. Otherwise, \mathcal{A} aborts and outputs a random bit $b' \leftarrow \{0, 1\}$.

Step 1: Discovering important public keys. For $i = 1$ to p , adversary \mathcal{A} does the following:

1. Compute $I_{f_i} = A_2(1^n, f_i)$, and choose random $b \leftarrow \{0, 1\}$ and $r \leftarrow \{0, 1\}^n$.
2. Compute $\text{Dec}_{SK_{f_i}}^{\mathcal{O}}(\text{Enc}_{MPK}^{\mathcal{O}}(I_{f_i}, b; r))$, storing all \mathcal{O} -queries in the list L .

Step 2: Discovering frequent queries for I^* . \mathcal{A} repeats the following $q \cdot p^3$ times: Choose random $b \leftarrow \{0, 1\}$ and $r \leftarrow \{0, 1\}^n$; compute $\text{Enc}_{MPK}^{\mathcal{O}}(I^*, b; r)$, storing all \mathcal{O} -queries in L .

Step 3: Discovering secret queries and decrypting the challenge. \mathcal{A} chooses $k \leftarrow [q \cdot p^3]$ and runs the following k times.

1. \mathcal{A} uniformly generates a secret key MSK' and a consistent partial oracle \mathcal{O}' for which (1) $\text{Setup}^{\mathcal{O}'}(MSK') = MPK$; (2) for all i it holds that $\text{KeyGen}_{MSK'}^{\mathcal{O}'}(f_i) = SK_{f_i}$; (3) the oracle \mathcal{O}' is consistent with L ; and (4) the key $SK'_{f^*} \stackrel{\text{def}}{=} \text{KeyGen}_{MSK'}^{\mathcal{O}'}(f^*)$ is well-defined.

We denote by L' the set of queries in \mathcal{O}' that are not in L (the “invented queries”). Note that $|L'| \leq q \cdot (p+2)$, since at most q queries are made by Setup and KeyGen(f) makes at most q queries for each of $SK_{f^*}, SK_{f_1}, \dots, SK_{f_p}$.

2. \mathcal{A} chooses $b \leftarrow \{0, 1\}$ and $r \leftarrow \{0, 1\}^n$, and computes $C := \text{Enc}_{MPK}^{\mathcal{O}}(I^*, b; r)$ (storing all \mathcal{O} -queries in L). For an oracle \mathcal{O}' defined below, \mathcal{A} then does:
 - (a) In iteration $k' < k$, adversary \mathcal{A} computes $\text{Dec}_{SK_{f^*}}^{\mathcal{O}'}(C)$.
 - (b) In iteration k , adversary \mathcal{A} computes $b' = \text{Dec}_{SK_{f^*}}^{\mathcal{O}'}(C^*)$.

Output: \mathcal{A} Outputs the bit b' computed in the k^{th} iteration of step 3.

Before defining the oracle \mathcal{O}' used above, we introduce some notation. Let L , \mathcal{O}' , and MSK' be as above, and note that we can view L and \mathcal{O}' as a tuple of (partial) functions (g, e, d) and (g', e', d') where g', e' , and d' extend g, e , and d , respectively. Define the following:

- \mathcal{Q}'_S is the set of pk for which $[g'(sk) = pk]$ is queried during computation of $\text{Setup}^{\mathcal{O}'}(MSK')$.
- \mathcal{Q}'_K is the set of pk for which $[g'(sk) = pk]$ is queried during computation of $\text{KeyGen}_{MSK'}^{\mathcal{O}'}(f)$ for some $f \in \{f^*, f_1, \dots, f_p\}$.
- $\mathcal{Q}'_{K-S} = \mathcal{Q}'_K \setminus \mathcal{Q}'_S$.
- L_g is the set of pk for which the query $[g(sk) = pk]$ is in L .

Note that \mathcal{A} can compute each of these sets from its view. Note further that $\mathcal{Q}'_S, \mathcal{Q}'_K, \mathcal{Q}'_{K-S}, \mathcal{O}'$ are fixed throughout an iteration of step 3, but L_g may change as queries are answered.

Oracle \mathcal{O}' is defined as follows. For any query whose answer is defined by \mathcal{O}' , return that answer. Otherwise:

1. For an encryption query $e(pk, x)$ with $pk \in \mathcal{Q}'_{K-S} \setminus L_g$, return a random y consistent with the rest of \mathcal{O}' . Act analogously for a decryption query $d(sk, y)$ with $pk \in \mathcal{Q}'_{K-S} \setminus L_g$ (where $pk = g(sk)$).
2. For a decryption query $d(sk, y)$, if there exists a pk with $[g(sk) = pk] \in \mathcal{O}'$ but $\not\in L$ there exists an $sk' \neq sk$ with $[g(sk') = pk] \in L$, then use \mathcal{O}' to answer the query $d(sk', y)$.
3. In any other case, query the real oracle \mathcal{O} and return the result. Store the query/answer in L (note that this might affect L_g as well).

An analysis of \mathcal{A} , proving Theorem [1](#), appears in Appendix [A](#) and the full version of our paper. The analysis is very similar to the one given in [5](#), with the main difference being Proposition [1](#).

4 Impossibility for Specific Cases

We use Theorem [1](#) to rule out black-box constructions of predicate encryption schemes in several specific cases of interest. Specifically, we consider the cases of identity-based encryption, forward-secure encryption, and broadcast encryption. We begin with a useful lemma.

² Although \mathcal{O}' is chosen to be consistent, a conflict can occur since L is updated as \mathcal{A} makes additional queries to the real oracle \mathcal{O} .

Lemma 1. Fix $q(\cdot)$, and assume $\{(\mathcal{F}_n, \mathbb{A}_n)\}_{n \in \mathbb{N}}$ has the following property: For sufficiently large n , there exist $f_1, \dots, f_{5q} \in \mathcal{F}_n$ and $I_1, \dots, I_{5q} \in \mathbb{A}_n$ such that:

For all $i \in \{1, \dots, 5q\}$ it holds that $f_i(I_i) = 1$ but $f_j(I_i) = 0$ for $j > i$.

Then $\{(\mathcal{F}_n, \mathbb{A}_n)\}_{n \in \mathbb{N}}$ can be q -covered. If the above holds for every polynomial q , then $\{(\mathcal{F}_n, \mathbb{A}_n)\}_{n \in \mathbb{N}}$ is easily covered.

Proof. We show that, under the stated assumption, $\{(\mathcal{F}_n, \mathbb{A}_n)\}_{n \in \mathbb{N}}$ satisfies Definition 5. Fix q and n large enough so that the condition of the lemma holds, and let f_1, \dots, f_{5q} and I_1, \dots, I_{5q} be as stated. Define algorithms A_1, A_2, A_3 as follows:

1. $A_1(1^n)$ chooses $i \leftarrow \{0, \dots, 5q\}$ and outputs $f^* = f_i$.
2. $A_2(1^n, f^*)$ finds i for which $f^* = f_i$ and outputs $I^* = I_i$.
3. $A_3(1^n, f^*)$ finds i for which $f^* = f_i$ and outputs f_{i+1}, \dots, f_{5q} . (If $i = 5q$ then output nothing.)

Note that $A_2(1^n, f^*)$ always outputs I^* with $f^*(I^*) = 1$. We show that for any \mathcal{F}_n -set system $\{S_f\}_{f \in \mathcal{F}_n}$ over $[q]$, the conditions of Definition 5 hold. We begin with the following claim:

Claim. For any \mathcal{F}_n -set system $\{S_f\}_{f \in \mathcal{F}_n}$ over $[q]$, there are at most q values $i \in \{1, \dots, 5q\}$ for which $S_{f_i} \not\subseteq \bigcup_{i < j \leq 5q} S_{f_j}$. (By convention, the union is the empty set if $j = 5q$.)

Proof. Define $\mathbf{S}_i \stackrel{\text{def}}{=} \bigcup_{i < j \leq 5q} S_{f_j}$, with $\mathbf{S}_{5q} = \emptyset$. Note that $\mathbf{S}_{i-1} = \mathbf{S}_i \cup S_{f_i}$, and so $S_{f_i} \not\subseteq \bigcup_{i < j \leq 5q} S_{f_j} = \mathbf{S}_i$ iff $\mathbf{S}_i \subsetneq \mathbf{S}_{i-1}$. Since

$$\mathbf{S}_{5q} \subseteq \mathbf{S}_{5q-1} \subseteq \dots \subseteq \mathbf{S}_1 \subseteq [q],$$

there can be at most q indices i where this occurs. □

Fixing an arbitrary \mathcal{F}_n -set system $\{S_f\}_{f \in \mathcal{F}_n}$ over $[q]$, let $\mathbb{I} \subset \{1, \dots, 5q\}$ be the set of indices for which $S_{f_i} \subseteq \bigcup_{i < j \leq q} S_{f_j}$; the claim above shows that $|\mathbb{I}| \geq 4q$. If A_1 chooses $i \in \mathbb{I}$ then:

1. $S_{f^*} = S_{f_i} \subseteq \bigcup_{i < j \leq q} S_{f_j}$.
2. $f_j(I^*) = f_j(I_i) = 0$ for all the predicates f_{i+1}, \dots, f_q output by A_3 .

Since A_1 chooses $i \in \mathbb{I}$ with probability $4/5$, this proves the lemma. □

We now apply Lemma 1 to several specific cases.

Identity-based encryption. It is easy to see that IBE for identities $\{\mathcal{I}_n\}$ can be viewed as an instance of predicate encryption by setting $\mathbb{A}_n = \mathcal{I}_n$ and $\mathcal{F}_n = \{f_{ID}\}_{ID \in \mathcal{I}_n}$ where

$$f_{ID}(ID') \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } ID' = ID \\ 0 & \text{otherwise} \end{cases}.$$

Let $N = |\mathcal{I}_n|$ denote the size of the identity space. Boneh et al. [5] already rule out black-box constructions of IBE from trapdoor permutations for $N = \omega(\text{poly}(n))$; the next theorem shows that our Theorem 1 generalizes their result:

Theorem 2. *There is no black-box construction (from trapdoor permutations or CCA-secure encryption) of an IBE scheme for $5N$ identities where each algorithm makes fewer than N queries to its oracle.*

As a corollary, there is no black-box construction of an IBE scheme (from trapdoor permutations or CCA-secure encryption) for a super-polynomial number of identities.

Proof. Let $\mathcal{I}_n = \{ID_1, \dots, ID_{5N}\}$. It is not hard to see that $\{(\mathcal{F}_n, \mathbb{A}_n)\}_{n \in \mathbb{N}}$ can be N -covered: take $f_{ID_1}, \dots, f_{ID_{5N}}$ and set $I_i = ID_i$ for all i . Then apply Theorem [11](#). □

Forward-secure public-key encryption. In a forward-secure public-key encryption scheme [7](#) secret keys are associated with time periods; the secret key at time period i enables decryption for ciphertexts encrypted at any time $j \geq i$. (We refer the reader to [7](#) for further discussion.) A forward-secure encryption scheme supporting $N = N(n)$ time periods can be cast as a predicate encryption scheme by letting $\mathbb{A}_n = \{1, \dots, N\}$ and $\mathcal{F}_n = \{f_i\}_{1 \leq i \leq N}$ where

$$f_i(j) \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } j \geq i \\ 0 & \text{otherwise} \end{cases} .$$

(A forward-secure encryption scheme imposes the additional requirement that $SK_{f_{i+1}}$ can be derived from SK_{f_i} ; since we do not impose this requirement our impossibility result is even stronger.) A black-box construction of a forward-secure encryption scheme from any CPA-secure encryption scheme exists for any $N = \text{poly}(n)$: the master public key contains public keys $\{pk_1, \dots, pk_N\}$, and the secret key at period i is $SK_{f_i} = \{sk_i, \dots, sk_N\}$; encryption at period j uses pk_j . While such a scheme is trivial as far as forward-secure encryption goes (since the public/secret key lengths are linear in N), it satisfies the definition. The next theorem indicates that, in some sense, this trivial construction is almost optimal as far as black-box constructions are concerned; moreover, there is no black-box construction supporting a super-polynomial number of time periods. (In contrast, there exist schemes based on specific assumptions [7.43](#) that support an unbounded number of time periods.)

Theorem 3. *There is no black-box construction (from trapdoor permutations or CCA-secure encryption) of a forward-secure encryption scheme for $5N$ periods where each algorithm in the scheme makes fewer than N queries to its oracle.*

As a corollary, there is no black-box construction of a forward-secure encryption scheme (from trapdoor permutations or CCA-secure encryption) supporting a super-polynomial number of time periods.

Proof. $\{(\mathcal{F}_n, \mathbb{A}_n)\}_{n \in \mathbb{N}}$ can be N -covered, as taking f_1, \dots, f_{5N} and setting $I_i = i$ for all i satisfies the conditions of Lemma [11](#). Then apply Theorem [11](#). □

Broadcast encryption. Finally, we look at the case of (public-key) broadcast encryption [9](#). Here, there is a fixed public key and a set of users $\mathcal{U} = \{1, \dots, U\}$

each with their own personal secret key; it should be possible for a sender to encrypt a message in such a way that only some subset $\mathcal{U}' \subset \mathcal{U}$ of users can decrypt. Consider the case where at most $k = k(n) < U$ users are excluded; we refer to this as *k-exclusion broadcast encryption*. This can also be modeled by predicate encryption, if we let $\mathbb{A}_n = \{\mathcal{U}' \subseteq \mathcal{U} \mid |\mathcal{U}'| \geq U - k\}$ and define $\mathcal{F}_n = \{f_i\}_{i \in \mathcal{U}}$ where

$$f_i(\mathcal{U}') \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } i \in \mathcal{U}' \\ 0 & \text{otherwise} \end{cases} .$$

Theorem 4. *There is no black-box construction (from trapdoor permutations or CCA-secure encryption) of a $(5k)$ -exclusion broadcast encryption scheme where each algorithm in the scheme makes k or fewer queries to its oracle.*

As a corollary, there is no black-box construction of a k -exclusion broadcast encryption scheme (from trapdoor permutations or CCA-secure encryption) for super-polynomial k .

Proof. We show that $\{(\mathcal{F}_n, \mathbb{A}_n)\}_{n \in \mathbb{N}}$ can be k -covered. Take f_1, \dots, f_{5k} and define

$$I_i \stackrel{\text{def}}{=} \mathcal{U} \setminus \{i, \dots, 5k\}$$

for $i \in \{1, \dots, 5k\}$. (So $I_{5k} = \mathcal{U}$.) Note that $|I_i| \geq U - 5k$ always, and these satisfy the conditions of Lemma 11. Applying Theorem 12 concludes the proof. \square

5 A Possibility Result for Predicate Encryption

Here we show that for the class of predicates and attributes $\{(\mathcal{F}_n, \mathbb{A}_n)\}$ where (roughly) for each $I \in \mathbb{A}_n$ there are at most polynomially-many $f \in \mathcal{F}_n$ with $f(I) = 0$, there is a black-box construction of a predicate encryption scheme over $\{(\mathcal{F}_n, \mathbb{A}_n)\}$ based on any CPA-secure encryption scheme. We remark that while we only prove payload hiding, our construction can in fact be shown to be attribute hiding [13] as well.

Our construction relies on the notion of an (N, k) -cover free family [8]:

Definition 6. *An (N, k) -cover free family over $[U]$ is a family $\mathcal{S} = \{S_1, \dots, S_N\}$, with $S_i \subseteq [U]$, such that for any distinct sets $S, S_1, \dots, S_k \in \mathcal{S}$ it holds that $S \setminus \bigcup_{i=1}^k S_i \neq \emptyset$.*

For any $k = \text{poly}(n)$ and $N = 2^{\text{poly}(n)}$ there exist [14][16] explicit, polynomial-time constructions of an (N, k) -cover free family over $[U]$ with $|U| = \text{poly}(n)$. (The specific results of [14][16] can be used to improve the efficiency of the construction that follows, but our only goal here is to show a construction that can be implemented in polynomial time.)

Theorem 5. *Fix $\{(\mathcal{F}_n, \mathbb{A}_n)\}$ and set $\text{Neg}_I \stackrel{\text{def}}{=} \{f \in \mathcal{F}_n : f(I) = 0\}$ for $I \in \mathbb{A}_n$. If there is a poly-time algorithm ListNeg for which $\text{ListNeg}(1^n, I) = \text{Neg}_I$, then there is a black-box construction of a predicate encryption scheme over $\{(\mathcal{F}_n, \mathbb{A}_n)\}$ from any CPA-secure encryption scheme.*

Proof. Since ListNeg runs in polynomial time, there is a polynomial k for which $|\text{Neg}_I| \leq k(n)$ for all $I \in \mathbb{A}_n$. Say predicates in \mathcal{F}_n can be represented using $\ell(n) = \text{poly}(n)$ bits. Let $\{U_n\}$ be such that $U_n = \text{poly}(n)$ and such that, for each n , there is an explicit $(2^{\ell(n)}, k(n))$ -cover free family $\mathcal{S} = \{S_1, \dots, S_{2^{\ell(n)}}\}$ over $[U_n]$. Identifying \mathcal{F}_n with a subset of $[2^{\ell(n)}]$, we can view the cover-free family as $\mathcal{S} = \{S_f\}_{f \in \mathcal{F}_n}$.

Let $(\text{Gen}', \text{Enc}', \text{Dec}')$ be a CPA-secure encryption scheme. Our construction of a predicate encryption scheme over $\{(\mathcal{F}_n, \mathbb{A}_n)\}$ is as follows:

- **Setup**, on input 1^n and a sufficiently long random string MSK , runs $\text{Gen}'(1^n)$ a total of $U = U_n$ times to generate keys $(pk_1, sk_1), \dots, (pk_U, sk_U)$. The master public key is $\{pk_1, \dots, pk_U\}$.
- **KeyGen**, given the secret keys $\{sk_i\}_{i=1}^U$ and a predicate $f \in \mathcal{F}_n$, outputs the subset $\{sk_i\}_{i \in S_f}$.
- **Enc**, given the public key, an attribute $I \in \mathbb{A}_n$, and a message m , computes $\text{Neg}_I = \text{ListNeg}(I)$ and sets $\bar{U} = [U] \setminus \left(\bigcup_{f \in \text{Neg}_I} S_f\right)$. The ciphertext is $(I, \{C_i\}_{i \in \bar{U}})$ where $C_i \leftarrow \text{Enc}'_{pk_i}(m)$.
- **Dec**, given the secret key $\{sk_i\}_{i \in S_f}$ for a predicate f and a ciphertext $(I, \{C_i\}_{i \in \bar{U}})$ for which $f(I) = 1$, first finds an index i for which $i \in S_f \cap \bar{U}$. (Such an index must exist, since

$$S_f \setminus \bar{U} = S_f \setminus \bigcup_{f': f'(I)=0} S_{f'},$$

and there are at most k predicates f' that the union is taken over.) The output is $\text{Dec}'_{sk_i}(C_i)$.

It is easy to see that the above construction satisfies correctness. We now prove security (in the sense of Definition 3). Let \mathcal{A} be an adversary attacking the scheme. We may assume without loss of generality that \mathcal{A} never requests a secret key for a predicate f for which $f(I^*) = 1$ (where I^* is the attribute used to encrypt the challenge ciphertext), since \mathcal{A} cannot succeed if that occurs.

For simplicity we prove security in a non-uniform model, but the proof can be modified easily to hold in the uniform model in the standard way. We consider $U+1$ hybrid experiments H_0, \dots, H_{U+1} , where H_0 corresponds to the experiment of Definition 3 when $b = 0$ is encrypted, and H_{U+1} corresponds to the experiment of Definition 3 when $b = 1$ is encrypted. Let δ_i denote the probability that \mathcal{A} outputs ‘0’ in H_i . We show that $|\delta_i - \delta_{i+1}|$ is negligible for all i ; since $U = U_n$ is polynomial in n , this proves that $|\delta_0 - \delta_{U+1}|$ is negligible and thus completes the proof.

Experiment H_i is defined as follows: Steps 1 and 2 are exactly as in Definition 3. In step 3, however, when encrypting the challenge ciphertext for the attribute I^* , let $\bar{U}^* = [U] \setminus \text{Neg}_{I^*}$ and set the ciphertext equal to $(I, \{C_j\}_{j \in \bar{U}^*})$, where

$$C_j \leftarrow \begin{cases} \text{Enc}'_{pk_j}(1) & j < i \\ \text{Enc}'_{pk_j}(0) & j \geq i \end{cases}.$$

\mathcal{A} may continue to request secret keys as in Definition 3.

We now prove that $|\delta_j - \delta_{j+1}|$ is negligible for any j . Fix j and consider the following adversary \mathcal{A}' attacking the underlying encryption scheme $(\text{Gen}', \text{Enc}', \text{Dec}')$. Given public key pk and ciphertext C (which is either an encryption of 0 or 1), the adversary \mathcal{A}' proceeds as follows:

1. Set $pk_j = pk$. For $i \neq j$, compute $(pk_i, sk_i) \leftarrow \text{Gen}'(1^n)$. Give the master public key $\{pk_1, \dots, pk_U\}$ to \mathcal{A} .
2. When \mathcal{A} requests a secret key for a predicate f , then if $j \notin S_f$ give to \mathcal{A} the secret keys $\{sk_i\}_{i \in S_f}$. Otherwise, abort and output a random bit.
3. When \mathcal{A} outputs I^* , compute $\text{Neg}_{I^*} = \text{ListNeg}(I^*)$ and then set

$$\bar{U}^* = [U] \setminus \left(\bigcup_{f \in \text{Neg}_{I^*}} S_f \right).$$

If $j \notin \bar{U}^*$ then abort and output a random bit. Otherwise, give \mathcal{A} the ciphertext $(I, \{C_i\}_{i \in \bar{U}^*})$ where

$$C_i \leftarrow \begin{cases} \text{Enc}'_{pk_i}(1) & i < j \\ C & i = j \\ \text{Enc}'_{pk_i}(0) & i > j \end{cases}.$$

4. Subsequent secret key queries made by \mathcal{A} are answered as before. Finally, \mathcal{A}' outputs whatever bit is output by \mathcal{A} .

Let $\text{Pr}_j[\cdot]$ denote the probability of an event in experiment H_j . We have

$$\begin{aligned} & |\text{Pr}[\mathcal{A}' \text{ outputs } 0 \mid C \leftarrow \text{Enc}'_{pk}(0)] - \text{Pr}[\mathcal{A}' \text{ outputs } 0 \mid C \leftarrow \text{Enc}'_{pk}(1)]| \\ &= |\text{Pr}[j \in \bar{U}^*] \cdot \text{Pr}_j[\mathcal{A} \text{ outputs } 0 \mid j \in \bar{U}^*] \\ &\quad - \text{Pr}[j \in \bar{U}^*] \cdot \text{Pr}_{j+1}[\mathcal{A} \text{ outputs } 0 \mid j \in \bar{U}^*]|, \end{aligned}$$

using the facts that (1) $\text{Pr}[j \in \bar{U}^*]$ is independent of whether C is an encryption of 0 or 1 and (2) when C is an encryption of 0 (resp., 1) then the view of \mathcal{A} (assuming $j \in \bar{U}^*$) is identical to its view in H_j (resp., H_{j+1}). Note further that

$$\text{Pr}_j[\mathcal{A} \text{ outputs } 0 \mid j \notin \bar{U}^*] = \text{Pr}_{j+1}[\mathcal{A} \text{ outputs } 0 \mid j \notin \bar{U}^*]$$

since the challenge ciphertext is distributed identically in each case. It follows that

$$\begin{aligned} & |\text{Pr}[\mathcal{A}' \text{ outputs } 0 \mid C \leftarrow \text{Enc}'_{pk}(0)] - \text{Pr}[\mathcal{A}' \text{ outputs } 0 \mid C \leftarrow \text{Enc}'_{pk}(1)]| \\ &= |\text{Pr}[j \in \bar{U}^*] \cdot \text{Pr}_j[\mathcal{A} \text{ outputs } 0 \mid j \in \bar{U}^*] \\ &\quad - \text{Pr}[j \in \bar{U}^*] \cdot \text{Pr}_{j+1}[\mathcal{A} \text{ outputs } 0 \mid j \in \bar{U}^*]| \\ &= |\delta_j - \delta_{j+1}|, \end{aligned}$$

concluding the proof. □

Acknowledgments

We thank the authors of [5] for providing us with the full version of their paper. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing official positions or policies, either expressed or implied, of the US Government, the US Army Research Laboratory, DARPA, the UK Government, or the UK Ministry of Defence. No official endorsement of any kind should be inferred. The US and UK Governments are authorized to reproduce and distribute reprints for Government purposes, notwithstanding any copyright notation herein.

References

1. 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)
2. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: IEEE Symposium on Security & Privacy, pp. 321–334. IEEE, Los Alamitos (2007)
3. 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)
4. Boneh, D., Franklin, M.K.: Identity based encryption from the Weil pairing. *SIAM Journal on Computing* 32(3), 586–615 (2003)
5. Boneh, D., Papakonstantinou, P.A., Rackoff, C., Vahlis, Y., Waters, B.: On the impossibility of basing identity-based encryption on trapdoor permutations. In: 49th Annual Symposium on Foundations of Computer Science (FOCS), pp. 283–292. IEEE, Los Alamitos (2008)
6. 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)
7. Canetti, R., Halevi, S., Katz, J.: A forward-secure public-key encryption scheme. *Journal of Cryptology* 20(3), 265–294 (2007)
8. Erdős, P., Frankl, P., Füredi, Z.: Families of finite sets in which no set is covered by the union of r others. *Israeli Journal of Mathematics* 51, 79–89 (1985)
9. Fiat, A., Naor, M.: Broadcast encryption. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 480–491. Springer, Heidelberg (1994)
10. Gennaro, R., Gertner, Y., Katz, J., Trevisan, L.: Bounds on the efficiency of generic cryptographic constructions. *SIAM J. Computing* 35(1), 217–246 (2005)
11. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: ACM CCS 2006: 13th ACM Conference on Computer and Communications Security, pp. 89–98. ACM Press, New York (2006)
12. Impagliazzo, R., Rudich, S.: Limits on the provable consequences of one-way permutations. In: 21st ACM Annual ACM Symposium on Theory of Computing (STOC), pp. 44–61. ACM Press, New York (1989)
13. 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)

14. Kumar, R., Rajagopalan, S., Sahai, A.: Coding constructions for blacklisting problems without computational assumptions. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 609–623. Springer, Heidelberg (1999)
15. Ostrovsky, R., Sahai, A., Waters, B.: Attribute-based encryption with non-monotonic access structures. In: ACM CCS 2007: 14th ACM Conference on Computer and Communications Security, pp. 195–203. ACM Press, New York (2007)
16. Porat, E., Rothschild, A.: Explicit non-adaptive combinatorial group testing schemes. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfssdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part I. LNCS, vol. 5125, pp. 748–759. Springer, Heidelberg (2008)
17. Reingold, O., Trevisan, L., Vadhan, S.P.: Notions of reducibility between cryptographic primitives. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 1–20. Springer, Heidelberg (2004)
18. Sahai, A., Waters, B.R.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)
19. 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)

A Proof Details

We analyze the success probability of the adversary \mathcal{A} from Section 3.1. Due to space limitations, the proof cannot be reproduced here in its entirety; we have instead aimed to describe those parts of our proof that differ most prominently from the proof of Boneh et al. [5]. The most significant new element in our proof is Proposition 1.

Toward analyzing the success probability of \mathcal{A} , we describe a series of experiments, the first of which corresponds to adversary \mathcal{A} interacting in the experiment from Definition 2. We show that, as long as no “bad” events (to be defined later) occur, the statistical distance between the transcripts generated in each of these experiments is not too large. This allows us to bound \mathcal{A} ’s success probability by comparing it to an appropriate event in the final experiment.

Expt₀: This corresponds to \mathcal{A} interacting in the experiment from Definition 2.

Expt₁: This is the same as Expt₀ except that \mathcal{O}'' (as defined after the k^{th} repetition of step 3) is used instead of \mathcal{O} to compute the challenge ciphertext C^* .

Expt₂: This is the same as Expt₁ except that \mathcal{O}'' never queries \mathcal{O} (cf. step 3 in the definition of \mathcal{O}''); instead, any such queries are answered randomly (subject to ensuring that \mathcal{O}'' remains consistent).

Expt₃: This is the following experiment with no adversary and using the real oracle \mathcal{O} :

Setup and challenge

1. Compute $f^* \leftarrow A_1(1^n)$, $I^* = A_2(1^n, f^*)$, and $\{f_1, \dots, f_p\} \leftarrow A_3(1^n, f^*)$.
2. Choose at random $MSK \leftarrow \{0, 1\}^n$ and compute $MPK := \text{Setup}^{\mathcal{O}}(MSK)$. If $f_i(I^*) = 1$ for some i , abort and output a random bit.
3. For every predicate $f \in \{f^*, f_1, \dots, f_p\}$ compute $SK_f := \text{KeyGen}_{MSK}^{\mathcal{O}}(f)$.

Step 1: Discovering important public keys. For $i = 1$ to p do:

1. Compute $I_{f_i} \leftarrow A_2(1^n, f_i)$, and choose random $b_i \leftarrow \{0, 1\}$ and $r_i \leftarrow \{0, 1\}^n$.
2. Compute $\text{Dec}_{SK_{f_i}}^{\mathcal{O}}(\text{Enc}_{MPK}^{\mathcal{O}}(I_{f_i}, b_i; r_i))$.

Step 2: Decrypting the challenge

1. Choose $r \leftarrow \{0, 1\}^n$, $b \leftarrow \{0, 1\}$ and compute $C^* := \text{Enc}_{MPK}^{\mathcal{O}}(I^*, b; r)$.
2. Compute $b' := \text{Dec}_{SK_{f^*}}^{\mathcal{O}}(C^*)$ and output b' . Note that $b' = b$ always.

This completes the description of Expt_3 .

For $i \in \{0, 1, 2\}$ we will be interested in the following transcripts defined in the course of Expt_i . These transcripts contain, in particular, all oracle queries/answers.

- trans_{setup}^i : The transcript of the setup phase. This includes the computation of MPK and $SK_{f_1}, \dots, SK_{f_p}$, as well as the computation of SK_{f^*} for the f^* chosen by the adversary. (Even though SK_{f^*} is not computed in the experiment, SK_{f^*} is well defined given f^* , MSK , and \mathcal{O} .)
- trans_{pks}^i : The transcript of step 1 (“discovering important public keys”).
- trans_{freq}^i : The transcript of step 2 (“discovering frequent queries for I^* ”).
- $\text{trans}_{sim-setup}^i$: This is the transcript defined by the adversary’s choice of MSK' and \mathcal{O}' in the k^{th} repetition of step 3, and can be viewed as the adversary’s “guess” for trans_{setup}^i .
- trans_*^i : The transcript of the encryption of C /decryption of C^* in the k^{th} repetition of step 3.
- $\text{trans}^i = (\text{trans}_{setup}^i, \text{trans}_{pks}^i, \text{trans}_{sim-setup}^i, \text{trans}_*^i)$.

For Expt_3 we define

- $\text{trans}_{sim-setup}^3$: The transcript of the “setup and challenge” step.
- trans_{pks}^3 : The transcript of step 1 (“discovering important public keys”).
- trans_*^3 : The transcript of step 2 (“decrypting the challenge”).
- $\text{trans}^3 = (\text{trans}_{pks}^3, \text{trans}_{sim-setup}^3, \text{trans}_*^3)$.

For a given transcript, we partition the set of public keys used (i.e., the set of pk ’s for which $[g(\cdot) = pk] \in \text{trans}$) into the following sets:

- We let $\mathcal{Q}_S(\text{trans})$ denote the public keys queried during execution of Setup:

$$\mathcal{Q}_S(\text{trans}) \stackrel{\text{def}}{=} \{pk \mid \text{the query } [g(\cdot) = pk] \in \text{trans} \text{ is asked by Setup}\}.$$

Intuitively, these are the pk ’s whose corresponding sk ’s are “useful” for decrypting ciphertexts.

- We let $\mathcal{Q}_K(\text{trans})$ denote the public keys queried by the KeyGen algorithm when some personal secret key is derived:

$$\mathcal{Q}_K(\text{trans}) \stackrel{\text{def}}{=} \{pk \mid [g(\cdot) = pk] \in \text{trans} \text{ is asked by KeyGen}_{MSK}(\cdot)\}$$

$$\mathcal{Q}_{K-S}(\text{trans}) \stackrel{\text{def}}{=} \mathcal{Q}_K(\text{trans}) \setminus \mathcal{Q}_S(\text{trans}).$$

- Finally, we will also look at the public keys “discovered” during encryption and decryption (cf. step 3 of the experiments):

$$\mathcal{Q}_{ENC+DEC}(\text{trans}, I, f) \stackrel{\text{def}}{=} \{pk \mid [g(\cdot) = pk] \text{ asked by Dec}_{SK_f}(\text{Enc}_{MPK}(I, \cdot; \cdot))\}$$

A.1 Bounding Probabilities of Bad Events

Fixing the master secret key MSK and the oracle \mathcal{O} (this fixes MPK as well as $\{SK_f\}_{f \in \mathcal{F}}$), we define four “bad” events and bound the probabilities of each of them. Here, we will only describe and bound one of these events; we refer to the full version of our paper for the remainder of the proof.

Let E_{NC}^i be the event that either of the following is true (in Expt_i):

1. $\exists f_i \in \{f_1, \dots, f_p\}$ such that $f_i(I^*) = 1$.
2. The following condition holds:

$$\mathcal{Q}_{ENC+DEC}(\text{trans}_{*}^i, I^*, f^*) \cap \mathcal{Q}_S(\text{trans}_{sim-setup}^i) \\ \not\subseteq \left(\bigcup_{f \in \{f_1, \dots, f_p\}} \mathcal{Q}_{ENC+DEC}(\text{trans}_{pks}^i, I_f, f) \right) \cap \mathcal{Q}_S(\text{trans}_{sim-setup}^i),$$

where $I_f := A_2(1^n, f)$.

Intuitively, the second condition above is the event that the public keys that are “useful” for f_1, \dots, f_p does not contain the public keys that are “useful” for f^* .

We bound the probability of E_{NC}^3 using the assumed easily-covered property of $\{(\mathcal{F}_n, \mathbb{A}_n)\}$; this is the crux of our proof, and is what motivates Definition 5.

Proposition 1. $\Pr[E_{NC}^3] \leq 1/5$.

Proof. Fix \mathcal{O} and $MSK \in \{0, 1\}^n$, thus fixing $\text{trans}_{sim-setup}^3$. If for each $f \in \mathcal{F}_n$ we fix a random tape r_f that is sufficiently long to run $\text{Dec}_{SK_f}(\text{Enc}_{MPK}(I, b; r))$ (where $I \stackrel{\text{def}}{=} A_2(f)$), then this defines, for each f , the set

$$S_f \\ \stackrel{\text{def}}{=} \left\{ pk \mid [g(\cdot) = pk] \text{ asked by } \text{Dec}_{SK_f}(\text{Enc}_{MPK}(I, b; r)) \right\} \cap \mathcal{Q}_S(\text{trans}_{sim-setup}^3).$$

Numbering the (at most q) public keys in $\mathcal{Q}_S(\text{trans}_{sim-setup}^3)$ in lexicographic order, we can view these $\{S_f\}_{f \in \mathcal{F}_n}$ as an \mathcal{F}_n -set system over $[q]$. The fact that $\{(\mathcal{F}_n, \mathbb{A}_n)\}$ can be q -covered implies that there exists a polynomial p such that

$$\Pr \left[\begin{array}{l} \forall f \in \mathcal{F}_n : r_f \leftarrow \{0, 1\}^* \\ f^* \leftarrow A_1, I^* := A_2(1^n, f^*) : \left(S_{f^*} \subseteq \bigcup_{i=1}^p S_{f_i} \right) \wedge \left(\forall i : f_i(I^*) = 0 \right) \end{array} \right] \geq \frac{4}{5}.$$

The above is a lower bound on the probability that E_{NC}^3 does not occur. \square

Hierarchical Predicate Encryption for Inner-Products

Tatsuaki Okamoto¹ and Katsuyuki Takashima²

¹ NTT, 3-9-11 Midori-cho, Musashino-shi, Tokyo, 180-8585 Japan
okamoto.tatsuaki@lab.ntt.co.jp

² Mitsubishi Electric, 5-1-1, Ofuna, Kamakura, Kanagawa, 247-8501 Japan
Takashima.Katsuyuki@aj.MitsubishiElectric.co.jp

Abstract. This paper presents a hierarchical predicate encryption (HPE) scheme for inner-product predicates that is secure (selectively attribute-hiding) in the standard model under new assumptions. These assumptions are non-interactive and of fixed size in the number of adversary’s queries (i.e., not “ q -type”), and are proven to hold in the generic model. To the best of our knowledge, this is the first HPE (or delegatable PE) scheme for inner-product predicates that is secure in the standard model. The underlying techniques of our result are based on a new approach on bilinear pairings, which is extended from bilinear pairing groups over linear spaces. They are quite different from the existing techniques and may be of independent interest.

1 Introduction

1.1 Background

The notion of *predicate encryption* (PE) was explicitly presented by Katz, Sahai and Waters [16] as a generalized (fine-grained) notion of encryption that covers identity-based encryption (IBE) [23, 5, 9, 10, 15], hidden-vector encryption (HVE) [7] and attribute-based encryption (ABE) [11, 13, 19, 20, 21].

Informally, secret keys in a predicate encryption scheme correspond to predicates in some class \mathcal{F} , and a sender associates a ciphertext with an attribute in a set Σ ; a ciphertext associated with the attribute $I \in \Sigma$ can be decrypted by secret key sk_f corresponding to the predicate $f \in \mathcal{F}$ if and only if $f(I) = 1$.

In addition, a stronger security notion for PE, *attribute-hiding*, than basic security requirement, *payload-hiding*, was defined in [16]. Roughly speaking, attribute-hiding requires that a ciphertext conceal the associated attribute as well as the plaintext, while payload-hiding only requires that a ciphertext conceal the plaintext. If attributes are identities, i.e., PE is IBE, attribute hiding PE implies *anonymous* IBE.

Katz, Sahai and Waters [16] also presented a concrete construction of PE for a class of predicates called *inner-product* predicates, which represents a wide class of predicates that includes an equality test (for IBE and HVE), disjunctions or conjunctions of equality tests, and, more generally, arbitrary CNF or

DNF formulas (for ABE). Informally, an attribute of inner-product predicates is expressed as vector \vec{x} and predicate $f_{\vec{v}}$ is associated with vector \vec{v} , where $f_{\vec{v}}(\vec{x}) = 1$ iff $\vec{x} \cdot \vec{v} = 0$. (Here, $\vec{x} \cdot \vec{v}$ denotes the standard inner-product.)

Although the Katz-Sahai-Waters scheme [16] is the most expressive attribute-hiding PE among the existing schemes, no delegation functionality was proposed. Shi and Waters [22] presented a delegation mechanism for a class of PE, but the admissible predicates of the system, which is a class of equality tests for HVE, are more limited than inner-product predicates in [16]. Okamoto and Takashima [18] presented hierarchical delegation of PE for inner-product predicates, but the security proof was only given in the generic model.

1.2 Our Results

This paper addresses the above problems in [16,22,18].

- This paper proposes a *hierarchical* predicate encryption (HPE) scheme for *inner-product* predicates, where a (natural) hierarchical delegation system of inner-product predicates is provided e.g., our hierarchical system is consistent with that for hierarchical IBE (HIBE) [4,8,11,12] (i.e., our HPE is specialized to anonymous HIBE, if the predicate of HPE is specified to the equality test of identities).
- The proposed HPE scheme is selectively attribute-hiding against chosen-plaintext-attacks (CPA) in the standard model under two new assumptions, the RDSP and IDSP assumptions. These assumptions are non-interactive, falsifiable and of fixed size in the number of adversary’s queries (i.e., not “ q -type”), and are proven to hold in the generic model.
- To achieve the result, this paper advances an approach recently developed in [17,18]. This approach is extended from bilinear pairing groups into higher dimensional vector spaces, and a notion, *dual pairing vector spaces* (DPVS), is employed in this paper. (We will explain this approach below.)

One of the most basic decisional assumptions in this approach is the decisional subspace problem (DSP) assumption. (It is a higher-dimensional generalization of the decisional DH and Linear assumptions, and the relationships of this assumption with the traditional ones are studied in [17].)

The assumptions introduced in this paper, the RDSP and IDSP assumptions, are variants of the DSP assumption in DPVS.

- The performance of the proposed HPE scheme is almost the same as (or slightly worse than) that in [18], where the dimension of DPVS for our HPE scheme is $n + 3$, whereas that for [18] is $n + 2$, when n is the dimension of predicate/attribute vectors.
- Since HPE is a generalized (fine-grained) version of anonymous HIBE (AHIBE) (or includes AHIBE as a special case), HPE covers (a generalized version of) applications described in [8], fully private communication and search on encrypted data. For example, we can use a two-level HPE scheme where the first level corresponds to the predicate/attribute of (single-layer) PE and the second level corresponds to those of “attribute search by a predicate” (generalized “key-word search”).

1.3 A New Approach – Dual Pairing Vector Spaces

We now explain how the approach works by using a typical construction example on direct products of pairing groups $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, g_T, e)$, where q is a prime, $\mathbb{G}_1, \mathbb{G}_2$ and \mathbb{G}_T are cyclic groups of order q , g_i is a generator of \mathbb{G}_i ($i = 1, 2$), $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is a non-degenerate bilinear pairing operation, and $g_T := e(g_1, g_2) \neq 1$. Here we denote the group operation of $\mathbb{G}_1, \mathbb{G}_2$ and \mathbb{G}_T by multiplication. Note that this construction also works on *symmetric* pairing groups, where $\mathbb{G}_1 = \mathbb{G}_2$. As for the definitions of some notations, see Section [1.5](#).

Vector spaces \mathbb{V} and \mathbb{V}^* : $\mathbb{V} := \overbrace{\mathbb{G}_1 \times \cdots \times \mathbb{G}_1}^N$ and $\mathbb{V}^* := \overbrace{\mathbb{G}_2 \times \cdots \times \mathbb{G}_2}^N$, whose elements are expressed by N -dimensional vectors, $\mathbf{x} := (g_1^{x_1}, \dots, g_1^{x_N})$ and $\mathbf{y} := (g_2^{y_1}, \dots, g_2^{y_N})$, respectively ($x_i, y_i \in \mathbb{F}_q$ for $i = 1, \dots, N$).

Canonical bases \mathbb{A} and \mathbb{A}^* : $\mathbb{A} := (\mathbf{a}_1, \dots, \mathbf{a}_N)$ of \mathbb{V} , where $\mathbf{a}_1 := (g_1, 1, \dots, 1)$, $\mathbf{a}_2 := (1, g_1, 1, \dots, 1), \dots, \mathbf{a}_N := (1, \dots, 1, g_1)$. $\mathbb{A}^* := (\mathbf{a}_1^*, \dots, \mathbf{a}_N^*)$ of \mathbb{V}^* , where $\mathbf{a}_1^* := (g_2, 1, \dots, 1)$, $\mathbf{a}_2^* := (1, g_2, 1, \dots, 1), \dots, \mathbf{a}_N^* := (1, \dots, 1, g_2)$.

Pairing operation: $e(\mathbf{x}, \mathbf{y}) := \prod_{i=1}^N e(g_1^{x_i}, g_2^{y_i}) = e(g_1, g_2)^{\sum_{i=1}^N x_i y_i} = g_T^{\mathbf{x} \cdot \mathbf{y}} \in \mathbb{G}_T$ for the above $\mathbf{x} \in \mathbb{V}$ and $\mathbf{y} \in \mathbb{V}^*$.

Base change: Canonical basis \mathbb{A} is changed to basis $\mathbb{B} := (\mathbf{b}_1, \dots, \mathbf{b}_N)$ of \mathbb{V} using a uniformly chosen (regular) linear transformation, $X := (\chi_{i,j}) \stackrel{\cup}{\leftarrow} GL(N, \mathbb{F}_q)$, such that $\mathbf{b}_i = \sum_{j=1}^N \chi_{i,j} \mathbf{a}_j$, ($i = 1, \dots, N$). \mathbb{A}^* is also changed to basis $\mathbb{B}^* := (\mathbf{b}_1^*, \dots, \mathbf{b}_N^*)$ of \mathbb{V}^* , such that $(\vartheta_{i,j}) := (X^T)^{-1}$, $\mathbf{b}_i^* = \sum_{j=1}^N \vartheta_{i,j} \mathbf{a}_j^*$, ($i = 1, \dots, N$). We see that $e(\mathbf{b}_i, \mathbf{b}_j^*) = g_T^{\delta_{i,j}}$, ($\delta_{i,j} = 1$ if $i = j$, and $\delta_{i,j} = 0$ if $i \neq j$) i.e., \mathbb{B} and \mathbb{B}^* are dual orthonormal bases of \mathbb{V} and \mathbb{V}^* .

Intractable Problem: One of the most natural *decisional* problems in our approach is the *decisional subspace problem* (DSP) [\[17\]](#). The DSP $_{(N_1, N_2)}$ assumption is: it is hard to tell $\mathbf{v} := v_{N_2+1} \mathbf{b}_{N_2+1} + \cdots + v_{N_1} \mathbf{b}_{N_1}$ from $\mathbf{u} := v_1 \mathbf{b}_1 + \cdots + v_{N_1} \mathbf{b}_{N_1}$, where $(v_1, \dots, v_{N_1}) \stackrel{\cup}{\leftarrow} \mathbb{F}_q^{N_1}$ and $N_2 + 1 < N_1$. DSP is intractable if the generalized DDH or DLIN problem is intractable [\[17\]](#).

Trapdoor: Although the DSP problem is assumed to be intractable, it can be efficiently solved by using *trapdoor* $\mathbf{t}^* \in \text{span}(\mathbf{b}_1^*, \dots, \mathbf{b}_{N_2}^*)$. Given $\mathbf{v} := v_{N_2+1} \mathbf{b}_{N_2+1} + \cdots + v_{N_1} \mathbf{b}_{N_1}$ or $\mathbf{u} := v_1 \mathbf{b}_1 + \cdots + v_{N_1} \mathbf{b}_{N_1}$, we can tell \mathbf{v} from \mathbf{u} using \mathbf{t}^* since $e(\mathbf{v}, \mathbf{t}^*) = 1$ and $e(\mathbf{u}, \mathbf{t}^*) \neq 1$ with high probability.

1.4 Related Works on Our Approach

Higher dimensional vector treatment of bilinear pairing groups have been already employed in the literature especially in the areas of IBE, ABE and BE (e.g., [\[4\]\[16\]\[8\]\[13\]\[14\]\[21\]](#)). For example, in a typical vector treatment, two vector forms of $P := (g_1^{x_1}, \dots, g_1^{x_n})$ and $Q := (g_2^{y_1}, \dots, g_2^{y_n})$ are set and pairing for P and Q is operated as $e(P, Q) := \prod_{i=1}^n e(g_1^{x_i}, g_2^{y_i})$. Such a treatment can be rephrased in our approach using the (symmetric pairing) notations shown in Section [1.3](#) such that $P = x_1 \mathbf{a}_1 + \cdots + x_n \mathbf{a}_n$ and $Q = y_1 \mathbf{a}_1^* + \cdots + y_n \mathbf{a}_n^*$ over canonical basis \mathbb{A} and \mathbb{A}^* .

The major drawback of this approach is the easily *decomposable* property over \mathbb{A} (and \mathbb{A}^*). That is, it is easy to decompose $x_i \mathbf{a}_i = (1, \dots, 1, g_1^{x_i}, 1, \dots, 1)$ from $P := x_1 \mathbf{a}_1 + \dots + x_n \mathbf{a}_n = (g_1^{x_1}, \dots, g_1^{x_n})$.

In contrast, the current approach employs basis \mathbb{B} that is linearly transformed from \mathbb{A} using a secret random matrix $X \in \mathbb{F}_q^{n \times n}$. A remarkable property over \mathbb{B} is that it seems hard to decompose $x_i \mathbf{b}_i$ from $P' := x_1 \mathbf{b}_1 + \dots + x_n \mathbf{b}_n$. In addition, the dual orthonormal basis \mathbb{B}^* of \mathbb{V}^* can be used as a source of the trapdoors to the decomposability (see Section 1.3) through the pairing operation over \mathbb{B} and \mathbb{B}^* . The hard decomposability and its trapdoors are the key trick in this paper. Note that composite order pairing groups are often employed with similar tricks, hard decomposability of a composite order group into the prime order subgroups and its trapdoors through factoring (e.g., [16,22]).

1.5 Notations

When A is a random variable or distribution, $y \stackrel{R}{\leftarrow} A$ denotes that y is randomly selected from A according to its distribution. When A is a set, $y \stackrel{U}{\leftarrow} A$ denotes that y is uniformly selected from A . $y := z$ denotes that y is set, defined or substituted by z . When a is a fixed value, $A(x) \rightarrow a$ (e.g., $A(x) \rightarrow 1$) denotes the event that machine (algorithm) A outputs a on input x . A function $f : \mathbb{N} \rightarrow \mathbb{R}$ is *negligible* in λ , if for every constant $c > 0$, there exists an integer n such that $f(\lambda) < \lambda^{-c}$ for all $\lambda > n$.

We denote the finite field of order q by \mathbb{F}_q . A vector symbol denotes a vector representation over \mathbb{F}_q , e.g., \vec{x} denotes $(x_1, \dots, x_n) \in \mathbb{F}_q^n$. $\vec{x} \cdot \vec{v}$ denotes the inner-product $\sum_{i=1}^n x_i v_i$ of two vectors $\vec{x} = (x_1, \dots, x_n)$ and $\vec{v} = (v_1, \dots, v_n)$. X^T denotes the transpose of matrix X . A bold face letter denotes an element of vector space \mathbb{V} (resp. \mathbb{V}^*), e.g., $\mathbf{x} \in \mathbb{V}$ (resp. $\mathbf{x}^* \in \mathbb{V}^*$). $\text{span}\langle \mathbf{b}_1, \dots, \mathbf{b}_n \rangle$ (resp. $\text{span}\langle \vec{x}_1, \dots, \vec{x}_n \rangle$) denotes the subspace generated by $\mathbf{b}_1, \dots, \mathbf{b}_n$ (resp. $\vec{x}_1, \dots, \vec{x}_n$).

2 Dual Pairing Vector Spaces

Definition 1. “Dual pairing vector spaces (DPVS)” $(q, \mathbb{V}, \mathbb{V}^*, \mathbb{G}_T, \mathbb{A}, \mathbb{A}^*)$ are a tuple of a prime q , two N -dimensional vector spaces \mathbb{V} and \mathbb{V}^* over \mathbb{F}_q , a cyclic group \mathbb{G}_T of order q , and their canonical bases i.e., $\mathbb{A} := (\mathbf{a}_1, \dots, \mathbf{a}_N)$ of \mathbb{V} and $\mathbb{A}^* := (\mathbf{a}_1^*, \dots, \mathbf{a}_N^*)$ of \mathbb{V}^* that satisfy the following conditions:

1. [Non-degenerate bilinear pairing] There exists a polynomial-time computable nondegenerate bilinear pairing $e : \mathbb{V} \times \mathbb{V}^* \rightarrow \mathbb{G}_T$ i.e., $e(\mathbf{s}\mathbf{x}, \mathbf{t}\mathbf{y}) = e(\mathbf{x}, \mathbf{y})^{st}$ and if $e(\mathbf{x}, \mathbf{y}) = 1$ for all $\mathbf{y} \in \mathbb{V}$, then $\mathbf{x} = \mathbf{0}$.
2. [Dual orthonormal bases] \mathbb{A}, \mathbb{A}^* , and e satisfy $e(\mathbf{a}_i, \mathbf{a}_j^*) = g_T^{\delta_{i,j}}$ for all i and j , where $\delta_{i,j} = 1$ if $i = j$, and 0 otherwise, and $g_T \neq 1 \in \mathbb{G}_T$.
3. [Distortion maps] Endomorphisms $\phi_{i,j}$ of \mathbb{V} s.t. $\phi_{i,j}(\mathbf{a}_j) = \mathbf{a}_i$ and $\phi_{i,j}(\mathbf{a}_k) = \mathbf{0}$ if $k \neq j$ are polynomial-time computable. Moreover, endomorphisms $\phi_{i,j}^*$ of \mathbb{V}^* s.t. $\phi_{i,j}^*(\mathbf{a}_j^*) = \mathbf{a}_i^*$ and $\phi_{i,j}^*(\mathbf{a}_k^*) = \mathbf{0}$ if $k \neq j$ are also polynomial-time computable. We call $\phi_{i,j}$ and $\phi_{i,j}^*$ “distortion maps”.

Three typical constructions are given in [17]; a product of bilinear pairing groups, or a Jacobian variety of a supersingular curve of genus ≥ 1 [23]. See Section 1.3 as well (where the description of distortion maps is omitted).

3 Assumptions

This section defines two variants of the DSP assumption, the RDSP and IDSP assumptions. An intuition behind these assumptions are given in Remark below.

DPVS generation algorithm $\mathcal{G}_{\text{dpvs}}$ takes input 1^λ ($\lambda \in \mathbb{N}$) and $N \in \mathbb{N}$, and outputs a description of $\text{param} := (q, \mathbb{V}, \mathbb{V}^*, \mathbb{G}_T, \mathbb{A}, \mathbb{A}^*)$ with security parameter λ and N -dimensional \mathbb{V} and \mathbb{V}^* . It can be constructed in a manner shown in [17]. We describe a random orthonormal basis generator \mathcal{G}_{ob} below, which is used as a subroutine in the RDSP and IDSP instance generators.

$$\begin{aligned} \mathcal{G}_{\text{ob}}(1^\lambda, N) : \text{param} &:= (q, \mathbb{V}, \mathbb{V}^*, \mathbb{G}_T, \mathbb{A}, \mathbb{A}^*) \xleftarrow{\text{R}} \mathcal{G}_{\text{dpvs}}(1^\lambda, N), \\ X &:= (\chi_{i,j}) \xleftarrow{\text{U}} GL(N, \mathbb{F}_q), (\vartheta_{i,j}) := (X^T)^{-1}, \\ \mathbf{b}_i &:= \sum_{j=1}^N \chi_{i,j} \mathbf{a}_j, \mathbb{B} := (\mathbf{b}_1, \dots, \mathbf{b}_N), \mathbf{b}_i^* := \sum_{j=1}^N \vartheta_{i,j} \mathbf{a}_j^*, \mathbb{B}^* := (\mathbf{b}_1^*, \dots, \mathbf{b}_N^*), \\ &\text{return } (\text{param}, \mathbb{B}, \mathbb{B}^*) \end{aligned}$$

We now define the RDSP and IDSP instance generators, $\mathcal{G}_\beta^{\text{RDSP}}$ and $\mathcal{G}_\beta^{\text{IDSP}}$.

$$\begin{aligned} \mathcal{G}_\beta^{\text{RDSP}}(1^\lambda, n) : (\text{param}, \mathbb{B}, \mathbb{B}^*) &\xleftarrow{\text{R}} \mathcal{G}_{\text{ob}}(1^\lambda, n+3), \vec{y} := (y_1, \dots, y_n) \xleftarrow{\text{U}} \mathbb{F}_q^n \setminus \{\vec{0}\}, \\ \delta_1, \delta_2, \zeta_1, \zeta_2 &\xleftarrow{\text{U}} \mathbb{F}_q, \mathbf{d}_{n+1} := \mathbf{b}_{n+1} + \mathbf{b}_{n+2}, \widehat{\mathbb{B}} := (\mathbf{b}_1, \dots, \mathbf{b}_n, \mathbf{d}_{n+1}, \mathbf{b}_{n+3}), \\ (\omega^{(k)}, \gamma_1^{(k)}, \gamma_2^{(k)})_{k=1,2,3} &\xleftarrow{\text{U}} GL(\mathbb{F}_q, 3), \\ \text{For } i = 1, \dots, n; k = 1, 2, 3; & \\ \mathbf{h}_i^{(k)*} &:= \omega^{(k)} \mathbf{b}_i^* + \gamma_1^{(k)} y_i \mathbf{b}_{n+1}^* + \gamma_2^{(k)} y_i \mathbf{b}_{n+2}^*, \tau_i^{(k)} := (\gamma_1^{(k)} + \gamma_2^{(k)}) y_i, \\ \mathbf{e}_0 &:= \delta_1 (\sum_{i=1}^n y_i \mathbf{b}_i) + \delta_2 \mathbf{b}_{n+3}, \\ \mathbf{e}_1 &:= \delta_1 (\sum_{i=1}^n y_i \mathbf{b}_i) + \zeta_1 \mathbf{b}_{n+1} + \zeta_2 \mathbf{b}_{n+2} + \delta_2 \mathbf{b}_{n+3}, \\ &\text{return } (\text{param}, \widehat{\mathbb{B}}, \{\mathbf{h}_i^{(k)*}, \tau_i^{(k)}\}_{i=1, \dots, n; k=1, 2, 3}, \vec{y}, \mathbf{e}_\beta). \end{aligned}$$

$$\begin{aligned} \mathcal{G}_\beta^{\text{IDSP}}(1^\lambda, n) : (\text{param}, \mathbb{B}, \mathbb{B}^*) &\xleftarrow{\text{R}} \mathcal{G}_{\text{ob}}(1^\lambda, n+3), \\ \vec{y} := (y_1, \dots, y_n) &\xleftarrow{\text{U}} \mathbb{F}_q^n \setminus \{\vec{0}\}, \vec{u} := (u_1, \dots, u_n) \xleftarrow{\text{U}} \mathbb{F}_q^n \setminus \{\vec{0}\}, \\ \delta_1, \delta_2, \zeta_1, \zeta_2 &\xleftarrow{\text{U}} \mathbb{F}_q, \mathbf{d}_{n+1} := \mathbf{b}_{n+1} + \mathbf{b}_{n+2}, \widehat{\mathbb{B}} := (\mathbf{b}_1, \dots, \mathbf{b}_n, \mathbf{d}_{n+1}, \mathbf{b}_{n+3}), \\ \text{For } i = 1, \dots, n; &(\omega^{(k)}, \gamma_{i,1}^{(k)}, \gamma_{i,2}^{(k)})_{k=1,2,3} \xleftarrow{\text{U}} GL(\mathbb{F}_q, 3), \\ \text{For } i = 1, \dots, n; k = 1, 2, 3; & \\ \mathbf{h}_i^{(k)*} &:= \omega^{(k)} \mathbf{b}_i^* + \gamma_{i,1}^{(k)} \mathbf{b}_{n+1}^* + \gamma_{i,2}^{(k)} \mathbf{b}_{n+2}^*, \tau_i^{(k)} := \gamma_{i,1}^{(k)} + \gamma_{i,2}^{(k)}, \\ \mathbf{e}_0 &:= \delta_1 (\sum_{i=1}^n y_i \mathbf{b}_i) + \zeta_1 \mathbf{b}_{n+1} + \zeta_2 \mathbf{b}_{n+2} + \delta_2 \mathbf{b}_{n+3}, \\ \mathbf{e}_1 &:= \delta_1 (\sum_{i=1}^n u_i \mathbf{b}_i) + \zeta_1 \mathbf{b}_{n+1} + \zeta_2 \mathbf{b}_{n+2} + \delta_2 \mathbf{b}_{n+3}, \\ &\text{return } (\text{param}, \widehat{\mathbb{B}}, \{\mathbf{h}_i^{(k)*}, \tau_i^{(k)}\}_{i=1, \dots, n; k=1, 2, 3}, \vec{y}, \mathbf{e}_\beta). \end{aligned}$$

Definition 2 (RDSP: Decisional Subspace Problem with Relevant Dual Vector Tuples). For all security parameter $\lambda \in \mathbb{N}$, we define RDSP advantage of a probabilistic machine \mathcal{B} as follows:

$$\text{Adv}_{\mathcal{B}}^{\text{RDSP}}(\lambda) := \left| \Pr \left[\mathcal{B}(1^\lambda, \rho) \rightarrow 1 \mid \rho \xleftarrow{R} \mathcal{G}_0^{\text{RDSP}}(1^\lambda, n) \right] - \Pr \left[\mathcal{B}(1^\lambda, \rho) \rightarrow 1 \mid \rho \xleftarrow{R} \mathcal{G}_1^{\text{RDSP}}(1^\lambda, n) \right] \right|.$$

The RDSP assumption is: for any probabilistic polynomial-time adversary \mathcal{B} , $\text{Adv}_{\mathcal{B}}^{\text{RDSP}}(\lambda)$ is negligible in λ .

Definition 3 (IDSP: Decisional Subspace Problem with Irrelevant Dual Vector Tuples). The IDSP advantage of \mathcal{B} , $\text{Adv}_{\mathcal{B}}^{\text{IDSP}}(\lambda)$, and the IDSP assumption are defined similarly as in Definition 2.

In the generic DPVS model, basic operations in \mathbb{V}, \mathbb{V}^* , and \mathbb{G}_T , i.e., vector additions in \mathbb{V} and \mathbb{V}^* , multiplication in \mathbb{G}_T , pairing, and distortion maps w.r.t. \mathbb{A} or \mathbb{A}^* , are given by “generic” algorithms that act independently of the representations of vectors or group elements.

Theorem 1. The advantages $\text{Adv}_{\mathcal{B}}^{\text{RDSP}}(\lambda)$ and $\text{Adv}_{\mathcal{B}}^{\text{IDSP}}(\lambda)$ are $O(d/2^\lambda)$ for any adversary \mathcal{B} in the generic DPVS model, where d is the maximum of the degrees of polynomials of formal variables (in the generic model game).

We will describe the proof of Theorem 1 in the full version of this paper.

Remark (Intuition behind the Assumptions)

Here we informally explain the RDSP assumption by using a simplified one. In the simplified RDSP assumption, $(\mathbf{h}_1^*, \dots, \mathbf{h}_n^*)$ is given to \mathcal{A} in addition to $(\mathbb{B} := (\mathbf{b}_1, \dots, \mathbf{b}_{n+2}), \vec{y} := (y_1, \dots, y_n), \mathbf{e}_\beta)$, such that $\mathbf{h}_i^* := \omega \mathbf{b}_i^* + y_i \mathbf{b}_{n+1}^*$ ($i = 1, \dots, n$; $\omega \xleftarrow{U} \mathbb{F}_q$) and $\mathbf{e}_\beta := \delta_1 (\sum_{i=1}^n y_i \mathbf{b}_i) + \beta \zeta \mathbf{b}_{n+1} + \delta_2 \mathbf{b}_{n+2}$ ($\beta \xleftarrow{U} \{0, 1\}$, $\delta_1, \delta_2, \zeta \xleftarrow{U} \mathbb{F}_q$). The simplified RDSP assumption is that it is hard for any adversary \mathcal{A} , given $(\mathbb{B}, \vec{y}, \mathbf{e}_\beta)$ along with $(\mathbf{h}_1^*, \dots, \mathbf{h}_n^*)$, to correctly guess β . (In the DSP assumption, only $(\mathbb{B}, \vec{y}, \mathbf{e}_\beta)$ is given to \mathcal{A} .)

$(\mathbf{h}_1^*, \dots, \mathbf{h}_n^*)$ is added in the RDSP assumption in order to simulate the key generation oracle in the security proof of our encryption scheme as follows: for any $\vec{v} := (v_1, \dots, v_n)$ with $\vec{v} \cdot \vec{y} \neq 0$, the simulator can compute a secret key \mathbf{k}^* for \vec{v} such that $\mathbf{k}^* := \frac{1}{\vec{v} \cdot \vec{y}} \sum_{i=1}^n v_i \mathbf{h}_i = \frac{\omega}{\vec{v} \cdot \vec{y}} (\sum_{i=1}^n v_i \mathbf{b}_i^*) + \mathbf{b}_{n+1}^* = \omega' (\sum_{i=1}^n v_i \mathbf{b}_i^*) + \mathbf{b}_{n+1}^*$ where $\omega' := \frac{\omega}{\vec{v} \cdot \vec{y}}$.

This secret key generation procedure, however, does not work for \vec{v} with $\vec{v} \cdot \vec{y} = 0$, since $\frac{1}{\vec{v} \cdot \vec{y}}$ cannot be computed. Therefore, $(\mathbf{h}_1^*, \dots, \mathbf{h}_n^*)$ does not seem helpful to break the RDSP assumption, since a secret-key \mathbf{k}^* for \vec{v} with “ $\vec{v} \cdot \vec{y} = 0$ ” is of use to guess β by checking whether $e(\mathbf{e}_\beta, \mathbf{k}^*) = 1$ or not. Hence, the RDSP assumption seems to hold if the DSP assumption does.

Similarly the IDSP assumption is introduced as a variant of the DSP assumption. In the RDSP and IDSP assumptions employed in this paper, we use a public element $\mathbf{d}_{n+1} := \mathbf{b}_{n+1} + \mathbf{b}_{n+2}$ (in place of \mathbf{b}_{n+1} in basis \mathbb{B} in the simplified

one), and \mathbf{b}_{n+1} and \mathbf{b}_{n+2} are not published. Such a modification is required for the IDSP assumption since the simplified IDSP assumption does not hold.

In addition, in our RDSP (and IDSP) assumption, $\{\mathbf{h}_i^{(k)*}\}_{i=1,\dots,n; k=1,2,3}$ is employed in place of $\{\mathbf{h}_i^*\}_{i=1,\dots,n}$. This modification is introduced to *re-randomize* the coefficients for each key generation of the simulation by a random linear combination of $\mathbf{h}_i^{(1)*}$, $\mathbf{h}_i^{(2)*}$ and $\mathbf{h}_i^{(3)*}$.

4 Definition of Hierarchical Predicate Encryption (HPE)

This section defines hierarchical predicate encryption (HPE) for the class of hierarchical inner-product predicates and its security¹.

In a delegation system, it is required that a user who has a capability can delegate to another user a more restrictive capability. In addition to this requirement, our hierarchical inner-product encryption introduces a format of hierarchy $\vec{\mu}$ to define common delegation structure in a system.

We call a tuple of positive integers $\vec{\mu} := (n, d; \mu_1, \dots, \mu_d)$ s.t. $\mu_0 = 0 < \mu_1 < \mu_2 < \dots < \mu_d = n$ a format of hierarchy of depth d attribute spaces. Let Σ_ℓ ($\ell = 1, \dots, d$) be the sets of attributes, where each $\Sigma_\ell := \mathbb{F}_q^{\mu_\ell - \mu_{\ell-1}} \setminus \{\vec{0}\}$. Let the hierarchical attributes $\Sigma := \bigcup_{\ell=1}^d (\Sigma_1 \times \dots \times \Sigma_\ell)$, where the union is a disjoint union. Then, for $\vec{v}_i \in \mathbb{F}_q^{\mu_i - \mu_{i-1}} \setminus \{\vec{0}\}$, the hierarchical predicate $f_{(\vec{v}_1, \dots, \vec{v}_\ell)}$ on hierarchical attributes $(\vec{x}_1, \dots, \vec{x}_h) \in \Sigma$ is defined as follows: $f_{(\vec{v}_1, \dots, \vec{v}_\ell)}(\vec{x}_1, \dots, \vec{x}_h) = 1$ iff $\ell \leq h$ and $\vec{x}_i \cdot \vec{v}_i = 0$ for all i s.t. $1 \leq i \leq \ell$.

Let the space of hierarchical predicates $\mathcal{F} := \{f_{(\vec{v}_1, \dots, \vec{v}_\ell)} \mid \vec{v}_i \in \mathbb{F}_q^{\mu_i - \mu_{i-1}} \setminus \{\vec{0}\}\}$. We call h (resp. ℓ) the level of $(\vec{x}_1, \dots, \vec{x}_h)$ (resp. $(\vec{v}_1, \dots, \vec{v}_\ell)$).

Definition 4. Let $\vec{\mu} := (n, d; \mu_1, \dots, \mu_d)$ s.t. $\mu_0 = 0 < \mu_1 < \mu_2 < \dots < \mu_d = n$ be a format of hierarchy of depth d attribute spaces. A hierarchical predicate encryption (HPE) scheme for the class of hierarchical inner-product predicates \mathcal{F} over the set of hierarchical attributes Σ consists of probabilistic polynomial-time algorithms Setup, GenKey, Enc, Dec, and Delegate $_\ell$ for $\ell = 1, \dots, d-1$. They are given as follows:

- Setup takes as input security parameter 1^λ and format of hierarchy $\vec{\mu}$, and outputs (master) public key \mathbf{pk} and (master) secret key \mathbf{sk} .
- GenKey takes as input the master public key \mathbf{pk} , secret key \mathbf{sk} , and predicate vectors $(\vec{v}_1, \dots, \vec{v}_\ell)$. It outputs a corresponding secret key $\mathbf{sk}_{(\vec{v}_1, \dots, \vec{v}_\ell)}$.
- Enc takes as input the master public key \mathbf{pk} , attribute vectors $(\vec{x}_1, \dots, \vec{x}_h)$, where $1 \leq h \leq d$, and plaintext m in some associated plaintext space, msg . It returns ciphertext c .
- Dec takes as input the master public key \mathbf{pk} , secret key $\mathbf{sk}_{(\vec{v}_1, \dots, \vec{v}_\ell)}$, where $1 \leq \ell \leq d$, and ciphertext c . It outputs either plaintext m or the distinguished symbol \perp .

¹ More general delegation structures (partial order structures) than tree hierarchical structures can be easily realized in our HPE scheme. See Remark in Section 5.

- Delegate_ℓ takes as input the master public key pk , ℓ -th level secret key $\text{sk}_{(\vec{v}_1, \dots, \vec{v}_\ell)}$, and $(\ell + 1)$ -th level predicate vector $\vec{v}_{\ell+1}$. It returns $(\ell + 1)$ -th level secret key $\text{sk}_{(\vec{v}_1, \dots, \vec{v}_{\ell+1})}$.

A HPE scheme should have the following correctness property: for all correctly generated pk and $\text{sk}_{(\vec{v}_1, \dots, \vec{v}_\ell)}$, generate $c \xleftarrow{R} \text{Enc}(\text{pk}, m, (\vec{x}_1, \dots, \vec{x}_h))$ and $m' := \text{Dec}(\text{pk}, \text{sk}_{(\vec{v}_1, \dots, \vec{v}_\ell)}, c)$. If $f_{(\vec{v}_1, \dots, \vec{v}_\ell)}(\vec{x}_1, \dots, \vec{x}_h) = 1$, then $m' = m$. Otherwise, $m' \neq m$ except for negligible probability.

For f and f' in \mathcal{F} , we denote $f' \leq f$ if the predicate vector for f is a prefix of that for f' . For the following definition for key queries, see [22].

Remark: We will explain the hierarchical structure by using a small (toy) example that has three levels and each level consists of 2-dimensional space, i.e., 6-dimensional space is employed in total. That is, $\vec{\mu} := (n, d; \mu_1, \dots, \mu_d) = (6, 3; 2, 4, 6)$ in this example.

A user who possesses a secret key sk_1 in the top level, associated with the top level predicate vector $\vec{v}_1 := (v_1, v_2)$, can delegate any value (say $\vec{v}_2 := (v_3, v_4)$) of the second level key sk_2 such that the predicate vector for sk_2 is (\vec{v}_1, \vec{v}_2) . Similarly, a user who possesses a secret key in the second level, sk_2 with (\vec{v}_1, \vec{v}_2) , can delegate any value (say $\vec{v}_3 := (v_5, v_6)$) of the third level key sk_3 with $(\vec{v}_1, \vec{v}_2, \vec{v}_3)$.

Secret key sk_1 with \vec{v}_1 , can decrypt a ciphertext associated with attribute vector $(\vec{x}_1, (*, *), (*, *)) := ((x_1, x_2), (*, *), (*, *))$ if $\vec{x}_1 \cdot \vec{v}_1 = 0$, where $*$ denotes an arbitrary value. Secret key sk_2 with (\vec{v}_1, \vec{v}_2) can decrypt a ciphertext with attribute vector $(\vec{x}_1, \vec{x}_2, (*, *))$ if $\vec{x}_1 \cdot \vec{v}_1 = 0$ and $\vec{x}_2 \cdot \vec{v}_2 = 0$. However sk_2 cannot decrypt a ciphertext with higher level (top level) attribute vector $\vec{x}_1 := (x_1, x_2)$ (or $(\vec{x}_1, (*, *), (*, *))$). Therefore, the capability of a delegated key sk_2 is more limited than the parent key sk_1 .

Hence, when $(\vec{v}_1, \vec{v}_2) := ((v_1, v_2), (v_3, v_4))$ is a predicate vector for a secret key, (\vec{v}_1, \vec{v}_2) is considered to be $(\vec{v}_1, \vec{v}_2, (0, 0))$, and when $\vec{x}_1 := (x_1, x_2)$ is an attribute vector for a ciphertext, \vec{x}_1 is considered to be $(\vec{x}_1, (*, *), (*, *))$, where $(*, *) \cdot (0, 0) = 0$ and $(*, *) \cdot \vec{v}_2 \neq 0$ unless $\vec{v}_2 = (0, 0)$.

Definition 5. A hierarchical inner-product predicate encryption scheme for hierarchical predicates \mathcal{F} over hierarchical attributes Σ is selectively attribute-hiding (AH) against chosen plaintext attacks if for all probabilistic polynomial-time adversaries \mathcal{A} , the advantage of \mathcal{A} in the following experiment is negligible in the security parameter.

1. \mathcal{A} outputs challenge attribute vectors $\mathcal{X}^{(0)} := (\vec{x}_1^{(0)}, \dots, \vec{x}_{h(0)}^{(0)})$, $\mathcal{X}^{(1)} := (\vec{x}_1^{(1)}, \dots, \vec{x}_{h(1)}^{(1)})$.
2. Setup is run to generate keys pk and sk , and pk is given to \mathcal{A} .
3. \mathcal{A} may adaptively makes a polynomial number of queries of the following type:
 - [Create key] \mathcal{A} asks the challenger to create a secret key for a predicate $f \in \mathcal{F}$. The challenger creates a key for f without giving it to \mathcal{A} .

- [Create delegated key] \mathcal{A} specifies a key for predicate f that has already been created, and asks the challenger to perform a delegation operation to create a child key for $f' \leq f$. The challenger computes the child key without giving it to the adversary.
- [Reveal key] \mathcal{A} asks the challenger to reveal an already-created key for predicate f s.t. $f(\mathcal{X}^{(0)}) = f(\mathcal{X}^{(1)}) = 0$.

Note that when key creation requests are made, \mathcal{A} does not automatically see the created key. \mathcal{A} sees a key only when it makes a reveal key query.

4. \mathcal{A} outputs challenge plaintexts $m^{(0)}, m^{(1)}$.
5. A random bit b is chosen. \mathcal{A} is given $c^{(b)} \stackrel{R}{\leftarrow} \text{Enc}(\text{pk}, m^{(b)}, \mathcal{X}^{(b)})$.
6. The adversary may continue to request keys for additional predicate vectors subject to the restrictions given in step 3.
7. \mathcal{A} outputs a bit b' , and succeeds if $b' = b$.

We define the advantage of \mathcal{A} as the quantity $\text{Adv}_{\mathcal{A}}^{\text{HPE}, \text{AH}}(\lambda) := |\Pr [b' = b] - 1/2|$.

Remark: In Definition 5, adversary \mathcal{A} is not allowed to ask a key-query for $(\vec{v}_1, \dots, \vec{v}_\ell)$ such that $f_{(\vec{v}_1, \dots, \vec{v}_\ell)}(\mathcal{X}^{(b)}) = 1$ for some $b \in \{0, 1\}$, while in the security definition in 16, such a key-query is allowed provided that $m^{(0)} = m^{(1)}$ and $f_{(\vec{v}_1, \dots, \vec{v}_\ell)}(\mathcal{X}^{(0)}) = f_{(\vec{v}_1, \dots, \vec{v}_\ell)}(\mathcal{X}^{(1)}) = 1$. This restriction is introduced to prove the security of the proposed HPE scheme only under the RDSP and IDSP assumptions. If we introduce another variant of the assumptions, we can relax this restriction. We will describe this case in the full version of this paper.

5 The Proposed HPE Scheme

5.1 Key Idea in Constructing the Proposed HPE

We will explain a key idea of the proposed HPE scheme.

First, as a special (1-level) case of the proposed construction of HPE, we will show a predicate encryption (PE) construction for the inner-product predicate. Through the orthonormal property of (random) dual bases $(\mathbb{B} := (\mathbf{b}_1, \dots, \mathbf{b}_{n+3}), \mathbb{B}^* := (\mathbf{b}_1^*, \dots, \mathbf{b}_{n+3}^*))$ in DPVS, $(g, \mathbb{V}, \mathbb{V}^*, \mathbb{G}_T, \mathbb{A}, \mathbb{A}^*)$, (Sections 1.3, 2 and 3), the PE scheme for the (n -dimensional) inner-product predicate can be constructed as below, where \mathbb{V} and \mathbb{V}^* are $(n + 3)$ -dimensional spaces, the public parameter is $(\mathbf{b}_1, \dots, \mathbf{b}_n, \mathbf{d}_{n+1} := \mathbf{b}_{n+1} + \mathbf{b}_{n+2}, \mathbf{b}_{n+3})$ as well as the parameters of DPVS, and the master secret key is $(X$ and) \mathbb{B}^* . Ciphertext (c_1, c_2) for attribute $\vec{x} := (x_1, \dots, x_n) \in \mathbb{F}_q^n$ and plaintext $m \in \mathbb{G}_T$ is $\mathbf{c}_1 := \delta_1(x_1\mathbf{b}_1 + \dots + x_n\mathbf{b}_n) + \zeta\mathbf{d}_{n+1} + \delta_2\mathbf{b}_{n+3}$ and $c_2 := g_T^\zeta m$, where $\delta_1, \delta_2, \zeta \stackrel{U}{\leftarrow} \mathbb{F}_q$. Secret key \mathbf{k}^* with predicate $\vec{v} := (v_1, \dots, v_n) \in \mathbb{F}_q^n$ is $\mathbf{k}^* := \sigma(v_1\mathbf{b}_1^* + \dots + v_n\mathbf{b}_n^*) + \eta\mathbf{b}_{n+1}^* + (1 - \eta)\mathbf{b}_{n+2}^*$, where $\sigma, \eta \stackrel{U}{\leftarrow} \mathbb{F}_q$. If $\vec{x} \cdot \vec{v} = 0$, plaintext m can be computed by $m = c_2 / e(\mathbf{c}_1, \mathbf{k}^*)$, since $e(\mathbf{c}_1, \mathbf{k}^*) = (\prod_{i=1}^n e(\delta_1 x_i \mathbf{b}_i, \sigma v_i \mathbf{b}_i^*)) \cdot e(\zeta \mathbf{b}_{n+1}, \eta \mathbf{b}_{n+1}^*) \cdot e(\zeta \mathbf{b}_{n+2}, (1 - \eta) \mathbf{b}_{n+2}^*) = g_T^{\delta_1 \sigma (\sum_{i=1}^n x_i v_i) + \zeta \eta + \zeta (1 - \eta)} = g_T^{\delta_1 \sigma (\vec{x} \cdot \vec{v}) + \zeta} = g_T^\zeta$.

We now explain the key idea of the proposed HPE scheme by using a small (toy) example. Let the dimension of (predicate/attribute) vectors be 6, in which

there are three levels and each level has 2-dimensions, \mathbb{V} and \mathbb{V}^* be 9-dimensional spaces, the public parameter be $\mathbb{B} := (\mathbf{b}_1, \dots, \mathbf{b}_6, \mathbf{d}_7, \mathbf{b}_9)$ as well as the parameters of DPVS, and the master secret key be $(X \text{ and } \mathbb{B}^* := (\mathbf{b}_1^*, \dots, \mathbf{b}_9^*))$, where $\mathbf{d}_7 := \mathbf{b}_7 + \mathbf{b}_8$.

Ciphertext $(\mathbf{c}_1, \mathbf{c}_2)$ for attribute $\vec{x} := (\vec{x}_1, \vec{x}_2, \vec{x}_3) := ((x_1, x_2), (x_3, x_4), (x_5, x_6)) \in \mathbb{F}_q^6$ and plaintext m is constructed as $\mathbf{c}_1 := \delta_1(x_1\mathbf{b}_1 + x_2\mathbf{b}_2) + \dots + \delta_3(x_5\mathbf{b}_5 + x_6\mathbf{b}_6) + \zeta\mathbf{d}_7 + \delta_4\mathbf{b}_8$ and $\mathbf{c}_2 := g_T^\zeta m$, where $\delta_1, \dots, \delta_4, \zeta \stackrel{\cup}{\leftarrow} \mathbb{F}_q$. If the attribute is a higher level such as $\vec{x}_1 := (x_1, x_2)$, generate a modified attribute $\vec{x}^+ := ((x_1, x_2), (x_3^+, x_4^+), (x_5^+, x_6^+))$, where $(x_3^+, x_4^+, x_5^+, x_6^+) \stackrel{\cup}{\leftarrow} \mathbb{F}_q^4$. Then, ciphertext \mathbf{c}_1 for attribute \vec{x}_1 is computed as ciphertext \mathbf{c}_1 for the modified attribute \vec{x}^+ .

Top level secret key $\vec{\mathbf{k}}_1^* := (\mathbf{k}_{1,0}^*, \dots, \mathbf{k}_{1,6}^*)$, for predicate $\vec{v} := (v_1, v_2) \in \mathbb{F}_q^2$ consists of three parts, $\mathbf{k}_{1,0}^*$, $(\mathbf{k}_{1,1}^*, \mathbf{k}_{1,2}^*)$ and $(\mathbf{k}_{1,3}^*, \dots, \mathbf{k}_{1,6}^*)$, where the first one is used for decryption of ciphertexts, the second one for re-randomization (of delegated key), and the last one for delegation. Each part is: $\mathbf{k}_{1,0}^* := \sigma_{1,0}(v_1\mathbf{b}_1^* + v_2\mathbf{b}_2^*) + \eta_0\mathbf{b}_7^* + (1 - \eta_0)\mathbf{b}_8^*$, $\mathbf{k}_{1,j}^* := \sigma_{1,j}(v_1\mathbf{b}_1^* + v_2\mathbf{b}_2^*) + \eta_j\mathbf{b}_7^* - \eta_j\mathbf{b}_8^*$ ($j = 1, 2$), and $\mathbf{k}_{1,j}^* := \sigma_{1,j}(v_1\mathbf{b}_1^* + v_2\mathbf{b}_2^*) + \psi\mathbf{b}_j + \eta_j\mathbf{b}_7^* - \eta_j\mathbf{b}_8^*$ ($j = 3, \dots, 6$), where $\sigma_{1,j}, \psi \stackrel{\cup}{\leftarrow} \mathbb{F}_q$ for $j = 0, \dots, 6$. The first one, $\mathbf{k}_{1,0}^*$, can decrypt ciphertext $(\mathbf{c}_1, \mathbf{c}_2)$ by $\mathbf{c}_2/e(\mathbf{c}_1, \mathbf{k}_{1,0}^*)$, since $e(\mathbf{c}_1, \mathbf{k}_{1,0}^*) = g_T^\zeta$ if an attribute of \mathbf{c}_1 is $((x_1, x_2), (*, *), (*, *))$ with $(x_1, x_2) \cdot (v_1, v_2) = 0$. To delegate a secret key for the 2nd level vector (v_3, v_4) , $\sigma_{2,j}(v_3\mathbf{k}_{1,3}^* + v_4\mathbf{k}_{1,4}^*)$ is added to $\mathbf{k}_{1,0}^*$ ($j = 0$), $\mathbf{0}$ ($j = 1, 2, 3$), and $\psi^+\mathbf{k}_{1,j}^*$ ($j = 5, 6$). To re-randomize the coefficients of $(v_1\mathbf{b}_1^* + v_2\mathbf{b}_2^*)$, \mathbf{b}_7^* and \mathbf{b}_8^* in the delegated key, $(\alpha_{j,1}\mathbf{k}_{1,1}^* + \alpha_{j,2}\mathbf{k}_{1,2}^*)$ is also added. So, the delegated key (the second level key) $\vec{\mathbf{k}}_2^* := (\mathbf{k}_{2,0}^*, \dots, \mathbf{k}_{2,3}^*, \mathbf{k}_{2,5}^*, \mathbf{k}_{2,6}^*)$, (where $\mathbf{k}_{2,0}^*$ is for decryption, $(\mathbf{k}_{2,1}^*, \dots, \mathbf{k}_{2,3}^*)$ for re-randomization, and $(\mathbf{k}_{2,5}^*, \mathbf{k}_{2,6}^*)$ for delegation) is computed as $\mathbf{k}_{2,0}^* := \mathbf{k}_{1,0}^* + (\alpha_{0,1}\mathbf{k}_{1,1}^* + \alpha_{0,2}\mathbf{k}_{1,2}^*) + \sigma_{2,0}(v_3\mathbf{k}_{1,3}^* + v_4\mathbf{k}_{1,4}^*)$, $\mathbf{k}_{2,j}^* := (\alpha_{j,1}\mathbf{k}_{1,1}^* + \alpha_{j,2}\mathbf{k}_{1,2}^*) + \sigma_{2,j}(v_3\mathbf{k}_{1,3}^* + v_4\mathbf{k}_{1,4}^*)$ ($j = 1, 2, 3$), and $\mathbf{k}_{2,j}^* := \psi^+\mathbf{k}_{1,j}^* + (\alpha_{j,1}\mathbf{k}_{1,1}^* + \alpha_{j,2}\mathbf{k}_{1,2}^*) + \sigma_{2,j}(v_3\mathbf{k}_{1,3}^* + v_4\mathbf{k}_{1,4}^*)$ ($j = 5, 6$), where $\alpha_{j,1}, \alpha_{j,2}, \sigma_{2,j}, \psi^+ \stackrel{\cup}{\leftarrow} \mathbb{F}_q$ ($j = 0, 1, 2, 3, 5, 6$). Then, the distribution of the delegated key (by Delegate) is equivalent to that obtained by the key generation query (GenKey) except negligible probability (i.e., the simulation of ‘create delegated key query’ can be equivalent to that of ‘create key query’).

In general, as for the ℓ -th level secret key, $\vec{\mathbf{k}}_\ell^* := (\mathbf{k}_{\ell,0}^*, \dots, \mathbf{k}_{\ell,\ell+1}^*, \mathbf{k}_{\ell,\mu_\ell+1}^*, \dots, \mathbf{k}_{\ell,n}^*)$, the first one, $\mathbf{k}_{\ell,0}^*$, is used for decryption, the second part of components, $\mathbf{k}_{\ell,1}^*, \dots, \mathbf{k}_{\ell,\ell+1}^*$, are for re-randomization (of a delegated key), and the last part of components, $\mathbf{k}_{\ell,\mu_\ell+1}^*, \dots, \mathbf{k}_{\ell,n}^*$, are for delegation.

5.2 HPE Scheme

Setup($1^\lambda, \vec{\mu} := (n, d; \mu_1, \dots, \mu_d)$) : (param, \mathbb{B}, \mathbb{B}^*) $\stackrel{R}{\leftarrow} \mathcal{G}_{\text{ob}}(1^\lambda, n + 3)$,
 $\mathbf{d}_{n+1} := \mathbf{b}_{n+1} + \mathbf{b}_{n+2}$, $\widehat{\mathbb{B}} := (\mathbf{b}_1, \dots, \mathbf{b}_n, \mathbf{d}_{n+1}, \mathbf{b}_{n+3})$,
 return sk := (X, \mathbb{B}^*) , pk := $(1^\lambda, \text{param}, \widehat{\mathbb{B}})$.

GenKey(pk, sk, $(\vec{v}_1, \dots, \vec{v}_\ell) := ((v_1, \dots, v_{\mu_1}), \dots, (v_{\mu_{\ell-1}+1}, \dots, v_{\mu_\ell}))$:
 $\sigma_{j,i}, \psi, \eta_j \xleftarrow{\cup} \mathbb{F}_q$ for $j = 0, \dots, \ell + 1, \mu_\ell + 1, \dots, n$; $i = 1, \dots, \ell$,
 $\mathbf{k}_{\ell,0}^* := \sum_{t=1}^{\ell} \sigma_{0,t} (\sum_{i=\mu_{t-1}+1}^{\mu_t} v_i \mathbf{b}_i^*) + \eta_0 \mathbf{b}_{n+1}^* + (1 - \eta_0) \mathbf{b}_{n+2}^*$,
 $\mathbf{k}_{\ell,j}^* := \sum_{t=1}^{\ell} \sigma_{j,t} (\sum_{i=\mu_{t-1}+1}^{\mu_t} v_i \mathbf{b}_i^*) + \eta_j \mathbf{b}_{n+1}^* - \eta_j \mathbf{b}_{n+2}^*$
for $j = 1, \dots, \ell + 1$,
 $\mathbf{k}_{\ell,j}^* := \sum_{t=1}^{\ell} \sigma_{j,t} (\sum_{i=\mu_{t-1}+1}^{\mu_t} v_i \mathbf{b}_i^*) + \psi \mathbf{b}_j^* + \eta_j \mathbf{b}_{n+1}^* - \eta_j \mathbf{b}_{n+2}^*$
for $j = \mu_\ell + 1, \dots, n$,
return $\vec{\mathbf{k}}_\ell^* := (\mathbf{k}_{\ell,0}^*, \dots, \mathbf{k}_{\ell,\ell+1}^*, \mathbf{k}_{\ell,\mu_\ell+1}^*, \dots, \mathbf{k}_{\ell,n}^*)$.

Enc(pk, $m \in \mathbb{G}_T$, $(\vec{x}_1, \dots, \vec{x}_\ell) := ((x_1, \dots, x_{\mu_1}), \dots, (x_{\mu_{\ell-1}+1}, \dots, x_{\mu_\ell}))$:
 $(\vec{x}_{\ell+1}, \dots, \vec{x}_d) \xleftarrow{\cup} \mathbb{F}_q^{\mu_{\ell+1}-\mu_\ell} \times \dots \times \mathbb{F}_q^{n-\mu_{d-1}}$, $\delta_1, \dots, \delta_d, \delta_{n+3}, \zeta \xleftarrow{\cup} \mathbb{F}_q$,
 $\mathbf{c}_1 := \sum_{t=1}^d \delta_t (\sum_{i=\mu_{t-1}+1}^{\mu_t} x_i \mathbf{b}_i) + \zeta \mathbf{d}_{n+1} + \delta_{n+3} \mathbf{b}_{n+3}$, $\mathbf{c}_2 := g_T^\zeta m$,
return $(\mathbf{c}_1, \mathbf{c}_2)$.

Dec(pk, $\mathbf{k}_{\ell,0}^*$, $\mathbf{c}_1, \mathbf{c}_2$) : $m' := \mathbf{c}_2 / e(\mathbf{c}_1, \mathbf{k}_{\ell,0}^*)$,
return m' .

Delegate $_\ell$ (pk, $\vec{\mathbf{k}}_\ell^*$, $\vec{v}_{\ell+1} := (v_{\mu_\ell+1}, \dots, v_{\mu_{\ell+1}})$) :

$\alpha_{j,i}, \sigma_j, \psi' \xleftarrow{\cup} \mathbb{F}_q$ for $j = 0, \dots, \ell + 2, \mu_{\ell+1} + 1, \dots, n$; $i = 1, \dots, \ell + 1$,
 $\mathbf{k}_{\ell+1,0}^* := \mathbf{k}_{\ell,0}^* + \sum_{i=1}^{\ell+1} \alpha_{0,i} \mathbf{k}_{\ell,i}^* + \sigma_0 (\sum_{i=\mu_\ell+1}^{\mu_{\ell+1}} v_i \mathbf{k}_{\ell,i}^*)$,
 $\mathbf{k}_{\ell+1,j}^* := \sum_{i=1}^{\ell+1} \alpha_{j,i} \mathbf{k}_{\ell,i}^* + \sigma_j (\sum_{i=\mu_\ell+1}^{\mu_{\ell+1}} v_i \mathbf{k}_{\ell,i}^*)$ for $j = 1, \dots, \ell + 2$,
 $\mathbf{k}_{\ell+1,j}^* := \sum_{i=1}^{\ell+1} \alpha_{j,i} \mathbf{k}_{\ell,i}^* + \sigma_j (\sum_{i=\mu_\ell+1}^{\mu_{\ell+1}} v_i \mathbf{k}_{\ell,i}^*) + \psi' \mathbf{k}_{\ell,j}^*$ for $j = \mu_{\ell+1} + 1, \dots, n$,
return $\vec{\mathbf{k}}_{\ell+1}^* := (\mathbf{k}_{\ell+1,0}^*, \dots, \mathbf{k}_{\ell+1,\ell+2}^*, \mathbf{k}_{\ell+1,\mu_{\ell+1}+1}^*, \dots, \mathbf{k}_{\ell+1,n}^*)$.

[Correctness] Assume that ciphertext $(\mathbf{c}_1, \mathbf{c}_2)$ is generated by Enc(pk, $m, (\vec{x}_1, \dots, \vec{x}_h)$) and secret key $\mathbf{k}_{\ell,0}^*$ is generated by GenKey(pk, sk, $(\vec{v}_1, \dots, \vec{v}_\ell)$). Note that $e(\mathbf{c}_1, \mathbf{k}_{\ell,0}^*) = g_T^{\sum_{1 \leq i \leq \ell} \sigma_i \delta_i \vec{x}_i \cdot \vec{v}_i + \zeta}$. If $\ell \leq h$ and $\vec{x}_i \cdot \vec{v}_i = 0$ for all i s.t. $1 \leq i \leq \ell$, then $e(\mathbf{c}_1, \mathbf{k}_{\ell,0}^*) = g_T^\zeta$. Otherwise, $e(\mathbf{c}_1, \mathbf{k}_{\ell,0}^*)$ is uniformly distributed. Hence, correctness holds for secret keys generated by GenKey, and it also holds for keys generated by Delegate by Claim [II](#).

Remark: A generalized delegation (not limited to a hierarchical delegation) system can be constructed on (1-level) PE described in the first part of Section [5.1](#), where the parameters are the same as above.

In the generalized delegatable PE scheme, secret key generation procedure GenKey(pk, sk, $\vec{v}_1 := (v_{1,1}, \dots, v_{1,n})$) outputs $\vec{\mathbf{k}}_1^* := (\mathbf{k}_{1,\text{dec}}^*, \mathbf{k}_{1,\text{ran},1}^*, \mathbf{k}_{1,\text{ran},2}^*, \mathbf{k}_{1,\text{del},1}^*, \dots, \mathbf{k}_{1,\text{del},n}^*)$, where $\mathbf{k}_{1,\text{dec}}^* := \sigma_{\text{dec}} (\sum_{i=1}^n v_{1,i} \mathbf{b}_i^*) + \eta_{\text{dec}} \mathbf{b}_{n+1}^* + (1 - \eta_{\text{dec}}) \mathbf{b}_{n+2}^*$; $\mathbf{k}_{1,\text{ran},j}^* := \sigma_{\text{ran},j} (\sum_{i=1}^n v_{1,i} \mathbf{b}_i^*) + \eta_{\text{ran},j} \mathbf{b}_{n+1}^* - \eta_{\text{ran},j} \mathbf{b}_{n+2}^*$ ($j = 1, 2$); $\mathbf{k}_{1,\text{del},j}^* := \sigma_{\text{del},j} (\sum_{i=1}^n v_{1,i} \mathbf{b}_i^*) + \psi \mathbf{b}_j^* + \eta_{\text{del},j} \mathbf{b}_{n+1}^* - \eta_{\text{del},j} \mathbf{b}_{n+2}^*$ ($j = 1, \dots, n$).

To delegate secret key $\vec{\mathbf{k}}_1^*$ for $\vec{v}_2 := (v_{2,1}, \dots, v_{2,n})$, where $\vec{v}_2 \notin \text{span}(\vec{v}_1)$, Delegate $_1$ (pk, $\vec{\mathbf{k}}_1^*$, \vec{v}_2) outputs $\vec{\mathbf{k}}_2^* := (\mathbf{k}_{2,\text{dec}}^*, \mathbf{k}_{2,\text{ran},1}^*, \mathbf{k}_{2,\text{ran},2}^*, \mathbf{k}_{2,\text{del},1}^*, \dots,$

$\mathbf{k}_{2,\text{del},n}^*$). Here, $\mathbf{k}_{2,\text{dec}}^* := \mathbf{k}_{1,\text{dec}}^* + \sum_{i=1}^2 \alpha_{\text{dec},i} \mathbf{k}_{1,\text{ran},i}^* + \sigma_{2,\text{dec}}(\sum_{i=1}^n v_{2,i} \mathbf{k}_{1,\text{del},i}^*)$; $\mathbf{k}_{2,\text{ran},j}^* := \sum_{i=1}^2 \alpha_{\text{ran},i} \mathbf{k}_{1,\text{ran},i}^* + \sigma_{2,\text{ran},j}(\sum_{i=1}^n v_{2,i} \mathbf{k}_{1,\text{del},i}^*)$ ($j = 1, 2, 3$); $\mathbf{k}_{2,\text{del},j}^* := \sum_{i=1}^2 \alpha_{\text{del},i} \mathbf{k}_{1,\text{ran},i}^* + \sigma_{2,\text{del},j}(\sum_{i=1}^n v_{2,i} \mathbf{k}_{1,\text{del},i}^*) + \psi' \mathbf{k}_{1,\text{del},j}^*$ ($j = 1, \dots, n$). Further delegation for $\vec{\mathbf{k}}_\ell^*$ ($\ell = 2, 3, \dots$) can be done in the same manner.

Ciphertext $(\mathbf{c}_1, \mathbf{c}_2)$ for attribute $\vec{x} := (x_1, \dots, x_n)$ and plaintext $m \in \mathbb{G}_T$ is the same as that of the 1-level PE. Key $\vec{\mathbf{k}}_1^*$ can decrypt $(\mathbf{c}_1, \mathbf{c}_2)$ if $\vec{v}_1 \cdot \vec{x} = 0$, and key $\vec{\mathbf{k}}_2^*$ can decrypt $(\mathbf{c}_1, \mathbf{c}_2)$ if $(\vec{v}_1 \cdot \vec{x} = 0) \wedge (\vec{v}_2 \cdot \vec{x} = 0)$. Namely the capability of delegated key $\vec{\mathbf{k}}_2^*$ is more limited than that of its parent key $\vec{\mathbf{k}}_1^*$. In general, the ℓ -th delegated secret key $\vec{\mathbf{k}}_\ell^*$ can decrypt $(\mathbf{c}_1, \mathbf{c}_2)$ if $(\vec{v}_1 \cdot \vec{x} = 0) \wedge \dots \wedge (\vec{v}_\ell \cdot \vec{x} = 0)$, where $\vec{v}_j \notin \text{span}\langle \vec{v}_1, \dots, \vec{v}_{j-1} \rangle$ for $2 \leq j \leq \ell$.

5.3 Security

Theorem 2. *The proposed HPE scheme is selectively attribute-hiding against chosen plaintext attacks under the RDSP and IDSP assumptions. For any adversary \mathcal{A} , there exist probabilistic machines \mathcal{B}_1 and \mathcal{B}_2 , whose running times are essentially the same as that of \mathcal{A} , such that for any security parameter λ ,*

$$\text{Adv}_{\mathcal{A}}^{\text{HPE,AH}}(\lambda) \leq \text{Adv}_{\mathcal{B}_1}^{\text{RDSP}}(\lambda) + \text{Adv}_{\mathcal{B}_2}^{\text{IDSP}}(\lambda) + 3\nu/q$$

where ν is the number of adversary’s queries.

Proof Outline: To prove the security, we employ five games, Game 0 (original selective-security game) to Game 4 whose advantage is 0, where, roughly, Game 1 is conceptually changed (the timing of challenger’s coin flips is changed) from Game 0, a delegated key query (i.e., a reveal query of an already-created delegated key) is replied by using GenKey (in place of Delegate) in Game 2, the plaintext part of the target ciphertext is randomized in Game 3, and the attribute vector part of the target ciphertext is randomized in Game 4.

Since the distribution regarding each revealed key query in Game 2 is equivalent to that in Game 1 except with probability at most $3/q$, the gap between Games 1 and 2 is bounded by $3\nu/q$.

To prove that the gap between Games 2 and 3 is bounded by the advantage of the RDSP assumption, target ciphertext $(\mathbf{c}_1, \mathbf{c}_2)$ for $m^{(b)}$ is generated by using \mathbf{e}_β from the RDSP assumption such that $\mathbf{c}_1 := \mathbf{e}_\beta + \zeta \mathbf{d}_{n+1}$ and $\mathbf{c}_2 := g_T^\zeta m^{(b)}$. Then $(\mathbf{c}_1, \mathbf{c}_2)$ is a ciphertext in Game 2 when $\beta = 0$, and it is a ciphertext in Game 3 when $\beta = 1$. The key generation oracle simulation can be perfectly executed by using $\{\mathbf{h}_i^{(k)*}, \tau_i^{(k)}\}_{i=1,\dots,n;k=1,2,3}$ from the RDSP assumption (see Remark after Theorem [1](#)). It can be done similarly to evaluate the gap between Games 3 and 4 (through the IDSP assumption).

Proof of Theorem [2](#)

To prove Theorem [2](#), we consider the following five games.

Game 0: Original game (Definition 5).

Game 1: Game 1 is the same as Game 0 except the following procedures:

1. When challenger \mathcal{C} gets challenge attributes $(\vec{x}_1^{(0)}, \dots, \vec{x}_{h(0)}^{(0)})$ and $(\vec{x}_1^{(1)}, \dots, \vec{x}_{h(1)}^{(1)})$ in the first step of the game, \mathcal{C} selects (challenge) bit $b \xleftarrow{\text{U}} \{0, 1\}$, and computes

$$(x_1^+, \dots, x_n^+) := (\delta_1 \vec{x}_1, \dots, \delta_d \vec{x}_d),$$

where $h := h^{(b)}$, $(\vec{x}_1, \dots, \vec{x}_h) := (\vec{x}_1^{(b)}, \dots, \vec{x}_h^{(b)})$, $(\vec{x}_{h+1}, \dots, \vec{x}_d) \xleftarrow{\text{U}} \mathbb{F}_q^{\mu_{h+1} - \mu_h} \times \dots \times \mathbb{F}_q^{n - \mu_{d-1}}$, and $\delta_1, \dots, \delta_d \xleftarrow{\text{U}} \mathbb{F}_q$.

2. When \mathcal{C} gets challenge plaintexts $(m^{(0)}, m^{(1)})$ from adversary \mathcal{A} , challenger \mathcal{C} computes (c_1, c_2) as below and returns it to \mathcal{A} .

$$c_1 := \sum_{i=1}^n x_i^+ \mathbf{b}_i + \zeta \mathbf{d}_{n+1} + \delta_{n+3} \mathbf{b}_{n+3}, \quad c_2 := g_T^{\zeta} m^{(b)},$$

where $\delta_{n+3}, \zeta \xleftarrow{\text{U}} \mathbb{F}_q$.

Game 2: Game 2 is the same as Game 1 except the following procedures.

1. When a create key query is issued by \mathcal{A} , challenger \mathcal{C} only records the specified predicates, and when a create delegated key query is issued, \mathcal{C} only records the specified keys and predicates. In this step, \mathcal{C} just records, but creates no corresponding keys.
2. When a reveal key query is issued for a hierarchical (level- ℓ) predicate $(\vec{v}_1, \dots, \vec{v}_\ell)$ which has been already recorded, \mathcal{C} creates the queried key by using GenKey. In addition, there is a special rule such that $(\sigma_{0,1}, \dots, \sigma_{0,\ell}) \xleftarrow{\text{U}} \mathbb{F}_q^\ell$ is selected again if $\sum_{t=1}^\ell \sigma_{0,t} \delta_t \vec{x}_t \cdot \vec{v}_t = 0$ in the computation process of GenKey.

Game 3: Game 3 is the same as Game 2 except the target ciphertext (c_1, c_2) is generated as follows:

$$c_1 := \sum_{i=1}^n x_i^+ \mathbf{b}_i + \zeta_1 \mathbf{b}_{n+1} + \zeta_2 \mathbf{b}_{n+2} + \delta_{n+3} \mathbf{b}_{n+3}, \quad c_2 := g_T^{\zeta} m^{(b)},$$

where $\delta_{n+3}, \zeta, \zeta_1, \zeta_2 \xleftarrow{\text{U}} \mathbb{F}_q$.

Game 4: Game 4 is the same as Game 3 except the target ciphertext (c_1, c_2) is generated as follows:

$$c_1 := \sum_{i=1}^n u_i \mathbf{b}_i + \zeta_1 \mathbf{b}_{n+1} + \zeta_2 \mathbf{b}_{n+2} + \delta_{n+3} \mathbf{b}_{n+3}, \quad c_2 := g_T^{\zeta} m^{(b)},$$

where $\delta_{n+3}, \zeta, \zeta_1, \zeta_2 \xleftarrow{\text{U}} \mathbb{F}_q$ and $\vec{u} := (u_1, \dots, u_n) \xleftarrow{\text{U}} \mathbb{F}_q^n \setminus \{\vec{0}\}$.

Let $\text{Adv}_{\mathcal{A}}^{(0)}(\lambda)$ be $\text{Adv}_{\mathcal{A}}^{\text{HPE,AH}}(\lambda)$ in Game 0, and $\text{Adv}_{\mathcal{A}}^{(i)}(\lambda)$ ($i = 1, \dots, 4$) be the advantage of \mathcal{A} in Game i . It is clear that $\text{Adv}_{\mathcal{A}}^{(0)}(\lambda) = \text{Adv}_{\mathcal{A}}^{(1)}(\lambda)$, since it is a conceptual change. It is also clear that $\text{Adv}_{\mathcal{A}}^{(4)}(\lambda) = 0$ by Lemma 4.

We will show three lemmas (Lemmas 1, 2, 3) that evaluate the gaps between pairs of $\text{Adv}_{\mathcal{A}}^{(i)}(\lambda)$ ($i = 1, 2, 3, 4$). From these lemmas, we obtain $\text{Adv}_{\mathcal{A}}^{\text{HPE,AH}}(\lambda) = \text{Adv}_{\mathcal{A}}^{(0)}(\lambda) = \text{Adv}_{\mathcal{A}}^{(1)}(\lambda) \leq \sum_{i=1}^3 \left| \text{Adv}_{\mathcal{A}}^{(i)}(\lambda) - \text{Adv}_{\mathcal{A}}^{(i+1)}(\lambda) \right| + \text{Adv}_{\mathcal{A}}^{(4)}(\lambda) \leq \text{Adv}_{\mathcal{B}_1}^{\text{RDSP}}(\lambda) + \text{Adv}_{\mathcal{B}_2}^{\text{IDSP}}(\lambda) + 3\nu/q$. \square

Lemma 1. For any adversary \mathcal{A} , $|\text{Adv}_{\mathcal{A}}^{(1)}(\lambda) - \text{Adv}_{\mathcal{A}}^{(2)}(\lambda)| \leq 3\nu/q$.

Proof. The distribution of $\vec{\mathbf{k}}_{\ell+1}^*$ generated by GenKey for a level- $(\ell+1)$ predicate is equivalent to that by the combination of GenKey for the level- ℓ predicate and Delegate_{ℓ} except with probability $2/q$, from Claim [1](#). Moreover, the special rule in Game 2 causes probability gap at most $1/q$ for each GenKey operation. Therefore, the revealed key distribution in Game 1 is equivalent to that in Game 2 except with probability at most $(1 - (1 - 3/q)^\nu) \leq 3\nu/q$, since the number of delegate queries is upper-bounded by ν . Hence (by using Shoup's difference lemma), the difference of $\text{Adv}_{\mathcal{A}}^{(1)}(\lambda)$ and $\text{Adv}_{\mathcal{A}}^{(2)}(\lambda)$ is upper-bounded by $3\nu/q$. \square

Claim 1. If $\vec{\mathbf{k}}_{\ell}^*$ is generated by $\text{GenKey}(\text{pk}, \text{sk}, (\vec{v}_1, \dots, \vec{v}_{\ell}))$, the distribution of $\vec{\mathbf{k}}_{\ell+1}^*$ generated by $\text{Delegate}(\text{pk}, \vec{\mathbf{k}}_{\ell}^*, \vec{v}_{\ell+1})$ is equivalent to that of $\vec{\mathbf{k}}_{\ell+1}^*$ generated by $\text{GenKey}(\text{pk}, \text{sk}, (\vec{v}_1, \dots, \vec{v}_{\ell}, \vec{v}_{\ell+1}))$ except with probability at most $2/q$.

Proof. The distribution of level- ℓ key $\mathbf{k}_{\ell,j}^*$ ($j = 1, \dots, \ell + 1$) is represented by that of the $\ell + 1$ coefficients, $(\sigma_{j,1}, \dots, \sigma_{j,\ell}, \eta_j)$, of $\sum_{i=\mu_{t-1}+1}^{\mu_t} v_i \mathbf{b}_{\ell,i}^*$ ($t = 1, \dots, \ell$) and \mathbf{b}_{n+1}^* (and the coefficient, ψ , of \mathbf{b}_j^* in addition when $j = \mu_{\ell} + 1, \dots, n$), since the coefficient of \mathbf{b}_{n+2}^* is dependent of that of \mathbf{b}_{n+1}^* .

Similarly, the distribution of level- $(\ell + 1)$ key $\mathbf{k}_{\ell+1,j}^*$ ($j = 1, \dots, \ell + 2$) is represented by that of the $\ell + 2$ coefficients, $(\sigma_{j,1}, \dots, \sigma_{j,\ell+1}, \eta_j)$.

When level- ℓ key $\mathbf{k}_{\ell,j}^*$ ($j = 1, \dots, \ell + 1$) is generated by $\text{GenKey}(\text{pk}, \text{sk}, (\vec{v}_1, \dots, \vec{v}_{\ell}))$, $(\sigma_{j,1}, \dots, \sigma_{j,\ell}, \eta_j)_{j=1, \dots, \ell+1}$ is uniformly distributed.

If coefficient matrix $(\sigma_{j,1}, \dots, \sigma_{j,\ell}, \eta_j)_{j=1, \dots, \ell+1}$ ($(\ell + 1) \times (\ell + 1)$ matrix) of $(\mathbf{k}_{\ell,j}^*)_{j=1, \dots, \ell+1}$ is regular and $\psi \neq 0$, then the coefficients, $(\sigma_{j,1}, \dots, \sigma_{j,\ell+1}, \eta_j)$, of $\text{Delegate}(\text{pk}, \vec{\mathbf{k}}_{\ell}^*, \vec{v}_{\ell+1})$ is uniformly distributed, i.e., $\text{Delegate}(\text{pk}, \vec{\mathbf{k}}_{\ell}^*, \vec{v}_{\ell+1})$ is equivalently distributed as $\text{GenKey}(\text{pk}, \text{sk}, (\vec{v}_1, \dots, \vec{v}_{\ell+1}))$.

Here, $(\sigma_{j,1}, \dots, \sigma_{j,\ell}, \eta_j)_{j=1, \dots, \ell+1}$ ($(\ell + 1) \times (\ell + 1)$ matrix) of $(\mathbf{k}_{\ell,j}^*)_{j=1, \dots, \ell+1}$ is regular and $\psi \neq 0$ except with probability at most $2/q$. \square

Lemma 2. For any adversary \mathcal{A} , there exists a probabilistic machine \mathcal{B}_1 , whose running time is essentially the same as that of \mathcal{A} , such that for any security parameter λ , $|\text{Adv}_{\mathcal{A}}^{(2)}(\lambda) - \text{Adv}_{\mathcal{A}}^{(3)}(\lambda)| = \text{Adv}_{\mathcal{B}_1}^{\text{RDSP}}(\lambda)$.

Proof. In order to prove Lemma [2](#), we construct a probabilistic machine \mathcal{B}_1 against the RDSP problem by using any adversary \mathcal{A} in a security game (Game 2 or 3) as a black box as follows:

1. \mathcal{B}_1 is given RDSP instance $(\text{param}, \widehat{\mathbb{B}}, \{\mathbf{h}_i^{(k)*}, \tau_i^{(k)}\}_{i=1, \dots, n; k=1, 2, 3}, \vec{y}, \mathbf{e}_{\beta})$.
2. \mathcal{B}_1 plays a role of challenger \mathcal{C} in the security game against adversary \mathcal{A} .
3. When \mathcal{B}_1 (or challenger \mathcal{C}) gets challenge attributes $(\vec{x}_1^{(0)}, \dots, \vec{x}_{h^{(0)}}^{(0)})$ and $(\vec{x}_1^{(1)}, \dots, \vec{x}_{h^{(1)}}^{(1)})$ in the first step of the game, \mathcal{B}_1 selects (challenge) bit $b \stackrel{\mathcal{U}}{\leftarrow} \{0, 1\}$, and computes

$$(x_1^+, \dots, x_n^+) := (\delta_1 \vec{x}_1, \dots, \delta_d \vec{x}_d),$$

where $h := h^{(b)}$, $(\vec{x}_1, \dots, \vec{x}_h) := (\vec{x}_1^{(b)}, \dots, \vec{x}_h^{(b)})$, $(\vec{x}_{h+1}, \dots, \vec{x}_d) \stackrel{\cup}{\leftarrow} \mathbb{F}_q^{\mu_{h+1} - \mu_h}$
 $\times \dots \times \mathbb{F}_q^{n - \mu_{d-1}}$, and $\delta_1, \dots, \delta_d \stackrel{\cup}{\leftarrow} \mathbb{F}_q$.

Let $(\pi_{i,j}) \stackrel{\cup}{\leftarrow} \{ \Pi \in GL(n, \mathbb{F}_q) \mid \vec{y} = \vec{x}^+ \cdot \Pi, \Pi^T = \Pi \}$, and $\Pi^* := (\pi_{i,j}^*) := ((\pi_{i,j})^T)^{-1}$. Note that $\vec{x}^+ = \vec{y} \cdot \Pi^*$. Public parameter pk is then calculated as follows and \mathcal{B}_1 returns pk to \mathcal{A} :

$$\begin{aligned} \tilde{\mathbf{b}}_j &:= \sum_{\varrho=1}^n \pi_{j,\varrho} \mathbf{b}_{\varrho}, \quad \tilde{\mathbf{b}}_j^* := \sum_{\varrho=1}^n \pi_{j,\varrho}^* \mathbf{b}_{\varrho}^* \quad (j = 1, \dots, n), \\ \tilde{\mathbb{B}} &:= (\tilde{\mathbf{b}}_1, \dots, \tilde{\mathbf{b}}_n, \mathbf{d}_{n+1}, \mathbf{b}_{n+3}), \quad \text{pk} := (1^\lambda, \text{param}, \tilde{\mathbb{B}}). \end{aligned}$$

4. When a reveal key query is issued for a hierarchical (level- ℓ) predicate $(\vec{v}_1, \dots, \vec{v}_\ell)$ which has been already recorded, \mathcal{B}_1 answers as follows: for $j = 0, \dots, \ell + 1, \mu_\ell + 1, \dots, n$, \mathcal{B}_1 calculates

$$\vec{v}_j^+ := (v_{j,1}^+, \dots, v_{j,\mu_\ell}^+) := (\sigma_{j,1} \vec{v}_1, \dots, \sigma_{j,\ell} \vec{v}_\ell), \quad (1)$$

where $\sigma_{j,1}, \dots, \sigma_{j,\ell} \stackrel{\cup}{\leftarrow} \mathbb{F}_q$. Then, \mathcal{B}_1 calculates and returns $\vec{\mathbf{k}}_\ell^* := (\mathbf{k}_{\ell,0}^*, \dots, \mathbf{k}_{\ell+1,\ell+1}^*, \mathbf{k}_{\ell,\mu_\ell+1}^*, \dots, \mathbf{k}_{\ell,n}^*)$ using $\{\mathbf{h}_i^{(k)*}, \tau_i^{(k)}\}$ in the RDSP instance:

$$\begin{aligned} \theta_0 &:= \sum_{k=1}^3 a_{0,k} \sum_{i=1}^{\mu_\ell} v_{0,i}^+ \sum_{\varrho=1}^n \pi_{i,\varrho}^* \tau_\varrho^{(k)}, \\ \mathbf{k}_{\ell,0}^* &:= \theta_0^{-1} \sum_{k=1}^3 a_{0,k} \sum_{i=1}^{\mu_\ell} v_{0,i}^+ \sum_{\varrho=1}^n \pi_{i,\varrho}^* \mathbf{h}_\varrho^{(k)*}, \\ \text{For } j &= 1, \dots, \ell + 1, \mu_\ell + 1, \dots, n; s = 1, 2, \\ \theta_{j,s} &:= \sum_{k=1}^3 a_{j,k,s} \sum_{i=1}^{\mu_\ell} v_{j,i}^+ \sum_{\varrho=1}^n \pi_{i,\varrho}^* \tau_\varrho^{(k)}, \\ \mathbf{f}_{j,s}^* &:= \sum_{k=1}^3 a_{j,k,s} \sum_{i=1}^{\mu_\ell} v_{j,i}^+ \sum_{\varrho=1}^n \pi_{i,\varrho}^* \mathbf{h}_\varrho^{(k)*}, \\ \mathbf{k}_{\ell,j}^* &:= \theta_{j,2} \mathbf{f}_{j,1}^* - \theta_{j,1} \mathbf{f}_{j,2}^*, \end{aligned}$$

For $j = \mu_\ell + 1, \dots, n$,

For $i = 1, \dots, \mu_\ell, j$,

$$\begin{aligned} \varphi_i &:= \sum_{k=1}^3 \tilde{a}_k \sum_{\varrho=1}^n \pi_{i,\varrho}^* \tau_\varrho^{(k)}, \quad \mathbf{m}_i^* := \sum_{k=1}^3 \tilde{a}_k \sum_{\varrho=1}^n \pi_{i,\varrho}^* \mathbf{h}_\varrho^{(k)*}, \\ z_j &:= \varphi_j \left(\sum_{i=1}^{\mu_\ell} v_{j,i}^+ \varphi_i \right)^{-1}, \quad \mathbf{k}_{\ell,j}^* := \mathbf{k}_{\ell,j}^* + \mathbf{m}_j^* - z_j \sum_{i=1}^{\mu_\ell} v_{j,i}^+ \mathbf{m}_i^*, \end{aligned}$$

where $a_{0,k}, a_{j,k,s}, \tilde{a}_k \stackrel{\cup}{\leftarrow} \mathbb{F}_q$ for $j = 1, \dots, \ell + 1, \mu_\ell + 1, \dots, n; k = 1, 2, 3; s = 1, 2$.

If $\theta_0 = 0$, $\{\sigma_{0,t}, a_{0,k} \stackrel{\cup}{\leftarrow} \mathbb{F}_q\}_{k=1,2,3;t=1,\dots,\ell}$ is selected again. For $j = \mu_\ell + 1, \dots, n$, if $\sum_{i=1}^{\mu_\ell} v_{j,i}^+ \varphi_i = 0$, $\{\sigma_{j,t}, \tilde{a}_k \stackrel{\cup}{\leftarrow} \mathbb{F}_q\}_{k=1,2,3;t=1,\dots,\ell}$ is selected again.

5. When \mathcal{B}_1 (or \mathcal{C}) gets challenge plaintexts $(m^{(0)}, m^{(1)})$ (from \mathcal{A}), \mathcal{B}_1 calculates and returns (c_1, c_2) s.t. $c_1 := \mathbf{e}_\beta + \zeta \mathbf{d}_{n+1}$ and $c_2 := \zeta \mathbf{g}_T m^{(b)}$ using \mathbf{e}_β in the RDSP instance, ζ , and $m^{(b)}$, where $\zeta \stackrel{\cup}{\leftarrow} \mathbb{F}_q$.
6. After the encryption query, GenKey oracle simulation for a reveal key query is executed as above.
7. \mathcal{A} outputs bit b' . If $b = b'$, \mathcal{B}_1 outputs $\beta' := 1$. Otherwise, \mathcal{B}_1 outputs $\beta' := 0$.

To prove Lemma 2, we show Claims 2, 3, and 4.

Claim 2. *Public parameter pk generated in step 3 above has the same distribution as that in Game 2 (and Game 3).*

Proof. Let $D := \begin{pmatrix} \Pi & 0 \\ 0 & I_3 \end{pmatrix}$ be square $(n + 3) \times (n + 3)$ matrix composed of Π and the identity matrix I_3 . Then basis $(\tilde{\mathbf{b}}_1, \dots, \tilde{\mathbf{b}}_n, \mathbf{b}_{n+1}, \mathbf{b}_{n+2}, \mathbf{b}_{n+3})$ of \mathbb{V} is obtained from basis \mathbb{B} by the linear transformation determined by D . Hence, its distribution is uniform. Therefore, $\tilde{\mathbb{B}} = (\tilde{\mathbf{b}}_1, \dots, \tilde{\mathbf{b}}_n, \mathbf{d}_{n+1}, \mathbf{b}_{n+3})$ in step 3 has the same distribution as that in Game 2 (and Game 3). \square

Claim 3. *Secret key $\vec{\mathbf{k}}_\ell^*$ generated in steps 4 and 6 above has the same distribution as that in Game 2 (and Game 3).*

Proof. First, we verify that basis $(\tilde{\mathbf{b}}_1^*, \dots, \tilde{\mathbf{b}}_n^*, \mathbf{b}_{n+1}^*, \mathbf{b}_{n+2}^*, \mathbf{b}_{n+3}^*)$ of \mathbb{V}^* is obtained by the linear transformation $(D^T)^{-1}$, where D is defined in the proof of Claim 2. That is, it is dual orthonormal to basis $(\tilde{\mathbf{b}}_1, \dots, \tilde{\mathbf{b}}_n, \mathbf{b}_{n+1}, \mathbf{b}_{n+2}, \mathbf{b}_{n+3})$. Therefore, we can consider $\mathbf{k}_{\ell,j}^*$ w.r.t. this dual orthonormal basis.

Secret key $\mathbf{k}_{\ell,0}^*$ generated in steps 4 and 6 is $\theta_0^{-1} (\sum_{k=1}^3 a_{0,k} \omega^{(k)}) \sum_{i=1}^{\mu_\ell} v_{0,i}^+ \tilde{\mathbf{b}}_i^* + \theta_0^{-1} \theta_1 \mathbf{b}_{n+1}^* + \theta_0^{-1} \theta_2 \mathbf{b}_{n+2}^*$, where $\theta_1 := (\sum_{k=1}^3 a_{0,k} \gamma_1^{(k)}) \vec{v}_0^+ \cdot \vec{x}^+$, $\theta_2 := (\sum_{k=1}^3 a_{0,k} \gamma_2^{(k)}) \vec{v}_0^+ \cdot \vec{x}^+$, and $\theta_0 = \theta_1 + \theta_2$. Let $\sigma := \theta_0^{-1} (\sum_{k=1}^3 a_{0,k} \omega^{(k)})$. Then, $\sigma, \theta_1, \theta_2$ are independently uniform, since $a_{0,k}$ are independently uniform, and $\theta_0^{-1} \theta_1 + \theta_0^{-1} \theta_2 = 1$. Also, from (II), the coefficients of $\sum_{i=\mu_{t-1}+1}^{\mu_t} v_i \tilde{\mathbf{b}}_i^*$ in $\mathbf{k}_{\ell,0}^*$ for each $1 \leq t \leq \ell$ are all uniformly and independently distributed. Therefore, generated $\mathbf{k}_{\ell,0}^*$ has the same distribution as in Game 2 and Game 3.

Similarly, for $j = 1, \dots, \ell + 1, \mu_\ell + 1, \dots, n$, the j -th key $\mathbf{k}_{\ell,j}^*$ has independently uniform coefficients w.r.t. $\sum_{i=\mu_{t-1}+1}^{\mu_t} v_i \tilde{\mathbf{b}}_i^*$ for each $1 \leq t \leq \ell$, and the sum of the coefficients of $\tilde{\mathbf{b}}_{n+1}^*$ and $\tilde{\mathbf{b}}_{n+2}^*$ is zero.

Finally, we investigate the distribution of the coefficients of $\tilde{\mathbf{b}}_j^*$ in $\mathbf{k}_{\ell,j}^*$ for $j = \mu_\ell + 1, \dots, n$. The additional term $\mathbf{m}_j^* - z_j \sum_{i=1}^{\mu_\ell} v_{j,i}^+ \mathbf{m}_i^*$ is

$$\begin{aligned}
 & -z_j \left(\sum_{k=1}^3 \tilde{a}_k \omega^{(k)} \right) \sum_{i=1}^{\mu_\ell} v_{j,i}^+ \tilde{\mathbf{b}}_i^* + \left(\sum_{k=1}^3 \tilde{a}_k \omega^{(k)} \right) \tilde{\mathbf{b}}_j^* \\
 & + (\varphi_{1,j} - z_j \sum_{i=1}^{\mu_\ell} v_{j,i}^+ \varphi_{1,i}) \mathbf{b}_{n+1}^* + (\varphi_{2,j} - z_j \sum_{i=1}^{\mu_\ell} v_{j,i}^+ \varphi_{2,i}) \mathbf{b}_{n+2}^*, \quad (2)
 \end{aligned}$$

where $\varphi_{1,i} := (\sum_{k=1}^3 \tilde{a}_k \gamma_1^{(k)}) x_i^+$, $\varphi_{2,i} := (\sum_{k=1}^3 \tilde{a}_k \gamma_2^{(k)}) x_i^+$ and $\varphi_i = \varphi_{1,i} + \varphi_{2,i}$. Therefore, for $j = \mu_\ell + 1, \dots, n$, the sum of the coefficients of \mathbf{b}_{n+1}^* and \mathbf{b}_{n+2}^* in (2) is zero, and the coefficients of $\tilde{\mathbf{b}}_j^*$ in $\mathbf{k}_{\ell,j}^*$ are common, $\sum_{k=1}^3 \tilde{a}_k \omega^{(k)}$, which is uniformly distributed. \square

Claim 4. *If $\beta = 0$, the distribution of $(\mathbf{c}_1, \mathbf{c}_2)$ generated in step 5 is the same as that in Game 2. If $\beta = 1$, the distribution of $(\mathbf{c}_1, \mathbf{c}_2)$ generated in step 5 is the same as that in Game 3.*

Proof. If $\beta = 0$, $\mathbf{c}_1 = \delta_1 \sum_{i=1}^n y_i \mathbf{b}_i + \zeta \mathbf{d}_{n+1} + \delta_2 \mathbf{b}_{n+3} = \delta_1 \sum_{i=1}^n x_i^+ \tilde{\mathbf{b}}_i + \zeta \mathbf{d}_{n+1} + \delta_2 \mathbf{b}_{n+3}$ and $\mathbf{c}_2 := \gamma_T^{\zeta} m^{(b)}$. This is the target ciphertext in Game 2 with $\text{pk} :=$

$(1^\lambda, \text{param}, \widetilde{\mathbb{B}})$. If $\beta = 1$, $\mathbf{c}_1 = \delta_1 \sum_{i=1}^n x_i^+ \widetilde{\mathbf{b}}_i + (\zeta + \zeta_1)\mathbf{b}_{n+1} + (\zeta + \zeta_2)\mathbf{b}_{n+2} + \delta_2\mathbf{b}_{n+3}$ and $\mathbf{c}_2 := g_T^\zeta m^{(b)}$. Because $\zeta + \zeta_1$, $\zeta + \zeta_2$, and ζ are independently uniform, this is the target ciphertext in Game 3 with $\text{pk} := (1^\lambda, \text{param}, \widetilde{\mathbb{B}})$. \square

From Claims 2, 3, and 4, when $\beta = 0$, the advantage of \mathcal{A} in the above game is equal to that in Game 2, i.e., $\text{Adv}_{\mathcal{A}}^{(2)}(\lambda)$, and also is equal to $\text{Pr}_0 := \Pr \left[\mathcal{B}_1(1^\lambda, \rho) \rightarrow 1 \mid \rho \xleftarrow{\mathbb{R}} \mathcal{G}_0^{\text{RDSP}}(1^\lambda, n) \right]$. Similarly, when $\beta = 1$, we see that the advantage of \mathcal{A} in the above game is equal to $\text{Adv}_{\mathcal{A}}^{(3)}(\lambda)$, and also is equal to $\text{Pr}_1 := \Pr \left[\mathcal{B}_1(1^\lambda, \rho) \rightarrow 1 \mid \rho \xleftarrow{\mathbb{R}} \mathcal{G}_1^{\text{RDSP}}(1^\lambda, n) \right]$. Therefore, $|\text{Adv}_{\mathcal{A}}^{(2)}(\lambda) - \text{Adv}_{\mathcal{A}}^{(3)}(\lambda)| = |\text{Pr}_0 - \text{Pr}_1| = \text{Adv}_{\mathcal{B}_1}^{\text{RDSP}}(\lambda)$. This completes the proof of Lemma 2. \square

Lemma 3. *For any adversary \mathcal{A} , there exists a probabilistic machine \mathcal{B}_2 , whose running time is essentially the same as that of \mathcal{A} , such that for any security parameter λ , $|\text{Adv}_{\mathcal{A}}^{(3)}(\lambda) - \text{Adv}_{\mathcal{A}}^{(4)}(\lambda)| = \text{Adv}_{\mathcal{B}_2}^{\text{IDSP}}(\lambda)$.*

Proof. Lemma 3 is similarly proved as Lemma 2. The proof will be given in the full version of this paper. \square

Lemma 4. *For any adversary \mathcal{A} , $\text{Adv}_{\mathcal{A}}^{(4)}(\lambda) = 0$.*

Proof. The value of b is independent from the adversary's view in Game 4. Hence, $\text{Adv}_{\mathcal{A}}^{(4)}(\lambda) = 0$. \square

References

1. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: 2007 IEEE Symposium on Security and Privacy, pp. 321–334. IEEE Press, Los Alamitos (2007)
2. 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)
3. Boneh, D., Boyen, X.: Secure identity based encryption without random oracles. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 443–459. Springer, Heidelberg (2004)
4. Boneh, D., Boyen, X., Goh, E.: Hierarchical identity based encryption with constant size ciphertext. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 440–456. Springer, Heidelberg (2005)
5. 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)
6. Boneh, D., Hamburg, M.: Generalized identity based and broadcast encryption scheme. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 455–470. Springer, Heidelberg (2008)
7. 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)

8. 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)
9. Cocks, C.: An identity based encryption scheme based on quadratic residues. In: Honary, B. (ed.) Cryptography and Coding 2001. LNCS, vol. 2260, pp. 360–363. Springer, Heidelberg (2001)
10. Gentry, C.: Practical identity-based encryption without random oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 445–464. Springer, Heidelberg (2006)
11. Gentry, C., Halevi, S.: Hierarchical identity-based encryption with polynomially many levels. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 437–456. Springer, Heidelberg (2009)
12. Gentry, C., Silverberg, A.: Hierarchical ID-based cryptography. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 548–566. Springer, Heidelberg (2002)
13. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: ACM Conference on Computer and Communication Security 2006, pp. 89–98. ACM, New York (2006)
14. 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)
15. Horwitz, J., Lynn, B.: Towards hierarchical identity-based encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 466–481. Springer, Heidelberg (2002)
16. 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)
17. Okamoto, T., Takashima, K.: Homomorphic encryption and signatures from vector decomposition. In: Galbraith, S.D., Paterson, K.G. (eds.) Pairing 2008. LNCS, vol. 5209, pp. 57–74. Springer, Heidelberg (2008)
18. Okamoto, T., Takashima, K.: A geometric approach on pairings and hierarchical predicate encryption. In: Poster session, EUROCRYPT 2009 (2009)
19. Ostrovsky, R., Sahai, A., Waters, B.: Attribute-based encryption with non-monotonic access structures. In: ACM Conference on Computer and Communication Security 2007, pp. 195–203. ACM, New York (2007)
20. Pirretti, M., Traynor, P., McDaniel, P., Waters, B.: Secure attribute-based systems. In: ACM Conference on Computer and Communication Security 2006, pp. 99–112. ACM, New York (2006)
21. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)
22. Shi, E., Waters, B.: Delegating capability in predicate encryption systems. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfssdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 560–578. Springer, Heidelberg (2008)
23. Takashima, K.: Efficiently computable distortion maps for supersingular curves. In: van der Poorten, A.J., Stein, A. (eds.) ANTS-VIII 2008. LNCS, vol. 5011, pp. 88–101. Springer, Heidelberg (2008)
24. Waters, B.: Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization. ePrint, IACR,
<http://eprint.iacr.org/2008/290>

Hedged Public-Key Encryption: How to Protect against Bad Randomness

Mihir Bellare¹, Zvika Brakerski², Moni Naor², Thomas Ristenpart¹,
Gil Segev², Hovav Shacham¹, and Scott Yilek¹

¹ Dept. of Computer Science & Engineering, University of California at San Diego,
9500 Gilman Drive, La Jolla, CA 92093, USA

{mihir, tristenp, hovav, syilek}@cs.ucsd.edu

² Dept. of Computer Science and Applied Mathematics,
Weizmann Institute of Science, Rehovot 76100, Israel

{zvika.brakerski, moni.naor, gil.segev}@weizmann.ac.il

Abstract. Public-key encryption schemes rely for their IND-CPA security on per-message fresh randomness. In practice, randomness may be of poor quality for a variety of reasons, leading to failure of the schemes. Expecting the systems to improve is unrealistic. What we show in this paper is that we can, instead, improve the cryptography to offset the lack of possible randomness. We provide public-key encryption schemes that achieve IND-CPA security when the randomness they use is of high quality, but, when the latter is not the case, rather than breaking completely, they achieve a weaker but still useful notion of security that we call IND-CDA. This hedged public-key encryption provides the best possible security guarantees in the face of bad randomness. We provide simple RO-based ways to make in-practice IND-CPA schemes hedge secure with minimal software changes. We also provide non-RO model schemes relying on lossy trapdoor functions (LTDFs) and techniques from deterministic encryption. They achieve adaptive security by establishing and exploiting the anonymity of LTDFs which we believe is of independent interest.

1 Introduction

Cryptography ubiquitously assumes that parties have access to sufficiently good randomness. In practice this assumption is often violated. This can happen because of faulty implementations, side-channel attacks, system resets or for a variety of other reasons. The resulting cryptographic failures can be spectacular [22, 24, 29, 2, 15]. What can we do about this? One answer is that system designers should build “better” systems, but this is clearly easier said than done. The reality is that random number generation is a complex and difficult task, and it is unrealistic to think that failures will never occur. We propose a different approach: designing schemes in such a way that poor randomness will have as little as possible impact on the security of the scheme in the following sense. With good randomness the scheme achieves whatever (strong) security notion

one is targeting, but when the same scheme is fed bad (even adversarially chosen) randomness, rather than breaking completely, it achieves some weaker but still useful notion of security that is the best possible under the circumstances. We call this “hedged” cryptography.

Previous work by Rogaway [32], Rogaway and Shrimpton [33], and Kamara and Katz [27] considers various forms of hedging for the symmetric encryption setting. In this paper, we initiate a study of hedged public-key encryption. We address two central foundational questions, namely to find appropriate definitions and to efficiently achieve them. Let us now look at all this in more detail.

THE PROBLEM. Achieving the standard IND-CPA notion of privacy [23] *requires* the encryption algorithm to be randomized. In addition to the public key and message, it takes as input a random string that needs to be freshly and independently created for *each and every* encryption.

Weak (meaning, low-entropy) randomness does not merely imply a loss of theoretical security. It can lead to catastrophic attacks. For example, weak-randomness based encryption is easily seen to allow recovery of the plaintext from the ciphertext for the quadratic residuosity scheme of [23] as well as the El Gamal encryption scheme [21]. Brown [15] presents such an attack on RSA-OAEP [10] with encryption exponent 3. Ouafi and Vaudenay [30] present such an attack on Rabin-SAEP [13]. We present an alternative attack in [7].

The above would be of little concern if we could guarantee good randomness. Unfortunately, this fails to be true in practice. Here, an “entropy-gathering” process is used to get a seed which is then stretched to get “random” bits for the application. The theory of cryptographically strong pseudorandom number generators [11] implies that the stretching can in principle be sound, and extractors further allow us to reduce the requirement on the seed from being uniformly distributed to having high min-entropy, but we still need a sufficiently good seed. (No amount of cryptography can create randomness out of nothing!) In practice, entropy might be gathered from timing-related operating system events or user keystrokes. As evidence that this process is error-prone, consider the recent randomness failure in Debian Linux, where a bug in the OpenSSL package led to insufficient entropy gathering and thence to practical attacks on the SSH [29] and SSL [2,36] protocols. Other exploits include [25,19].

THE NEW NOTION. The idea is to provide two tiers of security. First, when the “randomness” is really random, the scheme should meet the standard IND-CPA notion of security. Otherwise, rather than failing completely, it should gracefully achieve some weaker but as-good-as-possible notion of security. The first important question we then face is to pick and formally define this fallback notion.

Towards this, we begin by suggesting that the *message* being encrypted may also have entropy or uncertainty from the point of view of the adversary. (If not, what privacy is there to be preserved by encryption?) We propose to harvest this. In this regard, the first requirement that might come to mind is that encryption with weak (even adversarially-known) randomness should be as secure as deterministic encryption, meaning achieve an analog of the PRIV notion of [6]. But

achieving this would require that the message by itself have high min-entropy. We can do better. Our new target notion of security, that we call Indistinguishability under a Chosen Distribution Attack (IND-CDA), asks that security is guaranteed as long as the *joint* distribution of the message and randomness has sufficiently high min-entropy. In this way, we can exploit for security whatever entropy might be present in the randomness *or* the message, and in particular achieve security even if neither taken alone is random enough.

Notice that if the message and randomness together have low min-entropy, then we cannot hope to achieve security, because an adversary can recover the message with high probability by trial encryption with all message-randomness pairs that occur with a noticeable probability. In a nutshell, our new notion asks that this necessary condition is also sufficient, and in this way is requiring security that is as good as possible.

We denote by H-IND our notion of *hedged* security that is satisfied by encryption schemes that are secure both in the sense of IND-CPA and in the sense of IND-CDA.

ADAPTIVITY. Our IND-CDA definition generalizes the indistinguishability-style formalizations of PRIV-secure deterministic encryption [8][12], which in turn extended entropic security [18]. But we consider a new dimension, namely, adaptivity. Our adversary is allowed to specify joint message-randomness distributions on to-be-encrypted challenges. The adversary is said to be adaptive if these queries depend on the replies to previous ones. Non-adaptive H-IND means IND-CPA plus non-adaptive IND-CDA and adaptive H-IND means IND-CPA plus adaptive IND-CDA.

Non-adaptive IND-CDA is a notion of security for randomized schemes that becomes identical to PRIV in the special case that the scheme is deterministic. Adaptive IND-CDA, when restricted to deterministic schemes, is an adaptive strengthening of PRIV that we think is interesting in its own right. As a consequence of the results discussed below, we get the first deterministic encryption schemes that achieve this stronger notion.

SCHEMES WITH RANDOM ORACLES. Our random oracle (RO) model schemes and their attributes are summarized in the first two rows of the table of Figure 1. Both REwH1 and REwH2 efficiently transform an arbitrary (randomized) IND-CPA scheme into a H-IND scheme with the aid of the RO. They are simple ways to make in-practice encryption schemes H-IND secure with minimal software changes. REwH1 has the advantage of not changing the public key and thus not requiring new certificates. It always provides non-adaptive H-IND security. It provides adaptive H-IND security if the starting scheme has the extra property of being anonymous in the sense of [4]. Anonymity is possessed by some deployed schemes like DHIES [1], making REwH1 attractive in this case. But some in-practice schemes, notably RSA ones, are not anonymous. If one wants adaptive H-IND security in this case we suggest REwH2, which provides it assuming only that the starting scheme is IND-CPA. It does this by adding a randomizer to

	Non-adaptive H-IND	Adaptive H-IND
REwH1	IND-CPA	IND-CPA + ANON-CPA
REwH2	IND-CPA	IND-CPA
RtD	IND-CPA, PRIV	IND-CPA, (u-)LTDF
PtD	(u-)LTDF	(u-)LTDF

Fig. 1. Table entries for the first two rows indicate the assumptions made on the (randomized) encryption scheme that underlies the RO-model hedged schemes in question. The entries for standard model scheme RtD are the assumptions on the underlying randomized and deterministic encryption schemes, respectively, and for PtD, on the underlying deterministic encryption scheme, which is the only primitive it uses.

the public key, so it does require new certificates. The schemes are extensions of the EwH deterministic encryption scheme of [6] and similar to [20].

SCHEMES WITHOUT RANDOM ORACLES. It is easy to see that even the existence of a non-adaptively secure IND-CDA encryption scheme implies the existence of a PRIV-secure deterministic encryption (DE) scheme. Achieving PRIV without ROs is already hard. Indeed, fully PRIV-secure DE without ROs has not yet been built. Prior work, however, does show how to construct PRIV-secure DE without ROs for block sources [12]. (Messages being encrypted have high min-entropy even conditioned on previous messages.) But H-IND introduces three additional challenges: (1) the min-entropy guarantee is on the joint message-randomness distribution rather than merely on the message; (2) we want a single scheme that is not only IND-CDA secure but also IND-CPA-secure; and (3) the adversary’s queries may be adaptive.

We are able to overcome these challenges to the best extent possible. We provide schemes that are H-IND-secure in the same setting as the best known PRIV ones, namely, for block sources, where we suitably extend the latter notion to consider both randomness and messages. Furthermore, we achieve these results under the same assumptions as previous work.

Our standard model schemes and their attributes are summarized in the last two rows of the table of Figure 1. RtD is formed by the generic composition of a deterministic scheme and a randomized scheme and achieves non-adaptive H-IND security as long as the base schemes meet their regular conditions. (That is, the former is PRIV-secure for block sources and the latter is IND-CPA.) Adaptive security requires that the deterministic scheme be a u-LTDF. (A lossy trapdoor function whose lossy branch is a universal hash function [31][12].) PtD is simpler, merely concatenating the message to the randomness and then applying deterministic encryption. It achieves both non-adaptive and adaptive H-IND under the assumption that the deterministic scheme is a u-LTDF. For both schemes, the universality assumption on the LTDF can be dropped by modifying the scheme and using the crooked leftover hash lemma as per [12]. (This is why the “u” is parenthesized in the table of Figure 1.)

ANONYMOUS LTDFs. Also of independent interest, we show that any u-LTDF is anonymous. Here we refer to a new notion of anonymity for trapdoor functions that we introduce, one that strengthens the notion of [4]. This step exploits an adaptive variant of the leftover hash lemma of [26].

Why anonymity? It is exploited in our proofs of adaptive security. Our new notion of anonymity for trapdoor functions is matched by a corresponding one for encryption schemes. We show that any encryption scheme that is both anonymous and non-adaptive H-IND secure is also adaptively H-IND secure. Anonymity of the u-LTDF, in our encryption schemes based on the latter primitive, allows us to show that these schemes are anonymous and thereby lift their non-adaptive security to adaptive.

RELATED WORK. In the symmetric setting, several works have recognized and addressed the problem of security in the face of bad randomness. Concern over the quality of available randomness is one of Rogaway’s motivations for introducing nonce-based symmetric encryption [32], where security relies on the nonce never repeating rather than being random. Rogaway and Shrimpton [33] provide a symmetric authenticated encryption scheme that defaults to a PRF when the randomness is known.

Kamara and Katz [27] provide symmetric encryption schemes secure against chosen-randomness attack (CRA). Here the adversary can obtain encryption under randomness of its choice but privacy is only required for messages encrypted with perfect, hidden randomness. Entropy in the messages is not considered or used. We in contrast seek privacy even when the randomness is bad as long as there is compensating entropy in the message. Also we deal with the public key setting.

Many works consider achieving strong cryptography given only a “weak random source” [28,16,14]. This is a source that does have high min-entropy but may not produce truly random bits. They show that many cryptographic tasks including symmetric encryption [28], commitment, secret-sharing, and zero knowledge [16] are impossible in this setting. We are not in this setting. We do assume a small amount of initial good randomness to produce keys. (This makes sense because it is one-time and because otherwise we can’t hope to achieve anything anyway.) On the other hand our assumption on the randomness available for encryption is even weaker than in the works mentioned. (We do not even assume it has high min-entropy.) Our key idea is to exploit the entropy in the message, which is not done in [28,16,14]. This allows us to circumvent their negative results.

Waters independently proposed hedge security as well as the PtD construction as a way to achieve it [35].

2 Preliminaries

NOTATION. Vectors are written in boldface, e.g. \mathbf{x} . If \mathbf{x} is a vector then $|\mathbf{x}|$ denotes its length and $\mathbf{x}[i]$ denotes its i^{th} component for $1 \leq i \leq |\mathbf{x}|$. We say that \mathbf{x} is a vector over D if $\mathbf{x}[i] \in D$ for all $1 \leq i \leq |\mathbf{x}|$. Throughout, $k \in \mathbb{N}$ denotes the

security parameter and 1^k its unary encoding. Unless otherwise indicated, an algorithm is randomized. The set of possible outputs of algorithm A on inputs x_1, x_2, \dots is denoted $[A(x_1, x_2, \dots)]$. “PT” stands for polynomial-time.

GAMES. Our security definitions and proofs use code-based games [9], and so we recall some background from [9]. A game (look at Figure 2 for examples) has an **Initialize** procedure, procedures to respond to adversary oracle queries, and a **Finalize** procedure. A game G is executed with an adversary A as follows. First, **Initialize** executes, and its outputs are the inputs to A . Then A executes, its oracle queries being answered by the corresponding procedures of G . When A terminates, its output becomes the input to the **Finalize** procedure. The output of the latter is called the output of the game, and we let $G^A \Rightarrow y$ denote the event that this game output takes value y . Our convention is that the running time of an adversary is the time to execute the adversary with the game that defines security, so that the running time of all game procedures is included.

PUBLIC-KEY ENCRYPTION. A public-key encryption (PKE) scheme is a tuple of PT algorithms $\mathcal{AE} = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ with associated message length parameter $n(\cdot)$ and randomness length parameter $\rho(\cdot)$. The parameter generation algorithm \mathcal{P} takes as input 1^k and outputs a parameter string par . The key generation algorithm \mathcal{K} takes input par and outputs a key pair (pk, sk) . The encryption algorithm \mathcal{E} takes inputs pk , message $m \in \{0, 1\}^{n(k)}$ and coins $r \in \{0, 1\}^{\rho(k)}$ and returns the ciphertext denoted $\mathcal{E}(pk, m; r)$. The deterministic decryption algorithm \mathcal{D} takes input sk and ciphertext c and outputs either \perp or a message in $\{0, 1\}^{n(k)}$. For vectors \mathbf{m}, \mathbf{r} with $|\mathbf{m}| = |\mathbf{r}| = v$ we denote by $\mathcal{E}(pk, \mathbf{m}; \mathbf{r})$ the vector $(\mathcal{E}(pk, \mathbf{m}[1]; \mathbf{r}[1]), \dots, \mathcal{E}(pk, \mathbf{m}[v]; \mathbf{r}[v]))$. We say that \mathcal{AE} is deterministic if \mathcal{E} is deterministic. (That is, $\rho(\cdot) = 0$.)

We consider the standard IND-CPA notion of security, captured by the game $\text{IND}_{\mathcal{AE}}$ where $\mathcal{AE} = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ is an encryption scheme. In the game, **Initialize** chooses a random bit b , generates parameters $par \leftarrow \mathcal{P}(1^k)$ and generates a key pair $(pk, sk) \leftarrow \mathcal{K}(par)$ before returning pk to the adversary. Procedure **LR**, on input messages m_0 and m_1 , returns $c \leftarrow \mathcal{E}(pk, m_b)$. Lastly, procedure **Finalize** takes as input a guess bit b' and outputs true if $b = b'$ and false otherwise. An IND-CPA adversary makes a single query (m_0, m_1) to **LR** with $|m_0| = |m_1|$. For IND-CPA adversary A we let $\text{Adv}_{\mathcal{AE}, A}^{\text{ind-cpa}}(k) = 2 \cdot \Pr[\text{IND}_{\mathcal{AE}, k}^A \Rightarrow \text{true}] - 1$. We say \mathcal{AE} is IND-CPA secure if $\text{Adv}_{\mathcal{AE}, A}^{\text{ind-cpa}}(\cdot)$ is negligible for all PT IND-CPA adversaries A .

SOURCES. We generalize the notion of a source to consider a joint distribution on the messages and the randomness with which they will be encrypted. A t -source ($t \geq 1$) with message length $n(\cdot)$ and randomness length $\rho(\cdot)$ is a probabilistic algorithm \mathcal{M} that on input 1^k returns a $(t+1)$ -tuple $(\mathbf{m}_0, \dots, \mathbf{m}_{t-1}, \mathbf{r})$ of equal-length vectors, where $\mathbf{m}_0, \dots, \mathbf{m}_{t-1}$ are over $\{0, 1\}^{n(k)}$ and \mathbf{r} is over $\{0, 1\}^{\rho(k)}$. We say that \mathcal{M} has min-entropy $\mu(\cdot)$ if

$$\Pr[(\mathbf{m}_b[i], \mathbf{r}[i]) = (m, r)] \leq 2^{-\mu(k)}$$

for all $k \in \mathbb{N}$, all $b \in \{0 \dots, t - 1\}$, all i and all $(m, r) \in \{0, 1\}^{n(k)} \times \{0, 1\}^{\rho(k)}$. We say it has conditional min-entropy $\mu(\cdot)$ if

$$\Pr [(\mathbf{m}_b[i], \mathbf{r}[i]) = (m, r) \mid \forall j < i (\mathbf{m}_b[j], \mathbf{r}[j]) = (\mathbf{m}'[j], \mathbf{r}'[j])] \leq 2^{-\mu(k)}$$

for all $k \in \mathbb{N}$, all $b \in \{0 \dots, t - 1\}$, all i , all (m, r) , and all vectors \mathbf{m}', \mathbf{r}' . A t -source with message length $n(\cdot)$, randomness length $\rho(\cdot)$, and min-entropy $\mu(\cdot)$ is referred to as a (μ, n, ρ) -mr-source when $t = 1$ and $\rho(\cdot) > 0$; a (μ, n) -m-source when $t = 1$ and $\rho(\cdot) = 0$; a (μ, n, ρ) -mmr-source when $t = 2$ and $\rho(\cdot) > 0$; and (μ, n) -mm-source when $t = 2$ and $\rho(\cdot) = 0$. Each “m” indicates the source outputting one message vector and an “r” indicates a randomness vector. When the source has *conditional* min-entropy $\mu(\cdot)$ we write block-source instead of source for each of the above. A $v(\cdot)$ -vector source outputs vectors of size $v(k)$ for all k .

UNIVERSAL HASH FUNCTIONS. A family of functions is a tuple $\mathcal{H} = (\mathcal{P}, \mathcal{K}, F)$ with associated message length $n(\cdot)$. It is required that the domain of $F(K, \cdot)$ is $\{0, 1\}^n$ for every k , every $par \in [\mathcal{P}(1^k)]$, and every $K \in [\mathcal{K}(par)]$. We say that \mathcal{H} is universal if for every k , all $par \in [\mathcal{P}(1^k)]$, and all distinct $x_1, x_2 \in \{0, 1\}^{n(k)}$, the probability that $F(K, x_1) = F(K, x_2)$ is at most $1/|R(par)|$ where $R(par) = \{ F(K, x) : K \in [\mathcal{K}(par)] \text{ and } x \in \{0, 1\}^n \}$ and the probability is over $K \leftarrow_s \mathcal{K}(par)$.

LOSSY TRAPDOOR FUNCTIONS (LTDFs). To a deterministic PKE scheme (recall that a family of injective trapdoor functions and a deterministic encryption scheme are, syntactically, the same object) $\mathcal{AE} = (\mathcal{P}_d, \mathcal{K}_d, \mathcal{E}_d, \mathcal{D}_d)$ with message length $n_d(\cdot)$ we can associate an (n_d, ℓ) -lossy key generator \mathcal{K}_l . This is a PT algorithm that, on input par , outputs a value pk for which the map $\mathcal{E}_d(pk, \cdot)$ has image size at most $2^{n_d(k) - \ell(k)}$. The parameter ℓ is called the lossiness of the lossy key generator. We associate to \mathcal{AE} , lossy key generator \mathcal{K}_l , and a LOS adversary A the function $\mathbf{Adv}_{\mathcal{AE}, \mathcal{K}_l, A}^{\text{los}}(k) = 2 \cdot \Pr [\text{LOS}_{\mathcal{AE}, \mathcal{K}_l, k}^A \Rightarrow \text{true}] - 1$, where game $\text{LOS}_{\mathcal{AE}, \mathcal{K}_l}$ works as follows. **Initialize** chooses a random bit b and generates parameters $par \leftarrow_s \mathcal{P}_d(1^k)$, if $b = 0$ runs $(pk, sk) \leftarrow_s \mathcal{K}_d(par)$ and if $b = 1$ runs $pk \leftarrow_s \mathcal{K}_l(par)$. It then returns pk (to the adversary A). When A finishes, outputting guess b' , **Finalize** returns true if $b = b'$. We say \mathcal{K}_l is *universal-inducing* if $\mathcal{H} = (\mathcal{P}_d, \mathcal{K}_l, \mathcal{E}_d)$ is a family of universal hash functions with message length n_d .

A deterministic encryption scheme \mathcal{AE} is a (n_d, ℓ) -lossy trapdoor function (LTDF) if there exists a (n_d, ℓ) -lossy key generator such that $\mathbf{Adv}_{\mathcal{AE}, \mathcal{K}_l, A}^{\text{los}}(\cdot)$ is negligible for all PT A . We say it is a universal (n_d, ℓ) -lossy trapdoor function (u-LTDF) if in addition \mathcal{K}_l is universal-inducing.

Lossy trapdoor functions were introduced by Peikert and Waters [31], and can be based on a variety of number-theoretic assumptions, including the hardness of the decisional Diffie-Hellman problem, the worst-case hardness of lattice problems, and the hardness of Paillier’s composite residuosity problem [31, 12, 34]. Boldyreva et al. [12] observed that the DDH-based construction is universal.

<p>proc. Initialize(1^k):</p> $par \leftarrow \mathcal{P}(1^k)$ $(pk, sk) \leftarrow \mathcal{K}(par)$ $b \leftarrow \{0, 1\}$ Ret par	<p>proc. LR(\mathcal{M}):</p> If $pkout = \text{true}$ then Ret \perp $(\mathbf{m}_0, \mathbf{m}_1, \mathbf{r}) \leftarrow \mathcal{M}(1^k)$ Ret $\mathcal{E}(pk, \mathbf{m}_b; \mathbf{r})$	<p>proc. RevealPK(\cdot):</p> $pkout \leftarrow \text{true}$ Ret pk <p>proc. Finalize(b'):</p> Ret $(b = b')$
---	---	--

Fig. 2. Game $CDA_{\mathcal{A}\mathcal{E},k}$

3 Security against Chosen Distribution Attack

Let $\mathcal{A}\mathcal{E} = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme. A CDA adversary is one whose **LR** queries are all *mmr*-sources. Game $CDA_{\mathcal{A}\mathcal{E}}$ of Figure 2 provides the adversary with two oracles. The advantage of CDA adversary A is

$$\text{Adv}_{\mathcal{A}\mathcal{E},A}^{\text{cda}}(k) = 2 \cdot \Pr [CDA_{\mathcal{A}\mathcal{E},k}^A \Rightarrow \text{true}] - 1.$$

In the random oracle model we allow all algorithms in Game CDA to access the random oracle; importantly, this includes the *mmr*-sources.

DISCUSSION. Adversary A can query **LR** with an *mmr*-source of its choice, an output $(\mathbf{m}_0, \mathbf{m}_1, \mathbf{r})$ of which represents choices of message vectors to encrypt and randomness with which to encrypt them. (An alternative formulation might have CDA adversaries query two *mr*-sources, and distinguish between the encryption of samples taken from one of these. But this would mandate that schemes ensure privacy of messages *and* randomness.) This allows A to dictate a joint distribution on the messages and randomness. In this way it conservatively models even adversarially-subverted random number generators. Multiple **LR** queries are allowed. In the most general case these queries may be adaptive, meaning depend on answers to previous queries.

Given that multiple **LR** queries are allowed, one may ask why an *mmr*-source needs to produce message and randomness vectors rather than simply a single pair of messages and a single choice of randomness. The reason is that the coordinates in a vector all depend on the same coins underlying an execution of \mathcal{M} , but the coins underlying the execution of the sources in different queries are independent.

Note that **Initialize** does not return the public key pk to A . A can get it at any time by calling **RevealPK** but once it does this, **LR** will return \perp . The reason is that we inherit from deterministic encryption the unavoidable limitation that encryption cannot hide public-key related information about the plaintexts [6]. (When the randomness has low entropy, the ciphertext itself is such information.)

As we saw in the previous section, no encryption scheme is secure when both messages and randomness are predictable. Formally, this means chosen-distribution attacks are trivial when adversaries can query *mmr*-sources of low min-entropy. Our notions (below) will therefore require security only for sources that have high min-entropy or high conditional min-entropy.

EQUALITY PATTERNS. Suppose A makes a query \mathcal{M} which returns $(\mathbf{m}_0, \mathbf{m}_1, \mathbf{r}) = ((a, a), (a, a'), (r, r))$ for some $a \neq a'$ and random r . Then it can win trivially because the (two) components of the returned vector \mathbf{c} are equal if $b = 0$ and unequal otherwise. This limitation, again inherited from deterministic encryption [6], is inherent. To capture it we associate to an mmr -source \mathcal{M} an equality-pattern probability

$$\zeta(k) = \Pr [\text{eq}((\mathbf{m}_0, \mathbf{r}), (\mathbf{m}_1, \mathbf{r})) = 0 : (\mathbf{m}_0, \mathbf{m}_1, \mathbf{r}) \leftarrow_{\mathcal{M}} (1^k)]$$

where $\text{eq}((\mathbf{x}_1, \mathbf{x}_2), (\mathbf{y}_1, \mathbf{y}_2))$ is 1 if for all i, j

$$(\mathbf{x}_1[i], \mathbf{x}_2[i]) = (\mathbf{x}_1[j], \mathbf{x}_2[j]) \text{ iff } (\mathbf{y}_1[i], \mathbf{y}_2[i]) = (\mathbf{y}_1[j], \mathbf{y}_2[j]),$$

and 0 otherwise. We point out that **LR** queries that are mmr -block-sources (and not, just, mmr -sources) with high conditional min-entropy have negligible equality-pattern probability.

NOTIONS. We can assume (without loss of generality) that a CDA adversary makes a single **RevealPK** query and then no further **LR** queries. We say A is a (μ, n, ρ) -adversary if all of its **LR** queries are (μ, n, ρ) - mmr -sources. We say that a PKE scheme \mathcal{AE} with message length $n(\cdot)$ and randomness length $\rho(\cdot)$ is IND-CDA secure for (μ, n, ρ) - mmr -sources if for all PT (μ, n, ρ) adversaries A the function $\text{Adv}_{\mathcal{AE}, A}^{\text{cda}}(\cdot)$ is negligible. Scheme \mathcal{AE} is H-IND secure for (μ, n, ρ) - mmr -sources if it is IND-CPA secure and IND-CDA secure for (μ, n, ρ) - mmr -sources. We can extend these notions to mmr -block-sources by restricting to adversaries that query mmr -block-sources.

ON ADAPTIVITY. We can consider non-adaptive IND-CDA security by restricting attention in the notions above to adversaries that only make a single **LR** query. Why do we not focus solely on this (simpler) security goal? The standard IND-CPA setting (implicitly) provides security against multiple, adaptive **LR** queries. This is true because in that setting a straightforward hybrid argument shows that security against multiple adaptive **LR** queries is implied by security against a single **LR** query [5,3]. We wish to maintain the same standard of adaptive security in the IND-CDA setting. Unfortunately, in the IND-CDA setting, unlike the IND-CPA setting, adaptive security is not implied by non-adaptive security. In short this is because a CDA adversary necessarily cannot learn the public key before (or while) making **LR** queries. To see the separation, consider a PKE scheme that appends to every ciphertext the public key used. This will not affect the security of the scheme when an adversary can only make a single query. However, an adaptive CDA adversary can query an mmr -source, learn the public key, and craft a second source that uses the public key to ensure ciphertexts which leak the challenge bit.

Given this, our primary goal is the stronger notion of adaptive security. That said, non-adaptive hedge security is also relevant because in practice adaptive adversaries might be rare and (as we will see in Section 5) one can find non-adaptively-secure schemes that are more efficient and/or have proofs under weaker assumptions.

ADAPTIVE PRIV. A special case of our framework occurs when the PKE scheme \mathcal{AE} being considered has randomness length $\rho(k) = 0$ for all k (meaning also that adversaries query mm-sources, instead of mnr-sources). In this case we are considering deterministic encryption, and the IND-CDA definition and notions give a strengthening (by way of adaptivity) of the PRIV security notion from [6,8,12]. (For non-adaptive adversaries the definitions are equivalent.) For clarity we will use PRIV to refer to this special case, and let $\mathbf{Adv}_{\mathcal{AE},A}^{\text{priv}}(k) = \mathbf{Adv}_{\mathcal{AE},A}^{\text{cda}}(k)$.

RESOURCE USAGE. Recall that by our convention, the running time of a CDA adversary is the time for the execution of the adversary with game $\text{CDA}_{\mathcal{AE},k}$. Thus, A being PT implies that the mnr-sources that comprise A 's LR queries are also PT. This is a distinction from [12] which will be important in our results. Note that in practice we do not expect to see sources that are not PT, so our definition is not restrictive. Non-PT sources were needed in [12] for showing that single-message security implied (non-adaptive) multi-message security for deterministic encryption of block sources.

4 Constructions

Here we present several constructions for hedged encryption. The first scheme uses a random oracle and an IND-CPA secure probabilistic encryption scheme. The next two schemes derive from composing a randomized encryption scheme with a deterministic one (there are two ways of ordering composition). Interestingly, only one ordering will end up providing security. The final scheme converts a deterministic encryption scheme to a hedged one by padding the message with random bits. For the following, let $\mathcal{AE}_r = (\mathcal{P}_r, \mathcal{K}_r, \mathcal{E}_r, \mathcal{D}_r)$ be a (randomized) PKE scheme with message length $n_r(\cdot)$ and randomness length $\rho(\cdot)$. Let $\mathcal{AE}_d = (\mathcal{P}_d, \mathcal{K}_d, \mathcal{E}_d, \mathcal{D}_d)$ be a (deterministic) PKE scheme with message length $n_d(\cdot)$ and randomness length always 0. Associate to \mathcal{AE}_c for $c \in \{d, r\}$ the function $\text{maxclen}_c(k)$ mapping any k to the maximum length (over all possible public keys, messages, and if applicable, randomness) of a ciphertext output by \mathcal{E}_c .

RANDOMIZED-ENCRYPT-WITH-HASH. Let $\mathcal{R} : \{0, 1\}^* \rightarrow \{0, 1\}^*$ be a random oracle. Let $\text{REwH}[\mathcal{AE}_r] = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ be the scheme parameterized by randomizer length κ that works as follows. Parameter generation, and decryption are the same as in \mathcal{AE}_r . Key generation runs $\mathcal{K}_r(\text{par}_r)$ to get (pk_r, sk_r) , chooses $K \leftarrow_{\$} \{0, 1\}^{\kappa(k)}$, and lets $pk = (pk_r \parallel K)$ and $sk = sk_r$. Algorithm $\mathcal{E}^{\mathcal{R}}$, on input (pk, m) where $pk = (pk_r \parallel K)$, chooses $r \leftarrow_{\$} \{0, 1\}^{\rho(k)}$ and computes $r' \leftarrow \mathcal{R}(pk_r \parallel K \parallel r \parallel m)$ (where here we take \mathcal{R} 's output to be of length $\rho(k)$) and outputs $\mathcal{E}_r(pk_r, m; r')$. Intuitively, the random oracle provides perfect and (as long as m and r are hard to predict) private randomness. When the key length $\kappa(k) = 0$ for all k , we refer to the scheme as REwH1, while when $\kappa(k) > 0$ for all k we refer to the scheme as REwH2. The scheme extends the Encrypt-with-Hash deterministic encryption scheme from [6], which is a special case of REwH1 when r has length 0, and is also reminiscent of constructions in the symmetric setting that utilize a PRF to ensure good randomness [27,33], as well as schemes using the Fujisaki-Okamoto transform [20].

DETERMINISTIC-THEN-RANDOMIZED. Our first standard model attempt is to perform hedged encryption via first applying deterministic encryption and then randomized. More formally let $\text{DtR}[\mathcal{AE}_r, \mathcal{AE}_d] = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ be the scheme that works as follows. The parameter generation algorithm \mathcal{P} runs $par_r \leftarrow \mathcal{P}_r(1^k)$ and $par_d \leftarrow \mathcal{P}_d(1^k)$ and outputs $par = (par_r, par_d)$. Key generation \mathcal{K} just runs $(pk_r, sk_r) \leftarrow \mathcal{K}_r(par_r)$ and $(pk_d, sk_d) \leftarrow \mathcal{K}_d(par_d)$ and outputs $pk = (pk_r, pk_d)$ and $sk = (sk_r, sk_d)$. We define encryption by

$$\mathcal{E}((pk_r, pk_d), m ; r) = \mathcal{E}_r(pk_r, c \parallel 10^\ell ; r) ,$$

where $c = \mathcal{E}_d(pk_d, m)$ and $\ell = n_r - |c| - 1$. Here we need that $n_r(k) > \max\text{clen}_d(k)$ for all k . Decryption is defined in the natural way. The scheme will clearly inherit IND-CPA security from the application of \mathcal{E}_r . If the deterministic encryption scheme is PRIV secure for min-entropy μ , then the composition will also be secure if the *message* has min-entropy at least μ . However, our strong notion of IND-CDA security requires that schemes be secure if the *joint* distribution on the message and randomness has high min-entropy. If the entropy is unfortuitously split between both the randomness and the message, then there is no guarantee that the composition will be secure. In fact, many choices for instantiating \mathcal{AE}_r and \mathcal{AE}_d lead to a composition for which attacks can be exhibited (even when the schemes are, separately, secure).

RANDOMIZED-THEN-DETERMINISTIC. We can instead apply randomized encryption first, and then apply deterministic encryption. Define $\text{RtD}[\mathcal{AE}_r, \mathcal{AE}_d] = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ to work as follows. The parameter and key generation algorithms are as for scheme DtR. Encryption is defined by

$$\mathcal{E}((pk_r, pk_d), m ; r) = \mathcal{E}_d(pk_d, c \parallel 10^\ell) .$$

where $c = \mathcal{E}_r(pk_r, m ; r)$ and $\ell = n_d - |c| - 1$. Here we need that $n_d(k) > \max\text{clen}_r(k)$ for all k . The decryption algorithm \mathcal{D} works in the natural way. As we will see, this construction avoids the security issues of the previous, as long as the randomized encryption scheme preserves the min-entropy of its inputs. (For example, if for all k , all $par_r \in [\mathcal{P}_r(1^k)]$, and all $(pk_r, sk_r) \in [\mathcal{K}_r(par_r)]$, $\mathcal{E}_r(pk_r, \cdot)$ is injective in (m, r) .) Many encryption schemes have this property; El Gamal [21] is one example.

PAD-THEN-DETERMINISTIC. Our final construction dispenses entirely with the need for a dedicated randomized encryption scheme, instead using simple padding to directly construct a (randomized) encryption scheme from a deterministic one. Let $\text{PtD}[\mathcal{AE}_d] = (\mathcal{P}_d, \mathcal{K}_d, \mathcal{E}, \mathcal{D})$ work as follows. Parameter and key generation are inherited from the underlying (deterministic) encryption scheme. Encryption is defined by

$$\mathcal{E}(pk_d, m ; r) = \mathcal{E}_d(pk_d, r \parallel m) .$$

Decryption proceeds by applying \mathcal{D}_d , to retrieve $r \parallel m$, and then returning m .

5 Non-adaptive Hedge Security

In this section we investigate the non-adaptive hedge security of REwH, RtD and PtD, leaving adaptive security to future sections.

RANDOMIZED-ENCRYPT-WITH-HASH. Intuitively, the security of $\text{REwH}[\mathcal{AE}_r]$ follows from the IND-CPA security of \mathcal{AE}_r and the random oracle providing “perfect” randomness. Following [6], for any k let $\text{maxpk}_{\mathcal{AE}}(k)$ be the maximum of $\Pr[pk = w : (pk, sk) \leftarrow_s \mathcal{K}(par)]$, where the maximum is taken over all $w \in \{0, 1\}^*$ and all $par \in [\mathcal{P}(1^k)]$.

Theorem 1. [REwH is non-adaptive H-IND secure]. *Let $\mathcal{AE}_r = (\mathcal{P}_r, \mathcal{K}_r, \mathcal{E}_r, \mathcal{D}_r)$ be a PKE scheme with message length $n(\cdot)$ and randomness length ρ and let $\mathcal{AE} = \text{REwH}[\mathcal{AE}_r] = (\mathcal{P}_r, \mathcal{K}_r, \mathcal{E}, \mathcal{D}_r)$ be the PKE scheme constructed from it.*

- (IND-CPA) *Let A be an IND-CPA adversary. Then there exists an IND-CPA adversary B such that for all k*

$$\text{Adv}_{\mathcal{AE}, A}^{\text{ind-cpa}}(k) = \text{Adv}_{\mathcal{AE}_r, B}^{\text{ind-cpa}}(k)$$

where B runs in time that of A and makes the same number of queries.

- (IND-CDA) *Let A be an adversary that makes a single **LR** query consisting of a $v(\cdot)$ -vector (μ, n, ρ) -mmr-source with equality-pattern probability $\zeta(\cdot)$ and making at most $h(\cdot)$ random oracle queries. Then there exists an IND-CPA adversary B such that for all k*

$$\text{Adv}_{\mathcal{AE}, A}^{\text{cda}}(k) \leq v(k) \left(\text{Adv}_{\mathcal{AE}_r, B}^{\text{ind-cpa}}(k) + \frac{2 \cdot h(k)}{2^{\mu(k)}} + 8 \cdot \text{maxpk}_{\mathcal{AE}_r}(k) \right) + \zeta(k)$$

Adversary B runs in time that of A and $\text{maxpk}_{\mathcal{AE}_r}$ is the maximum public key probability of \mathcal{AE}_r . □

The first part of the theorem is straightforward to prove. The second follows from an adaptation of the proof of security for the similar Encrypt-with-Hash deterministic encryption scheme in [6]. Notice that the theorem holds for both REwH1 and REwH2; the only difference is that with the latter the $\text{maxpk}_{\mathcal{AE}}(k)$ term improves depending on the length κ .

RANDOMIZED-THEN-DETERMINISTIC. Intuitively, the non-adaptive hedged security of the RtD construction is inherited from the IND-CPA security of the underlying randomized scheme \mathcal{AE}_r and the (non-adaptive) PRIV security of the underlying deterministic scheme \mathcal{AE}_d . As alluded to before, we have one technical requirement on \mathcal{AE}_r for the IND-CDA proof to work. We say $\mathcal{AE}_r = (\mathcal{P}_r, \mathcal{K}_r, \mathcal{E}_r, \mathcal{D}_r)$ with message length $n_r(\cdot)$ and randomness length $\rho(\cdot)$ is *min-entropy preserving* if for any k , any $par_r \in [\mathcal{P}_r(1^k)]$, any $(pk_r, sk_r) \in [\mathcal{K}_r(par_r)]$, and for all $c \in \{0, 1\}^*$ it is the case for any (μ, n_r, ρ) -mr-source \mathcal{M} outputting vectors of size one that $\Pr[c = \mathcal{E}_r(pk_r, m; r) : (m, r) \leftarrow_s \mathcal{M}(1^k)] \leq 2^{-\mu}$. In words, encryption preserves the min-entropy of the input message and randomness. We have the following theorem.

Theorem 2. [RtD is non-adaptive H-IND secure]. Let $\mathcal{AE}_r = (\mathcal{P}_r, \mathcal{K}_r, \mathcal{E}_r, \mathcal{D}_r)$ be a min-entropy preserving PKE scheme with message length $n_r(\cdot)$ and randomness length $\rho(\cdot)$. Let $\mathcal{AE}_d = (\mathcal{P}_d, \mathcal{K}_d, \mathcal{E}_d, \mathcal{D}_d)$ be a (deterministic) encryption scheme with message length $n_d(\cdot)$ so that $n_d(\cdot) \geq \max\text{clen}_r(\cdot)$. Let $\mathcal{AE} = \text{RtD}[\mathcal{AE}_r, \mathcal{AE}_d] = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ be the PKE scheme defined in Section 4.

- (IND-CPA) Let A be an IND-CPA adversary. Then there exists an IND-CPA adversary B such that for any k

$$\text{Adv}_{\mathcal{AE}, A}^{\text{ind-cpa}}(k) = \text{Adv}_{\mathcal{AE}_r, B}^{\text{ind-cpa}}(k)$$

where B runs in time that of A plus the time to run \mathcal{E}_d once.

- (IND-CDA) Let A be a CDA adversary that makes one LR query consisting of a $v(\cdot)$ -vector (μ, n_r, ρ) -mmr-source (resp. block-source). Then there exists a PRIV adversary B such that for any k

$$\text{Adv}_{\mathcal{AE}, A}^{\text{cda}}(k) \leq \text{Adv}_{\mathcal{AE}_d, B}^{\text{priv}}(k)$$

where B runs in time that of A plus the time to run $v(k)$ executions of \mathcal{E}_r and makes one LR query consisting of a $v(\cdot)$ -vector $(\mu, \max\text{clen}_r)$ -mm-source (resp. block-source). □

Note that the second part of the theorem states the result for either sources or just block-sources. We briefly sketch the proof. The first part of the theorem is immediate from the IND-CPA security of \mathcal{AE}_r . For the second part, any mmr-source \mathcal{M} queried by A is converted into an mm-source \mathcal{M}' to be queried by B . This is done by having \mathcal{M}' run \mathcal{M} to get $(\mathbf{m}_0, \mathbf{m}_1, \mathbf{r})$ and then outputting the pair of vectors $(\mathcal{E}_r(pk, \mathbf{m}_0; \mathbf{r}), \mathcal{E}_r(pk, \mathbf{m}_1; \mathbf{r}))$. (The ciphertexts are the “messages” for \mathcal{E}_d .) Because \mathcal{AE}_r is min-entropy preserving, \mathcal{M}' is a source of the appropriate type.

PAD-THEN-DETERMINISTIC. The security of the PtD scheme is more difficult to establish. The IND-CDA security is inherited immediately from the PRIV security of the \mathcal{AE}_d scheme. Here the challenge is, in fact, proving IND-CPA security. For this we will need a stronger assumption on the underlying deterministic encryption scheme — that it is a u-LTDF.

Theorem 3. [PtD is non-adaptive H-IND secure]. Let $\mathcal{AE}_d = (\mathcal{P}_d, \mathcal{K}_d, \mathcal{E}_d, \mathcal{D}_d)$ be a deterministic encryption scheme with message length $n_d(\cdot)$. Let $\mathcal{AE} = \text{PtD}[\mathcal{AE}_d] = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ be the PKE scheme defined in Section 4 with message length $n(\cdot)$ and randomness length $\rho(\cdot)$ such that $n(k) = n_d(k) - \rho(k)$ for all k .

- (IND-CPA) Let \mathcal{K}_l be a universal-inducing (n_d, ℓ) -lossy key generation algorithm for \mathcal{AE}_d . Let A be an IND-CPA adversary. Then there exists a LOS adversary B such that for all k

$$\text{Adv}_{\mathcal{AE}, A}^{\text{ind-cpa}}(k) \leq \text{Adv}_{\mathcal{AE}_d, \mathcal{K}_l, B}^{\text{los}}(k) + \sqrt{2^{3n(k) - \ell(k) + 2}}.$$

B runs in time that of A .

- (IND-CDA) Let A be a CDA adversary that makes one LR query consisting of a $v(\cdot)$ -vector (μ, n, ρ) -mmr-source (resp. block-source). Then there exists a PRIV adversary B such that for all k

$$\mathbf{Adv}_{\mathcal{AE},A}^{\text{cda}}(k) \leq \mathbf{Adv}_{\mathcal{AE}_d,B}^{\text{priv}}(k)$$

where B runs in time that of A and makes one LR query consisting of a $v(\cdot)$ -vector (μ, n_d) -mm-source (resp. block-source). \square

One might think that concluding IND-CPA can be based just on PtD being IND-CDA secure, since the padded randomness provides high min-entropy. However, this approach does not work because an IND-CPA adversary expects knowledge of the public-key *before* making any LR queries, while a CDA adversary only learns the public-key *after* making its LR queries. This issue is discussed in more detail in [8]. We use a different approach (which may be of independent interest) to prove this part of Theorem 3; the details are given in the full version [7]. Our proof strategy, intuitively, corresponds to using the standard LHL $2^{n(k)}$ times, once for each possible message the IND-CPA adversary might query.

6 Anonymity for Chosen Distribution Attacks

In the previous section we proved non-adaptive security for the RtD and PtD constructions. But, as established in Section 3, we actually want to meet the stronger goal of adaptive security. In the adaptive setting, adversaries can make multiple LR queries, specifying sources that are generated as a function of previously-seen ciphertexts. Recall that one reason adaptivity is difficult to achieve is because ciphertexts might leak information about the public key. In turn, knowledge of the public key leads to trivial IND-CDA attacks. This suggests a natural relationship with key privacy, also called anonymity [4]. Anonymity requires (informally) that ciphertexts leak no information about the public key used to perform encryption. In this section we formalize a notion of anonymity for chosen-distribution attacks. In the next section we'll use this definition as a step towards adaptive IND-CDA security.

DEFINITIONS. Let $\mathcal{AE} = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme. Game ANON $_{\mathcal{AE}}$ shown in Figure 3 provides the adversary with two oracles. An ANON adversary A is one whose queries are all mr-sources. The advantage of ANON adversary A is

$$\mathbf{Adv}_{\mathcal{AE},A}^{\text{anon}}(k) = 2 \cdot \Pr [\text{ANON}_{\mathcal{AE},k}^A \Rightarrow \text{true}] - 1 .$$

We say that a PKE scheme \mathcal{AE} with message length $n(\cdot)$ and randomness length $\rho(\cdot)$ is ANON secure for (μ, n, ρ) -mr-sources if for all PT adversaries A that only query (μ, n, ρ) -mr-sources the function $\mathbf{Adv}_{\mathcal{AE},A}^{\text{anon}}(\cdot)$ is negligible. We can extend this notion to mr-block-sources in the obvious way. In the special case that the randomness length of \mathcal{AE} is always zero, the ANON definition formalizes anonymity for deterministic encryption or, equivalently, trapdoor functions, generalizing a definition from [4].

DISCUSSION. Anonymity for PKE in the sense of key privacy was first formalized by Bellare et al. [4], but their notion (analogously to traditional semantic security) only works in the context of good randomness. The ANON notion, akin to IND-CDA, formalizes key privacy in the face of bad randomness. While

<u>proc. Initialize(k):</u>	<u>proc. Enc(\mathcal{M}):</u>	<u>proc. LR(\mathcal{M}):</u>	<u>proc. Finalize(a'):</u>
$par \leftarrow \mathcal{P}(1^k)$	If $pkout = true$	$(\mathbf{m}, \mathbf{r}) \leftarrow \mathcal{M}(1^k)$	Ret $(a = a')$
$(pk_0, sk_0) \leftarrow \mathcal{K}(par)$	Ret \perp	$\mathbf{c} \leftarrow \mathcal{E}(pk_a, \mathbf{m}; \mathbf{r})$	
$(pk_1, sk_1) \leftarrow \mathcal{K}(par)$	$(\mathbf{m}, \mathbf{r}) \leftarrow \mathcal{M}(1^k)$	$pkout \leftarrow true$	
$a \leftarrow \{0, 1\}$	Ret $\mathcal{E}(pk_0, \mathbf{m}; \mathbf{r})$	Ret (pk_0, pk_1, \mathbf{c})	
Ret par			

Fig. 3. Game ANON _{\mathcal{AE}, k}

we will use it mainly as a technical tool to simplify showing that schemes meet adaptive IND-CDA, it is also of independent interest as a new security target for PKE schemes when key privacy is important. (That is, one might want to hedge against bad randomness for anonymity as well as message privacy.)

7 Adaptive Hedge Security

The following theorem, whose proof appears in the full version [7], shows that achieving ANON security and non-adaptive IND-CDA security are sufficient for achieving adaptive IND-CDA security.

Theorem 4. *Let $\mathcal{AE} = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme with message length $n(\cdot)$ and randomness length $\rho(\cdot)$. Let A be a IND-CDA adversary making $q(\cdot)$ LR queries, each being a $v(\cdot)$ -vector (μ, n, ρ) -mmr-source (resp. block-source). Then there exist IND-CDA adversary B and ANON adversary C such that for all k*

$$\text{Adv}_{\mathcal{AE}, A}^{\text{cda}}(k) \leq 2q(k) \cdot \text{Adv}_{\mathcal{AE}, B}^{\text{cda}}(k) + 4q(k) \cdot \text{Adv}_{\mathcal{AE}, C}^{\text{anon}}(k).$$

B makes one LR query consisting of a $v(\cdot)$ -vector (μ, n, ρ) -mmr-source (resp. block-source). C makes at most $q(k) - 1$ Enc queries and one LR query, all these consisting of $v(\cdot)$ -vector (μ, n, ρ) -mr-sources (resp. block-sources). Both B and C run in the same time as A . □

Given a non-adaptively IND-CDA secure scheme, Theorem 4 reduces the task of showing it adaptively secure to that of showing it meets the ANON definition. Of course, ANON is still an adaptive notion. (Adversaries can formulate their LR query to be a source that’s a function of previously seen ciphertexts.) Nevertheless, it formalizes a sufficient condition for adaptive CDA security of any PKE scheme and captures the relationship between adaptivity and anonymity. We believe this is an interesting (and novel) application of anonymity.

We can show that our random oracle scheme REwH is ANON secure when the underlying randomized scheme meets the traditional notions of anonymity for PKE [4]. We also want to show that the RtD and PtD schemes are ANON secure. We first show something more general: that any u-LTDF is anonymous. Then, that RtD and PtD are anonymous follows when using deterministic schemes that are also u-LTDFs.

UNIVERSAL LTDFs ARE ANONYMOUS. Intuitively u-LTDFs are anonymous because the lossy mode admits a universal hash, implying that no information about the public key is leaked by outputs (generated from sources with high conditional min-entropy). One might expect that formalizing this intuition would follow from straightforward application of the Leftover Hash Lemma (LHL) [26]. However our anonymity definitions are adaptive, so one cannot apply the LHL (or even the generalized LHL [17]) directly. Rather, we first show an adaptive variant of the LHL is implied by the standard LHL via a hybrid argument. See the full version for details. Here we use it to prove the following theorem; details appear in the full version [7].

Theorem 5. *Let $\mathcal{AE}_d = (\mathcal{P}_d, \mathcal{K}_d, \mathcal{E}_d, \mathcal{D}_d)$ be a (deterministic) encryption scheme with message length $n(\cdot)$ and an associated universal-inducing (n, ℓ) -lossy key generator \mathcal{K}_ℓ . Let A be an ANON adversary making $q(\cdot)$ **Enc** queries and a single **LR** query, all of these being $v(\cdot)$ -vector (μ, n) - m -block-sources. Then there exists LOS adversary B such that for all k*

$$\mathbf{Adv}_{\mathcal{AE}_d, A}^{\text{anon}}(k) \leq 2 \cdot \mathbf{Adv}_{\mathcal{AE}_d, B}^{\text{los}}(k) + 3 \cdot q(k) \cdot v(k) \cdot \sqrt{2^{n(k)-\ell(k)-\mu(k)}}.$$

B runs in time that of A . □

Consider RtD and PtD when instantiated with a deterministic encryption scheme that is a u-LTDF. We can apply Theorem 5 to conclude ANON security for both schemes. Combining this with Theorems 2 and 4 yields proof of adaptive hedge security for RtD. Likewise, combining it with Theorems 3 and 4 yields proof of adaptive hedge security for PtD. Also Theorems 4 and 5 combine with [12, Th. 5.1] to give the first adaptively-secure deterministic encryption scheme (based on u-LTDFs).

REwH2 IS ADAPTIVELY SECURE. As we show above, we can get adaptive security from REwH when the underlying IND-CPA randomized scheme is anonymous in the sense of [4]. We observe that scheme REwH2 is adaptively secure when instantiated with *any* IND-CPA randomized scheme (not just anonymous ones). To show this, we give a direct proof in the full version [7]. Since popular encryption schemes such as RSA are not anonymous, we believe scheme REwH2 could be relevant in practice. That being said, we still think REwH1 is important since non-adaptive security is still a strong notion, and the scheme does not require any changes to the structure of the public key.

EXTENSIONS. In the full version [7] we discuss extensions and variants of RtD and PtD, where we improve the (adaptive) concrete security and show how to securely use LTDFs that are not necessarily universal.

Acknowledgements

We thank the Asiacrypt 2009 reviewers for detailed and thoughtful comments. Mihir Bellare and Thomas Ristenpart are supported by NSF grant CNS-0627779 and a gift from Intel Corporation. Moni Naor is Incumbent of the Judith Kleeman

Professorial Chair. His research is supported in part by a grant from the Israel Science Foundation. Gil Segev is supported by the Adams Fellowship Program of the Israel Academy of Sciences and Humanities. Hovav Shacham is supported in part by a MURI grant administered by the Air Force Office of Scientific Research. Scott Yilek is supported by NSF grant CNS-0831536.

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, Springer, Heidelberg (2001)
2. Abeni, P., Bello, L., Bertacchini, M.: Exploiting DSA-1571: How to break PFS in SSL with EDH (July 2008), http://www.lucianobello.com.ar/exploiting_DSA-1571/index.html
3. Baudron, O., Pointcheval, D., Stern, J.: Extended notions of security for multicast public key cryptosystems. In: Welzl, E., Montanari, U., Rolim, J.D.P. (eds.) ICALP 2000. LNCS, vol. 1853, p. 499. Springer, Heidelberg (2000)
4. Bellare, M., Boldyreva, A., Desai, A., Pointcheval, D.: Key-privacy in public-key encryption. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, p. 566. Springer, Heidelberg (2001)
5. Bellare, M., Boldyreva, A., Micali, S.: Public-key encryption in a multi-user setting: Security proofs and improvements. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, p. 259. Springer, Heidelberg (2000)
6. 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)
7. Bellare, M., Brakerski, Z., Naor, M., Ristenpart, T., Segev, G., Shacham, H., Yilek, S.: Hedged public-key encryption: How to protect against bad randomness. IACR ePrint Archive (2009), Full Version of this paper
8. Bellare, M., Fischlin, M., O'Neill, A., Ristenpart, T.: Deterministic encryption: Definitional equivalences and constructions without random oracles. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 360–378. Springer, Heidelberg (2008)
9. 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)
10. 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)
11. Blum, M., Micali, S.: How to generate cryptographically strong sequences of pseudo random bits. In: FOCS 1982. IEEE, Los Alamitos (1982)
12. Boldyreva, A., Fehr, S., O'Neill, A.: On notions of security for deterministic encryption, and efficient constructions without random oracles. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 335–359. Springer, Heidelberg (2008)
13. Boneh, D.: Simplified OAEP for the RSA and Rabin functions. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, p. 275. Springer, Heidelberg (2001)
14. Bosley, C., Dodis, Y.: Does privacy require true randomness? In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 1–20. Springer, Heidelberg (2007)
15. Brown, D.R.: A weak randomizer attack on RSA-OAEP with $e=3$. IACR ePrint Archive (2005)

16. Dodis, Y., Ong, S.J., Prabhakaran, M., Sahai, A.: On the (im)possibility of cryptography with imperfect randomness. In: FOCS 2004. IEEE, Los Alamitos (2004)
17. Dodis, Y., Ostrovsky, R., Reyzin, L., Smith, A.: Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data. *SIAM Journal of Computing* 38(1), 97–139 (2008)
18. Dodis, Y., Smith, A.: Entropic security and the encryption of high entropy messages. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 556–577. Springer, Heidelberg (2005)
19. Dorrendorf, L., Gutterman, Z., Pinkas, B.: Cryptanalysis of the windows random number generator. In: CCS 2007. ACM, New York (2007)
20. Fujisaki, E., Okamoto, T.: How to enhance the security of public-key encryption at minimum cost. In: Imai, H., Zheng, Y. (eds.) PKC 1999. LNCS, vol. 1560, Springer, Heidelberg (1999)
21. Gamal, T.E.: 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)
22. Goldberg, I., Wagner, D.: Randomness in the Netscape browser. *Dr. Dobbs's Journal* (January 1996)
23. Goldwasser, S., Micali, S.: Probabilistic encryption. *Journal of Computer and System Sciences* 28(2), 270–299 (1984)
24. Gutterman, Z., Malkhi, D.: Hold your sessions: An attack on Java session-id generation. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 44–57. Springer, Heidelberg (2005)
25. Gutterman, Z., Pinkas, B., Reinman, T.: Analysis of the linux random number generator. In: IEEE Symposium on Security and Privacy, pp. 371–385 (2006)
26. Impagliazzo, R., Levin, L.A., Luby, M.: Pseudo-random generation from one-way functions. In: STOC 1989. ACM, New York (1989)
27. Kamara, S., Katz, J.: How to encrypt with a malicious random number generator. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 303–315. Springer, Heidelberg (2008)
28. McInnes, J.L., Pinkas, B.: On the impossibility of private key cryptography with weakly random keys. In: Menezes, A., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 421–435. Springer, Heidelberg (1991)
29. Mueller, M.: Debian OpenSSL predictable PRNG bruteforce SSH exploit (May 2008), <http://milw0rm.com/exploits/5622>
30. Ouafi, K., Vaudenay, S.: Smashing SQUASH-0. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 134–152. Springer, Heidelberg (2009)
31. Peikert, C., Waters, B.: Lossy trapdoor functions and their applications. In: STOC 2008. ACM, New York (2008)
32. Rogaway, P.: Nonce-based symmetric encryption. In: Roy, B., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 348–359. Springer, Heidelberg (2004)
33. Rogaway, P., Shrimpton, T.: A provable-security treatment of the key-wrap problem. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 373–390. Springer, Heidelberg (2006)
34. Rosen, A., Segev, G.: Efficient lossy trapdoor functions based on the composite residuosity assumption. *Cryptology ePrint Archive, Report 2008/134* (2008)
35. Waters, B.: Personal Communication to Hovav Shacham (December 2008)
36. Yilek, S., Rescorla, E., Shacham, H., Enright, B., Savage, S.: When private keys are public: Results from the 2008 Debian OpenSSL vulnerability. In: IMC 2009. ACM, New York (to appear, 2009)

Secure Two-Party Computation Is Practical

Benny Pinkas¹, Thomas Schneider², Nigel P. Smart³, and Stephen C. Williams³

¹ Dept. of Computer Science,
University of Haifa
Haifa 31905, Israel
`benny@pinkas.net`

² Horst Görtz Institute for IT-Security,
Ruhr-University Bochum,
D-44780 Bochum, Germany
`thomas.schneider@trust.rub.de`

³ Dept. Computer Science,
University of Bristol,
Woodland Road,
Bristol, BS8 1UB, United Kingdom
`{nigel,williams}@cs.bris.ac.uk`

Abstract. Secure multi-party computation has been considered by the cryptographic community for a number of years. Until recently it has been a purely theoretical area, with few implementations with which to test various ideas. This has led to a number of optimisations being proposed which are quite restricted in their application. In this paper we describe an implementation of the two-party case, using Yao's garbled circuits, and present various algorithmic protocol improvements. These optimisations are analysed both theoretically and empirically, using experiments of various adversarial situations. Our experimental data is provided for reasonably large circuits, including one which performs an AES encryption, a problem which we discuss in the context of various possible applications.

1 Introduction

That secure multi-party computation can be executed at all is considered one of the main results of the theory of cryptography. Starting with Yao's seminal work [30] many authors have looked at various optimisations and extensions to the basic concept, for both the two-party and the multi-party settings, see for example [7, 10, 11, 18, 20, 23, 29]. Until recently all work on secure multi-party computation has been essentially of a theoretical nature, focusing on feasibility results. However in the last few years a number of practical implementations have appeared [3, 5, 6, 22, 24].

There are many different protocols for secure multi-party computation. Our work focuses on implementation of secure computation and therefore we only mention protocols which have been previously implemented. Secure multi-party computation essentially comes in two flavours. The first approach is typically

based upon secret sharing and operates on an arithmetic circuit representation of the computed function, such as in the BGW (Ben-Or, Goldwasser and Wigderson) or CCD (Chaum, Crepeau and Damgård) protocols [4, 8]. This approach is usually applied when there is an honest majority among the participants (which can only exist if more than two parties participate in the protocol). An alternative approach represents the function as a binary circuit. This approach was used in the original two-party garbled circuit construction of Yao [30], and in the GMW (Goldreich, Micali and Wigderson) multi-party protocol [11].

The arithmetic circuit method is better at representing addition and multiplication operations, where parties have additive shares of secret values, but cannot be used to compute comparisons unless the shares are converted to shares of the binary representation of the values. This approach has been used to great effect in the SIMAP project [6], which has resulted in a “real-life” application of secure multi-party computation to the Danish sugar beet industry [5].

The binary circuit approach handles arithmetic operations, especially multiplications, less efficiently, but can easily compute binary operations such as comparisons. This second approach, which forms the basis of Yao’s construction for the two party case, has been implemented by Malkhi et al. in the Fairplay system [24]. That system also provides a method to compile a given functionality from a representation in a high-level language into a circuit, which is then interpreted by a run-time environment that performs the secure evaluation of this functionality. FairplayMP, an extension of Fairplay to the case of more than two parties using a modified version of the protocol of Beaver et al. [2] has recently been released [3]. All these implementations provide security against semi-honest adversaries only. A major advantage of the binary circuit based systems (Fairplay and FairplayMP) is that they run in a constant number of communication rounds, whereas the SIMAP system has the advantage of being able to process arithmetic operations very efficiently.

Efficient extensions of Yao’s construction to more relevant adversarial models have been a topic of research interest in the last few years. There are several constructions which aim to secure the protocol against malicious adversaries without using generic zero-knowledge protocols. We will focus on the construction of Lindell and Pinkas [20] which is efficient and provides fully simulatable security according to the definition of Canetti [7]. A definition of a weaker class of corruption, “covert adversaries”, and a protocol secure against this type of behavior, was provided by Aumann and Lindell [1]. In [22] an implementation of the basic Lindell–Pinkas protocol was reported upon and experimental data in various security models was provided.

¹ This construction may be preferable over other two-party protocols with security against malicious adversaries. The construction of Mohassel and Franklin [23] only protects privacy and is not fully simulatable. The construction of Jarecki and Shmatikov [18] requires the use of public-key operations, rather than symmetric key operations, for any gate of the circuit. The construction of Nielsen and Orlandi [26], too, uses public key operations, or rather public-key based commitments, for each key of every wire of the circuit. A precise practical comparison between the different approaches is beyond the scope of the current paper.

In this paper we improve on the implementation of [22] in a number of ways. The resulting set of quantitative improvements results in qualitative conclusions: (1) We demonstrate that two-party computation, secure against malicious adversaries, is truly practical, and we experimentally identify the performance bottlenecks which remain after our optimisations. This result should direct further research to the issues which have the largest effect on performance. (2) We experiment with a secure computation of the AES standard, and show that it is indeed feasible, even with security against malicious adversaries. There are a number of applications of such an implementation, some of which we describe below. (3) We provide the first implementation of a protocol with security against covert adversaries and we compare the performance of all 3 types of protocols: malicious, covert and semi-honest.

A more detailed summary of our main results is as follows:

- We improve the communication cost for transmitting the circuits between the parties. In the case when we model the underlying key derivation functions (KDFs) as correlation robust (see discussion below), using the technique of [19] we are able to transmit no information for the XOR gates within the circuit. In this situation we are also able to reduce the data which needs to be sent by 25% for the other gates. When we are not willing to model the KDFs as correlation robust, and we only assume they are pseudo-random functions, we are unable to perform the free XOR optimisation. However we are able to reduce the communication cost for all gates by 50%. Unlike other methods used to improve communication, like [13], our improvement makes a marginal impact on computational costs. We will return to this in a later section.
- In addition to the theoretical analysis we provide experimental data for evaluating “real life” circuits, in both the honest-but-curious, covert and malicious adversary cases; also for the two different methods in the literature that construct the auxillary circuits in the covert and malicious cases (see [22] and the full version). The implementation for the malicious setting is based on the construction of Lindell and Pinkas [20] which provides security in the sense of full simulatability. Therefore the resulting construction can be used as a black-box primitive in more complex applications. The use of our optimisations results in a considerable performance boost compared to previous experimental results published in [22].

Our optimisations change the performance bottleneck to a different part of the computation; namely, the verification of garbled circuits generated by the circuit constructor. This observation is important for focusing future research on the issues that affect the overhead the most.

- We experiment with secure evaluation of a circuit which computes an AES encryption of a single block. The secure computation of AES involves one party which knows the key, and a different party which has an input block. The second party learns the encryption of the block, while the first party learns nothing. We demonstrate the feasibility of computing this function in the semi-honest, covert and malicious settings.

Secure evaluation of AES has an impact in a number of scenarios which we will discuss in short here and elaborate on in the full version. The fact that a secure computation of AES is feasible, and can run in a matter of seconds, is quite surprising.

Application 1, OPRF: A secure computation of a pseudo-random function, denoted OPRF for “oblivious prf”, has been defined in [9] for the purpose of secure keyword based searches, and was subsequently used in different applications. The OPRF protocol in [9] is based on the Naor-Reingold prf, which is a number theoretic construction. Our construction has different advantages over the NR based construction, which we detail in the full version.

Application 2, Side Channel Protection: In [12] the authors introduce “one-time programs”, which are programs that can only be executed once and then “self-destruct”. An important advantage of this construction is that the execution of the program reveals no side-channel information. Most of the computation in that construction is essentially done using a garbled Yao circuit.

One of the main applications of smart cards is to compute symmetric encryptions, and therefore the ability to compute AES encryptions by Yao circuits has immediate application in the above scenario. It enables smart cards to perform a one-time computation, secure against side-channel attacks, of AES. This is particularly interesting since in that setting the circuit evaluation need only be secure against semi-honest adversaries, while we show below that semi-honest computation of AES can be run very efficiently, taking only a few seconds.

Application 3, Blind MACs and Blind Encryption: One can think of the operation of obtaining the AES encryption of a message, under the other party’s secret key, as a blind MAC or a blind symmetric encryption. These operations have different applications in secure computation.

Application 4, Third Party Operations on Encrypted Data: We essentially show that encryption and decryption can be implemented using circuits. This enables secure computation of homomorphic operations on encrypted data. This operation is done by a circuit which receives two ciphertexts from one party and a key from the other party, decrypts the ciphertexts, applies some arbitrary mathematical operation to the plaintexts, and then encrypts the result.

2 Yao’s Garbled Circuit Construction

Two-party secure function evaluation makes use of the famous garbled circuit construction of Yao [30] which we briefly overview in this section. The basic idea is to encode the function to be computed via a binary circuit and then to securely evaluate the circuit on the players’ inputs.

2.1 Garbled Circuits

We consider two parties, denoted as P_1 and P_2 , who wish to compute a function securely which is represented as a simple binary circuit. First assume the circuit consists of only a single gate with two input wires and one output wire. We denote the input wires by w_1 and w_2 , and the output wire by w_3 . The input to w_1 is denoted by b_1 and is known to P_1 , similarly P_2 knows the input to w_2 and this is given by b_2 . Each gate has a unique identifier Gid ; this enables a circuit fan out of greater than one, i.e., it enables the output wire of one gate to be used in more than one other gate. We require that P_2 evaluates the gate on the two inputs, without P_1 learning anything, and without P_2 determining the value b_1 , bar what it can deduce from the output of the gate and its own input. We define the output of the gate by the function $G(b_1, b_2) \in \{0, 1\}$.

The construction of Yao works as follows. P_1 encodes, or garbles, each wire w_i by selecting two different cryptographic keys k_i^0 and k_i^1 of length t . Here t is a computational security parameter which suffices for the length of a symmetric encryption scheme. A random permutation π_i of $\{0, 1\}$ is associated to each wire. The garbled value of wire w_i is then represented by $k_i^{b_i} \| c_i$, where $c_i = \pi_i(b_i)$. We call the value c_i the “external value” of the wire, note that this value is completely independent of the actual value of the wire b_i .

An encryption function $E_{k_1, k_2}^s(m)$ is selected which has as input two keys of length t , a message m , and some additional information s . The additional information s must be unique per invocation of the encryption function, i.e., it is used only once for any choice of keys. The gate itself is then replaced by a four entry table indexed by the values of c_1 and c_2 , and given by

$$c_1, c_2 : E_{k_1^{b_1}, k_2^{b_2}}^{\text{Gid} \| c_1 \| c_2} \left(k_3^{G(b_1, b_2)} \| c_3 \right),$$

where $c_1 = \pi_1(b_1)$, $c_2 = \pi_2(b_2)$, and $c_3 = \pi_3(G(b_1, b_2))$. Each entry in the table corresponds to a combination of the values of the input wires and contains the encryption of the corresponding garbled output value. The resulting look up table, or set of look up tables in general, is called the “garbled circuit”.

Player P_1 then sends to P_2 the garbled circuit, the key corresponding to its input value $k_1^{b_1}$, the value $c_1 = \pi_1(b_1)$, and the permutation π_3 . The parties engage in an oblivious transfer (OT) protocol so that P_2 learns the value of $k_2^{b_2} \| c_2$, where $c_2 = \pi_2(b_2)$. Player P_2 can then decrypt the entry in the look up table indexed by (c_1, c_2) using $k_1^{b_1}$ and $k_2^{b_2}$; revealing the value of $k_3^{G(b_1, b_2)} \| c_3$. P_2 determines the value of $G(b_1, b_2)$ by using the mapping π_3^{-1} from c_3 to $\{0, 1\}$.

In the general case the circuit consists of multiple gates. Player P_1 chooses random garbled values for all wires and uses them for constructing tables for all gates. It sends these tables, i.e., the garbled circuit, to P_2 and in addition provides P_2 with the garbled values and the c values of P_1 's inputs, and with the permutations π used to encode the *output* wires of the circuit. Player P_2 uses invocations of oblivious transfer to learn the garbled values and c values of its

own inputs to the circuit. Given these values, P_2 can evaluate the gates in the first level of the circuit, compute the garbled values and the c values of their output wires. Player P_2 can then continue with this process and compute the garbled values of all wires in the circuit. Finally P_2 uses the π permutations of the output wires of the circuit to compute the real output values of the circuit. If P_1 additionally requires some output from the circuit then this can be dealt with by standard mechanisms, as described in the full version.

One could use more general gates than 2-to-1 gates, such as n -to- m gates with 2^n entries. However the optimisations we shall present in this paper are most effective when applied to 2-to-1 gates. While we found that more general gates can improve the performance of a naive Yao circuit protocol, they actually decrease the performance of the optimisations. Hence the rest of this paper is restricted to 2-to-1 gates.

2.2 Required Implementation Details

Having described the basic theoretical description of Yao's protocol and its extensions, we now present a number of implementation details which are needed to understand some of our optimisations. The basic implementation choice of the underlying encryption scheme to be used is the same as the implementation described in [22].

Oblivious transfer: Unlike [22] we do not use the OT scheme of Hazay and Lindell (HL) [15]. Instead we use the OT scheme of Peikert et al. (PVW) [27]. This scheme is UC-secure and hence requires the setup of a Common Reference String (CRS) of a few hundred bits. For our experiments we assume that this is given to the parties. (Alternatively, the parties can run a coin-tossing protocol to generate the CRS, which is possible due to the nature of the CRS used in the PVW scheme.) The batched method of PVW is more efficient per OT than the batched method of HL, especially on the receiver's side. In particular the CRS can be used for any number of invocations of the OT, whereas the method in HL requires the maximum number of OT's being executed to be known before the setup is performed. (The setup in HL also requires two ZK-proofs as opposed to a CRS being created in PVW.) The OT stage is not our computational bottleneck, and is unlikely to be, unless one is in the rare situation of having a circuit with a large number of inputs for P_2 and yet a relatively small number of gates. Thus we do not consider optimisations of OT schemes which are secure against only semi-honest or covert adversaries, since the fully secure OT is efficient enough.

Encryption scheme: The only implementation detail we will need from [22] is that the encryption scheme is implemented via

$$E_{k_1, k_2}^s(m) = m \oplus \text{KDF}^{|m|}(k_1, k_2, s)$$

where KDF is a key derivation function, whose $|m|$ bits of output are independent of the two input keys in isolation, and which depends on the value of s . We will instantiate this function as follows²

$$\text{KDF}^\ell(k_1, k_2, s) = H(k_1 \| s)_{1\dots\ell} \oplus H(k_2 \| s)_{1\dots\ell}.$$

Even if H is a Merkle–Damgård type hash function this will be secure (with the associated issues of length extension), since we are only applying the function to fixed length inputs. Indeed, in our experiments we implement H using SHA-256.

Modeling the hash function, and correlation robustness: In this paper we need to model the underlying hash function H in two ways. In the first we make the usual assumption that it behaves as a pseudo-random function, namely that $H(k \| s)$ is an invocation of a pseudo-random function keyed by k , with the input s . However one of our optimisations requires that we make a stronger assumption on the hash function, namely that it is correlation robust. This later property can be stated formally as follows:

Definition 1 (Correlation robustness [16]). *An efficiently computable function $H : \{0, 1\}^* \rightarrow \{0, 1\}^\ell$ is correlation robust if the following distribution is pseudo-random: $(t_1, \dots, t_m, H(t_1 \oplus r), \dots, H(t_m \oplus r))$, where t_1, \dots, t_m and r are chosen at random, and m is polynomial in the security parameter.*

This can also be stated by saying that the function $f_r(x) = H(x \oplus r)$ is a weak pseudo-random function. The definition also implies that the distribution of $(H(t_1), \dots, H(t_m), H(t_1 \oplus r), \dots, H(t_m \oplus r))$ is pseudo-random.

The correlation-robustness assumption is satisfied by a random oracle (or rather by a very weak form of it: a non-programmable, non-extractable random oracle). However, assuming correlation robustness seems as a much weaker requirement than assuming the existence of random oracles. This assumption has been introduced in [16] and was used there for providing security against malicious adversaries for a method of extending oblivious transfer. The correlation robustness assumption has been recently used in the context of oblivious transfer [14, 17] and in the context of secure computation [19, 26].

For our construction, as we deal with circuits with arbitrary fan out, we require a slightly modified definition. Namely that for any set $S = \{s_1, \dots, s_{|S|}\}$

² In [22] two instantiations were presented, depending on whether we are working in the random oracle model (ROM) or standard model, via truncating, or extending, the output of a suitable hash function H in the standard way as follows

$$\text{KDF}^\ell(k_1, k_2, s) = \begin{cases} H(k_1 \| k_2 \| s)_{1\dots\ell} & H \text{ is modeled as an RO,} \\ H(k_1 \| s)_{1\dots\ell} \oplus H(k_2 \| s)_{1\dots\ell} & H \text{ is modeled as a PRF.} \end{cases}$$

The difference is that the security analysis in the ROM works even if we feed related keys to different invocations of the function. Namely, it is possible to compute, say, $H(k_1 \| k_2), H(k_1 \| k'_2), H(k'_1 \| k_2)$ and $H(k'_1 \| k'_2)$ and claim that knowledge of k_1, k_2 does not disclose information about any of the values except $H(k_1 \| k_2)$. This is impossible in the standard model. Therefore if $H()$ is modeled as a prf it must be invoked separately with each key.

of size which is of the same order as the number gates, the distribution of $(t_1, \dots, t_m, \langle H((t_1 \oplus r) \| s_1), \dots, H((t_m \oplus r) \| s_1) \rangle, \langle H((t_1 \oplus r) \| s_2), \dots, H((t_m \oplus r) \| s_2) \rangle, \dots, \langle H((t_1 \oplus r) \| s_{|S|}), \dots, H((t_m \oplus r) \| s_{|S|}) \rangle)$ is pseudo-random, where t_1, \dots, t_m and r are chosen at random. In other words, all the pads that are used for encrypting table entries are pseudo-random. If one is willing to assume this then our optimisations provide highly efficient protocols. We also provide optimisations for when the user is unwilling to make such an assumption.

3 Structural Optimisations of the Circuit

Yao's protocol operates on functions which are described as a boolean circuit, and its overhead depends on the size of the circuit. A convenient way of generating a representation of a function in this form is to use a compiler which translates a description of a function in a high-level language to a description as a binary circuit. The Fairplay system provides a compiler for this task which operates on functions described in a high-level language called Secure Function Description Language (SFDL) [3, 24]. We use that compiler as the basis of our experiments, but use our own run-time environment to execute the protocol.

There are a number of general circuit simplifications which can be performed to the output of the Fairplay compiler. We have implemented a number of these, based on two basic ideas: (1) identifying component circuits which can be replaced by simpler combinations of gates, and (2) identifying complicated components whose output must always be zero, or one; this allows for the component to be removed and other subsequent components to be further simplified. A combination of these techniques is surprisingly effective, and allows us to produce circuits which are often 60 percent more efficient than the circuit produced by the Fairplay compiler.

Many of the techniques used are ad-hoc, but the following technique is particularly effective. First, by a technique akin to common sub-expression elimination, we identify sets of gates which can be replaced by a single 3-to-1 gate, and then replace the 3-to-1 gate with a set of 2-to-1 gates which was chosen to minimize the number of non-XOR gates. This is particularly effective when combined with our later technique of Section 4, in the case of correlation robust KDFs, to remove the cost of any XOR gates; however the technique is also successful in the more general case as well. We call a gate *even* if its truth table has an even number of '1' entries (for example, a XOR gate is even), otherwise it is called *odd* (an OR gate, for example, is odd). We show in the full version that it is possible to replace any 3-to-1 even gate with at most a single 2-to-1 non-XOR gate and at most three XOR gates. The optimal transformation rules, which we found by exhaustive search, are listed in the full version.

4 Optimisations with Free XORs, When the KDF Is Correlation Robust

In [19] Kolesnikov and Schneider present an optimisation based on the correlation robustness assumption, which allows XOR gates to be evaluated for free, thus

doing away with the need to evaluate or transmit the garbled tables for such gates. The optimisation requires that there is a global random value R of bit length t , known only to P_1 , such that for all garbled wires w_i it holds that $k_i^1 = k_i^0 \oplus R$. In other words, the garbling of the 1 value of a wire, is determined purely from XOR-ing the garbled 0 value with the value R . Note that a similar property holds for the external values of the wire: $\pi_i(1) = \pi_i(0) \oplus 1$. With this convention we have that a XOR gate can be implemented by simply XOR-ing together the two garbled input values, and the two external values. Namely, for a XOR gate mapping wires w_1 and w_2 to wire w_3 , it holds that $k_3 = k_1 \oplus k_2$ and $c_3 = c_1 \oplus c_2$. For a full proof of this optimisation see [19]. Note that [19] states the proof in the random oracle model, but it can be easily seen, as noted in [19], that the proof can be based on the correlation robustness assumption.

Garbled Row Reduction – GRR: The above solution is ideal for XOR gates, but in addition we would like to reduce the size of the tables of the non-XOR gates as well. The following simple optimisation (which was pointed out in [25]) provides a 25 percent reduction in the sizes of the tables needed to represent two-input gates. We can do this in a way which still allows the use of the above trick for free XOR gates. (In general, this method provides a $1/2^n$ reduction in the size of n -to-1 gates, but we will only describe it in detail for the two input case.)

The observation is that instead of defining the two garbled values of the output wires randomly, we can define one of them as a function of garbled values of the two input wires which result in this output value. In other words, we choose an input pair $(b_1, b_2) \in \{0, 1\}^2$, and define the garbled output value of $G(b_1, b_2)$ to be a function of the garbled values of b_1 and b_2 . The gate table therefore need not store an entry for the input combination (b_1, b_2) . In the evaluation phase, if the evaluator has the garbled values of the pair (b_1, b_2) it can compute the corresponding garbled output directly, without consulting the gate table.

Suppose the gate maps wire w_1 and wire w_2 to wire w_3 . As before we let k_i^0 and k_i^1 denote the garbled wire values, $G(b_1, b_2)$ denote the function being implemented by the gate, and we set the external value of the wire to be $c_i = \pi_i(b_i)$. We then define the garbled output value corresponding to the output resulting from the external input values $(c_0, c_1) = (0, 0)$ as

$$k_3^{G(\pi_1^{-1}(0), \pi_2^{-1}(0))} \parallel c_3 = \text{KDF}^{t+1} \left(k_1^{\pi_1^{-1}(0)}, k_2^{\pi_2^{-1}(0)}, \text{Gid} \parallel 0 \parallel 0 \right).$$

In other words, the garbled value is exactly equal to the pseudo-random mask that was used to hide it in the basic protocol. Note that this operation also defines the external value c_3 of this output value. We therefore define π_3 such that $c_3 = \pi_3(G(\pi_1^{-1}(0), \pi_2^{-1}(0)))$. The other garbled value of the output wire, $k_3^{1-G(\pi_1^{-1}(0), \pi_2^{-1}(0))}$ is then chosen as in the free XOR method above, to enable the evaluation of XOR gates for free. The table is then constructed in the standard way except that we do not store, or transmit, its first entry.

On evaluating the garbled gate the evaluator proceeds as in the standard algorithm except when it wishes to access the first entry of the table, i.e., when

the external values of both input wires are 0, namely $c_1 = c_2 = 0$. In that case it possesses the garbled values $k_1^{b_1}$ and $k_2^{b_2}$, where $b_1 = \pi_1^{-1}(0)$ and $b_2 = \pi_2^{-1}(0)$. It uses them to compute $k_3^{G(b_1, b_2)}$ and $c_3 = \pi_3(G(b_1, b_2))$, by computing $\text{KDF}^{t+1}(k_1^{b_1}, k_2^{b_2}, 0 || 0 || \text{Gid})$ as defined in the equation above.

We will denote this optimisation as Garbled Row Reduction, GRR for short, in our future discussions.

Security: We sketch why the above optimisation maintains security. Recall that the proof of security for Yao's protocol given in [21] shows security against a corrupt P_2 based on a hybrid argument, and on a claim that for each gate it is infeasible to distinguish between a correct garbled table of this gate and a table which encrypts the same value in all four entries. In order for this argument to apply to the GRR optimisation, it is required to show that it is infeasible to find out if the garbled value assigned to the first table entry, $k_3^{G(\pi_1^{-1}(0), \pi_2^{-1}(0))} || c_3$ is equal to the values encrypted in the other entries. However this value is equal to the mask that is used to encrypt the first entry in Yao's original protocol, and we know that if a polynomial adversary is given only a single pair of garbled input values then the masks that are used for encrypting the other entries of the table are pseudo-random. Therefore the claim follows.

5 Optimisations without Free Xors, When the KDF Is Not Correlation Robust

One may not want to assume the KDF is correlation robust, or perhaps the proportion of XOR gates in the circuit is so low that making this assumption is not as effective. In these situations, too, we would like to reduce the overhead required by the Yao circuit. This section describes an optimisation which reduces the size of every two-input gate by 50%, but which, unfortunately, cannot be combined with the free XOR method of Section 4.

The underlying idea is that if we are not using the free XOR trick then the two values of the output wire can be chosen independently.³ The 50% reduction in the size of the gate tables is based on Shamir secret sharing [28]. It makes use of a finite field \mathbb{F}_{2^t} . Recall that t is the bit length of the keys used to represent the garbled values of the wires. We can therefore interpret keys as elements of \mathbb{F}_{2^t} and vice versa. We also interpret small integers such as 1, 2, 3 etc. as elements in \mathbb{F}_{2^t} . For example if we think of \mathbb{F}_{2^t} as $\mathbb{F}_2[X]/(f(X))$, for some polynomial of degree t , then the integer 3 can be interpreted as $x + 1$.

As before we assume a garbled table indexed by the external values, c_1 and c_2 , and each entry corresponds to the value being output, on input of the values $k_1^{b_1}$ and $k_2^{b_2}$ where $b_i = \pi_i^{-1}(c_i)$. We set the rows of the gate table to be numbered

³ This allows for possible extensions of the GRR method, and in the full version we detail another optimisation method, which we call Garbled Table Reduction (GTR), which reduces the size for the garbled tables needed to represent odd 2-to-1 gates by 1/3, and the size of tables of even 2-to-1 gates by 1/2.

1, . . . , 4, and therefore set $r = 2c_1 + c_2 + 1$ to be the row number of table entry (c_1, c_2) . We define the value used to mask this entry as

$$K_r || M_r = \text{KDF}^{t+1}(k_1^{b_1}, k_2^{b_2}, s) \tag{1}$$

where $s = \text{Gid} || c_1 || c_2$, K_r is a bit string of length t bits and M_r is a single bit used to mask the external value of the output. We use a different method for optimising odd and even gates. The truth table of each gate, and therefore also the information whether the gate is odd or even, is known to the circuit evaluator. Therefore it can compute each gate according to the right method. (The only information hidden from the evaluator is the values passing on intermediate wires of the circuit.)

5.1 Odd 2-to-1 Gates

Suppose we are implementing an OR-gate, where the external values of $c_1 = 0$ and $c_2 = 0$ correspond to the real input values $(0, 0)$, the other cases will follow immediately from the following. This means that the values $r = 2, 3$ and 4 should evaluate to the same output value k_3^1 , whilst $r = 1$ should evaluate to the output value k_3^0 . We first define over \mathbb{F}_2^t a polynomial $P(X)$ of degree two, by interpolating the polynomial which intersects the three points $(2, K_2)$, $(3, K_3)$ and $(4, K_4)$, where each K_r value was defined according to equation (II). (This is the value which in the other constructions was used to mask entry r of the table.) The garbled output value k_3^1 is defined to be $k_3^1 = P(0)$. We also compute $K_5 = P(5)$ and $K_6 = P(6)$. We then define a second polynomial $Q(X)$, also of degree two, by interpolating the polynomial which intersects the three points $(1, K_1)$, $(5, K_5)$ and $(6, K_6)$, where K_1 was defined according to equation (II). The garbled output value k_3^0 is now defined by $k_3^0 = Q(0)$. The garbled table is replaced by the two values (K_5, K_6) . In addition, for each of the four original rows, the external value for the output wire in the r th row is encrypted using the bit M_r , defined in equation (II). The total amount of data sent for the gate is therefore $2t + 4$ bits.

Player P_2 then, given two key values $k_1^{b_1}$ and $k_2^{b_2}$ plus two external values c_1 and c_2 , computes, using equation (II) the value of K_r and M_r for $r = 2c_1 + c_2 + 1$. Recall that the evaluator knows r but not b_1 or b_2 . It then uses the two supplied values of K_5 and K_6 to interpolate the polynomial passing through the points (r, K_r) , $(5, K_5)$ and $(6, K_6)$. The result is either $Q(X)$ or $P(X)$, depending on whether $r = 1$ or not. Player P_2 then recovers the associated secret value $k_3^{b_3}$, by evaluating the polynomial at the point $X = 0$. Using M_r the evaluator can also decrypt the encryption of the external value of the output wire and so obtains c_3 . Hence the evaluator recovers the correct value of the output wire.

5.2 Even 2-to-1 Gates

The only non-trivial even 2-to-1 gates are the XOR and NXOR gate, since all other gates can be replaced by wires. Again let us assume the external input values $c_1 = 0$ and $c_2 = 0$ correspond to the real input values $(0, 0)$, and assume

we are dealing with a XOR gate. Then the entries 1 and 4 in the standard garbled table will correspond to the same output key, namely $k_3^{\pi_3^{-1}(0)}$. Any other case will follow from the following description.

Player P_1 first creates a linear polynomial $P(X)$ over \mathbb{F}_{2^t} which interpolates the two points $(1, K_1)$ and $(4, K_4)$. The value of $k_3^{\pi_3^{-1}(0)}$ is defined to be equal to $P(0)$. If the external value of this output value is 0 then we store $P(5)$ into the first row of the new table of this gate, otherwise we store $P(5)$ as the second entry. Then P_1 creates another linear polynomial $Q(X)$ which interpolates the two points $(2, K_2)$ and $(3, K_3)$. The value of $k_3^{\pi_3^{-1}(1)}$ is then defined to be $Q(0)$, and the value $Q(5)$ is stored in the remaining row of our new table. The external values of the output wires are now encrypted and stored, using the M_r values as before as a separate sub-table of 4 bits in length. Thus, the total amount of data required to represent the gate is $2t + 4$ bits.

Player P_2 given two key values $k_1^{b_1}$ and $k_2^{b_2}$ plus two external values c_1 and c_2 , computes the value of K_r and M_r . Using M_r it can determine the external value of the output wire. If this external value is zero then using the first entry of our garbled table and the value of K_r , the evaluator recovers $P(X)$ and hence $P(0) = k_3^{\pi_3^{-1}(0)}$. If the external value is one then using the second entry of the table and the value K_r , the evaluator recovers $Q(X)$ and hence $Q(0) = k_3^{\pi_3^{-1}(1)}$.

Security: We sketch why the above optimisations maintain security. Given a pair of garbled values of the input wires, P_2 can compute a garbled output value, but cannot distinguish the other garbled output value from random. This is because that other garbled value is defined using a linear combination with a value which is unknown to P_2 . This fact can be used in a, somewhat modified, security proof in the spirit of the proof of Yao's protocol in [21].

6 Some Experimental Results

We now present some experimental results. In our results we separate out pre-computation time, i.e., generating the required garbled circuits, from the rest of the computation. This is because it depends on the application whether one should consider this time as part of the computation time or not.

There are two major conclusions of our experiments. Firstly, assuming the KDF is correlation robust then the GRR optimisation produces the most efficient implementation. Secondly we conclude that rather large circuits can be practically evaluated using the methods described. Thus secure two-party computation has become more of a reality than one might previously have thought.

Example 1 – Evaluation a Simple Circuit: First we present results for a simple circuit, where we took the circuit for which each of P_1 and P_2 's input is a 32-bit integer. The output for P_2 should be the single bit resulting from the application of the comparison operator on the inputs. The output for P_1 will be a six bit integer resulting from the scalar product of the bits of the two inputs,

Table 1. Experimental Results For Example 1 (Times are in seconds)

Adv.	Input Enc.	Method	No. Gates	% XOR Gates	Precomp Time	Send Time	OT Time	Calc Time	Total Time	Total KBytes
Semi-Honest		Base	251	11	0	0	2	0	2	46
		PRF-SS	537	55	0	0	1	0	1	34
		CoR-GRR	537	55	0	0	1	0	1	22
		ROM-GRR	537	55	0	0	1	0	1	22
Covert	Indep. Inputs	Base	419	38	7	1	4	6	18	1188
		PRF-SS	705	61	8	0	2	7	17	969
		CoR-GRR	705	61	6	1	3	5	15	682
		ROM-GRR	705	61	1	1	2	0	4	629
Covert	Random Comb.	Base	1247	79	9	2	4	7	22	2275
		PRF-SS	1535	82	9	1	3	7	20	1646
		CoR-GRR	1555	82	7	1	3	5	16	682
		ROM-GRR	1555	82	1	1	3	0	5	629
Malic.	Indep. Inputs	Base	1571	83	171	80	47	54	352	180599
		PRF-SS	1857	85	175	79	39	67	360	173942
		CoR-GRR	1857	85	147	78	37	39	301	164323
		ROM-GRR	1857	85	141	71	37	38	287	161741
Malic.	Random Comb.	Base	3029	89	163	75	19	64	321	167276
		PRF-SS	2799	90	161	74	16	69	320	158904
		CoR-GRR	2781	90	117	75	16	39	247	140265
		ROM-GRR	2802	90	117	69	16	37	239	137609

i.e. the number of ones in the string obtained from forming the bit-wise “and” of the two strings.

Applying the Fairplay compiler to this functionality we obtain a circuit with 689 gates. We produce two circuits from this output; the first, denoted $C_{2,3}$, is to allow comparison with the existing state of the art, namely the methods of [22]. This is a circuit which uses 2-to-1 and 3-to-1 gates and has 245 gates. The second circuit we use, denoted C_{xor} , replaces, via the techniques of Section 3, all complex gates with 2-to-1 gates, and tries to minimise the number of non-XOR gates in the circuit. This circuit has 531 gates, 240 of which are non-XOR gates. An extra six gates are needed in each circuit so as to encode P_1 's for transmission back to P_1 , without P_2 learning the value.

The above circuit sizes are purely to implement the functionality, they do not include the extra wires and gates required to transmit P_1 's output back to P_1 (for details of how this is done see the full version), nor do they include the extension of the circuit to cope with P_2 's input in the case of Covert and Malicious adversaries. (We refer to the two methods for encoding P_2 's input as the *independent inputs* and the *random combinations* methods. For the details of these methods see [20] or the full version. These methods add a set of XOR gates to the circuit, which transform P_2 's inputs using a random linear encoding.) The sizes of the extended circuits, and the resulting run-times are given in Table 1, which measures the total elapsed wall times in seconds for the various cases.

The calculations were performed on two machines with Intel Core 2 Duo's running at 3.0 GHz, with 4GB of RAM connected by a 1GB ethernet. The hash function $H()$ used in the protocol was implemented as SHA-256.

The column of "Total KBytes" contains the total number of kilobytes of data which were transferred during the run of the protocol. The column "Method" details the type of computation used, as follows:

- Base: Denotes the optimisations proposed in [22], extended to the case of Covert and Honest adversaries, which we use for comparison purposes, as our baseline implementation. This uses the $C_{2,3}$ circuit mentioned above, the KDF which is secure in the standard model, and the OT of Hazay-Lindell [15] as opposed to that of Peikert et al. [27].
- PRF-SS: This denotes using the secret sharing based method of Section 5, to reduce the size of the garbled tables. For this the KDF is assumed to be a PRF, but not correlation robust.
- CoR-GRR: This denotes an implementation which is only secure assuming the KDF is correlation robust. It uses the free XOR trick and the method of Garbled Row Reduction, from Section 4, to reduce the size of the remaining garbled tables.
- ROM-GRR: As above for CoR-GRR but all hash functions used are modelled as random oracles. This means we can implement our KDF via a single hash function call, based on the method described in Footnote 2.

The column denoted "No. of gates" describes the number of gates, and the percentage of XOR gates, in the extended circuit (which transfers P_1 's outputs and applies the extension described in the full version, encoding P_2 's input).

For the Covert and Malicious cases the "Input Enc." column denotes whether we use the Independent Inputs technique or the Random Combinations technique for the extended circuit construction. See the full version for details. From the table we can deduce the following conclusions:

- The running time in the semi-honest setting is about 10-20 times faster than in the covert setting, which is in turn about 15-20 times faster than in the malicious setting.
- A lot of the extra data needed to be transmitted in the Malicious case is related to the large number of commitments and decommitments which need to be transmitted. Thus our optimisation techniques are less effective in the Malicious case. This points to a clear direction for future research in optimising the Malicious case.
- If one is not willing to assume that the KDF is correlation robust we see that using our technique based on secret sharing can reduce the amount of data being transmitted, compared to the base scheme, without increasing the computational cost.
- In all cases we see that the correlation robust variant using Garbled-Row-Reduction is the most efficient variant. The extra efficiency comes from the free XOR's which reduce both the number of encryption/decryptions which need to be performed and also the amount of data needing to be transmitted.

- Note that if we assume the random oracle model, and so could implement our KDF via a single hash function call then for Covert adversaries the protocols run significantly faster. That this does not apply as much to the Malicious case is due to the fact that most of the time in the Malicious case is spent with creating, sending and verifying the various commitments.

We pause to compare our two optimisations with the optimisation in bandwidth suggested in [13]. In our system P_1 , the circuit constructor, sends commitments to all circuits that it constructs and to its own inputs, and a random subset of these committed values are checked by P_2 . In [13] it is suggested that P_1 commits to a random seed, and uses this to generate the circuit. Then only the commitment to this seed, and eventually its decommitment, need to be transmitted. This means that P_2 needs to compute the circuit given the seed. Whilst this optimisation clearly significantly reduces the consumed bandwidth, it actually leads to a significant increase in the time needed to perform the protocol. To see this consider our Covert experiments in Table 1. The optimisation in [13] would reduce practically to zero, the entry for the “Send Time” column, but P_2 would now need to recompute almost all of the calculations in the “Precomp Time” column. Thus the technique of [13] is only to be compared to ours in the situation where bandwidth is very expensive and CPU time is very cheap.

Before passing onto our larger example we note the following. If we let p denote the proportion of XOR gates within a circuit, and we let N denote the amount of data needed to be sent per circuit in the standard Yao construction, then the average amount of data needed to be sent per circuit gate when using the free XOR gates and GRR methods is $3/4 \cdot (1 - p) \cdot N$. Whereas if we do not use the free XOR gate method and instead use the method based on secret sharing, this value becomes $N/2$. Hence, if we are willing to assume correlation robust KDFs, then the method which uses secret sharing and does not use the free XOR method, will be more efficient as long as the fraction of XOR gates, p , is smaller than $1/3$. However as can be seen from the column entitled “% XOR Gates”, this proportion is generally much larger than $1/3$, especially in the case of Covert and Malicious adversaries where we have had to extend the circuit by a large linear component. This expansion is performed to cope with possible adversarial behaviour related to P_2 's input, see the full version for details. One should note that these theoretical estimates of bandwidth are never achieved fully in practice due to overheads in the underlying data transmission mechanism and the fact that they assume a bit-oriented communication mechanism, whereas practical communication is performed in bytes. Hence the saving we achieve in gate transmission is about 5-10% less than one would predict purely by theory.

Example 2 - Evaluating AES: As our second example we created a circuit which computes an AES encryption of a single 128-bit block with respect to a 128-bit key. Here P_1 's input is the secret key, and P_2 's input is the message block. We require that P_2 learns the encryption of its message under P_1 's secret key, and that P_1 learns nothing. Compiling such a circuit using the Fairplay compiler, and applying various optimisations, resulted in a circuit, which we

Table 2. Experimental Results for Example 2 (Again times are in seconds)

Adv.	Input Enc.	Method	No. Gates	% XOR Gates	Precomp Time	Send Time	OT Time	Calc Time	Total Time	Total KBytes
Semi-Honest		Base	28216	56	5	2	4	3	14	3162
		PRF-SS	33880	66	5	1	3	3	12	1752
		CoR-GRR	33880	66	2	1	2	2	7	503
		ROM-GRR	33880	66	1	1	3	2	7	503
Covert	Indep. Inputs	Base	28600	56	96	47	18	45	206	51899
		PRF-SS	34264	67	92	36	13	50	191	29380
		CoR-GRR	34264	67	40	21	11	23	95	9078
		ROM-GRR	34264	67	22	21	11	6	60	8942
Malic.	Random Comb.	Base	40253	69	1250	448	39	887	2624	987442
		PRF-SS	45944	75	1184	392	34	829	2439	711729
		CoR-GRR	45960	75	483	270	34	361	1148	406010
		ROM-GRR	45881	75	453	276	35	350	1114	417907

denote by $C_2^{(1)}$, with 33880 gates, where each gate is a 2-to-1 gate. This circuit was derived in a way to try to minimize the number of non-XOR gates. Again, we stress, the above circuit size purely implements the AES functionality, it does not include the extension of the circuit to cope with P_2 's input in the case of Covert and Malicious adversaries. Note that the key schedule takes up only about 15% of the circuit, hence encrypting a sequence of message blocks as in CBC-Mode encryption will scale almost linearly with respect to our data.

We repeated our experiments from above, but in Table 2 we only present the times for the most efficient choice for the input encoding.

We conclude that performing the Yao protocol is certainly feasible on complicated functionalities such as AES encryption. For the case of honest and covert adversaries we again see that the computation and bandwidth consumed, when we use correlation robust KDFs and the GRR method, greatly reduces in comparison to the base case. If one is not willing to assume correlation robust KDFs (or use the ROM) then our secret sharing based optimisation greatly reduces the bandwidth without affecting the run times. For the malicious case the improvement in the secret sharing based version is less pronounced due to the large number of commitments which need to be transmitted and opened. This clearly points to the place where future optimisation research needs to be performed, namely in reducing the number of commitments needed in the situation of malicious adversaries. However even without such future optimisation we note that performance can be significantly reduced by taking advantage of the inherent parallelism in the algorithm in the Malicious case (in which P_1 generates many commitments and P_2 verifies a subset of them). For web service or cloud computing applications, where server farms are common place, an improvement in computational time by a factor around s_1 could be expected.

We end by noting that many application domains of a secure evaluation of AES, for example the one-time program example from [12], require only security against semi-honest adversaries. Hence, such applications are already

within the reach of practical realisation. Furthermore, this application requires no computation of the OT or data to be sent. Thus the party generating the one-time-program will take the time needed in our *Precomp Time* column, and the evaluator (after querying the one-time-memory) will take the time needed in the *Calc Time* column.

Acknowledgments

All authors would like to thank the EU projects CACE and eCrypt-2 for partially funding the work in this paper. The first author was also partially funded by the ERC project SFEROT. The third author was partially funded by a Royal Society Wolfson Merit Award, and the fourth author was partially funded by EPSRC. We would also like to thank Benny Applebaum, Yehuda Lindell and Ahmad-Reza Sadeghi for various comments and discussions.

References

1. 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)
2. Beaver, D., Micali, S., Rogaway, P.: The round complexity of secure protocols. In: 22nd STOC, pp. 503–513 (1990)
3. Ben-David, A., Nisan, N., Pinkas, B.: FairplayMP: a system for secure multi-party computation. In: Computer and Communications Security – CCS 2008, pp. 257–266. ACM, New York (2008)
4. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation. In: 20th STOC, pp. 1–10 (1988)
5. Bogetoft, P., Christensen, D.L., Dâmgard, I., Geisler, M., Jakobsen, T., Krøigaard, M., Nielsen, J.D., Nielsen, J.B., Nielsen, K., Pagter, J., Schwartzbach, M., Toft, T.: Secure multiparty computation goes live. In: Dingleline, R., Golle, P. (eds.) FC 2009. LNCS, vol. 5628, pp. 325–343. Springer, Heidelberg (2009)
6. Bogetoft, P., Damgård, I., Jakobsen, T., Nielsen, K., Pagter, J., Toft, T.: A practical implementation of secure auctions based on multiparty integer computation. In: Di Crescenzo, G., Rubini, A. (eds.) FC 2006. LNCS, vol. 4107, pp. 142–147. Springer, Heidelberg (2006)
7. Canetti, R.: Security and composition of multiparty cryptographic protocols. *Journal of Cryptology* 13(1), 143–202 (2000)
8. Chaum, D., Crepeau, C., Damgård, I.: Multiparty unconditionally secure protocols. In: 20th STOC, pp. 11–19 (1988)
9. Freedman, M.J., Ishai, Y., Pinkas, B., Reingold, O.: Keyword search and oblivious pseudorandom functions. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 303–324. Springer, Heidelberg (2005)
10. Goldreich, O.: *Foundations of Cryptography: Volume 2 – Basic Applications*. Cambridge Univ. Press, Cambridge (2004)
11. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game – A completeness theorem for protocols with honest majority. In: 19th STOC, pp. 218–229 (1987)

12. 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)
13. Goyal, V., Mohassel, P., Smith, A.: Efficient two party and multi-party computation against covert adversaries. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 289–306. Springer, Heidelberg (2008)
14. Harnik, D., Ishai, Y., Kushilevitz, E., Nielsen, J.B.: OT-Combiners via secure computation. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 393–411. Springer, Heidelberg (2008)
15. Hazay, C., Lindell, Y.: Oblivious transfer, polynomial evaluation and set intersection. Manuscript (2008)
16. Ishai, Y., Kilian, J., Nissim, K., Petrank, E.: Extending oblivious transfers efficiently. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 145–161. Springer, Heidelberg (2003)
17. 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)
18. Jarecki, S., Shmatikov, V.: Efficient two-party secure computation on committed inputs. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 97–114. Springer, Heidelberg (2007)
19. Kolesnikov, V., Schneider, T.: Improved garbled circuit: Free XOR gates and applications. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfssdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 486–498. Springer, Heidelberg (2008)
20. Lindell, Y., Pinkas, B.: An efficient protocol for secure two-party computation in the presence of malicious adversaries. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 52–78. Springer, Heidelberg (2007)
21. Lindell, Y., Pinkas, B.: A proof of security of Yao’s protocol for two-party computation. *Journal of Cryptology* 22, 161–188 (2009)
22. Lindell, Y., Pinkas, B., Smart, N.P.: Implementing two-party computation efficiently with security against malicious adversaries. In: Ostrovsky, R., De Prisco, R., Visconti, I. (eds.) SCN 2008. LNCS, vol. 5229, pp. 2–20. Springer, Heidelberg (2008)
23. Mohassel, P., Franklin, M.K.: Efficiency tradeoffs for malicious two-party computation. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T.G. (eds.) PKC 2006. LNCS, vol. 3958, pp. 458–473. Springer, Heidelberg (2006)
24. Malkhi, D., Nisan, N., Pinkas, B., Sella, Y.: Fairplay — a secure two-party computation system. In: Proc. of 13th USENIX Security Symposium (2004)
25. Naor, M., Pinkas, B., Sumner, R.: Privacy Preserving Auctions and Mechanism Design. In: Proc. of the 1st ACM conf. on Electronic Commerce (November 1999)
26. Nielsen, J.B., Orlandi, C.: LEGO for two party secure computation. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 368–386. Springer, Heidelberg (2009)
27. Peikert, C., Vaikuntanathan, V., Waters, B.: A framework for efficient and composable oblivious transfer. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 554–571. Springer, Heidelberg (2008)
28. Shamir, A.: How to share a secret. *Communications of the ACM* 11, 612–613 (1979)
29. Woodruff, D.: Revisiting the efficiency of malicious two-party computation. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 79–96. Springer, Heidelberg (2007)
30. Yao, A.: How to generate and exchange secrets. In: 27th FOCS, pp. 162–167 (1986)

Secure Multi-party Computation Minimizing Online Rounds

Seung Geol Choi^{1,*}, Ariel Elbaz^{1,*}, Tal Malkin^{1,*}, and Moti Yung²

¹ Columbia University
{sgchoi, arielbaz, tal}@cs.columbia.edu
² Google Inc. & Columbia University
moti@cs.columbia.edu

Abstract. Multi-party secure computations are general important procedures to compute any function while keeping the security of private inputs. In this work we ask whether preprocessing can allow low latency (that is, small round) secure multi-party protocols that are universally-composable (UC). In particular, we allow any polynomial time preprocessing as long as it is independent of the exact circuit and actual inputs of the specific instance problem to solve, with only a bound k on the number of gates in the circuits known.

To address the question, we first define the model of “Multi-Party Computation on Encrypted Data” (MP-CED), implicitly described in [FH96, JJ00, CDN01, DN03]. In this model, computing parties establish a threshold public key in a preprocessing stage, and only then private data, encrypted under the shared public key, is revealed. The computing parties then get the computational circuit they agree upon and evaluate the circuit on the encrypted data. The MP-CED model is interesting since it is well suited for modern computing environments, where many repeated computations on overlapping data are performed.

We present two different round-efficient protocols in this model:

- The first protocol generates k garbled gates in the preprocessing stage and requires only two (online) rounds.
- The second protocol generates a garbled universal circuit of size $O(k \log k)$ in the preprocessing stage, and requires only one (online) round (i.e., an obvious lower bound), and therefore it can run asynchronously.

Both protocols are secure against an active, static adversary controlling any number of parties. When the fraction of parties the adversary can corrupt is less than half, the adversary cannot force the protocols to abort.

The MP-CED model is closely related to the general Multi-Party Computation (MPC) model and, in fact, both can be reduced to each other. The first (resp. second) protocol above naturally gives protocols for three-round (resp. two-round) universally composable MPC secure against active, static adversary controlling any number of parties (with preprocessing).

Keywords: Computing with Encrypted Data, Multi-Party Computation, Public key Cryptography, Cryptographic Protocols, Universal Composition.

* Supported in part by NSF Grants CCF-0347839, CNS-0716245 and CNS-0831094.

1 Introduction

Secure Multi-party Computation (MPC). Protocols for MPC enable a set of parties to correctly evaluate a function such that no information about the private inputs of the parties is revealed, beyond what is leaked by the output of the function. This notion was first presented by Yao [Y86] for the two-party case, and by Goldreich et al. [GMW87] for the multi-party case. However, implementations for MPC are notoriously inefficient. Many protocols implementing them have delays associated with the depth of the circuit and even constant round protocols produce very long delays. The question that we want to settle in this work is whether one can use preprocessing computation in order to “be ready” once the inputs and the actual circuit (problem) to compute on are given. Note that the world of computing is transforming into “cloud services” where parties can “rent” computational resources. Thus, it may make sense to perform a lengthy preprocessing in the background, with no specific input and problem to solve in mind, just as a preparation. To this end cloud resources can be employed on behalf of users, and massive computations and communication can be performed. Then in the online stage once the input is given and the circuit determined, it can be performed much faster given the preprocessing. As long as at least one of the servers in the cloud is not corrupted, the correctness and privacy of the online stage computation is guaranteed.

We consider the following variation on secure multi-party computation, called *multi-party computing with encrypted data* (MP-CED): (1) The computing parties publish a shared public key, and hold shares of the matching private key. (2) The parties also know some bound on the circuit size that they will be required to compute securely. The parties then perform a preprocessing stage. For this stage too, we may try to minimize the parties’ work and computation rounds, but this is not the main goal, which is the efficiency of the on-line stage. (3) The input distribution is a database of encrypted data that can be published by many parties (not necessarily those taking part in the computation); i.e., think about the parties as a service (like the census bureau) computing on behalf of a larger population. (4) The concrete computation circuit (or circuits) is given, and the input to use from the database (their indices in the database) are determined. Then and only then (5) the parties are engaged in a short computation to achieve the task and produce the output while protecting the private data. Note that the input database may be reused for many computations.

We remark that our model is somewhat related to a multi-party extension of the model by Rivest, Adleman and Dertouzos [RAD78]. They put forth a scenario for secure computation over database of encrypted data, called *Computing with Encrypted Data* (CED). This model is highly attractive since it represents the case where a database is first collected and maintained and only later a computation on it is decided upon and executed (e.g., data mining and statistical database computation done over the encrypted database). We discuss the encrypted data model and the multi-party version here, and in fact show that MP-CED and MPC can be reduced to each other (shown in Section 3.3).

1.1 Motivation

We consider protocols in the universal composability (UC) framework introduced by Canetti [C01]. UC secure protocols remain secure even when executed concurrently

with arbitrary other protocols running in some larger network, and can be used as sub-routines of larger protocols in a modular fashion.

Round-Efficient Protocols with Preprocessing. Round complexity is an important criterion for the efficiency of an MPC protocol. A long line of work, including [BMR90, IK00, GIKR01, DI05, DI06, DIK⁺08], focused on reducing both the round complexity and communication complexity.

Also, it is known that UC secure computation of general functions is not possible in the plain model in the case of honest minority. In particular, UC secure two-party computation of a wide class of functionalities was ruled out by [CF01, CKL03]. To circumvent these impossibility results, it is common to assume some pre-computation setup, and the most common assumption is that a common reference string (CRS) is made available to the parties before the computation. Canetti et al. [CLOS02] showed that (under suitable cryptographic assumptions) a CRS suffices for UC secure MPC of any well-formed functionality.

In our work, we consider stronger relaxation on the setup, called general preprocessing [DI05]¹, in which the parties perform some work as long as it is independent of the inputs and the circuit for which the actual computation is to be done later. The main motivation for this model is to reduce the amount of work during the execution of the protocol beyond a preprocessing phase.

Considering the two aspects above, we ask the following natural question:

Allowing any polynomial time preprocessing (in some input parameter) before the circuit (whose size is bound by the same input parameter) and the inputs are known, is there a very small constant round protocol?

1.2 Our Results

We address the aforementioned question affirmatively by constructing two different round-efficient protocols for MP-CED, which we call \mathcal{P}_1 and \mathcal{P}_2 . Both protocols can be naturally transformed into round-efficient protocols for MPC (c.f. Section 3.3). Each protocol has its own advantage depending on the following parameters:

1. round complexity in the online stage (our major concern),
2. round complexity in the preprocessing stage, and
3. the number of gates constructed throughout the protocol.

In terms of online round complexity, protocol \mathcal{P}_1 is “two rounds” whereas that of protocol \mathcal{P}_2 is “one round” (which is optimal, since even non-secure computation need to collect the data and it takes one round). There are some cases, however, in which the preprocessing round complexity of \mathcal{P}_1 is better, under some efficiency considerations. We use general constant-round MPC protocols [IPS08] for the preprocessing stage in \mathcal{P}_2 , whereas in \mathcal{P}_1 we can use the protocol given in Appendix A, which requires *exactly* $2n$ rounds. When n is small enough, preprocessing in \mathcal{P}_1 can be more round-efficient

¹ Preprocessing in [DI05] is independent only of the inputs (it depends on the circuit to be evaluated), whereas we require preprocessing to be independent both of the circuit and of the inputs.

(when n is large, a general MPC protocol can be used in \mathcal{P}_1 , too). Also, the number of gates constructed in \mathcal{P}_2 is larger than that in \mathcal{P}_1 . To evaluate a circuit with up to k gates, \mathcal{P}_1 constructs k garbled gates in the preprocessing stage, as explained below. In contrast, \mathcal{P}_2 generates a universal circuit [V76] in the preprocessing stage, which is later used (in the online stage) to evaluate a given circuit. The smallest known universal circuit that can evaluate a circuit with k gates has $O(k \log k)$ gates [KS08]. We overview the two protocols in the following.

First Protocol (\mathcal{P}_1). In a big picture, we follow the framework of Yao’s garbled circuit technique. However, the main difference is that, in our protocol, garbling is done *on the individual gate level* so that this procedure can be executed in the preprocessing level independently of the circuit to be given and computed later. In the online stage, construction of wires between gates according to the given circuit is performed.

- In the preprocessing stage, the parties generate a ‘garbled’ truth table for each individual gate. Truth tables are for NAND gates, and they have four rows and three columns – left-input, right-input, and output. Each row is randomly shuffled, and each element is an encryption of Boolean value. We emphasize that no party knows anything more than the fact that it’s a randomly shuffled encrypted table for NAND.

In addition, a fresh pair of public key and (encrypted) private key is generated for each row. This key is used for constructing encrypted wiring information in the online stage, when the circuit is given.

- In the online stage, given the encrypted data and a circuit, the computing parties ‘connect’ truth tables by adding *wiring information*. The wiring information tells, given two tables T_{pred}, T_{succ} according to the topology of the circuit, which row of T_{pred} ’s output column is equal to which row of T_{succ} ’s input column. We note that this information should be carefully revealed; otherwise, the adversary may try computing different rows of the truth tables using the wirings, and may learn more than is allowed. In fact, during the computation (online stage), exactly one row’s wirings for each table should be revealed.

To enable such wirings we introduce *Multi-Party Conditional Oblivious Decryption Exposure* (M-CODE) (in Section 2), which is a multi-party extension to the CODE functionality, introduced in [CEJ⁺07] for the two party case. M-CODE assumes a group of parties share a secret key x of a public key y . Three ciphertexts c_{out}, c_{in}, c_{key} — all encrypted under y — and a new public key z are given as input. For $\ell \in \{out, in, key\}$, let m_ℓ be the plaintext encrypted in c_ℓ . If m_{out} equals m_{in} , M-CODE outputs $E_z(m_{key})$. Otherwise, M-CODE chooses a random value r and outputs $E_z(r)$. The computing parties use M-CODE such that, for each row of a truth table, the three ciphertexts of the M-CODE are (1) output value of the previous table (2) the input value of this row and (3) the secret key for this row. We refer the reader to Section 3.1 for more details.

With two round implementation of M-CODE for ElGamal encryption, we obtain a two-round protocol for MP-CED and a three-round protocol for MPC.

Theorem 1. *Assuming the DDH assumption holds, protocol \mathcal{P}_1 is a two-round UC secure protocol for MP-CED in the \mathcal{F}_{zk} hybrid — and, thereby three-round UC secure protocol for MPC in the \mathcal{F}_{zk} hybrid in the general preprocessing model — against an active and static adversary as long as at most $t < n$ computing parties are corrupted.*

The protocols manipulate linear number of gates in the circuit size. Furthermore, if $t < n/2$ parties are corrupted, \mathcal{P}_1 is robust against abort.²

Second Protocol (\mathcal{P}_2). Protocol \mathcal{P}_2 follows Yao’s garbled technique more closely than \mathcal{P}_1 . However, the circuit that is to be garbled is a universal circuit [V76, KS08] to maintain independence of the circuit to be given. Optimal round complexity in the online stage is achieved by putting a simple constraint on the input-layer labels in the garbled circuit and by employing the multiplicative homomorphism of ElGamal encryption. As in the first protocol, a group of parties share a secret key x of a public key y .

- In the preprocessing stage, the parties generate a garbled circuit [Y86] of a universal circuit C_U , with some special restrictions on keys of input wires. In the garbled circuit C_U , there are two keys w_0^i and w_1^i for each wire i , where w_b^i corresponds to the wire carrying bit b (see Section 3.2 for more detail). The special restriction on input wires is that $w_1^i/w_0^i = h$ for a random global value h unknown to any party. The two keys can be constructed by picking w_0^i uniformly at random and letting $w_1^i = h \cdot w_0^i$. In addition to the garbled circuit of C_U , the following encryptions are generated: (1) the encryption $E_y(h)$ and (2) $E_y(w_0^i)$ for each input wire i . Construction of a garbled circuit along with aforementioned encryptions — i.e., $E_y(h)$ and $E_y(w_0^i)$ ’s — can be performed using a constant-round UC secure protocols for general MPC [KOS03, IPS08]. Input contribution of a bit 0 is done by $E_y(h^0)$, and for a bit 1, re-encrypted $E_y(h^1)$ is used via homomorphism.
- In the online stage, for each input wire i where a bit b is the contributed input for the wire, computing parties obtain w_b^i . The encryption $E_y(w_b^i)$ can be obtained via homomorphism given the encrypted input $c_i = E_y(h^b)$, giving $E_y(w_b^i) \cdot c_i = E_y(w_b^i h^b) = E_y(w_b^i)$, since $w_1^i = h \cdot w_0^i$. Now parties obtain the key w_b^i for each input wire i using threshold decryption and can locally evaluate the garbled circuit. Note that w_b^i does not leak any information on b since it’s randomly distributed (with w_{1-b}^i hidden).

Theorem 2. *Assuming the DDH assumption holds, protocol \mathcal{P}_2 is a one-round UC secure protocol for MP-CED in the \mathcal{F}_{zk} hybrid – and, thereby two-round UC secure protocol for MPC in the \mathcal{F}_{zk} hybrid in the general preprocessing model – against an active and static adversary as long as at most $t < n$ computing parties are corrupted. The protocol processes $k \log k$ gates where k is the circuit size. Furthermore, if $t < n/2$ parties are corrupted, \mathcal{P}_2 is robust against abort.*

1.3 Related Work

Round Complexity. Beaver et al. [BMR90] showed the first MPC protocol that required constant (but large) number of rounds, and Damgård and Ishai [DI05] presented the first adaptively UC secure protocol that achieves two rounds in the (linear) preprocessing model when the number of malicious parties $t < n/5$ and some higher constant rounds when $t < n/2$. Recently, Ishai et al. constructed UC secure protocol with malicious majority in the OT hybrid model running in (large) constant rounds [IPS08] (see Fig 1).

² Instantiation of protocol \mathcal{P}_1 (in particular, key setup in the preprocessing) is parameterized by t . Therefore protocol \mathcal{P}_1 is not a ‘best-of-both-worlds’ protocol [KLP06]. This is true of \mathcal{P}_2 , too.

MPC	circuit	rounds	security	#corr
[KOS03]	B	$O(1)$	St	$t < n$
[DI05]	B	2	Ad	$t < n/5$
[DI05, DI06]	B	$O(1)$	Ad	$t < n/2$
[DIK ⁺ 08]	B	$O(1)$	Ad	$t < n/2$
[IPS08]	B	$O(1)$	Ad	$t < n$
\mathcal{P}_1	B	3	St	$t < n$
\mathcal{P}_2	B	2	St	$t < n$

MP-CED	circuit	rounds	security	#corr
[CDN01]	Ar	$O(d)$	SA, St	$t < n/2$
[JJ00]	B	$O(n+d)$	SA, St	$t < n/2$
[DN03]	Ar	$O(d)$	UC, Ad	$t < n/2$
\mathcal{P}_1	B	2	UC, St	$t < n$
\mathcal{P}_2	B	1	UC, St	$t < n$

#corr = number of corrupted parties, B = Boolean, Ar = Arithmetic, St = static, Ad = adaptive, SA = stand-alone

Fig. 1. UC Secure Constant-Round MPC Protocols (Left) and MP-CED Protocols (Right). We denote by d the depth of a given circuit, by n the number of parties, and by t the number of corrupted parties. \mathcal{P}_1 and \mathcal{P}_2 denote the protocols proposed here. Here the column 'rounds' means the number of rounds in the online stage.

For the two-party setting, which is a special case of MPC, Katz and Ostrovsky [KO04] showed that it's impossible to construct a secure protocol running in four rounds using enhanced trapdoor permutation (eTDP) or homomorphic encryption in a black-box manner in the plain model, and they constructed a five-round protocol. To overcome this lower bound, Horvitz and Katz [HK07] used CRS to construct a UC secure two-party protocol in two rounds. Nielsen and Orlandi [NO09] gave a two party protocol using a cut-and-choose approach. In a big picture, their idea is somewhat similar to ours: after many garbled gates are generated, they are connected to each other according to the circuit to be evaluated.

In the (non-UC) stand-alone setting, the work of [KK00, AIK05] gave a general non-interactive reduction of any n -party functionality computed by a polynomial size Boolean circuit into a (possibly randomized) functionality of degree-3 over $GF(2)$. Combining this reduction with any secure protocol with malicious majority (for example, [GMW87]) leads to round-efficient protocols in the stand-alone setting.

MP-CED. Some nontrivial instantiations for CED were shown, originating with Sander et al. [SY99], who gave a protocol for circuits in NC^1 . Beaver [B00] extended this result to accommodate any function in $NLOGSPACE$ [BL96]. Recently, Gentry presented a construction for any polynomial size circuit by showing doubly-homomorphic encryption scheme from ideal lattices [G09], however it is not yet clear if this can give efficient protocols for MP-CED (see discussion in Section 3.3).

MP-CED was also considered by Franklin and Haber [FH96] and the subsequent works [JJ00, CDN01, DN03]. In their works, after a threshold encryption key is established, each party broadcasts the encryption of its input, and the parties evaluate the circuit on the encrypted data. However, they do not explicitly treat the setting as a unique model for MP-CED, with a specific setup state that is independent of the inputs and the circuits to be computed, and do not consider *input separation* – inputs can be contributed by parties that do not take part in the computation. Note that all these previous works in the model dealt with the two party case, which we extend herein to the multi-party case.

The protocol given by Cramer et al. [CDN01] computes an arithmetic circuit and achieves security in the case of honest majority, but the number of rounds is linear in the depth of the circuit. A UC adaptively secure protocol with the same round complexity was given by [DN03]. Jakobson and Juels [JJ00] use mix-and-match approach to compute on encrypted data, but their approach requires even more rounds (linear in the sum of the depth of the circuit and the number of parties). Figure 1 lists these previous works, in some relations to our protocols (while concentrating on on-line rounds, and omitting some of the advantages our results has beyond the table).

2 Preliminaries

For any integer t , let $[t] = \{0, 1, \dots, t - 1\}$. Let k be a security parameter. We choose a cyclic group \mathcal{G}_g^q of order $q \approx 2^k$ with a generator g where the DDH problem [DH76] is hard. For example, \mathcal{G}_g^q can be a subgroup of order q of a multiplicative group Z_p^* for a safe prime $p = 2q + 1$, i.e., $\mathcal{G}_g^q = \{g^0, g^1, \dots, g^{q-1}\} \pmod{p}$. We assume \mathcal{G}_g^q is known in advance.

ElGamal Encryption. ElGamal encryption [E85] is semantically secure under the DDH assumption over \mathcal{G}_g^q [TY98]. *The key generation algorithm* generates a public/secret key pair (y, x) where $x \in_R [q]$ and $y = g^x$. *Encryption* of a message $m \in \mathcal{G}_g^q$ under a public key y , denoted by $E_y(m)$, is (g^r, my^r) where $r \in_R [q]$. *Decryption* of a ciphertext $c = (\alpha, \beta)$ with the secret key x , denoted by $D_x(c)$, is β/α^x .

Homomorphism. Multiplication of two ciphertexts $E_y(m_1) = (g^{r_1}, m_1 y^{r_1})$ and $E_y(m_2) = (g^{r_2}, m_2 y^{r_2})$ is defined as $(g^{r_1+r_2}, m_1 m_2 y^{r_1+r_2})$, which shows the homomorphism of ElGamal encryption (i.e., $E_y(m_1) \cdot E_y(m_2) = E_y(m_1 \cdot m_2)$). In addition, encryption keys are also homomorphic in the sense that given key pairs $\{(y_i = g^{x_i}, x_i)\}_i$, the pair $(\prod_i y_i, \sum_i x_i)$ is a valid key pair. When two ciphertexts encrypt the same message, we denote $c_1 \equiv c_2$.

Zero-Knowledge Proofs of Knowledge (ZK-PoK). A proof of knowledge is a proof for a relation R , in which the prover convinces the verifier that an instance is in the language, and also that *the prover knows a witness for this instance*. We will use standard notation to denote proofs of knowledge related to discrete log. For example, $PK\{b : a = g^b\}$ denotes a proof of knowledge where the prover convinces the verifier that she knows the value of b , such that $a = g^b$, when a is known to both.

In the common reference string (CRS) model, we can use non-interactive zero-knowledge proofs (NIZK) due to De Santis et al. [SCO⁺01] (see the discussion in [CLOS02, Section 6]) which is UC-secure [C01]. In the random oracle model (ROM), the above proof systems can be efficient NIZK using the standard Fiat-Shamir technique [FS86] combined with OR proofs of Σ -protocols [CDS94].

Secret Sharing [S79, F87]. A secret sharing scheme allows a secret $s \in [q]$ to be shared among n parties, such that a threshold of $t + 1$ parties can recover the secret, whereas any smaller set of parties can not learn anything about the secret. In Shamir's secret sharing scheme, the shares are values of a degree- t polynomial, and the secret is the free coefficient of the polynomial.

We show below how the parties can share and recover the secret s . Moreover, the parties may choose to recover d^s for some $d \in \mathcal{G}_g^q$, or an ElGamal encryption of d^s (without learning anything about the secret s).

- *Sharing*: A dealer chooses at random a degree t polynomial $Q(x) := s + a_1x + \dots + a_t x^t$, where the free coefficient is the secret s . The share of party P_i is $s_i = Q(i)$.
- *Recovering s* : Let T be a set of $t + 1$ parties. They evaluate $Q'(0) = \sum_{i \in T} s_i L_i(0)$ to recover s , where L_i is a Lagrangian on the points in T ³.
- *Recovering an exponentiation d^s* : Similar to above, the parties can evaluate $d^s = d^{Q'(0)} = d^{\sum_{i \in T} s_i L_i(0)} = \prod_{i \in T} d^{s_i L_i(0)}$, using only $\{d^{s_i}\}_{i \in T}$.
- *Recovering $E_y(d^s)$* : Using multiplicative homomorphism of ElGamal, the parties evaluate $E_y(d^s) = E_y(d^{Q'(0)}) = \prod_{i \in T} E_y(d^{s_i L_i(0)}) = \prod_{i \in T} E_y(d^{s_i})^{L_i(0)}$, using only $\{E_y(d^{s_i})\}_{i \in T}$.

Multi-party Conditional Oblivious Decryption Exposure (M-CODE). We introduce *Multi-Party Conditional Oblivious Decryption Exposure* (M-CODE). M-CODE assumes a group of parties share a secret key x of a public key y . Three ciphertexts c_{out}, c_{in}, c_{key} — all encrypted under y — and a new public key z are given as input. For $\ell \in \{out, in, key\}$, let m_ℓ be the plaintext encrypted in c_ℓ . If m_{out} equals m_{in} , M-CODE outputs $E_z(m_{key})$. Otherwise, M-CODE chooses a random value r and outputs $E_z(r)$. A variant of this functionality for the two party case was initially introduced by [CEI⁺07]. The intuitive idea is to generate a ciphertext that encrypts m_{key} multiplied by $(m_{out}/m_{in})^r$ for a random r . If $m_{in} = m_{out}$, then the output would be m_{key} . We assume party P_i has x_i , all the parties know $c_{out}, c_{in}, c_{key}, z, (y, y_1 = g^{x_1}, \dots, y_n = g^{x_n})$, and let $c_{out} = E_y(m_{out}) = (\alpha, \beta), c_{in} = E_y(m_{in}) = (\gamma, \delta), c_{key} = E_y(m_{key}) = (\lambda, \mu)$. The protocol for M-CODE proceeds as follows:

1. Each party P_i chooses $e_i \in_R [q]$, and computes $\epsilon_i = (\alpha/\gamma)^{e_i}, \zeta_i = (\beta/\delta)^{e_i}, \pi_i = PK\{e_i : \epsilon_i = (\alpha/\gamma)^{e_i}, \text{ and } \zeta_i = (\beta/\delta)^{e_i}\}$, and broadcasts $(\epsilon_i, \zeta_i, \pi_i)$.
2. Let $\epsilon = \prod_{i \in S_1} \epsilon_i$ and $\zeta = \prod_{i \in S_1} \zeta_i$ where S_1 is the set of parties which sent valid messages. Each party P_i chooses r_i randomly and computes $d_i = (d_{i1}, d_{i2}) = E_z((\epsilon\lambda)^{x_i})$ and $\psi_i = PK\{(r_i, x_i) : d_{i1} = g^{r_i}, d_{i2} = z^{r_i}(\epsilon\lambda)^{x_i}, y_i = g^{x_i}\}$, and broadcasts (d_i, ψ_i) .
3. Let S_2 be the set of parties that sent valid messages in steps 1 & 2. If $|S_2| \leq t$, then the protocol aborts. Each party P_i , using the homomorphic multiplication, computes $d = (d_1, d_2) = E_z((\epsilon\lambda)^x) = \prod_{j \in S_2} d_j^{L_j(0)}$ where $L_j(\cdot)$ is a Lagrangian on the indices in S_2 . P_i uses homomorphic operations to compute $E_z(m_{key}) = (1/d_1, \zeta\mu/d_2)$, which is

$$E_z(\zeta\mu/(\epsilon\lambda)^x) = E_z\left(\left(\frac{\beta/\alpha^x}{\delta/\gamma^x}\right)^e \cdot (\mu/\lambda^x)\right) = E_z\left(\left(\frac{m_{out}}{m_{in}}\right)^e \cdot m_{key}\right),$$

where $e = \sum_{i \in S_1} e_i$.

³ Lagrangian L_i on the points in T is a degree t polynomial such that $L_i(x) = 1$ if $x = i$ and $L_i(x) = 0$ if $x \in T$ and $x \neq i$. The polynomial $Q'(x) = \sum_{i \in T} s_i L_i(x)$ is a degree t polynomial that goes through the points $(i, s_i)_{i \in T}$, and thus must be $Q(x)$.

3 Multi-party Computing with Encrypted Data

We assume the circuit C of interest is normalized: all intermediate gates are NAND gates, and output gates are IDENTITY gates⁴. We can easily attain this circuit by adding another layer of IDENTITY gates on top of a circuit that consists of NAND gates.

3.1 First Protocol (\mathcal{P}_1)

In the first protocol, called \mathcal{P}_1 , each gate is garbled, and then the computing parties ‘connect’ gates by adding *wiring information* using M-CODE.

Preprocessing Stage. The first step is to establish a global public key y for ElGamal encryption. The computing parties have shares of the corresponding secret key x . Once the public key is established, the next step is to generate truth tables for individual gates. The columns of input, output, and intermediate gates differ slightly, as can be seen in Figure 2 which shows the structure of truth tables.

1. Input and Output. These are encrypted with the global public key y .
2. Placeholders for the wiring information. This connects a row of the truth table to matching rows in successor gates.
3. The columns PK and SK contain a random ElGamal key pair, where the private key is encrypted under the global public key y (and the wiring information is encrypted using the secret keys in SK).
4. For output gates, ciphertexts in column Final encrypt the same plaintexts as ciphertexts in column In.

During the preprocessing stage, the parties can generate polynomial number of garbled gates, that can later be used for evaluating circuits. Therefore it suffices to know a bound on the sizes of circuits to be evaluated later. Preprocessing can be done in constant number of round using general MPC protocols [KOS03, IPS08]. If the number of computing parties is small, it can be done explicitly in $2n$ rounds, where n is the number of computing parties, using the protocol in Appendix A.

Input contribution is performed by publishing a ciphertext $c = (c_1, c_2) = E_y(g^b)$ for an input $b \in \{0, 1\}$. This can be done securely by adding $PK\{r : (c_1 = g^r, c_2 = y^r) \text{ or } (c_1 = g^r, c_2 = gy^r)\}$.

Online Stage: Generation of Wires Between Garbled Gates. In Figure 2, G_i is the left predecessor of G_k . The connection between the two gates should be established through some “wiring” such that during the computation the output of G_i can be propagated to the left input of G_k . So, rows of T_i with output value $b \in \{0, 1\}$ should be connected to rows of T_k with left input value b .

Requirements for Wiring. In our protocol, the following conditions are considered in generating wires.

⁴ An IDENTITY gate has single input bit (wire) and output bit, and it copies the input bit value to its output.

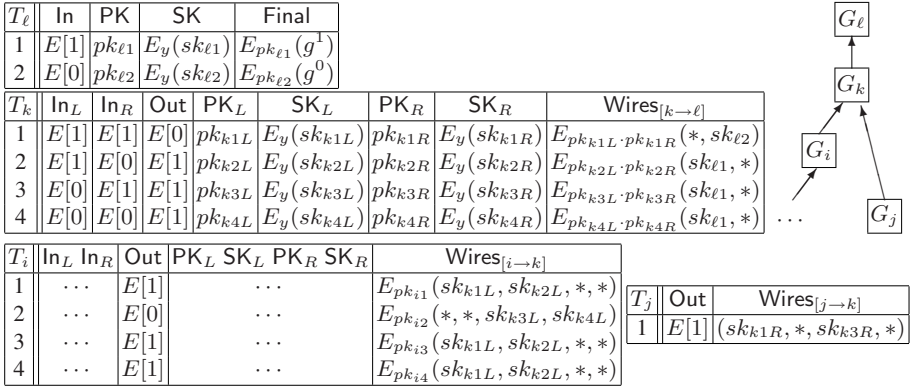


Fig. 2. Garbled Truth Tables for the Gates (G_i, G_j, G_k, G_ℓ). The topology of the gates is given on the right. G_j is an input gate, G_ℓ is an output gate, and G_i, G_k are intermediate gates. Table T_x is the truth table describing gate G_x . y is the global public key. Each row of an intermediate truth table has two sets of (secret, public) keys, and contains the wiring information, “connecting” it to the next gate, encrypted using these two keys. $E[0]$ and $E[1]$ are $E_y(g^0)$ and $E_y(g^1)$ respectively. In table T_i , $pk_{i1} = pk_{i1L} \cdot pk_{i1R}$, and pk_{i2}, \dots, pk_{i4} are defined similarly. In the Wires columns, $E(a, b, c, d)$ denotes concatenation of $E(a), \dots, E(d)$.

- (Encrypting the Wiring Information.) The wiring information, except wirings connecting an input gate to an intermediate gate, should be encrypted. Public wiring may help the (even semi-honest) adversary to learn more information than the output of C . Therefore, it is encrypted with the public key stored in columns PK_L and PK_R .
- (Conditional Exposure of Wiring Information.) For the computation to proceed, the protocol should reveal the wiring information for the rows along the computational path. In the beginning, wirings from input gates is public. Along the computational path, on each gate, exactly one row should allow decryption of the wiring information.
- (Oblivious Generation of Wiring Information.) The wiring information are added to garbled gates after they are built. It is essential that, even if the truth table is encrypted and shuffled, the parties should still be able to add the wiring information.

Computation of a Circuit Using Wires. Let $T_i[a][b]$ denote the element located at column a and row b in T_i . The column Wires contains wiring information, and we denote the column Wires from T_i to T_k by $Wires_{[i \rightarrow k]}$ ⁵. Looking at the column Wires alone, $Wires(v)$ denotes the v th row of this column in the plaintext form. For example, $Wires_{[i \rightarrow k]}(2) = (*, *, sk_{k3L}, sk_{k4L})$ in Figure 2. We also use $Wire(v, w)$ to denote the w th element of $Wires(v)$. If $Wire_{[i \rightarrow k]}(v, w) \neq *$, it means that $T_i[Out][v] \equiv T_k[In][w]$. In Figure 2, for example, we have $Wire_{[i \rightarrow k]}(2, 3) \neq *$ because $T_i[Out][2] \equiv T_k[In_L][3] \equiv E[0]$.

This wiring information helps the circuit computation to proceed correctly. The computation proceeds in order from input gates to output gates. In Figure 2, for example, if

⁵ If G_i has another outgoing wire, say to G_m , T_i will have another column $Wires_{[i \rightarrow m]}$.

Connecting the Gates: Fill in Wires Columns.

- For every $\text{Wire}_{[i \rightarrow k]}(v, w)$ of an intermediate gate T_i , run M-CODE for $c_{out} = T_i[\text{Out}][v]$, $c_{in} = T_k[\text{In}][w]$ and $c_{key} = T_k[\text{SK}][w]$, with the key $z = T_i[\text{PK}_L][v] \cdot T_i[\text{PK}_R][v]$.
- For every $\text{Wire}_{[j \rightarrow k]}(v, w)$ of an input gate T_j , run M-CODE for $c_{out} = T_j[\text{Out}][v]$, $c_{in} = T_k[\text{In}][w]$ and $c_{key} = T_k[\text{SK}][w]$, with the trivial key $z = g^0$.

Depending on the circuit topology, the subscript of a column may differ (e.g., In_L , In_R , or In).

Local Computation. Each party computes the output of C using the M-CODE transcripts on the input gates.

Fig. 3. Online Stage of \mathcal{P}_1

row 2 of T_i and row 1 of T_j are on the computation path, then row 3 of T_k is also on the computation path because $w = 3$ is the only row where $\text{Wire}_{[i \rightarrow k]}(2, w) \neq *$ and $\text{Wire}_{[j \rightarrow k]}(1, w) \neq *$.

Constructing Wires. We implement each $\text{Wire}_{[i \rightarrow k]}(v, w)$ using a M-CODE transcript for $c_{out} = T_i[\text{Out}][v]$, $c_{in} = T_k[\text{In}][w]$, $c_{key} = T_k[\text{SK}][w]$, and $z = T_i[\text{PK}][v]$ ⁶. This directly satisfies the requirements of encrypted wiring and oblivious wiring generation. Conditional exposure is achieved by executing M-CODE protocols in the input layer with a trivial public key $z = 1$, so that the wiring information in the input layer is known to every party.

The description of \mathcal{P}_1 can be found in Figure 3. Running the online stage takes two rounds. The communication complexity of \mathcal{P}_1 is $\mathcal{O}(nk|C|)$ (plus the NIZK, if we assume the CRS case) where $|C|$ is the size of the circuit.

3.2 Second Protocol (\mathcal{P}_2)

The idea of \mathcal{P}_2 is that in a preprocessing stage, the parties generate a garbled circuit, using Yao’s technique, of a universal circuit. The garbled circuit has a restriction on the keys of input wires, that allows the online computation to take only one round in our model, as opposed to the two-round OT based approach of Yao. The preprocessing stage can be done in constant number of rounds, using general MPC protocols [KOS03, IPS08].

Preprocessing Stage: Garbling Universal Circuit. The first step is to establish a global public key y for ElGamal encryption. The computing parties have shares of the corresponding secret key x . In contrast to protocol \mathcal{P}_1 , however, here, ElGamal encryption is used only for input layer.

Next, a garbled circuit for *universal circuit* is generated, using Yao’s garbled circuit technique [Y82]. In the generation procedure, for each wire i , two random keys, w_0^i and w_1^i are generated. The key w_0^i (resp., w_1^i) represents 0 (resp., 1) for wire i . For each gate G_j , a truth table T_j is generated. In each table, a private key encryption (denoted

⁶ Depending on the circuit topology, if this is a left input or right input to the gate, the pair (c_{in}, c_{key}) may also be $(T_b[\text{In}_L][w], T_b[\text{SK}_L][w])$ or $(T_b[\text{In}_R][w], T_b[\text{SK}_R][w])$.

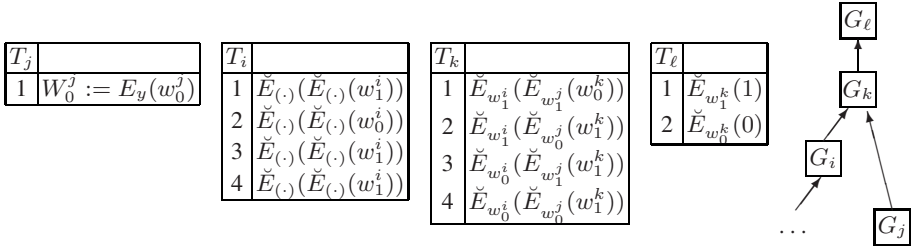


Fig. 4. Garbled Truth Tables for the Gates (G_i, G_j, G_k, G_ℓ). The topology of the gates is given on the right. G_j is an input gate, G_ℓ is an output gate, and G_i, G_k are intermediate gates. Table T_x is the truth table describing gate G_x . y is the global public key. Encryption \check{E} is a private key encryption based on pseudorandom function with efficient verifiable range [LP09].

$\check{G}, \check{E}, \check{D}$) with efficiently verifiable range (based on pseudorandom function) is used [LP09]. Figure 4 shows the structure of the garbled circuit.

- Recall that we assume all output gates are identity gates, with only one incoming wire and only two rows in the corresponding truth table. Each row encrypts the Boolean value represented by the corresponding wire, and the rows are randomly shuffled. An example is given in Figure 4 in the first row of table G_ℓ , the input value is 1 (the key w_1^k represents 1), and it encrypts 1, which is the output value of this row.
- For all other gates, each gate has two incoming wires and four rows. Each row encrypts a key for the outgoing wire, which represents the appropriate Boolean value of NAND of the incoming wires' values, and the rows are randomly shuffled. For example, in G_k of Figure 4 the first row encrypts w_0^k , representation of 0 for wire k , since NAND of the values that the keys of the incoming wires represent (i.e., the value 1 represented by w_1^i in wire i , and the value 1 represented by w_1^j in wire j) is 0.

To construct a secure protocol for MP-CED, we depart from the traditional Yao garbled circuit technique, by giving restriction on input wires.

- A random element $h \in \mathcal{G}_g^q$ is chosen, which no party knows, and $H = E_y(h)$ is published. We emphasize that H is generated *once and for all*. In other words, every instance of garbled universal circuit can use the same H .
- For input wire j , two keys $w_0^j, w_1^j \in \mathcal{G}_g^q$ are randomly generated, conditioned on $w_1^j = h \cdot w_0^j$. Only the encryption of the first key, $W_0^j := E_y(w_0^j)$ is published.

Since we garble a universal circuit, it suffices to know a bound on the sizes of circuits to be evaluated later. A universal circuit of size $O(k \log k)$ can accept circuits of size k as inputs [KS08].

Input contribution is performed such that for input $b \in \{0, 1\}$, a ciphertext $c = (c_1, c_2) = E_y(h^b)$ is published.

⁷ Roughly speaking, in such an encryption scheme, given a ciphertext and a key, it is efficiently verifiable whether the given ciphertext was encrypted under the given key. This helps computing parties to correctly compute the garbled circuit.

- When input is 0, publish $E_y(1)$.
- When input is 1, publish a re-encryption of H (recall $H = (H_1, H_2) = E_y(h)$).

A proof of knowledge is added, $PK\{r : (c_1 = g^r, c_2 = y^r) \text{ or } (c_1 = H_1g^r, c_2 = H_2y^r)\}$.

Online Stage: Obtaining keys for input-wires. Computing parties need to obtain a key, for each wire j , that represents the Boolean value b that the corresponding input ciphertext encrypts – that is, w_b^j . But the key should not leak any information about the input ciphertext. Our protocol meets such requirement by using homomorphism of ElGamal encryption⁸. Let c_j be the ciphertext of contributed input $b \in \{0, 1\}$ for input wire j . Computing parties work as follows:

- For every input wire j , compute $W^j = W_0^j \cdot c_j$ locally using homomorphism of ElGamal encryption. Then, decrypt W^j via threshold decryption by computing parties using their shares for x . This gives w_b^j , which matches the input b .
- Each party computes the output of C using the key w_b^j locally.

Running the online stage in \mathcal{P}_2 takes one round. The communication complexity of \mathcal{P}_2 is $\mathcal{O}(nk|C| \log |C|)$ (plus the NIZK, if we assume the CRS case) where $|C|$ is the size of the circuit.

3.3 Discussion

MP-CED vs. MPC with Preprocessing. General MPC and MP-CED can be reduced to each other.

- Given a protocol π for MP-CED, we can construct a protocol π' for MPC with preprocessing, as follows. In the preprocessing stage of π' , the parties share an encryption key. In the online stage of π' , each party publishes encryption of its input under the shared key, and the parties follow protocol π . The resulting MPC protocol π' requires one more online rounds than the underlying protocol π . This approach is implicitly used in [FH96, JJ00, CDN01, DN03].
- Given a protocol π' for MPC, we can construct a protocol π for MP-CED, as follows. In MP-CED, the parties share a secret key, and the inputs are encrypted. Protocol π should compute C on these given input ciphertexts. This can be done by the parties running protocol π' using a circuit C' derived from C . Circuit C' consists of two stages: the first stage of C' gets shares of the secret key and the ciphertexts, and decrypts the ciphertexts to give plaintexts. The second stage of C' essentially evaluates C on these plaintext inputs from the first stage. In running the protocol π , each party's input is its share for the secret key. Circuit C' has more gates than C . However, if the round complexity of π' does not depend on the depth of the circuit, then the round complexity of π is the same as the round complexity of π' .

On Basing MP-CED on Doubly-Homomorphic Encryption. Recently, Gentry constructed a doubly homomorphic encryption scheme using ideal lattices [G09], which

⁸ In fact, any homomorphic encryption can be used. We chose to use ElGamal encryption since it is already used in \mathcal{P}_1 .

solves the CED problem. Since our goal is to give a round-efficient protocol, it is an interesting question whether doubly-homomorphic encryption allows non-interactive secure computation. However, this seems unlikely.

- (Threshold Decryption.) It's not known whether Gentry's scheme supports threshold decryption. Thus, there has to be at least one party which can decrypt ciphertexts by itself. If this party sees the inputs (which are encrypted and published in the MP-CED model), it can decrypt private inputs of other parties and break the security. Thus, there must be a separation between parties who can decrypt and parties who get access to the input and intermediate ciphertexts.
- (Malicious Parties.) Parties without decryption capability would compute a circuit on encrypted inputs using double homomorphism. In order for the protocol to compute output in a plaintext form, they have to submit some ciphertexts to a party with decryption capability. In the malicious setting, to make sure that they applied doubly homomorphism correctly, some kind of zero-knowledge proof should be added to the ciphertexts they submit. However, it is not clear how such a proof can be constructed when the verifier has the decryption capability – as mentioned above, it must not see the input ciphertexts.

The above issue also stands against achieving MPC protocols against an active adversary with doubly homomorphic encryptions.

References

- [AIK05] Applebaum, B., Ishai, Y., Kushilevitz, E.: Computationally private randomizing polynomials and their applications. In: IEEE Conference on Computational Complexity, pp. 260–274 (2005)
- [B00] Beaver, D.: Minimal-latency secure function evaluation. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 335–350. Springer, Heidelberg (2000)
- [BL96] Boneh, D., Lipton, R.: Algorithms for black-box fields and their application to cryptography. In: Kobitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 283–297. Springer, Heidelberg (1996)
- [BMR90] Beaver, D., Micali, S., Rogaway, P.: The round complexity of secure protocols (extended abstract). In: Proc. 22nd Annual ACM Symposium on Theory of Computing (STOC), pp. 503–513 (1990)
- [C01] Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: Proc. 42nd IEEE Symposium on Foundations of Computer Science (FOCS), pp. 136–145 (2001)
- [CDN01] Cramer, R., Damgård, I., Nielsen, J.B.: Multiparty computation from threshold homomorphic encryption. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 280–299. Springer, Heidelberg (2001)
- [CDS94] 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)
- [CEJ⁺07] Choi, S.G., Elbaz, A., Juels, A., Malkin, T., Yung, M.: Two-party computing with encrypted data. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 298–314. Springer, Heidelberg (2007)
- [CF01] Canetti, R., Fischlin, M.: Universally composable commitments. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 19–40. Springer, Heidelberg (2001)

- [CKL03] Canetti, R., Kushilevitz, E., Lindell, Y.: On the limitations of universally composable two-party computation without set-up assumptions. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 68–86. Springer, Heidelberg (2003)
- [CLOS02] Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: Proc. 34th Annual ACM Symposium on Theory of Computing (STOC), pp. 494–503 (2002)
- [DH76] Diffie, W., Hellman, M.E.: New directions in cryptography. *IEEE Trans. on Information Theory* IT-22(6), 644–654 (1976)
- [DI05] Damgård, I., Ishai, Y.: Constant-round multiparty computation using a black-box pseudorandom generator. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 378–394. Springer, Heidelberg (2005)
- [DI06] Damgård, I., Ishai, Y.: Scalable secure multiparty computation. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 501–520. Springer, Heidelberg (2006)
- [DIK⁺08] Damgård, I., Ishai, Y., Krøigaard, M., Nielsen, J.B., Smith, A.: Scalable multiparty computation with nearly optimal work and resilience. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 241–261. Springer, Heidelberg (2008)
- [DN03] Damgård, I., 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)
- [E85] ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory* 31, 469–472 (1985)
- [F87] Feldman, P.: A practical scheme for non-interactive verifiable secret sharing. In: Proc. 28th IEEE Symposium on Foundations of Computer Science (FOCS), pp. 427–437 (1987)
- [FH96] Franklin, M.K., Haber, S.: Joint encryption and message-efficient secure computation. *jac*. 9(4), 217–232 (1996)
- [FS86] Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Massey, J.L. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987)
- [G09] Gentry, C.: Fully homomorphic encryption using ideal lattices. In: STOC (to appear, 2009)
- [GIKR01] Gennaro, R., Ishai, Y., Kushilevitz, E., Rabin, T.: The round complexity of verifiable secret sharing and secure multicast. In: Proc. 33rd Annual ACM Symposium on Theory of Computing (STOC), pp. 580–589 (2001)
- [GMW87] Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game. In: Proc. 19th Annual ACM Symposium on Theory of Computing (STOC), pp. 218–229. ACM Press, New York (1987)
- [HK07] Horvitz, O., Katz, J.: Universally-composable two-party computation in two rounds. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 111–129. Springer, Heidelberg (2007)
- [IK00] Ishai, Y., Kushilevitz, E.: Randomizing polynomials: A new representation with applications to round-efficient secure computation. In: FOCS, pp. 294–304 (2000)
- [IKLP06] Ishai, Y., Kushilevitz, E., Lindell, Y., Petrank, E.: On combining privacy with guaranteed output delivery in secure multiparty computation. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 483–500. Springer, Heidelberg (2006)
- [IPS08] 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)
- [JJ00] Jakobsson, M., Juels, A.: Mix and match: Secure function evaluation via ciphertexts. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, p. 162. Springer, Heidelberg (2000)

- [KO04] Katz, J., Ostrovsky, R.: Round-optimal secure two-party computation. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 335–354. Springer, Heidelberg (2004)
- [KOS03] Katz, J., Ostrovsky, R., Smith, A.: Round efficiency of multi-party computation with a dishonest majority. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 578–595. Springer, Heidelberg (2003)
- [KS08] Kolesnikov, V., Schneider, T.: A practical universal circuit construction and secure evaluation of private functions. In: Tsudik, G. (ed.) FC 2008. LNCS, vol. 5143, pp. 83–97. Springer, Heidelberg (2008)
- [LP09] Lindell, Y., Pinkas, B.: A proof of security of Yao’s protocol for two-party computation. *J. Cryptology* 22(2), 161–188 (2009)
- [NO09] Nielsen, J.B., Orlandi, C.: LEGO for two-party secure computation. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 368–386. Springer, Heidelberg (2009)
- [RAD78] Rivest, R., Adelman, L., Dertouzos, M.: On data banks and privacy homomorphisms. In: DeMillo, R., Dobkin, D., Jones, A., Lipton, R. (eds.) Foundations of Secure Computation, pp. 169–179. Academic Press, London (1978)
- [S79] Shamir, A.: How to share a secret. *Commun. ACM* 22(11), 612–613 (1979)
- [SCO⁺01] 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)
- [SYY99] Sander, T., Young, A., Yung, M.: Non-interactive cryptocomputing for NC¹. In: Proc. 40th IEEE Symposium on Foundations of Computer Science (FOCS), pp. 554–567 (1999)
- [TY98] Tsiounis, Y., Yung, M.: On the security of ElGamal based encryption. In: Imai, H., Zheng, Y. (eds.) PKC 1998. LNCS, vol. 1431, pp. 117–134. Springer, Heidelberg (1998)
- [V76] Valiant, L.G.: Universal circuits (preliminary report). In: Proc. 8th Annual ACM Symposium on Theory of Computing (STOC), pp. 196–203 (1976)
- [Y82] Yao, A.: Protocols for secure computations (extended abstract). In: Proc. 23rd IEEE Symposium on Foundations of Computer Science (FOCS), pp. 160–164 (1982)
- [Y86] Yao, A.: How to generate an exchange secrets. In: Proc. 27th IEEE Symposium on Foundations of Computer Science (FOCS), pp. 162–167 (1986)

A Explicit Preprocessing of \mathcal{P}_1

During the preprocessing stage, the parties can generate polynomial number of garbled gates, which can later be used for evaluating circuits. Therefore it suffices to know a bound on the sizes of circuits to be evaluated later. We show how to generate such truth tables explicitly given y as a global public key.

Throughout, each bit b will be encrypted with plaintext g^b . Denote by $\langle m \rangle$ a simple ElGamal ciphertext (with randomness $r = 0$): $(1, m)$. For an ElGamal ciphertext c for a bit, its negation $\neg c$ is defined as $\langle g^1 \rangle / c$. For two ElGamal ciphertext $a = (a_1, a_2)$ and $b = (b_1, b_2)$, define $\text{ZKe}_u(a, b)$ — the proof that b is a re-encryption a with public key u — as $\text{PK}\{r : b_1 = g^r a_1, b_2 = u^r a_2\}$. When public key is not specified, ZKe means ZKe_y . The construction details can be found in Appendix [A.3](#).

A.1 Preliminaries: Joint Generation of Garbled Gates

We associate a gate with the truth table for it. The entries of the truth tables are encrypted Boolean values, and the rows of each truth table are permuted, such that only a threshold of the parties can (1) recover any plaintext and (2) learn the permutation of the rows.

Sampling a Random Encrypted Boolean Value. In this protocol, n parties perform an oblivious analogue of XORing their respective random bits in n rounds. In our case, semantic security of ElGamal and the soundness of the attached proof guarantee they cannot.

1. Each party P_i selects $a_i \in_R \{0, 1\}$ and computes $\hat{a}_i = E_y(g^{a_i})$, $\pi_i = \text{ZKe}(\langle g^0 \rangle, \hat{a}_i) \vee \text{ZKe}(\langle g^1 \rangle, \hat{a}_i)$ and broadcasts (\hat{a}_i, π_i) . Let $S = \{j : \pi_j \text{ is valid}\}$. Set $\hat{a} \leftarrow \widehat{a_{\min}}$ where $\min = \min_{j \in S} j$.
2. For $j = 2, \dots, |S|$:

Let i be the j -th smallest element in S . P_i computes an encryption \hat{d}_i such that $\hat{d}_i = (d_{i1}, d_{i2})$ is a re-encryption of \hat{a} if $a_i = 0$ or a re-encryption of $-\hat{a}$ otherwise. Then P_i broadcasts (\hat{d}_i, ψ_i) where

$$\psi_i = \left(\text{ZKe}(\langle g^0 \rangle, \hat{a}_i) \wedge \text{ZKe}(\hat{a}, \hat{d}_i) \right) \vee \left(\text{ZKe}(\langle g^1 \rangle, \hat{a}_i) \wedge \text{ZKe}(-\hat{a}, \hat{d}_i) \right).$$

If ψ_i is valid, then each party sets $\hat{a} \leftarrow \hat{d}_i$.

As in computing xor, it is enough that one of the bits is random (or, in our case, that one party is honest) to guarantee a random output as long as corrupt parties can not have their bit choices depend on the bits of other parties. The invariant of the protocol is that at the end of each round the ciphertext \hat{a} encrypts exclusive-or of a_i 's so far.

Generating a Garbled IDENTITY Gate. First, run the procedure of sampling a random encrypted Boolean value. Let the output of the procedure is \hat{a} . The first row of an IDENTITY gate is \hat{a} , and the second row is computed by negating the value of \hat{a} .

In	Out
\hat{a}	\hat{a}
$-\hat{a}$	$-\hat{a}$

IDENTITY

Generating a Garbled NAND Gate.

1. Each party P_i selects $a_i, b_i \in_R \{0, 1\}$ and computes $\hat{a}_i = E_y(g^{a_i})$, $\hat{b}_i = E_y(g^{b_i})$, $\pi_i = \text{ZKe}(\langle g^0 \rangle, \hat{a}_i) \vee \text{ZKe}(\langle g^1 \rangle, \hat{a}_i)$, and $\phi_i = \text{ZKe}(\langle g^0 \rangle, \hat{b}_i) \vee \text{ZKe}(\langle g^1 \rangle, \hat{b}_i)$, and broadcasts $(\hat{a}_i, \hat{b}_i, \pi_i, \phi_i)$.
2. Run the procedure of sampling random encrypted Boolean values with \hat{a}_i 's. Let \hat{a} be the output of the procedure. Let $S = \{j : \pi_j \text{ and } \phi_j \text{ are valid}\}$. Set $\hat{b} \leftarrow \langle g^0 \rangle$ and $\hat{ab} \leftarrow \langle g^0 \rangle$.
3. For $j = 1, \dots, |S|$: Let i be the j -th smallest element in S . P_i computes encryptions \hat{d}_i and \hat{e}_i such that
 - If $b_i = 0$, then \hat{d}_i is a re-encryption of \hat{b} and \hat{e}_i is a re-encryption of \hat{ab} .
 - If $b_i = 1$, then \hat{d}_i is a re-encryption of $-\hat{b}$ and \hat{e}_i is a re-encryption of \hat{a}/\hat{ab} . Then P_i broadcasts $(\hat{d}_i, \hat{e}_i, \psi_i)$ where $\psi_i = \psi_i^0 \vee \psi_i^1$ for $\psi_i^0 = \text{ZKe}(\langle g^0 \rangle, \hat{b}_i) \wedge \text{ZKe}(\hat{b}, \hat{d}_i) \wedge \text{ZKe}(\hat{ab}, \hat{e}_i)$ and $\psi_i^1 = \text{ZKe}(\langle g^1 \rangle, \hat{b}_i) \wedge \text{ZKe}(-\hat{b}, \hat{d}_i) \wedge \text{ZKe}(\hat{a}/\hat{ab}, \hat{e}_i)$. If ψ_i is valid, then each party sets $\hat{b} \leftarrow \hat{d}_i$ and $\hat{ab} \leftarrow \hat{e}_i$.

The invariant of the loop is that at the end of each round the ciphertext \hat{ab} encrypts exclusive-or of ab_i 's so far. After $\hat{a}, \hat{b}, \hat{ab}$ are generated, each party P_i can complete the truth table, by locally negating the ciphertexts as described in the table.

In _L	In _R	Out
\hat{a}	\hat{b}	$-\hat{ab}$
\hat{a}	$-\hat{b}$	$\hat{ab} \cdot (-\hat{a})$
$-\hat{a}$	\hat{b}	$\hat{ab} \cdot (-\hat{b})$
$-\hat{a}$	$-\hat{b}$	$\hat{a} \cdot \hat{b}/\hat{ab}$

NAND

A.2 Preliminaries: Jointly Recoverable Encrypted ElGamal Key Pairs

Verifiable ElGamal Encryption of Discrete Logarithm. To generate a jointly recoverable encrypted ElGamal key pair, we first introduce the following verifiable encryption of discrete logarithm.

Let $\gamma := g^z$. We want to encrypt z in a verifiable manner. Let z_i be the i -th rightmost bit of z for $i \in [k]$. The verifiable encryption is $\hat{E}_y(z) = (\widehat{z}_0, \dots, \widehat{z}_{k-1}, \pi)$, where $\widehat{z}_i = E_y(g^{z_i \cdot 2^i})$ for $i \in [k]$. The proof π is

$$\left(\bigwedge_{i=0}^{k-1} \left(\text{ZKe}(\langle g^0 \rangle, \widehat{z}_i) \vee \text{ZKe}(\langle g^{2^i} \rangle, \widehat{z}_i) \right) \right) \wedge \text{ZKe} \left(\langle \gamma \rangle, \prod_{i=0}^{k-1} \widehat{z}_i \right).$$

When we get $(g^{z_0 \cdot 2^0}, \dots, g^{z_{k-1} \cdot 2^{k-1}})$ by decrypting $\hat{E}_y(z)$, z can be extracted via exhaustive search in polynomial time in k because each z_i is a bit.

Note that the encryption scheme is homomorphic if we ignore the proof part. Multiplication of two verifiable encryptions $\hat{E}_y(z) = (\widehat{z}_1, \dots, \widehat{z}_{k-1})$ and $\hat{E}_y(w) = (\widehat{w}_1, \dots, \widehat{w}_{k-1})$ is defined as $\hat{E}_y(z) \cdot \hat{E}_y(w) = (\widehat{z}_1 \cdot \widehat{w}_1, \dots, \widehat{z}_{k-1} \cdot \widehat{w}_{k-1})$.

Generation of Jointly Recoverable Encrypted ElGamal Key Pairs. For simplicity, we omit the proof part of the verifiable encryption from the presentation below. Generation of a key pair can be done as follows:

1. Each party P_j runs ElGamal key generation and obtains (k_j, g^{k_j}) . It broadcasts $(g^{k_j}, \hat{E}_y(k_j))$.
2. Let S be the set of parties whose encryptions are verified. In the PK column, $\prod_{j \in S} g^{k_j}$ is set.

In the SK column, $\prod_{j \in S} \hat{E}_y(k_j)$ is set.

Extraction of the secret key. Let $(Y_0, \dots, Y_{k-1}) := \prod_{j \in S} \hat{E}_y(k_j)$. Let g^{z_i} be the decryption of Y_i . Then given $(g^{z_0}, \dots, g^{z_{k-1}})$, we can extract the secret key $\sum_{j \in S} k_j = \sum_i z_i$ by finding each z_i via exhaustive search, which can be done efficiently since $g^{z_i} \in \{2^{0 \cdot 2^i}, 2^{1 \cdot 2^i}, \dots, 2^{n \cdot 2^i}\}$.

A.3 Preprocessing of \mathcal{P}_1

The preprocessing takes $2n$ rounds, since step 1.1 and step 1.2 can be executed concurrently. This protocol is UC-secure, but for lack of space, we defer the proof of security to full version.

Step 1.1: Garbled Circuit Generation - Intermediate Gates. For each NAND gate, run the procedure of joint generation of garbled NAND gate in Appendix [A.1](#) to fill in In and Out Columns. For each pair of columns PK and SK, run the procedure of jointly recoverable encrypted ElGamal key pairs in Appendix [A.2](#)⁹. The above tasks are executed in parallel.

Step 1.2: Garbled Circuit Generation - Output Gates

1. Run the procedure of sampling random encrypted Boolean values in Appendix [A.1](#) where each party P_i selects $a_i \in_R \{0, 1\}$. Let \hat{a} be the output of the procedure and let $S = \{j : P_j \text{ behaved honestly during the procedure}\}$. Fill in In and Out Columns as an IDENTITY gate.

⁹ Now in the online stage, k instances of M-CODE are executed since $T_b[\text{SK}][w]$ contains k ElGamal ciphertexts. The communication complexity blows up by multiplicative factor of k .

In addition, run the procedures of jointly recoverable encrypted ElGamal key pairs in Appendix A.2 to fill the columns PK and SK. Let z_1 and z_2 be the two keys in the column PK.

2. In order to fill Final column, each party P_i such that $i \in S$ broadcasts $(\widehat{a_{i,z_1}} = E_{z_1}(g^{a_i}), \widehat{a_{i,z_2}} = E_{z_2}(g^{a_i}))$. Set $\widehat{a_{z_1}} \leftarrow \langle g^0 \rangle, \widehat{a_{z_2}} \leftarrow \langle g^1 \rangle$. Parties jointly compute $E_{z_1}(g^{\oplus_i a_i})$ and $E_{z_2}(g^{1-\oplus_i a_i})$. In particular, for $i = 1, \dots, |S|$:

(a) Let i be the j -th smallest element in S . P_i computes encryptions $\widehat{d}_i, \widehat{e}_i$ such that \widehat{d}_i (resp. \widehat{e}_i) is a re-encryption of $\widehat{a_{i,z_1}}$ (resp. $\widehat{a_{i,z_2}}$) if $a_i = 0$ or a re-encryption of $\neg \widehat{a_{i,z_1}}$ (resp. $\neg \widehat{a_{i,z_2}}$) otherwise. Then P_i broadcasts $(\widehat{d}_i, \widehat{e}_i, \psi_i)$ where $\psi_i = \psi_i^0 \vee \psi_i^1$ for

$$\begin{aligned} \psi^b e &= \text{ZKe}(\langle g^0 \rangle, \widehat{a}_i) \wedge \text{ZKe}_{z_1}(\langle g^0 \rangle, \widehat{a_{i,z_1}}) \wedge \text{ZKe}_{z_2}(\langle g^0 \rangle, \widehat{a_{i,z_2}}) \wedge \\ &\quad \text{ZKe}_{z_1}(\widehat{a_{z_1}}, \widehat{d}_i) \wedge \text{ZKe}_{z_2}(\widehat{a_{z_2}}, \widehat{e}_i) \quad \text{and} \\ \psi^1 &= \text{ZKe}(\langle g^1 \rangle, \widehat{a}_i) \wedge \text{ZKe}_{z_1}(\langle g^1 \rangle, \widehat{a_{i,z_1}}) \wedge \text{ZKe}_{z_2}(\langle g^1 \rangle, \widehat{a_{i,z_2}}) \wedge \\ &\quad \text{ZKe}_{z_1}(\neg \widehat{a_{z_1}}, \widehat{d}_i) \wedge \text{ZKe}_{z_2}(\neg \widehat{a_{z_2}}, \widehat{e}_i). \end{aligned}$$

(b) If ψ_i is valid, then each party sets $\widehat{a_{z_1}} \leftarrow \widehat{d}_i$, and $\widehat{a_{z_2}} \leftarrow \widehat{e}_i$. Otherwise, in the case of honest majority, parties collectively compute a_i from threshold decryption using (y_1, \dots, y_n) and compute $\widehat{a_{z_1}}, \widehat{a_{z_2}}$ accordingly. In the case of honest minority, the protocol aborts. Finally, set $\widehat{a_{z_2}} \leftarrow \neg \widehat{a_{z_2}}$.

Improved Non-committing Encryption with Applications to Adaptively Secure Protocols

Seung Geol Choi^{1,*}, Dana Dachman-Soled^{1,*}, Tal Malkin^{1,*}, and Hoeteck Wee^{2,**}

¹ Columbia University
{sgchoi,dglasner,tal}@cs.columbia.edu
² Queens College, CUNY
hoeteck@cs.qc.cuny.edu

Abstract. We present a new construction of non-committing encryption schemes. Unlike the previous constructions of Canetti et al. (STOC '96) and of Damgård and Nielsen (Crypto '00), our construction achieves all of the following properties:

- **Optimal round complexity.** Our encryption scheme is a 2-round protocol, matching the round complexity of Canetti et al. and improving upon that in Damgård and Nielsen.
- **Weaker assumptions.** Our construction is based on *trapdoor simulatable cryptosystems*, a new primitive that we introduce as a relaxation of those used in previous works. We also show how to realize this primitive based on hardness of factoring.
- **Improved efficiency.** The amortized complexity of encrypting a single bit is $O(1)$ public key operations on a constant-sized plaintext in the underlying cryptosystem.

As a result, we obtain the first non-committing public-key encryption schemes under hardness of factoring and worst-case lattice assumptions; previously, such schemes were only known under the CDH and RSA assumptions. Combined with existing work on secure multi-party computation, we obtain protocols for multi-party computation secure against a malicious adversary that may adaptively corrupt an arbitrary number of parties under weaker assumptions than were previously known. Specifically, we obtain the first adaptively secure multi-party protocols based on hardness of factoring in both the stand-alone setting and the UC setting with a common reference string.

Keywords: public-key encryption, adaptive corruption, non-committing encryption, secure multi-party computation.

1 Introduction

Secure multi-party computation (MPC) allows several mutually distrustful parties to perform a joint computation without compromising, to the greatest extent possible,

* Supported in part by NSF Grants CCF-0347839, CNS-0716245, CNS-0831094 and SBE-0245014.

** Partially supported by a PSC-CUNY Award, and part of this work was done while a post-doc at Columbia University.

the privacy of their inputs or the correctness of the outputs. An important criterion in evaluating the security guarantee is *how many* parties an adversary is allowed to corrupt and *when* the adversary determines which parties to corrupt. Ideally, we want to achieve the strongest notion of security, namely, against an adversary that corrupts an arbitrary number of parties, and *adaptively* determines who and when to corrupt during the course of the computation (and without assuming erasures¹). Even though the latter is a very natural and realistic assumption about the adversary, most of the MPC literature only addresses security against a static adversary, namely one that chooses (and fixes) which parties to corrupt before the protocol starts executing. And if indeed such protocols do exist, it is important to answer the following question:

What are the cryptographic assumptions under which we can realize MPC protocols secure against a malicious, adaptive adversary that may corrupt a majority of the parties?

Towards answering this question, we revisit the problem of constructing *non-committing encryption schemes*, a cryptographic primitive first introduced by Canetti et al. [CFGN96] as a tool for building adaptively secure MPC protocols in the presence of an honest majority. Informally, non-committing encryption schemes are semantically secure, possibly interactive encryption schemes, with the additional property that a simulator can generate special ciphertexts that can be opened to both a 0 and a 1. In a more recent work, Canetti et al. [CLOS02] (extending [B98]) showed how to construct adaptively secure oblivious transfer protocols starting from non-committing public-key encryption schemes (i.e. the key generation algorithm must be non-interactive), which may in turn be used to construct MPC protocols secure against a malicious, adaptive adversary that may corrupt an arbitrary number of parties.

Unfortunately, the only known constructions of non-committing public-key encryption schemes (PKEs) are based on the CDH and RSA assumptions [CFGN96] and the construction exploits in a very essential way that these assumptions give rise to families of trapdoor permutations with a common domain. If we allow for an interactive key generation phase, Damgård and Nielsen [DN00], building on [B97, CFGN96], constructed 3-round non-committing encryption schemes based on a more general assumption, that of *simulatable PKEs*, which may in turn be realized from DDH, CDH, RSA and more recently, worst-case lattice assumptions [GPV08] (see figure 1).

1.1 Our Results

First, we present a new construction of non-committing encryption schemes, which simultaneously improves upon all of the previous constructions in [CFGN96, DN00]:

Optimal Round Complexity. We provide a construction of non-committing PKEs from simulatable cryptosystems. Our construction is surprisingly simple - a twist to the standard cut-and-choose techniques used in [DN00, KO04] - and also admits a fairly

¹ Refer to [C00, Section 5.2] for a discussion on how trusted erasures may be a problematic assumption.

straight-forward simulation and analysis. In particular, our construction and the analysis are conceptually and technically simpler than those in [CFGN96, DN00]; we avoid having to analyze the number of one's in certain Binomial distributions as in [CFGN96] and to consider a subtle failure mode as in [DN00].

Reducing the assumptions. Informally, a simulatable PKE is an encryption scheme with special algorithms for obliviously sampling public keys and random ciphertexts without learning the corresponding secret keys and plaintexts; in addition, both of these oblivious sampling algorithms should be efficiently invertible.

We define a weaker assumption, which we refer to as trapdoor simulatable cryptosystems, and prove that it is sufficient for our construction and analysis to go through. Roughly speaking, we provide the inverting algorithms in a simulatable cryptosystem with additional trapdoor information (hence the modifier “trapdoor”), which makes it *easier* to design a simulatable cryptosystem.

Improved efficiency. While the main focus of this work is feasibility results (notably, reducing the computational assumptions for both non-committing encryption schemes and adaptively secure MPC), we show how to combine a variant of our basic construction with the use of error-correcting codes to achieve better efficiency. That is, the amortized complexity of encrypting a single bit is $O(1)$ public-key operations on a constant-sized plaintext in the underlying cryptosystem.

Thus, we obtain the following.

Theorem 1 (informal). *There exists a black-box construction of a non-committing public-key encryption scheme, starting from any trapdoor simulatable cryptosystem.*

Factoring-Based constructions. Next, we derive trapdoor simulatable cryptosystems from a variant of Rabin’s trapdoor permutations (c.f. [H99, S96, FF02]) based on the hardness of factoring Blum integers.

Theorem 2 (informal). *Suppose factoring Blum integers is hard on average. Then, there exists a trapdoor simulatable cryptosystem.*

We stress that we do not know how to construct a simulatable cryptosystem under the same assumptions; specifically, inverting the sampling algorithm for ciphertexts in our construction without the trapdoor (the factorization of the Blum integer modulus) appears to be as hard as factoring Blum integers. This shows that trapdoor simulatable cryptosystems is indeed a meaningful and useful relaxation. In the process, we also obtain the first factoring-based dense cryptosystems². When combined with enhanced trapdoor permutations, this yields the first factoring-based non-interactive proofs of knowledge [DP92].

Oblivious Transfer and MPC. We consider the applications of our main result to the constructions of adaptively secure oblivious transfer and general MPC protocols in both the stand-alone setting and the UC setting (c.f. [CLOS02, IPS08, CDSMW09]).

² These are PKE schemes where a random string has a inverse polynomial probability of being a valid public key.

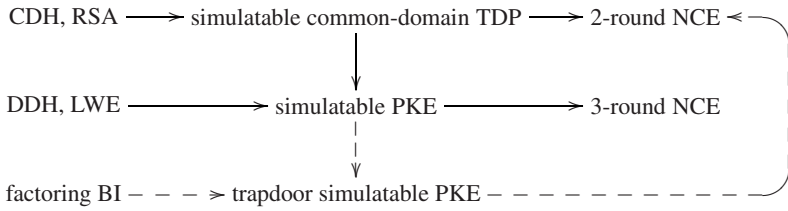


Fig. 1. Summary of previous results (solid lines) along with our contributions (dashed lines)

Theorem 3 (informal). *There exists a black-box construction of a 6-round 1-out-of- ℓ oblivious transfer protocol for strings in the \mathcal{F}_{COM} -hybrid model³ in the UC setting that is secure against a malicious, adaptive adversary, starting from any trapdoor simulatable cryptosystem.*

We add that if the oblivious key generation algorithm in the trapdoor simulatable cryptosystem achieves statistical indistinguishability (which is the case for all of the afore-mentioned constructions), then we obtain an OT protocol that is secure against a computationally unbounded malicious sender. While our OT protocol is not as efficient as that in the recent work of Garay, Wichs and Zhou [GWZ09] (we incur an additional multiplicative overhead that is linear in the security parameter), our protocol along with our general framework offers several advantages:

- In addition to relying on the \mathcal{F}_{COM} functionality and a simulatable PKE (to implement non-committing encryption) as in our work, the [GWZ09] framework requires a so-called enhanced dual-mode cryptosystem. This is a relatively high-level CRS-based primitive from [PVW08] augmented with two main additional properties: the first has a flavor of oblivious sampling; the second requires that the underlying CRS be a common *random* string (modulo some system parameters) and not just a common reference string. This requirement is inherent to their framework, since this CRS is generated using a coin-tossing protocol. This latter requirement is very restrictive, and the only known construction of an enhanced dual-mode cryptosystem is based on the quadratic residuosity assumption.
- Our protocol immediately handles 1-out-of- ℓ OT, whereas [GWZ09] only addresses 1-out-of-2 OT, a limitation inherited from [PVW08].

Combined with [CLOS02, IPS08, CDSMW09], we obtain the following corollaries:

Corollary 1 (informal). *Assuming the existence of trapdoor simulatable cryptosystems, there exists adaptively secure multi-party protocols in the stand-alone setting and in the \mathcal{F}_{COM} -hybrid model in the UC setting against a malicious adversary that may adaptively corrupt any number of parties.*

Specifically, we obtain the first adaptively secure multi-party protocols based on hardness of factoring in both the stand-alone setting and the UC setting with a common reference string.

³ \mathcal{F}_{COM} is an ideal functionality for commitment.

1.2 Additional Related Work

The problem of constructing encryption schemes that are secure against adaptive corruptions was first addressed in the work of Beaver and Haber [BH92]. They considered a simpler scenario where the honest parties have the ability to securely and completely erase previous states. For instance, an honest sender could erase the randomness used for encryption after sending the ciphertext, so that upon being corrupted, the adversary only gets to see the corresponding plaintext. An intermediate model, wherein we assume secure erasures for either the sender or receiver but not both (or, by limiting the adversary to corrupting at most one of the two parties), has been considered in several other works [JL00, CHK05, KO04].

Organization. We present an overview of our constructions in Section 2, preliminaries in Section 3, the formulation of a trapdoor simulatable PKE in Section 4, our factoring-based trapdoor simulatable PKE in Section 6, and our non-committing encryption scheme in Section 5. In Section 7, we show the construction of a 6-round oblivious transfer protocol.

2 Overview of Our Constructions

At a high level, our non-committing PKE is similar to that from previous works [CFGN96, DN00, KO04]. The receiver generates a collection of public keys in such a way that it only knows an α fraction of the corresponding secret keys; this can be achieved by generating an α fraction of the public keys using the key generation algorithm and the remaining $1 - \alpha$ fraction obliviously. Similarly, the sender generates a collection of ciphertexts in such a way that it only knows an α fraction of the corresponding plaintexts. Previous constructions all work with the natural choice of $\alpha = 1/2$ so that the simulator generates a collection of ciphertexts half of which are encryptions of 0 and the other half are encryptions of 1. As noted in [KO04], this is sufficient for obtaining non-committing PKEs wherein at most one party is corrupted. Roughly speaking, the difficulty in handling simultaneous corruptions of both the sender and the receiver with $\alpha = 1/2$ is that in the simulation, the sender's choice of the α fraction of keys completely determine the receiver's choice of the α fraction of ciphertexts whereas in an actual honest encryption, these choices are completely independent (we elaborate on this later in this section). The key insight in our construction is to work with a smaller value of α (turns out $1/4$ is good enough).

A Toy Construction. Consider the following encryption scheme, which is a simplification of that in [KO04, DN00]. The receiver generates a pair of public keys (PK_0, PK_1) by generating one key (selected at random) using the key-generation algorithm, and the other using the oblivious sampling algorithm. To encrypt a bit b , the sender generates a pair of ciphertexts (C_0, C_1) as follows: pick a random bit r , set C_r to be $\text{Enc}_{PK_r}(b)$ and choose C_{1-r} using the oblivious sampling algorithm. To decrypt, the receiver decrypts exactly one of C_0, C_1 using the secret key that it knows. This construction corresponds to $\alpha = 1/2$ where α is the fraction of public keys for which the receiver knows the secret key, and also the fraction of ciphertexts for which the sender knows the plaintext. Observe that this encryption scheme has the following properties:

- It has a constant decryption error of $1/4$ if an obviously sampled ciphertext is equally likely to decrypt to 0 or 1. As shown in [KO04], this error can be reduced by standard repetition techniques.
- It tolerates corruption of either the sender or the receiver, but not both. Consider a simulator that generates both of (PK_0, PK_1) (along with SK_0, SK_1) using the key-generation algorithm, and a ciphertext (C_0, C_1) as follows: pick a random bit β , and set C_0 to be $Enc_{PK_0}(\beta)$ and C_1 to be $Enc_{PK_1}(1 - \beta)$. Suppose the simulator later learns that this is an encryption of 0. If only the sender is corrupted, the simulator claims $r = \beta$ and that $C_{1-\beta}$ is obviously sampled. If only the receiver is corrupted, it claims that it knows SK_β and that $PK_{1-\beta}$ is obviously sampled.

We highlight two subtleties in the above simulation strategy. First, it achieves 0 decryption error (as opposed to $1/4$ in an honest encryption); this can be fixed with a somewhat more involved simulation strategy. This in turn becomes pretty complicated once we use standard repetition techniques to reduce the decryption error. Next, it is always the case in the simulation that either both PK_0 and C_0 are obviously sampled, or both PK_1 and C_1 are obviously sampled. As such, this simulation strategy fails if both the sender and the receiver are corrupted, because in an actual encryption, which of PK_0, PK_1 and which of C_0, C_1 are obviously sampled are determined independently.

Our Encryption Scheme. As noted in the introduction, the key insight in our construction is to work with a small value of α . In addition, following [DN00], we use a random k -bit encoding of 0 and 1, where k is the security parameter:

- The receiver generates $4k$ public keys PK_1, \dots, PK_{4k} : k of them are generated using the key-generation algorithm, and the remaining $3k$ are generated using the oblivious sampling algorithm. The receiver then sends PK_1, \dots, PK_{4k} along with two random k -bit messages M_0, M_1 .
- To encrypt a bit b , the sender sends $4k$ ciphertexts (one for each of PK_1, \dots, PK_{4k}), of which k are encryptions of M_b , and the remaining ones are obviously sampled.
- To decrypt, the receiver decrypts the k ciphertexts for which it knows the corresponding secret key. If any of the k plaintexts matches M_0 , it outputs 0 and otherwise, it outputs 1.

Encoding 0 and 1 randomly as M_0 and M_1 is useful for two reasons:

- That an obviously sampled ciphertext is equally likely to decrypt to 0 or 1 is no longer needed to guarantee correctness (c.f. [DN00]). Indeed, reasoning about decryptions of obviously sampled ciphertext is non-trivial for the lattice-based simulatable PKEs in [GPV08].
- Constructing a simulator becomes much easier as we avoid having to generate distributions over k independent biased bits conditioned on the majority of the bits being 0, say. Generating such distributions arises for instance in [CFGN96] and is related to the first subtlety associated with the naive simulation strategy. In our construction, the simulated ciphertext comprises k encryptions of M_0 , k encryptions of M_1 , and $2k$ obviously generated ciphertexts. Having these extra $2k$ obviously generated ciphertexts (which is possible because $\alpha < 1/2$) is crucial for handling simultaneous corruptions of the sender and the receiver.

Trapdoor Simulatable PKEs from Factoring. Our factoring-based trapdoor simulatable PKE construction consists of two main steps. First, we modify the Rabin trapdoor permutations based on squaring modulo Blum integer so that it remains a permutation over any arbitrary integer modulus. This relies on the following number-theoretical structural lemma implicit in [H99, S96, FF02]⁴:

Let N be an arbitrary odd k -bit integer, and let $Q_N = \{a^{2^k} \pmod{N} \mid a \in Z_N^*\}$. Then, the map $\psi : x \mapsto x^2$ defines a permutation over Q_N .

We also provide an efficient algorithm for inverting ψ given the factorization of N . Note that the standard algorithm for computing square roots does not guarantee that the output lies in Q_N . Moreover, the probability that a random square root lies in Q_N may be exponential small so we cannot repeatedly computing random square roots until we find one in Q_N ; it's also not clear a-priori how to test membership in Q_N even given the factorization of N .

The next step transforms the family of trapdoor permutations ψ acting on the domain Q_N into a family of “enhanced” trapdoor permutations with the same domain Q_N , using an idea from [G04, Section C.1]. The latter has the property that we can obviously sample a random element y in Q_N so that given y along with the coin tosses used to sample y , it is infeasible to compute the preimage of y under the permutation (note that the naive algorithm for sampling a random element of Q_N gives away its preimage under ψ). We will need the oblivious sampling algorithm for a random element in Q_N in our oblivious sampling algorithm for random ciphertexts. We will also need to realize trapdoor invertibility for the latter, which requires an efficient algorithm that given the factorization of N and an element y in Q_N , outputs a random 2^k 'th root of y ⁵. Note that iteratively computing random square roots k times does not work: after computing the first square root, we may not end up with a 2^{k-1} 'th power.

3 Preliminaries

If A is a probabilistic polynomial time (hereafter, ppt) algorithm that runs on input x , $A(x)$ denotes the random variable according to the distribution of the output of A on input x . We denote by $A(x; r)$ the output of A on input x and random coins r . To simplify the notation, we will often omit quantifying over the distribution for r ; it will usually be clear from the context when r is not fixed, that it is drawn from the uniform distribution over strings of the appropriate length.

We assume that the reader is familiar with the standard definitions of public-key encryption schemes and semantic security (c.f. [GM84, G04]). We stress that we allow decryption errors that are exponentially small in k :

⁴ It was shown in [H99] that ψ defines a permutation over the subgroup O_N of Z_N^* of odd order, and that O_N contains Q_N ; turns out $O_N = Q_N$. While Q_N is trivially sampleable, it is not clear a-priori how to sample from O_N .

⁵ If we are given just N and not its factorization, this problem is at least as hard as factoring random Blum integers. This is in essence why we only obtain a factoring-based trapdoor simulatable PKE and not a simulatable PKE.

Definition 1 (encryption scheme). A triple $(\text{Gen}, \text{Enc}, \text{Dec})$ is an encryption scheme, if Gen and Enc are ppt algorithms and Dec is a deterministic polynomial-time algorithm such that for every message $m \in \{0, 1\}^*$ of polynomial length, $\Pr[\text{Gen}(1^k) \rightarrow (\text{PK}, \text{SK}), \text{Enc}_{\text{PK}}(m) \rightarrow c; \text{Dec}_{\text{SK}}(c) \neq m] < 2^{-\Omega(k)}$.

Non-committing Encryption. For simplicity, we present the definition of a non-committing public-key encryption scheme for single-bit messages:

Definition 2 (non-committing encryption [CEGN96]). A non-committing (bit) encryption scheme consists of a tuple $(\text{NCGen}, \text{NCEnc}, \text{NCDec}, \text{NCSim})$ where $(\text{NCGen}, \text{NCEnc}, \text{NCDec})$ is an encryption scheme and NCSim is the simulation algorithm that on input 1^k , outputs $(e, c, \sigma_G^0, \sigma_E^0, \sigma_G^1, \sigma_E^1)$ with the following property: for $b = 0, 1$ the following distributions are computationally indistinguishable:

- the joint view of an honest sender and an honest receiver in a normal encryption of b :

$$\{(e, c, \sigma_G, \sigma_E) \mid (e, d) = \text{NCGen}(1^k; \sigma_G), c = \text{NCEnc}_e(b; \sigma_E)\}$$

- simulated view of an encryption of b :

$$\{(e, c, \sigma_G^b, \sigma_E^b) \mid \text{NCSim}(1^k) \rightarrow (e, c, \sigma_G^0, \sigma_E^0, \sigma_G^1, \sigma_E^1)\}$$

It follows from the definition that a non-committing encryption scheme is also semantically secure.

Encrypting longer messages. Starting with a non-committing bit encryption scheme $(\text{NCGen}, \text{NCEnc}, \text{NCDec}, \text{NCSim})$, we may encrypt a longer message of length n by generating n independent public keys using NCGen , encrypting each bit of the message using a different public key and then concatenating the n ciphertexts. Note that this is different from the case of semantically secure encryption, where we may encrypt each bit using the same public key.

4 Trapdoor Simulatable Public Key Encryption

A ℓ -bit trapdoor simulatable encryption scheme consists of an encryption scheme $(\text{Gen}, \text{Enc}, \text{Dec})$ augmented with $(\text{oGen}, \text{oRndEnc}, \text{rGen}, \text{rRndEnc})$. Here, oGen and oRndEnc are the oblivious sampling algorithms for public keys and ciphertexts, and rGen and rRndEnc are the respective inverting algorithms⁶. We require that, for all messages $m \in \{0, 1\}^\ell$, the following distributions are computationally indistinguishable:

$$\{\text{rGen}(r_G), \text{rRndEnc}(r_G, r_E, m), \text{PK}, c \mid (\text{PK}, \text{SK}) = \text{Gen}(1^k; r_G), c = \text{Enc}_{\text{PK}}(m; r_E)\} \\ \text{and } \{\hat{r}_G, \hat{r}_E, \hat{\text{PK}}, \hat{c} \mid (\hat{\text{PK}}, \perp) = \text{oGen}(1^k; \hat{r}_G), \hat{c} = \text{oRndEnc}_{\hat{\text{PK}}}(1^k; \hat{r}_E)\}$$

It follows from the definition that a trapdoor simulatable encryption scheme is also semantically secure.

⁶ Existence of such inverting algorithms is called *trapdoor invertibility*. Compared to the simulatable cryptosystem (without trapdoor) defined in [DN00], rGen (resp. rRndEnc) takes r_G (resp. (r_G, r_E, m)) as the additional trapdoor information.

Encrypting longer messages. We note that if we started only with a trapdoor simulatable PKE for single bits, we may encrypt a longer message of length n by generating a single public key PK using Gen, and concatenating each of the message encrypted under PK.

5 Non-Committing Encryption from Weaker Assumptions

Theorem 4. *Suppose there exists a trapdoor simulatable encryption scheme. Then, there exists a non-committing encryption scheme as well as a universally composable oblivious transfer protocol secure against semi-honest, adaptive adversaries.*

We show how to construct a non-committing bit encryption scheme (NCGen, NCEnc, NCDec, NCSim) from a k -bit trapdoor simulatable PKE (Gen, Enc, Dec) (augmented with (oGen, oRndEnc, rGen, rRndEnc)). This is sufficient to establish the theorem by the connection between encrypting single bits and multiple bits as discussed in Sections 3 and 4. Our construction is presented in Figures 2 and 3.

Correctness. We begin by establishing correctness.

- Assume that the input $[c_1, \dots, c_{4k}]$ to the decryption algorithm is a random encryption of 0. Recall that $J = \{\text{Dec}_{\text{SK}_i}(c_i) \mid i \in T\}$ and we will output 0 unless $M_0 \notin J$. It is easy to see that $\Pr[M_0 \notin J] \leq \binom{3k}{k} / \binom{4k}{k} + 2^{-\Omega(k)}$ where the first summand comes from the probability that $S \cap T = \emptyset$ and the second

Key Generation NCGen(1^k):

1. Pick M_0, M_1 at random from $\{0, 1\}^k$.
2. Choose a random subset $T \subseteq [4k]$ of size k .
3. For $i = 1, 2, \dots, 4k$, generate a pair $(\text{PK}_i, \text{SK}_i)$ as follows:

$$(\text{PK}_i, \text{SK}_i) = \begin{cases} \text{Gen}(1^k) & \text{if } i \in T \\ \text{oGen}(1^k) & \text{otherwise} \end{cases}$$

Set $e = [M_0, M_1, \text{PK}_1, \dots, \text{PK}_{4k}]$ and $d = [T, \text{SK}_1, \dots, \text{SK}_{4k}]$.

Encryption NCEnc_{PK}(b):

1. Choose a random subset $S \subseteq [4k]$ of size k .
2. For $i = 1, 2, \dots, 4k$, generate a ciphertext c_i as follows:

$$c_i = \begin{cases} \text{Enc}_{\text{PK}_i}(M_b) & \text{if } i \in S \\ \text{oRndEnc}_{\text{PK}_i}(1^k) & \text{otherwise} \end{cases}$$

Set $c = [c_1, \dots, c_{4k}]$.

Decryption NCDec_{PK}(c):

1. Compute $J = \{\text{Dec}_{\text{PK}_i}(c_i) \mid i \in T\}$.
2. If $M_0 \in J$, output 0; else, output 1.

Fig. 2. Non-Committing Encryption Scheme (NCGen, NCEnc, NCDec)

Simulation NCSim:

1. Pick M_0, M_1 at random from $\{0, 1\}^k$.
2. Picking the sets S_0, S_1, T_0, T_1 :
 - Pick two random subsets S_0, T_0 of $[4k]$ each of size k .
 - Pick two random subsets S_1, T_1 of $[4k] \setminus (S_0 \cup T_0)$ such that $|S_1 \cap T_1| = |S_0 \cap T_0|$.
3. Generating the keys: for $i = 1, 2, \dots, 4k$, set

$$(\text{PK}_i, \text{SK}_i) = \begin{cases} \text{Gen}(1^k; r_G^i) & \text{if } i \in T_0 \cup S_0 \cup T_1 \cup S_1 \\ \text{oGen}(1^k; \hat{r}_G^i) & \text{otherwise} \end{cases}$$
4. Generating the ciphertext: for $i = 1, 2, \dots, 4k$, set

$$c_i = \begin{cases} \text{Enc}_{\text{PK}_i}(M_0; r_E^i) & \text{if } i \in S_0 \\ \text{Enc}_{\text{PK}_i}(M_1; r_E^i) & \text{if } i \in S_1 \\ \text{oRndEnc}_{\text{PK}_i}(\hat{r}_E^i) & \text{otherwise} \end{cases}$$
5. Simulating an opening to b : set $\sigma_G^b = \{T_b, u_G^{b,1}, \dots, u_G^{b,4k}\}$ and $\sigma_E^b = \{S_b, u_E^{b,1}, \dots, u_E^{b,4k}\}$, where

$$u_G^{b,i} = \begin{cases} r_G^i & \text{if } i \in T_b \\ \text{rGen}(r_G^i) & \text{if } i \in T_0 \cup T_1 \cup S_0 \cup S_1 \setminus T_b \\ \hat{r}_G^i & \text{otherwise} \end{cases}$$

$$u_E^{b,i} = \begin{cases} r_E^i & \text{if } i \in S_b \\ \text{rRndEnc}(r_G^i, r_E^i, M_{1-b}) & \text{if } i \in S_{1-b} \\ \hat{r}_E^i & \text{otherwise} \end{cases}$$

Set $e = [M_0, M_1, \text{PK}_1, \dots, \text{PK}_{4k}]$, $c = [c_1, \dots, c_{4k}]$. Additionally output $\sigma_G^0, \sigma_E^0, \sigma_G^1, \sigma_E^1$.

Fig. 3. Non-Committing Encryption Scheme NCSim

bounds the probability of a decryption error in the underlying encryption scheme (Gen, Enc, Dec).

- Assume that the input $[c_1, \dots, c_{4k}]$ to the decryption algorithm is a random encryption of 1. Recall that $J = \{\text{Dec}_{\text{SK}_i}(c_i) \mid i \in T\}$ and we will output 1 unless $M_0 \in J$. To bound $\Pr[M_0 \in J]$, observe that the distribution of J depends only on $M_1, \text{PK}_1, \dots, \text{PK}_{4k}, T, \text{SK}_1, \dots, \text{SK}_{4k}$ and the coin tosses used to generate c_1, \dots, c_{4k} , and is therefore independent of the choice of a random M_0 . This means that for each $i \in T$, the probability that $\text{Dec}_{\text{SK}_i}(c_i)$ equals M_0 is 2^{-k} . Taking a union bound, we obtain $\Pr[M_0 \in J] \leq k \cdot 2^{-k}$.

Security. We need to show that for each $b = 0, 1$, a normal encryption of b and a simulated encryption of b are computationally indistinguishable. Note that the view in a normal encryption of b contains two sets T, S which we will label as T_b, S_b and we will append to the view two sets T_{1-b}, S_{1-b} that are determined as follows: pick two random subsets S_{1-b}, T_{1-b} of $[4k] \setminus (S_b \cup T_b)$ such that $|S_{1-b} \cap T_{1-b}| = |S_b \cap T_b|$; call this distribution

H_0 . We will also append to the view in a simulated encryption of b the sets T_{1-b}, S_{1-b} as determined by the experiment NCSim ; call this distribution H_{4k} . We will show that the augmented distributions H_0 and H_{4k} are computationally indistinguishable in two steps:

Reasoning about the sets. First, we claim that the 4-tuple (S_0, T_0, S_1, T_1) in the augmented distribution H_0 and in H_{4k} are identically distributed. If $b = 0$, this is obvious since the distributions are defined in exactly the same way. The case for $b = 1$ follows from a symmetry argument, namely that if we switch (S_0, T_0) with (S_1, T_1) in the experiment NCSim , we get exactly the same distribution. Henceforth, it suffices to argue that H_0 and H_{4k} are computationally indistinguishable, conditioned on some fixed (S_0, T_0, S_1, T_1) in both H_0 and H_{4k} . We may now wLOG focus on the case $b = 0$. In fact, we may as well also fix M_0, M_1 in both H_0 and H_{4k} . In addition to $S_0, T_0, S_1, T_1, M_0, M_1$, the distributions H_0, H_{4k} comprise:

- $4k$ public keys $\text{PK}_1, \dots, \text{PK}_{4k}$ (generated using either Gen or oGen);
- $4k$ ciphertexts c_1, \dots, c_{4k} (generated using either Enc or oRndEnc);
- $4k$ sets of coin tosses u_G^1, \dots, u_G^{4k} for generating the public/secret keys; and
- $4k$ sets of coin tosses u_E^1, \dots, u_E^{4k} for generating the ciphertexts.

That is, we have $4k$ tuples of the form $(\text{PK}_i, c_i, u_G^i, u_E^i), i = 1, \dots, 4k$ in each view. Since S_0, T_0, S_1, T_1 are fixed, each of these $4k$ tuples are independently sampled from some distribution that only depends on the index i . Denote by X_1, \dots, X_{4k} the random variables for the $4k$ tuples in H_0 , and Y_1, \dots, Y_{4k} the random variables for the $4k$ tuples in H_{4k} .

The hybrid argument. Next, we argue that X_i and Y_i are computationally indistinguishable for $i = 1, \dots, 4k$, from which the indistinguishability of H_0 and H_{4k} follows via a hybrid argument. There are several cases we need to consider:

- $i \in T_0$ or $i \in [4k] \setminus (T_0 \cup S_0 \cup T_1 \cup S_1)$. It is easy to verify that in either of these cases, X_i and Y_i are identically distributed.
- $i \in S_1$ (“ $\text{oGen}, \text{oRndEnc} \cong \text{Gen}, \text{Enc}$ ”). Here, X_i is the distribution

$$\{\hat{\text{PK}}, \hat{c}, \hat{r}_G, \hat{r}_E \mid (\hat{\text{PK}}, \perp) = \text{oGen}(\hat{r}_G), \hat{c} = \text{oRndEnc}_{\hat{\text{PK}}}(\hat{r}_E)\}$$

and Y_i is the distribution

$$\{\text{PK}, c, \text{rGen}(r_G), \text{rRndEnc}(r_G, r_E, M_1) \mid (\text{PK}, \text{SK}) = \text{Gen}(r_G), c = \text{Enc}_{\text{PK}}(M_1; r_E)\}.$$

Indistinguishability follows immediately from the security of the trapdoor simulatable PKE.

- $i \in S_0 \setminus T_0$ (“ $\text{oGen}, \text{Enc} \cong \text{Gen}, \text{Enc}$ ”). Here, X_i is the distribution

$$\{\hat{\text{PK}}, c, \hat{r}_G, r_E \mid (\hat{\text{PK}}, \perp) = \text{oGen}(\hat{r}_G), c = \text{Enc}_{\hat{\text{PK}}}(M_0; r_E)\}$$

and Y_i is the distribution

$$\{\text{PK}, c, \text{rGen}(r_G), r_E \mid (\text{PK}, \text{SK}) = \text{Gen}(r_G), c = \text{Enc}_{\text{PK}}(M_0; r_E)\}.$$

Indistinguishability follows again from the security of the trapdoor simulatable PKE.

- $i \in T_1 \setminus S_1$ (“oGen, oRndEnc \cong Gen, oRndEnc”). Here, X_i is the distribution

$$\{\hat{\text{PK}}, \hat{c}, \hat{r}_G, \hat{r}_E \mid (\hat{\text{PK}}, \perp) = \text{oGen}(\hat{r}_G), \hat{c} = \text{oRndEnc}_{\hat{\text{PK}}}(\hat{r}_E)\}$$

and Y_i is the distribution

$$\{\text{PK}, \hat{c}, \text{rGen}(r_G), \hat{r}_E \mid (\text{PK}, \text{SK}) = \text{Gen}(r_G), \hat{c} = \text{oRndEnc}_{\text{PK}}(\hat{r}_E)\}.$$

Indistinguishability follows again from the security of the trapdoor simulatable PKE.

Improving the Efficiency. Instead of using sets $S, T \subset [4k]$ of size k , we choose $S, T \subset [40]$ of size 10. The previous analysis still goes through, except we now have a constant decryption error. To address this problem, we first encode the message \hat{m} with a linear-rate error-correcting code that corrects a constant fraction of errors, and then encrypt the codeword with the encryption scheme with constant error.

6 Trapdoor Simulatable PKE from Hardness of Factoring

Theorem 5. *Suppose factoring Blum integers is hard on average, and that Blum integers are dense, then there exists a trapdoor simulatable PKE.*

For simplicity, we only present a 1-bit trapdoor simulatable encryption scheme; we may encrypt longer messages by encrypting bit by bit.

A number-theoretic lemma. Fix any k -bit integer modulus N and we will work with the group Z_N^* . We will use $\text{factor}(N)$ to denote the factorization of N , and we define $Q_N = \{a^{2^k} \mid a \in Z_N^*\}$. Now, consider the map $\psi_N : Q_N \rightarrow Q_N$ given by $\psi_N(x) = x^2 \pmod{N}$. As shown in [H99, Facts 3.5-3.7], ψ_N defines a permutation on Q_N . We provide a more direct proof which also yields an efficient algorithm to invert ψ_N given $\text{factor}(N)$.

Claim. The map ψ_N defines a permutation on Q_N .

Proof. Let q denote the largest odd divisor of $\phi(N)$, where $\phi(\cdot)$ is the Euler’s totient function. It is easy to see that $\phi(N)$ divides $2^k q$, since $N < 2^k$. Take any $y \in Q_N$, where $y = a^{2^k}$. Then by Euler’s theorem, $y^q = 1 \pmod{N}$ and thus $\psi_N(y^{(q+1)/2}) = y \pmod{N}$. Clearly, $y^{(q+1)/2} \in Q_N$, so the map ψ_N is surjective. Moreover, the range and domain of ψ_N have equal sizes, so ψ_N must define a bijection. \square

The construction. We sketch the construction here; the formal construction is shown in Figure 4.

⁷ The codeword length (or, equivalently the message length) should be $\Omega(k)$. Then, by Chernoff bound, the number of decryption errors remains a constant fraction of the codeword length with overwhelming probability.

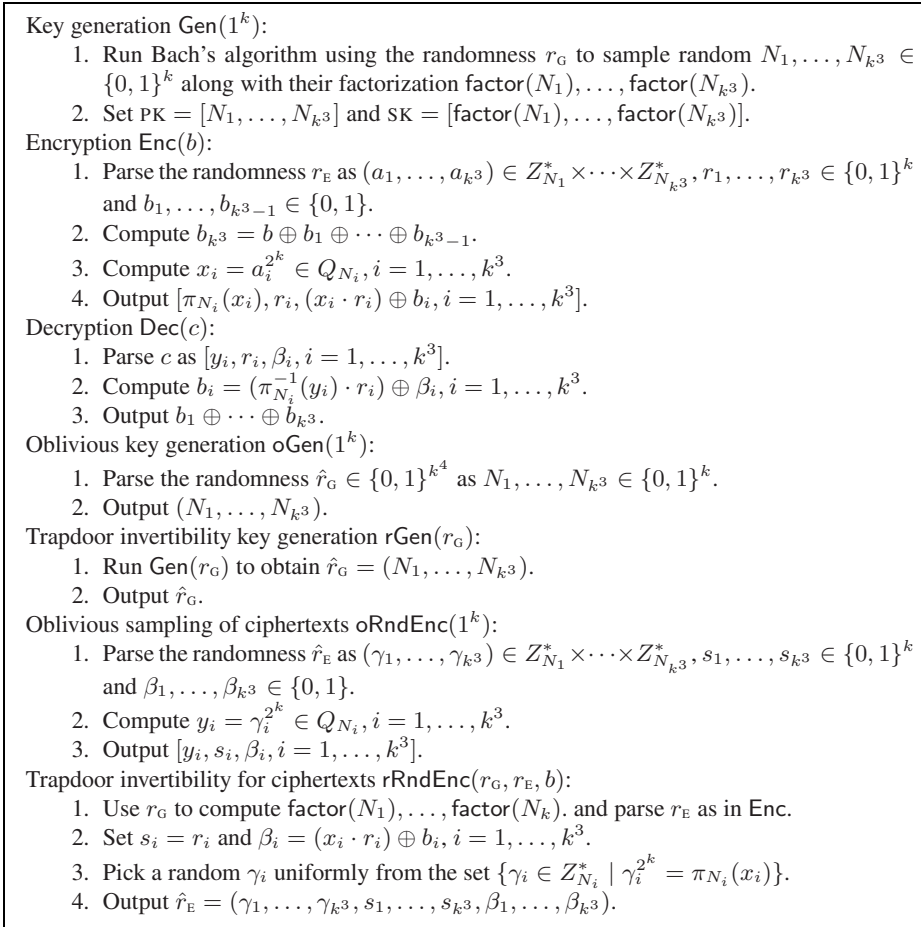


Fig. 4. Trapdoor Simulatable PKE from hardness of factoring Blum integers

STEP 1: First, we construct a family of “weakly one-way” enhanced trapdoor permutations. We start by modifying ψ_N to obtain a new family of permutations π_N ; the modification is analogous to that in [G04, Section C.1] to obtain enhanced trapdoor permutations from Rabin’s trapdoor permutations. The permutations $\pi_N : Q_N \rightarrow Q_N$ are indexed by a k -bit integer N and is given by:

$$\pi_N(x) \stackrel{\text{def}}{=} \psi_N^{k+1}(x) = x^{2^{k+1}} \pmod{N}$$

and the trapdoor is $\text{factor}(N)$. We may sample from this family by running Bach’s algorithm [B88, K02] to pick a random k -bit integer along with its factorization.

It is easy to verify π_N is a family of trapdoor permutations. Clearly, π_N is a permutation because it is the $(k+1)$ -fold iterate of a permutation ψ_N . Given the index N , π_N is efficiently computable by repeated squaring. Given the trapdoor $\text{factor}(N)$, π_N^{-1} is efficiently computable given $\text{factor}(N)$, by simply mapping y

to $y^{((q+1)/2)^{k+1}}$, i.e., raising y to the $(q + 1)/2$ 'th power $k + 1$ times. Here, q denotes the largest odd divisor of $\phi(N)$, which is easy to compute with the trapdoor. Moreover, we can show that if N is a Blum integer (which occurs with probability $\Omega(1/k^2)$ [GM04, RS94]), then inverting π_N given N is at least as hard as factoring N . This implies that π_N is one-way with probability $\Omega(1/k^2)$ over the choice of N .

STEP 2: Construct a “weak” encryption scheme using the standard construction of PKE from trapdoor permutations via the Goldreich-Levin hard-core predicate. The public key is N , the secret key is $\text{factor}(N)$, and to encrypt a bit b , we pick a random $x \in Q_N, r \in \{0, 1\}^k$ and output $(\pi_N(x), r, (x \cdot r) \oplus b)$, where $x \cdot r$ is the standard dot-product of k -bit strings. Again, this scheme will be semantically secure with probability $\Omega(1/k^2)$ over the choice of N .

STEP 3: To boost the security of the “weak” encryption scheme, we define a new scheme where the public key is k^3 random k -bit strings N_1, \dots, N_{k^3} (with overwhelming probability, one of these is a Blum integer), and to encrypt a bit b , we pick random b_1, \dots, b_{k^3} such that $b = b_1 \oplus \dots \oplus b_{k^3}$ and concatenate the encryptions of b_1, \dots, b_{k^3} under the respective public keys N_1, \dots, N_{k^3} . By a standard argument (c.f. [Y82, DP92]), this encryption scheme is semantically secure in the standard sense.

Analysis. Indeed, we claim something stronger – that the encryption scheme derived in Step 3 is a trapdoor simulatable PKE.

- (Oblivious sampling & trapdoor invertibility for key generation) This is trivial, since a random public key corresponds to a string in $\{0, 1\}^{4k}$. We can clearly sample such a public key without learning the secret key.
- (Oblivious sampling & trapdoor invertibility for random ciphertext) For simplicity, we present the algorithms for sampling random ciphertext for the scheme obtained in Step 2. Here, sampling is easy: on input the public key N , pick $\gamma \in Z_N^*, s \in \{0, 1\}^k, \beta \in \{0, 1\}$ and output (γ^{2^k}, s, β) . To implement reverse sampling, we need an efficient algorithm that given $\text{factor}(N)$ and $x \in Q_N$, output a random element of the set $\{\gamma \in Z_N^* \mid \gamma^{2^k} = \pi_N(x) = x^{2^{k+1}}\}$. This can be accomplished as follows: pick a random $\eta \in Z_N^*$ and output $x^2 \cdot \eta / (\eta^{2^k})^{((q+1)/2)^k}$, where q is as before the largest odd divisor of $\phi(N)$. This works because $\eta / (\eta^{2^k})^{((q+1)/2)^k}$ will be a random 2^k 'th root of 1 (mod N).

For the actual proof of security, we will need to show that if N is a random Blum integer, then the following distributions are computationally indistinguishable for every b :

$$\{(N, \gamma, \pi_N(x), r, (x \cdot r) \oplus b)\} \text{ and } \{(N, \gamma, \gamma^{2^k}, r, \beta)\}$$

The first distribution corresponds to an encryption of b using modulus N and randomness (x, r) along with γ the output of rRndEnc (a random solution to the equation $\gamma^{2^k} = \pi_N(x)$). The second corresponds to an obviously generated ciphertext along with the randomness. If there exists an efficient distinguisher, then there exists an efficient procedure A that on input N, γ , outputs $\pi_N^{-1}(\gamma^{2^k})$ with noticeable probability. Since squaring is a bijection on quadratic residues modulo Blum integers, the output of A is also the 4th root of γ^2 . We may then use a reduction in [G04, Section C.1] to derive from A an algorithm for factoring N with noticeable probability.

7 Oblivious Transfer and MPC

We describe the construction underlying Theorem 3 which proceeds in two steps:

STEP 1: We begin with the [CLOS02] construction of a semi-honest OT protocol as applied to our non-committing encryption scheme, and observe that the protocol is secure against malicious senders. For that, we just need to show how to extract the sender's input when the receiver is honest. In this case, the simulator will generate the public keys sent by the receiver in the first message along with the secret keys, so that it can then extract the malicious sender's input by decrypting.

STEP 2: Next, we apply the compiler in [CDSMW09] to "boost" the security guarantee from tolerating semi-honest receivers to tolerating malicious receivers. (Note that we will not need to apply OT reversal as in [CDSMW09].)

Acknowledgements. We thank Ran Canetti, Yuval Ishai, Jonathan Katz, and Chris Peikert for helpful discussions and clarifications.

References

- [B88] Bach, E.: How to generate factored random numbers. *SIAM J. Comput.* 17(2), 179–193 (1988)
- [B97] Beaver, D.: Plug and play encryption. In: Kaliski Jr., B.S. (ed.) *CRYPTO 1997*. LNCS, vol. 1294, pp. 75–89. Springer, Heidelberg (1997)
- [B98] Beaver, D.: Adaptively secure oblivious transfer. In: Ohta, K., Pei, D. (eds.) *ASIACRYPT 1998*. LNCS, vol. 1514, pp. 300–314. Springer, Heidelberg (1998)
- [BH92] 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)
- [C00] Canetti, R.: Security and composition of multiparty cryptographic protocols. *J. Cryptology* 13(1), 143–202 (2000)
- [CDSMW09] Choi, S.G., Dachman-Soled, 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)
- [CFGN96] Canetti, R., Feige, U., Goldreich, O., Naor, M.: Adaptively secure multi-party computation. In: *STOC*, pp. 639–648 (1996), Longer version http://www.wisdom.weizmann.ac.il/~naor/PAPERS/nce_abs.html
- [CHK05] Canetti, R., Halevi, S., Katz, J.: Adaptively-secure, non-interactive public-key encryption. In: Kilian, J. (ed.) *TCC 2005*. LNCS, vol. 3378, pp. 150–168. Springer, Heidelberg (2005)
- [CLOS02] Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: *STOC*, pp. 494–503 (2002)
- [DN00] Damgård, I.B., 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)
- [DP92] De Santis, A., Persiano, G.: Zero-knowledge proofs of knowledge without interaction. In: *FOCS*, pp. 427–436 (1992)
- [FF02] Fischlin, M., Fischlin, R.: The representation problem based on factoring. In: Preneel, B. (ed.) *CT-RSA 2002*. LNCS, vol. 2271, pp. 96–113. Springer, Heidelberg (2002)

- [G04] Goldreich, O.: Foundations of Cryptography: Basic Applications, vol. II. Cambridge University Press, Cambridge (2004)
- [GM84] Goldwasser, S., Micali, S.: Probabilistic encryption. *J. Comput. Syst. Sci.* 28(2), 270–299 (1984)
- [GM04] Granville, A., Martin, G.: Prime number races (2004), <http://arxiv.org/abs/math/0408319>
- [GPV08] Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: *STOC*, pp. 197–206 (2008)
- [GWZ09] 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)
- [H99] Halevi, S.: Efficient commitment schemes with bounded sender and unbounded receiver. *J. Cryptology* 12(2), 77–89 (1999)
- [IPS08] 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)
- [JL00] Jarecki, S., Lysyanskaya, A.: Adaptively secure threshold cryptography: Introducing concurrency, removing erasures (Extended abstract). In: Preneel, B. (ed.) *EUROCRYPT 2000*. LNCS, vol. 1807, pp. 221–242. Springer, Heidelberg (2000)
- [K02] Kalai, A.: Generating random factored numbers, easily. In: *SODA*, pp. 412–412 (2002)
- [KO04] Katz, J., Ostrovsky, R.: Round-optimal secure two-party computation. In: Franklin, M. (ed.) *CRYPTO 2004*. LNCS, vol. 3152, pp. 335–354. Springer, Heidelberg (2004)
- [PVW08] Peikert, C., Vaikuntanathan, V., Waters, B.: A framework for efficient and composable oblivious transfer. In: Wagner, D. (ed.) *CRYPTO 2008*. LNCS, vol. 5157, pp. 554–571. Springer, Heidelberg (2008)
- [RS94] Rubinfeld, M., Sarnak, P.: Chebyshev's bias. *Experiment. Math* 3(3), 173–197 (1994)
- [S96] Schnorr, C.-P.: Security of 2^t -root identification and signatures. In: Koblitz, N. (ed.) *CRYPTO 1996*. LNCS, vol. 1109, pp. 143–157. Springer, Heidelberg (1996)
- [Y82] Yao, A.C.-C.: Theory and applications of trapdoor functions. In: *FOCS*, pp. 80–91 (1982)

Non-malleable Statistically Hiding Commitment from Any One-Way Function

Zongyang Zhang, Zhenfu Cao, Ning Ding, and Rong Ma

Department of Computer Science and Engineering,
Shanghai Jiao Tong University, P.R. China
{zongyangzhang,zfcao,dingning,marong}@sjtu.edu.cn

Abstract. We give a construction of non-malleable statistically hiding commitments based on the existence of one-way functions. Our construction employs statistically hiding commitment schemes recently proposed by Haitner and Reingold [1], and special-sound WI proofs. Our proof of security relies on the message scheduling technique introduced by Dolev, Dwork and Naor [2], and requires only the use of black-box techniques.

1 Introduction

A commitment scheme is an interactive protocol between two parties, the committer, who holds a value, and the receiver. It usually consists of two phases: the commit phase and the reveal phase. During the commit phase, the committer puts a value in a “locked box” and sends it to the receiver. In the reveal phase, the committer sends the “key” to the receiver, then the receiver opens the box and retrieves the value. Two basic properties of a commitment scheme are the hiding property (the receiver cannot learn the committed value before the reveal phase) and the binding property (the committer is bounded to one value after the commit phase). There are two fundamental types of commitment schemes, statistical hiding and statistical binding. In this work, we focus mainly on statistically hiding commitment schemes, where the hiding property holds against unbounded receivers while the binding property is required to hold only against polynomially bounded senders.

The concept of non-malleability was first introduced by Dolev et al. [2]. The basic properties of commitment schemes cannot prevent malleable attacks mounted by a man-in-the-middle adversary who has full control of the communication channel between the committer and the receiver. Loosely speaking, a commitment scheme is non-malleable if one cannot transform the commitment of a value into a commitment of a related value. This kind of non-malleability is called *non-malleability with respect to commitment* [3]. The notion of non-malleability used by Di Crescenzo et al. [4] is called *non-malleability with respect to opening*, i.e., the adversary cannot construct a commitment from a given one, such that after having seen the opening of the original commitment, the adversary is able to correctly open his commitment with a related value. In the rest of this paper, when we say non-malleability, we actually mean non-malleability with respect to opening.

1.1 Related Work

Statistically hiding commitment schemes were first shown to exist based on number-theoretic assumptions [5,6], or more generally, based on any collection of claw-free permutations [7] with an efficiently-recognizable index set [8]. Subsequent work on constructing statistically hiding commitment schemes are based on collision-resistant hash functions [9], or based on any one-way permutation [10], or based on regular one-way functions [11]. Nguyen et al. [12] and Haitner and Reingold [1] made fundamental progress by constructing statistically hiding commitment schemes based on the minimal cryptographic assumption that one-way functions exist.

Based on number-theoretic assumptions, non-malleable statistically hiding commitment schemes were designed in [13,3] assuming the existence of a common reference string that is shared by the two players before the protocol execution. Thus, their schemes do not work in the plain model (i.e., without setup assumptions). More recently, Pass and Rosen [14] constructed a non-malleable commitment scheme that was statistically hiding based on a family of collision-resistant hash functions. Their scheme is round-efficient and needs only constant-round communication. However, the security proof relies on non-black-box techniques and is not efficient.

As one of the central goals of cryptography is to reduce complexity assumptions for various cryptographic primitives and construct them under more standard assumptions, there remain open questions as to *whether or not non-malleable statistically hiding commitment can be based solely on the existence of one-way functions, and be shown secure relying only on black-box techniques.*

1.2 Our Result

In this paper, we give affirmative answers to both of the questions posed above. We show that the existence of one-way function is a sufficient condition for the existence of non-malleable statistically hiding commitment.

Theorem 1. *If one-way functions exist, then there exists a non-malleable statistically hiding commitment scheme.*

Our commitment scheme uses the commitment scheme [1] to commit to the desired value, but modify the opening process by adding a “trapdoor” that can be extracted and used by the simulator to cheat in the reveal phase, and would not be known to the committer in a real execution. Although the extraction requires rewinding, we rely on the message scheduling technique of Lin et al. [15], which is a slight modification of the message scheduling technique introduced by Dolev et al. [2], to show this will suffice to prove the non-malleability. Our proof requires only standard black-box techniques. As a tradeoff, however, our protocol needs polynomial rounds of interaction.

The preliminaries and definitions are illustrated in section 2. Our non-malleable statistically hiding commitment scheme is shown in section 3.

2 Preliminaries and Definitions

For any NP languages L , note that there is a natural witness relation R_L containing pairs (x, w) where w is the witness for the membership of x in L . A function $\mu(\cdot)$, where $\mu : \mathbb{N} \rightarrow [0, 1]$ is called *negligible* if for every positive polynomial $p(\cdot)$, for all sufficiently large $n \in \mathbb{N}$, $\mu(n) < \frac{1}{p(n)}$. A *probability ensemble* is a sequence $X = \{X_i\}_{i \in I}$ of random variables, where I is a countable index set and X_i is a random variable ranging over $\{0, 1\}^{p(|i|)}$ for some polynomial $p(\cdot)$. Two probability ensembles $X = \{X_i\}_{i \in I}$ and $Y = \{Y_i\}_{i \in I}$ are *computationally indistinguishable*, if no probabilistic polynomial-time (PPT) algorithm distinguishes between them with more than negligible probability. For page limited, we assume the readers are familiar with interactive proofs.

Special-sound proofs. A 3-round public-coin interactive proof for the language $L \in \text{NP}$ with witness relation R_L is **special-sound** with respect to R_L , if for any two accepting transcripts (α, β, γ) and $(\alpha', \beta', \gamma')$ for some statement $x \in L$, such that $\alpha = \alpha'$ and $\beta \neq \beta'$, a witness w such that $(x, w) \in R_L$ can be computed by a polynomial-time deterministic procedure.

2.1 Witness Indistinguishability

The concept of witness indistinguishability was proposed by Feige and Shamir [16]. An interactive proof system is witness indistinguishable (WI) if the verifier cannot tell which of the witnesses is being used by the prover to carry out the proof, even if the verifier knows both witnesses. We focus on NP languages L with a corresponding witness relation R_L . The readers are referred to [16] for formal definition.

Special-sound WI proofs for NP languages can be based on the existence of non-interactive commitment schemes. Assuming only one-way functions, 4-round special-sound WI proofs for NP languages exist.¹ More precisely, there is a 3-round special-sound WI proof for the language of Hamiltonian Graphs [17], assuming one-way permutation families exist. If the commitment scheme used by the protocol [17] is replaced by Naor's commitment scheme [18], then it becomes a 4-round special-sound WI proof while the assumption is reduced to the existence of one-way functions. For simplicity, we use 3-round special-sound WI proofs in our protocol though our proof works also with 4-round special-sound WI proofs.

2.2 Commitment Schemes

In this work, we consider statistically hiding commitment schemes.

Definition 1 (Commitment Scheme). A pair of PPT interactive machines $\langle C, R \rangle$ is said to be a commitment scheme if the following two properties hold:

¹ A 4-round protocol is special sound if there exists polynomial-time deterministic procedure to extract the witness from any two accepting transcripts $(\tau, \alpha, \beta, \gamma)$ and $(\tau', \alpha, \beta, \gamma)$ such that $\tau = \tau', \alpha = \alpha'$ and $\beta \neq \beta'$.

Statistical hiding: For every unbounded interactive Turing machine R^* , it holds that the ensemble $\{\text{sta}_{\langle C,R \rangle}^{R^*}(v_1, z)\}_{v_1 \in \{0,1\}^n, n \in \mathbb{N}, z \in \{0,1\}^*}$ and the ensemble $\{\text{sta}_{\langle C,R \rangle}^{R^*}(v_2, z)\}_{v_2 \in \{0,1\}^n, n \in \mathbb{N}, z \in \{0,1\}^*}$ have negligible statistical difference,² where $\text{sta}_{\langle C,R \rangle}^{R^*}(v, z)$ denotes the random variable describing the output of R^* after receiving a commitment to v using $\langle C, R \rangle$.

Computational binding: A malicious (expected) PPT committer S^* can succeed in opening a given commitment in two different ways only with negligible probability. The reader is referred to [19, 7] for more details.

2.3 Non-malleable Commitments

As stated in [14], we formalize the notion of non-malleability by a comparison between a *man-in-the-middle* execution and a *simulated* execution. Just as [2, 15], we consider a tag-based variant of non-malleability.

Let $\langle C, R \rangle$ be a commitment scheme. Let $n \in \mathbb{N}$ be a security parameter. Let $\mathcal{R} \in \{0, 1\}^n \times \{0, 1\}^n$ be a polynomial-time computable valid relation [13] (i.e., for all $v \in \{0, 1\}^n$, $\mathcal{R}(v, \perp) = 0$). In the man-in-the-middle execution, the adversary A is simultaneously participating in a left and right interaction. In the left interaction, the man-in-the-middle adversary A interacts with the committer C to receive a commitment to a value v using tag tag . In the right interaction, A interacts with the receiver R and tries to commit to a related value using tag of its choice $\tilde{\text{tag}}$. After commit phase execution in both interactions, A receives decommitment keys from C and then generates the corresponding decommitment key for \tilde{v} . Prior to the interaction, the value v is given to C as local input. A receives an auxiliary input z , which might contain a priori information about v . If the right commitment or decommitment fails, or $\text{tag} = \tilde{\text{tag}}$, \tilde{v} is set to $= \perp$. Let the boolean random variable $\text{mim}_{\text{open}}^A(\mathcal{R}, v, z)$ denote whether A succeeds. Note $\text{mim}_{\text{open}}^A(\mathcal{R}, v, z) = 1$ if and only if A decommits to a value \tilde{v} such that $\mathcal{R}(v, \tilde{v}) = 1$.

In the simulated execution, a simulator S directly interacts with honest receiver R . As in the man-in-the-middle execution, the value v is chosen prior to the interaction, and S receives some a priori information about v as part of its auxiliary input z . S also receives tag tag . S first executes the commitment scheme with R . Once the commitment phase has been completed, S receives the value v and attempts to decommit to a value \tilde{v} with tag $\tilde{\text{tag}}$. If $\text{tag} = \tilde{\text{tag}}$, \tilde{v} is set to \perp . Let the boolean random variable $\text{sim}_{\text{open}}^S(\mathcal{R}, v, z)$ denote whether S succeeds. Note $\text{sim}_{\text{open}}^S(\mathcal{R}, v, z) = 1$ if and only if S decommits to a value \tilde{v} such that $\mathcal{R}(v, \tilde{v}) = 1$.

Definition 2 (Non-malleable Commitment [14]). A commitment scheme $\langle C, R \rangle$ is said to be non-malleable with respect to opening if for every PPT man-in-the-middle adversary A , there exists an expected PPT simulator S and a negligible function $\mu : \mathbb{N} \rightarrow [0, 1]$, such that for every polynomial-time computable

² The statistical difference between two ensembles $\{X_i\}_{i \in I}$ and $\{Y_i\}_{i \in I}$ is defined by $\frac{1}{2} \cdot \sum_{\alpha} |\Pr[X_i = \alpha] - \Pr[Y_i = \alpha]|$.

valid relation $\mathcal{R} \subseteq \{0, 1\}^n \times \{0, 1\}^n$, for all tags of polynomial length, for every $v \in \{0, 1\}^n$ and every $z \in \{0, 1\}^*$, the following holds:

$$\Pr[\text{mim}_{\text{open}}^A(\mathcal{R}, v, z) = 1] < \Pr[\text{sim}_{\text{open}}^S(\mathcal{R}, v, z) = 1] + \mu(n)$$

A commitment scheme that is non-malleable according to Definition 2 is liberal non-malleable rather than strict non-malleable [2,3]. Note we follow [14] in that non-malleability is guaranteed only if the commit phase and the reveal phase do not overlap.

3 Construction

We begin by presenting a high-level overview of our protocol. Our protocol is based on the statistically hiding commitment scheme [1] while relying on the messages scheduling technique [15] which is a slight modification of the message scheduling technique of [2]. The commit phase of our protocol is the same as that of the commitment protocol in [1]. The reveal phase, however, comes in two parts. Roughly, the reveal phase employs the two-witness technique by Feige [20] and the well known FLS-technique [21]. First, the receiver proves that it knows one of the preimages of either element s_0 or element s_1 computed by itself in the domain of a one-way function. Then, the committer sends the committed value v and proves it knows how to open the commitment or one of the preimages of either element s_0 or element s_1 . The proofs used by the prover and the verifier are all tag-based WI proofs elaborately scheduled as [15]. For simplicity of exposition, our description relies on the existence of one-way functions with efficiently recognizable range.³ We also assume the one-way function is length-preserving. Since any one-way function can be transformed into length-preserving one-way function [19].

3.1 Tag-Based Witness-Indistinguishable Proof

First, we propose a tag-based WI proof for every NP language L which is used as a basic tool in the final commitment scheme. The length of the tag is polynomial bounded to the length of the security parameter n . Denote the polynomial by $t(\cdot)$. In Fig. 1, both design_0 and design_1 contain two executions of special-sound WI proofs for L but with elaborately designed scheduling. The tag-based WI proof $\langle P_{\text{tag}}, V_{\text{tag}} \rangle$ for L is shown in Fig. 2. The protocol is composed of $4t$ -round special-sound WI proofs for language L . More precisely, there are t rounds, where in round j , the schedule $\text{design}_{\text{tag}_j}$ is followed by $\text{design}_{1-\text{tag}_j}$. The properties of $\langle P_{\text{tag}}, V_{\text{tag}} \rangle$ are easy to verify. The details are omitted.

One basic technique in proving the security of most zero-knowledge and commitment protocols is standard rewinding. However, the rewinding technique is problematic when extending to concurrent (here one-left one-right) execution environment as an adversary may adaptively schedule its messages that withstand any targeted simulator (i.e., the simulator may run super-polynomial time or is

³ The protocol can be easily modified to work with arbitrary one-way function by providing a witness hiding proof that an element is in the range of the one-way function.

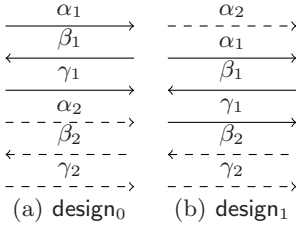


Fig. 1. Two schedules

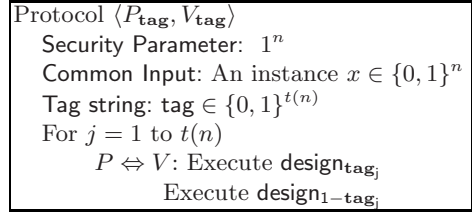


Fig. 2. Tag-based WI proof $\langle P_{\text{tag}}, V_{\text{tag}} \rangle$

exposed to malleability attack.). Considering the non-malleability property for commitment schemes, the pivot is to design the stand-alone simulator that satisfying Definition 2. Here we also come up with the problem of how to simulate when the adversary adaptively schedules its messages.

The scheduling in Fig. 1 which is identical to [15] is vital in achieving the non-malleability. The main advantage of this scheduling is that for the proof given by a man-in-the-middle adversary, there exists a point at which the adversary cannot answer the challenge from the verifier by simply modifying the proof on the other side (provided the tag of the proof is different from that of the proof on the other side.).

Related to the above scheduling is a notion called **safe-point**, from which it is possible to perform extraction by standard rewinding until we obtain a second proof transcript, without “affecting” the other side interaction. Below is the formal definition of safe-point, which is mainly taken from [15] and abridged to our setting.

Definition 3 (Safe-point [15]). A prefix ρ of a transcript τ is called a **safe-point**, if there exists an accepting proof $(\alpha_r, \beta_r, \gamma_r)$ in the right interaction, such that

1. α_r occurs in ρ , but not β_r (and γ_r).
2. For any proof $(\alpha_l, \beta_l, \gamma_l)$ in the left interaction, if only α_l occurs in ρ , then β_l occurs after γ_r .

When protocol $\langle P_{\text{tag}}, V_{\text{tag}} \rangle$ is run concurrently, it is guaranteed there is a **safe-point** for right interaction that has a tag different from the left interaction following from the next lemma.

Lemma 1 (Safe-point Lemma [15]⁴). In any one-one man-in-the-middle execution of $\langle P_{\text{tag}}, V_{\text{tag}} \rangle$, if the right interaction has a different tag from the tag of the left interaction, there exists a **safe-point** for the right interaction.

⁴ The safe-point lemma in [15] applies to any one-many concurrent execution environment, where the adversary participates in one left interaction and polynomial many right interactions. Here we use a simpler version of the safe-point lemma, where the adversary participates in one left interaction and one right interaction.

3.2 Non-malleable Statistically Hiding Commitment Scheme

Let $\langle \text{SHC}, \text{SHR} \rangle$ be the statistically hiding commitment scheme [1] from any one-way function [5] and let $\langle P_{\text{tag}}, V_{\text{tag}} \rangle$ be a tag-based WI proof for NP. The commitment protocol is shown in Fig. 3. The length of the tag is $m(n)$. Our construction in fact compiles any statistically hiding commitment scheme with non-interactive reveal phase into a non-malleable statistically hiding one with interactive reveal phase, assuming the existence of one-way functions.

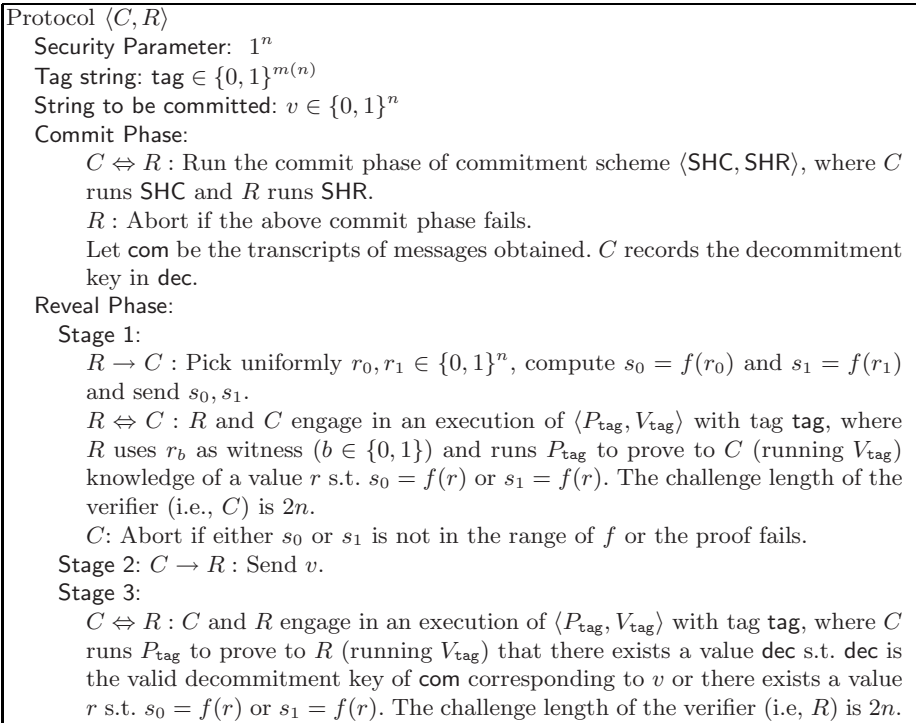


Fig. 3. Non-malleable statistically hiding commitment scheme $\langle C, R \rangle$

Theorem 2. *Suppose that $\langle \text{SHC}, \text{SHR} \rangle$ is a statistically hiding commitment scheme with non-interactive reveal phase and $\langle P_{\text{tag}}, V_{\text{tag}} \rangle$ is a tag-based WI proof. Then $\langle C, R \rangle$ is a non-malleable statistically hiding commitment scheme.*

Remark 1. The commitment scheme shown in Fig. 3 is tag-based non-malleable. Compared with existing tag-based commitment schemes [2, 15, 22], it seems a bit

⁵ Note the commitment scheme [1] is only for a single bit. By running their scheme in parallel, we obtain a commitment scheme of any polynomial length. Hence, we also assume that the basic statistically hiding commitment scheme is for a string.

strange that our construction uses tags only in the reveal phase. In fact, this approach is inspired by the work of [14,15]. Even tag-based non-malleable commitments can be transformed into content-based non-malleable commitments in a standard way [2], we explicitly present one in Appendix A for reference.

Remark 2. The high level approach of our commitment scheme is to combine [14] with [2,15]. That is, to commit to v , in the commit phase, a sender commits v using the statistically hiding commitment scheme [1], and in the reveal phase, a sender sends v and proves using a “simulation-extractable” argument [2,15] that the commit phase transcript opens to v . The simulation strategy at a high level is from [14]. For technical reasons, naively using the simulation-extractable arguments from [2,15] does not work. We need to modify the opening process by adding a “trapdoor” that can be extracted and used by the simulator to cheat in the reveal phase. This is the reason why we add one more phase (i.e., Stage 1). Whereas in [2,15], the trapdoor is only used in the hybrid experiment for analysis and may therefore hard-wired via a different analysis.

Proof (sketch). We need to prove the scheme satisfies the following three properties: statistical hiding, computational binding and non-malleability with respect to opening. We start by proving the hiding and non-malleability properties and then return to the proof of the binding property.

Statistical hiding. The hiding property follows directly from the hiding property of the commitment scheme $\langle \text{SHC}, \text{SHR} \rangle$. Note that $\langle \text{SHC}, \text{SHR} \rangle$ is statistically hiding, and so $\langle C, R \rangle$ is also statistically hiding.

Non-malleability. We show that for every PPT man-in-the-middle adversary A , there exists a probabilistic expected polynomial-time simulator S and a negligible function μ such that for every polynomial-time computable relation $\mathcal{R} \subseteq \{0, 1\}^n \times \{0, 1\}^n$, for every tag tag of length $m(n)$, for every $v \in \{0, 1\}^n$ and every $z \in \{0, 1\}^*$, it holds that

$$\Pr[\text{mim}_{\text{open}}^A(\mathcal{R}, v, z) = 1] < \Pr[\text{sim}_{\text{open}}^S(\mathcal{R}, v, z) = 1] + \mu(n) \quad (1)$$

Denote by A_{rev} the state of A after the the commit phase, i.e., A_{rev} contains A 's description along with its configuration at that time just before the reveal phase starts.

We proceed to describing the simulator S . S on input z and security parameter 1^n interacts with an honest receiver R and runs the adversary A internally. During the commit phase, on a high level, S internally incorporates A and emulates the commit phase of the left execution for adversary A by honestly committing to 0^n , while externally relaying messages in the right execution between A and R .

Once the commit phase is finished, S receives a value v and has to perform the reveal phase internally with A_{rev} . In Stage 1, S plays as an honest sender in the left reveal phase and as an honest receiver in the right reveal phase. Once the simulation of Stage 1 completes, S applies the safe-point lemma to find a

safe-point and extract a witness w to the statement proved by A_{rev} in the left reveal phase by standard rewinding.⁶ In Stage 2, S just sends v to A_{rev} in the left reveal phase. Then the simulation for Stage 3 begins. S uses a fake witness (i.e. the trapdoor w) to simulate the left interaction for A_{rev} , while emulating the right interaction as an honest receiver. When the simulation for Stage 3 completes, S again applies the safe-point lemma to find a safe-point and extract a witness \tilde{w} (i.e., the decommitment keys of A) in the right interaction. Finally, by using \tilde{w} , S can complete the reveal phase of the external execution with R .

More formally, S proceeds as follows on auxiliary input z and tag tag :

1. S internally incorporates $A(z)$.
2. During the commit phase S proceeds as follows:
 - (a) S internally emulates left interaction for A by honestly committing to 0^n .
 - (b) Messages from right execution are forwarded externally to R .
3. Once the commit phase has finished, S receives the value v . Let $\text{com}, \widetilde{\text{com}}$ denote the left and right execution transcripts respectively.
4. During the reveal phase S internally incorporates A_{rev} and proceeds as follows:
 - (a) **Stage 1 Main Execution Phase:** S emulates a one-one man-in-the-middle execution by playing as honest sender with tag tag on the left and as honest receiver on the right. After completing the execution, denote by Δ the transcripts of messages obtained. Denote the right tag by $\tilde{\text{tag}}$. We emphasize here that S can emulate left interaction independent of v in Stage 1.

Stage 1 Rewinding Phase: Next, S attempts to extract the witness used by A_{rev} on the left if $\text{tag} \neq \tilde{\text{tag}}$.

- i. In Δ , find the first point ρ that is a safe-point. Let the associated proof be $(\alpha_\rho, \beta_\rho, \gamma_\rho)$.
- ii. Repeat until a second proof transcript $(\alpha_\rho, \beta'_\rho, \gamma'_\rho)$ is obtained: Emulate the left interaction as in the Stage 1 Main Execution phase. For the right interaction:
 - If A_{rev} expects to get a new proof from the right receiver, S then emulates the proof by generating design_0 himself. Forward one of the two proofs internally.
 - If A_{rev} sends a challenge for a proof whose first message occurs in ρ : cancel the execution, rewind to ρ and continue.
- iii. If $\beta_\rho \neq \beta'_\rho$, extract and record the witness w from $(\alpha_\rho, \beta_\rho, \gamma_\rho)$ and $(\alpha_\rho, \beta'_\rho, \gamma'_\rho)$. Otherwise halt and output fail.

Finally, if the above (i.e. step 4a) runs for more than 2^n steps, halt and output fail.

⁶ In Stage 1, the committer acts as a prover and the receiver acts as a verifier. The safe-point and safe-point lemma still work by interchanging right and left.

- (b) **Stage 2:** Send v to the adversary A_{rev} .
- (c) **Stage 3 Main Execution Phase:** By using w as witness, S can easily simulate left interaction for A_{rev} . The right interaction is emulated by S adopting honest receiver strategy. After completing the execution, denote by Δ' the transcripts of messages obtained in the execution of Stage 2 and Stage 3 .

Stage 3 Rewinding Phase: S attempts to extract the decommitment key of A_{rev} on the right:

- i. In Δ' , find the first point $\tilde{\rho}$ that is a **safe-point**. Let the associated proof be $(\tilde{\alpha}_{\tilde{\rho}}, \tilde{\beta}_{\tilde{\rho}}, \tilde{\gamma}_{\tilde{\rho}})$.
- ii. Repeat until a second proof transcript $(\tilde{\alpha}_{\tilde{\rho}}, \tilde{\beta}'_{\tilde{\rho}}, \tilde{\gamma}'_{\tilde{\rho}})$ is obtained: Emulate the right interaction as in the **Stage 3 Main Execution Phase**. For the left interaction:
 - If A_{rev} expects to get a new proof from the committer, S is free to answer the request by using the witness w , except when A_{rev} sends a challenge for a proof whose first message occurs in $\tilde{\rho}$, S cancels the execution, rewinds to $\tilde{\rho}$ and continues.
- iii. If $\tilde{\beta}_{\tilde{\rho}} \neq \tilde{\beta}'_{\tilde{\rho}}$, extract a witness \tilde{w} from $(\tilde{\alpha}_{\tilde{\rho}}, \tilde{\beta}_{\tilde{\rho}}, \tilde{\gamma}_{\tilde{\rho}})$ and $(\tilde{\alpha}_{\tilde{\rho}}, \tilde{\beta}'_{\tilde{\rho}}, \tilde{\gamma}'_{\tilde{\rho}})$. Otherwise halt and output fail.
- iv. If \tilde{w} is a valid decommitment key for $\langle \text{SHC}, \text{SHR} \rangle$, i.e., $(\widetilde{\text{com}}, \tilde{w}, \tilde{v})$ is a legal transcript for $\langle \text{SHC}, \text{SHR} \rangle$, set $\widetilde{\text{rev}} = \tilde{w}$. Otherwise halt and output fail.

Finally, if the above (step 4b) runs for more than 2^n steps, halt and output fail.

- (d) If the right interaction is accepting and $\text{tag} \neq \widetilde{\text{tag}}$, and $\widetilde{\text{rev}}$ contains a valid decommitment key, run the honest committer strategy on input $\widetilde{\text{com}}$ and decommitment key $\widetilde{\text{rev}}$, value \tilde{v} with tag $\widetilde{\text{tag}}$.

Running time of S . We show that the running time of S is expected PPT. Note the time spent by S in the commit phase is $\text{poly}(n)$. After S extracts the witness \tilde{w} , the time spent by S in step 4d is also $\text{poly}(n)$. Next, we show that the expected time spent by S in the reveal phase (except running time in step 4d) is also $\text{poly}(n)$. For simplicity, we assume that S does not check the fail condition and may run for more than 2^n steps (since this only increases the total running time).

Recall that in the reveal phase, S rewinds A from two safe points. We need to show the time spent in step 4a and step 4c are all expected PPT. We first analyze the time spent in step 4a during the simulation. Then using the same method, we show that the time spent in step 4c is also expected PPT.

Note the time spent by S in the Stage 1 Main Execution Phase is $\text{poly}(n)$. We then show the time spent in Stage 1 Rewinding Phase is expected PPT. The analysis hereafter is similar to that in 115 but is simpler. Let $T(i)$ be the random variable that describes the time spent in rewinding a proof after i messages have been exchanged. We show that $E[T(i)] \leq \text{poly}(n)$ and then by linearity of expectation, we conclude that the expected time spent by S in the Stage 1 Rewinding Phase is $\sum_i E[T(i)] \leq \sum_i \text{poly}(n) \leq \text{poly}(n)$.

Next we will bound the time $E[T(i)]$. Given a partial transcript of messages ρ , let $\Pr[\rho]$ denote the probability that ρ occurs as a prefix of the execution emulated in **Stage 1 Main Execution Phase**. Let p_ρ denote the probability that ρ is a **safe-point**⁷ and is rewound. From the construction of S , we know that S keeping rewinding until it finds another accepting transcript $(\alpha_\rho, \beta'_\rho, \gamma'_\rho)$ for ρ , canceling each rewinding for which ρ is not a **safe-point**, i.e., A_{rev} requests the second message of a proof in the right-interaction whose first message occurs in ρ . As the emulated committer and receiver act identically as real committer and real receiver in this stage, conditioned on ρ , a view occurring in a rewinding from ρ is same as occurring in the **Stage 1 Main Execution Phase**. Thus, the probability of canceling a rewinding from ρ is at most $1-p_\rho$. Furthermore, the expected number of rewindings is at most $\frac{1}{p_\rho}$. Therefore, the expected number of rewindings from ρ is at most $p_\rho \cdot \frac{1}{p_\rho} = 1$ and each rewinding takes at most $\text{poly}(n)$ steps, i.e., $E[T(i)|\rho] \leq \text{poly}(n)$. Thus,

$$E[T(i)] = \sum_{\rho \text{ of length } i} E[T(i)|\rho] \cdot \Pr[\rho] \leq \text{poly}(n) \cdot \sum_{\rho \text{ of length } i} \Pr[\rho] \leq \text{poly}(n)$$

The expected running time of S in step **4c** is also polynomial-time using similar analysis as above. We omit the details.

Analysis of the simulator S . In order to show equation **(II)**, we define a hybrid stand-alone simulator HYB_1 that also receives v as auxiliary input. HYB_1 proceeds exactly as S except that in the commit phase, instead of feeding A a commitment to 0^n , HYB_1 feeds A a commitment to v .

Since both the experiment S and HYB_1 are efficiently computable, the following claim follows directly from the hiding property of $\langle \text{SHC}, \text{SHR} \rangle$.

Claim 1. *There exists some negligible function μ' such that*

$$\left| \Pr[\text{sim}_{\text{open}}^S(\mathcal{R}, v, z) = 1] - \Pr[\text{sim}_{\text{open}}^{\text{HYB}_1}(\mathcal{R}, v, z) = 1] \right| < \mu'(n)$$

Next we proceed to showing the following claim.

Claim 2. *There exists some negligible function μ'' such that*

$$\left| \Pr[\text{mim}_{\text{open}}^A(\mathcal{R}, v, z) = 1] - \Pr[\text{sim}_{\text{open}}^{\text{HYB}_1}(\mathcal{R}, v, z) = 1 | \text{-fail}] \right| < \mu''(n)$$

Proof (sketch). Note the view of A in the commit phase in a real interaction is identical to the view of A in HYB_1 . Furthermore, HYB_1 feeds A messages according to the correct distribution in **Stage 1**, the view of A_{rev} in the simulation

⁷ Note the roles of C and R interchange in **Stage 1** where C acts as a verifier and R acts as a prover. The **safe-point lemma** will be used by interchanging the right and the left.

of Stage 1 by experiment HYB_1 is identical to the view of A_{rev} in a real interaction. The view of A_{rev} in the simulation of Stage 3 by HYB_1 is computationally indistinguishable following from the witness-indistinguishability of $\langle P_{\text{tag}}, V_{\text{tag}} \rangle$. As the safe-point lemma shows, when the right interaction has a different tag from the left interaction, there is a safe-point. Hence, according to the actions of HYB_1 , it will either output fail or succeed in the extraction from A_{rev} . Conditioned on HYB_1 not outputting fail, by the computational-binding property of $\langle \text{SHC}, \text{SHR} \rangle$, except with negligible probability, the witness \tilde{w} and the value \tilde{v} extracted by HYB_1 are the valid decommitment key and committed value of A , respectively.

We next show $\left| \Pr[\text{sim}_{\text{open}}^{\text{HYB}_1}(\mathcal{R}, v, z) = 1] - \Pr[\text{sim}_{\text{open}}^{\text{HYB}_1}(\mathcal{R}, v, z) = 1 | \neg \text{fail}] \right|$ is negligible by proving that the probability that event fail happens is negligible. This together with Claim [1](#) and Claim [2](#) conclude Eq. [\(II\)](#).

Claim 3. HYB_1 outputs fail with negligible probability.

Proof. The proof of this claim is similar to that of [\[15\]](#). More precisely, HYB_1 outputs fail only in three cases: HYB_1 runs for more than 2^n steps; or the same proof transcript is obtained from some safe-point; or the witness extracted is not a valid decommitment. The arguments of the first two cases are almost the same as those in [\[15\]](#). The main difference lies in the analysis of the third case.

HYB₁ runs for more than 2^n steps: We know that the expected running time of HYB_1 and S are same, i.e., $\text{poly}(n)$. Using Markov inequality, we conclude that the probability that HYB_1 runs more than 2^n steps is at most $\frac{\text{poly}(n)}{2^n}$.

The same proof transcript is obtained from some safe-point: This case occurs if HYB_1 picks some challenge β (resp. $\tilde{\beta}$) in Stage 1 (resp. Stage 3) Rewinding Phase that appeared as a challenge in the Stage 1 (resp. Stage 3) Main Execution Phase. As HYB_1 runs for at most 2^n steps, it picks at most 2^n challenges. Furthermore, the length of each challenge is $2n$. By applying the union bound, we obtain that the probability that a β (resp. $\tilde{\beta}$) is picked twice is at most $\frac{2^n}{2^{2n}}$. Since there are at most polynomial many challenges in Stage 1 (resp. Stage 3), using union bound again, we conclude that the probability that it outputs fail in this case is negligible.

The witness extracted is not a valid decommitment:^{[8](#)} Suppose, on the contrary, the witness extracted is not the decommitment key for $\langle \text{SHC}, \text{SHR} \rangle$, then by the special-sound property, it follows that it must be a value r'

⁸ The proof in this case heavily relies on the “simulation-extractability” property of $\langle P_{\text{tag}}, V_{\text{tag}} \rangle$ in Stage 1. An ordinary WI proof of knowledge is not suffice here, as the problem in this case is reduced to the security of one-way functions or witness-indistinguishability of underlying subprotocols, in the presence of an expected PPT adversary who can *rewind* the same subprotocols.

such that $f(r') = s_{b'}$ for some $b' \in \{0, 1\}$. Denote by r_b ($b \in \{0, 1\}$) the witness used by HYB_1 in Stage 1 of right interaction. If $b' = 1 - b$, then we can break the one-way function f . Given A, z and v , we construct an algorithm B that inverts f . The input to B is an n -bit string $y = f(x)$ where x was chosen randomly from $\{0, 1\}^n$. B wants to output a pre-image of y under f . B proceeds as follows: B runs identically as HYB_1 with inputs z, v with the exception that when simulating the right receiver for A in Stage 1 of reveal phase, it picks a random bit $b \in \{0, 1\}$ and a random string $r_b \in \{0, 1\}^n$, and sets $s_b = f(r_b), s_{1-b} = y$. By using r_b as witness, it can simulate the right interaction with A_{rev} easily. Finally, if B extracts a witness r' where $f(r') = y$, then we break the one-wayness of f . The probability that B inverts f is identical to the probability that HYB_1 inverts f which is non-negligible. This contradicts the one-wayness of f .

We therefore have only to deal with the case that B always outputs r' such that $f(r') = s_b$, i.e., B always outputs same preimage it knows. Then we can break the witness indistinguishability of the underlying special-sound proofs as follows: Recall that the proof $\langle P_{\text{tag}}, V_{\text{tag}} \rangle$ in Stage 1 of right interaction contains $4m$ number of special-sound WI proofs. The above assumption is that B always extracts the same preimage used by itself in Stage 1 of right interaction. We know that if the $4m$ number proofs use r_0 , B outputs r_0 , and if the $4m$ number proofs use r_1 , B outputs r_1 . Applying standard hybrid arguments, there exists $i \in [4m]$, by using r_0 for the first $i - 1$ proofs and r_1 for the last $4m - i$ proofs, the witness used in the i -th special-sound proof is the same as that of the witness extracted by B . We can use this session to break the witness-indistinguishability of special-sound WI proof. The probability we break the witness-indistinguishability property of the underlying special-sound proof is $\frac{1}{4m}$ times the probability that HYB_1 inverts f which is non-negligible. This contradicts the witness-indistinguishability property of the underlying special-sound proof.

Computational binding. The binding property intuitively follows from the binding property of the underlying commitment scheme $\langle \text{SHC}, \text{SHR} \rangle$ and the special-sound property (or more precisely proof of knowledge property) of the underlying proof in $\langle P_{\text{tag}}, V_{\text{tag}} \rangle$. A formal proof proceeds along the lines of the proof of non-malleability. More precisely, suppose, there exists an adversary A that can violate the binding property of $\langle C, R \rangle$, then we design an algorithm A' that violates the binding property of $\langle \text{SHC}, \text{SHR} \rangle$. A' incorporates A and relays the commit phase messages to an external honest receiver SHR . In the reveal phase, there is no need of A' to simulate the left interaction for A . Note in the non-malleability proof, two extraction are executed. Here, we only execute one extraction by standard rewinding, and obtain the decommitment key. Using this information, A' can easily complete the reveal phase with SHR . It follows from the witness-indistinguishability property of $\langle P_{\text{tag}}, V_{\text{tag}} \rangle$ that the probability that A' breaks

the binding property of $\langle \text{SHC}, \text{SHR} \rangle$ is negligible close to the probability that A breaks the binding property of $\langle C, R \rangle$.

Schedule of messages: In the non-malleability proof, the design of S is based on an unspecified assumption, i.e., in the reveal phase, Stage 3 on both interactions will not start unless the simulations for Stage 1 are completed. Without loss of generality, this assumption is reasonable.

Consider the scenario where the simulation for Stage 1 of the left interaction and Stage 3 of the right interaction overlap. The simulation goes well as the adversary runs as a prover in Stage 3 of the right interaction, and the rewinding of Stage 1 of the left interaction will not “rewind” the Stage 3 of the right interaction (i.e., the adversary can only answer the left challenge by itself, without the help from the right interaction). By using the safe-point lemma, the simulator can still find a safe-point and extract the witness to the statement proved by the adversary by standard rewinding. Furthermore, the adversary also runs as a prover in Stage 1 of the left interaction, and the rewinding of Stage 3 of the right interaction will not “rewind” the Stage 1 of the left interaction. Due to a more simpler but similar reason, when the simulation for Stage 3 of the left interaction and Stage 1 of the right interaction overlap, the simulator has no difficulty and the two extractions also performs well. We take a special note of the fact that the safe-point lemma depicts the existence of safe-point in any one-one concurrent execution environment, and considers an environment where one-side of the interaction is empty as a special case.

Acknowledgement

We thank the anonymous reviewers for their valuable comments. Zongyang Zhang thank Huijia Lin for helpful discussions on the definition of safe-point. The work is supported by the National Nature Science Foundation of China No.60673079 and No.60773086, and by the National 973 Program No. 2007CB 311201.

References

1. Haitner, I., Reingold, O.: Statistically-hiding commitment from any one-way function. In: Johnson, D.S., Feige, U. (eds.) STOC, pp. 1–10. ACM, New York (2007)
2. Dolev, D., Dwork, C., Naor, M.: Nonmalleable cryptography. SIAM J. Comput. 30, 391–437 (2000)
3. Fischlin, M., Fischlin, R.: Efficient non-malleable commitment schemes. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 413–431. Springer, Heidelberg (2000)
4. Di Crescenzo, G., Ishai, Y., Ostrovsky, R.: Non-interactive and non-malleable commitment. In: STOC, pp. 141–150 (1998)
5. Boyar, J., Kurtz, S.A., Krentel, M.W.: A discrete logarithm implementation of perfect zero-knowledge blobs. J. Cryptology 2, 63–76 (1990)

6. Brassard, G., Chaum, D., Crépeau, C.: Minimum disclosure proofs of knowledge. *J. Comput. Syst. Sci.* 37(2), 156–189 (1988)
7. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.* 17, 281–308 (1988)
8. Goldreich, O., Kahan, A.: How to construct constant-round zero-knowledge proof systems for NP. *J. Cryptology* 9, 167–190 (1996)
9. Damgård, I., Pedersen, T.P., Pfitzmann, B.: On the existence of statistically hiding bit commitment schemes and fail-stop signatures. *J. Cryptology* 10, 163–194 (1997)
10. Naor, M., Ostrovsky, R., Venkatesan, R., Yung, M.: Perfect zero-knowledge arguments for NP using any one-way permutation. *J. Cryptology* 11, 87–108 (1998)
11. Haitner, I., Horvitz, O., Katz, J., Koo, C.Y., Morselli, R., Shaltiel, R.: Reducing complexity assumptions for statistically-hiding commitment. In: Cramer, R. (ed.) *EUROCRYPT 2005*. LNCS, vol. 3494, pp. 58–77. Springer, Heidelberg (2005)
12. Nguyen, M.H., Ong, S.J., Vadhan, S.P.: Statistical zero-knowledge arguments for NP from any one-way function. In: *FOCS*, pp. 3–14. IEEE Computer Society, Los Alamitos (2006)
13. Di Crescenzo, G., Katz, J., Ostrovsky, R., Smith, A.: Efficient and non-interactive non-malleable commitment. In: Pfitzmann, B. (ed.) *EUROCRYPT 2001*. LNCS, vol. 2045, pp. 40–59. Springer, Heidelberg (2001)
14. Pass, R., Rosen, A.: New and improved constructions of nonmalleable cryptographic protocols. *SIAM J. Comput.* 38, 702–752 (2008)
15. Lin, H., Pass, R., Venkatasubramanian, M.: Concurrent non-malleable commitments from any one-way function. In: Canetti, R. (ed.) *TCC 2008*. LNCS, vol. 4948, pp. 571–588. Springer, Heidelberg (2008)
16. Feige, U., Shamir, A.: Witness indistinguishable and witness hiding protocols. In: *STOC*, pp. 416–426. ACM, New York (1990)
17. Blum, M.: How to prove a theorem so no one else can claim it. In: *Proceedings of the International Congress of Mathematicians*, pp. 1444–1451 (1986)
18. Naor, M.: Bit commitment using pseudorandomness. *J. Cryptology* 4, 151–158 (1991)
19. Goldreich, O.: *The Foundations of Cryptography*, vol. 1. Cambridge University Press, UK (2001)
20. Feige, U.: *Alternative Models for Zero Knowledge Interactive Proofs*. PhD thesis, The Weizmann Institute of Science, Rehovot, Israel (1990)
21. Feige, U., Lapidot, D., Shamir, A.: Multiple noninteractive zero knowledge proofs under general assumptions. *SIAM J. Comput.* 29, 1–28 (1999)
22. MacKenzie, P., Yang, K.: On simulation-sound trapdoor commitments. *Cryptology ePrint Archive*, Report 2003/252 (2003), <http://eprint.iacr.org/>

A A Content-Based Non-malleable Commitment Scheme

Let (SHC, SHR) be the statistically hiding commitment scheme [1] from any one-way function and let $(P_{\text{tag}}, V_{\text{tag}})$ be a tag-based WI proof for all NP. Let $\text{SS} = (\text{SG}, \text{Sig}, \text{SVer})$ be a secure signature scheme. The content-based non-malleable statistically hiding commitment scheme is shown in Fig. 4. Due to page limit, the formal proof is omitted here.

Protocol $\langle C, R \rangle$
 Security Parameter: 1^n
 String to be committed: $v \in \{0, 1\}^n$
 Commit Phase:
 $C \Leftrightarrow R$: Run the commit phase of commitment scheme $\langle \text{SHC}, \text{SHR} \rangle$.
 R : Abort if the above commit phase fails.
 Denote the above transcript as com . C records the decommitment key in dec .
 Reveal Phase:
 Stage 1:
 $R \rightarrow C$: Set $(pk_0, sk_0) \leftarrow \text{SG}(1^n)$ and send pk_0 .
 $R \rightarrow C$: Pick uniformly $r_0, r_1 \in \{0, 1\}^n$, compute $s_0 = f(r_0)$ and $s_1 = f(r_1)$ and send s_0, s_1 .
 $C \Leftrightarrow R$: R and C engage in an execution of $\langle P_{pk_0}, V_{pk_0} \rangle$ with tag pk_0 , where R uses r_b as witness ($b \in \{0, 1\}$) and runs P_{pk_0} to prove to C (running V_{pk_0}) that there exists a value r s.t. $s_0 = f(r)$ or $s_1 = f(r)$. The challenge length of the verifier (i.e., C) is $2n$. C aborts if either s_0 or s_1 is not in the range of f or the proof fails.
 $R \rightarrow C$: Let tr_0 be the transcript so far. Set $\sigma_0 \leftarrow \text{Sig}(tr_0, sk_0)$ and send σ_0 .
 C : Abort if $\text{Sver}(pk_0, tr_0, \sigma_0) \neq 1$.
 Stage 2: $C \rightarrow R$: Send v .
 Stage 3:
 $C \rightarrow R$: Set $(pk_1, sk_1) \leftarrow \text{SG}(1^n)$ and send pk_1 .
 $C \Leftrightarrow R$: C and R engage in an execution of $\langle P_{pk_1}, V_{pk_1} \rangle$ with tag pk_1 , where C uses witness dec and runs P_{pk_1} to prove to R (running V_{pk_1}) that there exists a value dec s.t. dec is the decommitment key of com corresponding to v or there exists a value r s.t. $s_0 = f(r)$ or $s_1 = f(r)$. The challenge length of the verifier (i.e., R) is $2n$.
 $C \rightarrow R$: Let tr_1 be the transcript so far. Set $\sigma_1 \leftarrow \text{Sig}(tr_1, sk_1)$ and send σ_1 .
 R : Abort if $\text{Sver}(pk_1, tr_1, \sigma_1) \neq 1$.

Fig. 4. Non-malleable statistically hiding commitment scheme $\langle C, R \rangle$

Proofs of Storage from Homomorphic Identification Protocols

Giuseppe Ateniese¹, Seny Kamara^{2,*}, and Jonathan Katz^{3,**}

¹ The Johns Hopkins University
ateniese@cs.jhu.edu

² Microsoft Research
senyk@microsoft.com

³ University of Maryland
jkatz@cs.umd.edu

Abstract. Proofs of storage (PoS) are interactive protocols allowing a client to verify that a server faithfully stores a file. Previous work has shown that proofs of storage can be constructed from any homomorphic linear authenticator (HLA). The latter, roughly speaking, are signature/message authentication schemes where ‘tags’ on multiple messages can be homomorphically combined to yield a ‘tag’ on any linear combination of these messages.

We provide a framework for building public-key HLAs from any identification protocol satisfying certain homomorphic properties. We then show how to turn any public-key HLA into a publicly-verifiable PoS with communication complexity independent of the file length and supporting an unbounded number of verifications. We illustrate the use of our transformations by applying them to a variant of an identification protocol by Shoup, thus obtaining the first unbounded-use PoS based on factoring (in the random oracle model).

1 Introduction

Advances in networking technology and the rapid accumulation of information have fueled a trend toward outsourcing data management to external service providers (“servers”). By doing so, organizations can concentrate on their core tasks rather than incurring the substantial hardware, software and personnel costs involved in maintaining data “in house”.

Outsourcing storage prompts a number of interesting challenges. One problem is to verify that the server continually and faithfully stores the entire file f entrusted to it by the client. The server is untrusted in terms of both security and reliability: it might maliciously or accidentally erase the data or place it onto temporarily unavailable storage media. This could occur for numerous reasons including cost-savings or external pressures (e.g., government censure).

* Portions of this work done while at Johns Hopkins.

** Portions of this work done while at IBM. Research supported by NSF grant #0426683.

The server might also accidentally erase some data and choose not to notify the client. Exacerbating the problem (and precluding naïve approaches) are factors such as limited bandwidth between the client and server, as well as the client’s limited resources. See [1,11] for a more thorough discussion.

If we allow communication complexity linear in f , there is a simple mechanism allowing the client to verify that the server stores f at any given time: When the client uploads f , the client locally stores a hash of f ; to verify, the server simply sends all of f and the client checks that this hashes to the correct value. For our purposes, we are interested in solutions with communication complexity that is much smaller than (and, ideally, independent of) the file size.

Ateniese et al. [1] and Juels and Kaliski [11] independently introduced approaches to this problem having sub-linear communication complexity. (Earlier work by Naor and Rothblum [13] is related, but considers a somewhat weaker adversarial model.) Ateniese et al. also distinguish between the case of *private verifiability*, where only the original client (or anyone with whom that client shares a key) can verify the server’s storage, and *public verifiability*, where anyone knowing the client’s public key can perform verification. Extensions and improvements were given by Shacham and Waters [14], Dodis, Vadhan, and Wichs [5], and Bowers, Juels, and Oprea [4]. We refer to [5] for a more detailed comparison among the existing schemes.

Here, we are interested in *publicly-verifiable* schemes that can be used for an *unbounded* number of verifications. A useful tool for this, implicit in [1] and further studied in [14,5], is a *homomorphic linear authenticator* (HLA), which can be defined in either the private- or public-key setting. Roughly speaking, this primitive allows a client to ‘tag’ each block f_i of a file $f = f_1 | \dots | f_n$ in such a way that for any vector c the server can homomorphically construct a (short) tag authenticating the value $\sum c_i \cdot f_i$.

Two recent works have considered the dynamic setting, where the remotely-stored data can be updated [2,6]. We do not address this problem here.

1.1 Our Contributions

The main contribution of this paper is to show a general mechanism (in the random oracle model) for constructing publicly-key HLAs from any identification protocol that is suitably homomorphic. The RSA-based HLA used by Ateniese et al. [1] (see also [14, Appendix E]) can be viewed as an instance of our mechanism applied to the Guillou-Quisquater [10] identification protocol; similarly, the Shacham-Waters scheme [14] can be seen as being derived from an underlying identification protocol in bilinear groups. By applying our transformation to a variant of Shoup’s identification scheme based on factoring [15], we obtain the first publicly-verifiable HLA based on factoring (in the random oracle model).

We also show a generic transformation from any HLA to a publicly-verifiable proof of storage with communication complexity independent of the file size. This transformation is in the standard model, and answers an open question from [14]. An analogous transformation with similar properties was shown (independently)

by Dodis et al. [5] in the setting of simpler private verifiability; our technique is different from theirs and is of independent interest.

Combining our results, we obtain a publicly-verifiable proof of storage based on the factoring assumption in the random oracle model. In our PoS, the communication complexity and the size of the client’s state are independent [4] of the file size, and the server’s storage is a constant multiple of the file size. In the PoS we describe, the computation of both the client and the server is linear in the file size, but notice that public-key HLAs can be layered on top of erasure codes (as in [14,4]) or used in conjunction with a probabilistic approach for multiple audits (as in [1]) to obtain better performance while retaining public verifiability.

2 Definitions

We write $x \leftarrow X$ to represent an element x being sampled uniformly at random from a set X . The output y of a randomized algorithm \mathcal{A} running on input x is denoted by $x \leftarrow \mathcal{A}(x)$. We sometimes write $y := \mathcal{A}(x; r)$ to denote the (deterministic) result of running \mathcal{A} on input x and random coins r . We use boldface to denote vectors. Given a vector \mathbf{v} we let v_i denote its i th component.

Throughout, $k \in \mathbb{N}$ denotes the security parameter. A function $\nu : \mathbb{N} \rightarrow \mathbb{R}$ is *negligible* if for every polynomial $p(\cdot)$ and large enough k , we have $\nu(k) < 1/p(k)$.

2.1 Homomorphic Linear Authenticators

Homomorphic linear authenticators (HLAs) were introduced by Ateniese et al. [1] as a building block for constructing communication-efficient proofs of storage; they were further studied in [14,5]. At a high level, HLAs are used as follows: viewing the file \mathbf{f} as an n -dimensional vector, the client begins by tagging each element of \mathbf{f} and then sending both \mathbf{f} and the vector of tags \mathbf{t} to the server. To verify that the server is storing the entire file, the client sends a random challenge vector \mathbf{c} and the server returns $\mu = \sum_i c_i \cdot f_i$ along with a tag τ , computed using \mathbf{f} , \mathbf{t} , and \mathbf{c} , which is supposed to authenticate this value.

HLAs can be defined both in the private and public-key settings. We give a definition for public-key HLAs and refer the reader to [5] for a formalization of private-key HLAs.

Definition 1 (Homomorphic linear authenticator). A public-key homomorphic linear authenticator is a tuple of four PPT algorithms $(\text{Gen}, \text{Tag}, \text{Auth}, \text{Vrfy})$ such that:

- $(pk, sk) \leftarrow \text{Gen}(1^k)$ is a probabilistic algorithm used to set up the scheme. It takes as input the security parameter and outputs a public and private key pair (pk, sk) . We assume pk defines a k -bit prime p and a positive integer B .
- $(\mathbf{t}, st) \leftarrow \text{Tag}_{sk}(\mathbf{f})$ is a probabilistic algorithm that is run by the client in order to tag a file. It takes as input a secret key sk and a file $\mathbf{f} \in [B]^n$, and outputs a vector of tags \mathbf{t} and state information st .

¹ The communication complexity for a file of size n is $\mathcal{O}(\log n + k)$, and as in [5] we assume $k \gg \log n$.

$\tau := \text{Auth}_{pk}(\mathbf{f}, \mathbf{t}, \mathbf{c})$ is a deterministic algorithm that is run by the server to generate a tag. It takes as input a public key pk , a file $\mathbf{f} \in [B]^n$, a tag vector \mathbf{t} , and a challenge vector $\mathbf{c} \in \mathbb{Z}_p^n$; it outputs a tag τ .

$b := \text{Vrfy}_{pk}(st, \mu, \mathbf{c}, \tau)$: is a deterministic algorithm that is used to verify a tag. It takes as input a public key pk , state information st , an element $\mu \in \mathbb{N}$, a challenge vector $\mathbf{c} \in \mathbb{Z}_p^n$, and a tag τ . It outputs a bit, where ‘1’ indicates acceptance and ‘0’ indicates rejection.

For correctness, we require that for all $k \in \mathbb{N}$, all (pk, sk) output by $\text{Gen}(1^k)$, all $\mathbf{f} \in [B]^n$, all (\mathbf{t}, st) output by $\text{Tag}_{sk}(\mathbf{f})$, and all $\mathbf{c} \in \mathbb{Z}_p^n$, it holds that

$$\text{Vrfy}_{pk} \left(st, \sum_i c_i f_i, \mathbf{c}, \text{Auth}_{pk}(\mathbf{f}, \mathbf{t}, \mathbf{c}) \right) = 1.$$

We remark that in certain schemes correctness (and security) may hold even when Vrfy is given only $\sum_i c_i f_i \bmod p$ (assuming $B < p$). In such cases the communication from the server to the client can be further reduced.

Informally an HLA is secure if, for a given file \mathbf{f} and challenge vector \mathbf{c} , no adversary can output a valid authenticator for an element $\mu' \neq \sum_i c_i f_i$.

Definition 2 (Unforgeability for public-key HLAs). Let $\Lambda = (\text{Gen}, \text{Tag}, \text{Auth}, \text{Vrfy})$ be a public-key HLA and \mathcal{A} be an adversary, and consider the following experiment:

1. The challenger computes $(pk, sk) \leftarrow \text{Gen}(1^k)$, where pk defines p and B .
2. Given pk and oracle access to $\text{Tag}_{sk}(\cdot)$, adversary \mathcal{A} outputs a file $\mathbf{f} \in [B]^n$.
3. The challenger tags the file by computing $(\mathbf{t}, st) \leftarrow \text{Tag}_{sk}(\mathbf{f})$.
4. Given \mathbf{t} and st , the adversary \mathcal{A} outputs a challenge vector $\mathbf{c} \in \mathbb{Z}_p^n$, an element $\mu' \in \mathbb{Z}$, and a tag τ' .
5. The adversary succeeds if $\mu' \neq \sum_i c_i f_i$ and $\text{Vrfy}_{pk}(st, \mu', \mathbf{c}, \tau') = 1$.

Λ is unforgeable if the success probability of every PPT adversary \mathcal{A} in the above experiment is negligible.

The distinctions between the case of public verifiability (as defined above) and private verifiability (as defined in [5]) are that, in the former setting (1) verification does not require the original secret key sk but only the state st and the original public key; (2) unforgeability holds even against an adversary who knows the public information pk and st . Our definition is also stronger than the one given in [5] in that we initially give the adversary access to a tagging oracle.

2.2 Homomorphic Identification Protocols

An identification protocol allows a prover \mathcal{P} in possession of a secret key sk to prove its identity to a verifier \mathcal{V} that possesses the corresponding public key pk . We consider 3-move identification protocols where the prover generates the first message α using the public key pk and randomness r ; the verifier sends a random challenge β ; and the prover then computes a response γ using (pk, sk) , the randomness r , and the verifier’s challenge β . Given the transcript of the protocol, the verifier decides whether to accept or not.

Definition 3 (Identification protocol). An identification protocol is a three-move protocol between a PPT prover \mathcal{P} and a PPT verifier \mathcal{V} . The protocol consists of four polynomial-time algorithms (Setup, Comm, Resp, Vrfy) such that:

$(pk, sk) \leftarrow \text{Setup}(1^k)$ is a probabilistic algorithm that takes as input the security parameter and outputs a public and private key pair (pk, sk) .

$\alpha \leftarrow \text{Comm}(pk; r)$ is a probabilistic algorithm run by the prover \mathcal{P} to generate the first message. It takes as input the public key and random coins r , and outputs an initial message α . We stress that there is no need for sk .

$\gamma \leftarrow \text{Resp}(pk, sk, r, \beta)$ is a probabilistic algorithm that is run by the prover \mathcal{P} to generate the third message. It takes as input the public key pk , the secret key sk , a random string r , and a challenge β (from some associated challenge space), and outputs a response γ .

$b := \text{Vrfy}(pk, \alpha, \beta, \gamma)$ is a deterministic algorithm run by the verifier \mathcal{V} to decide whether to accept the interaction. It takes as input the public key pk , an initial message α , a challenge β , and a response γ . It outputs a bit b , where ‘1’ indicates acceptance and ‘0’ indicates rejection.

For correctness, we require that for all $k \in \mathbb{N}$, all (pk, sk) output by $\text{Setup}(1^k)$, all random coins r , and all β in the appropriate challenge space, it holds that

$$\text{Vrfy} \left(pk, \text{Comm}(pk; r), \beta, \text{Resp}(pk, sk, r, \beta) \right) = 1.$$

An identification protocol is *homomorphic* if the verification of several transcripts of the protocol can be “batched”:

Definition 4 (Homomorphic identification protocol). An identification protocol $\Sigma = (\text{Setup}, \text{Comm}, \text{Resp}, \text{Vrfy})$ is homomorphic if there exist efficient functions $\text{Combine}_1, \text{Combine}_3$ such that:

Completeness: For all (pk, sk) output by $\text{Setup}(1^k)$ and all $\mathbf{c} \in \mathbb{Z}_{2^k}^n$, if transcripts $\{(\alpha_i, \beta_i, \gamma_i)\}_{1 \leq i \leq n}$ are such that $\text{Vrfy}(pk, \alpha_i, \beta_i, \gamma_i) = 1$ for all i , then:

$$\text{Vrfy} \left(pk, \text{Combine}_1(\mathbf{c}, \alpha), \sum_i c_i \beta_i, \text{Combine}_3(\mathbf{c}, \gamma) \right) = 1.$$

Unforgeability: Consider the following experiment involving an adversary \mathcal{A} :

1. The challenger computes $(pk, sk) \leftarrow \text{Setup}(1^k)$ and gives pk to \mathcal{A} .
2. The following is repeated a polynomial number of times:
 - \mathcal{A} outputs β' in the challenge space. The challenger chooses random r , computes $\gamma := \text{Resp}(pk, sk, r, \beta')$, and gives (r, γ) to \mathcal{A} .
3. The adversary outputs a n -vector of challenges β . Then for each i the challenger chooses r_i at random, sets $\alpha_i := \text{Comm}(pk; r_i)$ and $\gamma_i := \text{Resp}(pk, sk, r_i, \beta_i)$, and gives (\mathbf{r}, γ) to \mathcal{A} .
4. \mathcal{A} outputs a triple $(\mathbf{c}, \mu', \gamma')$, where $\mathbf{c} \in \mathbb{Z}_{2^k}^n$. The adversary succeeds if (1) $\mu' \neq \sum_i c_i \beta_i$ and (2) $\text{Vrfy}(pk, \text{Combine}_1(\mathbf{c}, \alpha), \mu', \gamma') = 1$.

2.3 Proofs of Storage

Definition 5 (Proof of storage). A (publicly-verifiable) proof of storage is a tuple of five PPT algorithms $(\text{Gen}, \text{Encode}, \text{Prove}, \text{Vrfy})$ such that:

$(pk, sk) \leftarrow \text{Gen}(1^k)$ is a probabilistic algorithm that is run by the client to set up the scheme. It takes as input a security parameter, and outputs a public and private key pair (pk, sk) . We assume pk defines a k -bit prime p and a positive integer B .

$(\mathbf{f}', st) \leftarrow \text{Encode}_{sk}(\mathbf{f})$ is a probabilistic algorithm that is run by the client in order to encode the file. It takes as input the secret key sk , and a file $\mathbf{f} \in [B]^n$. It outputs an encoded file \mathbf{f}' and state information st .

$\pi := \text{Prove}(pk, \mathbf{f}', \mathbf{c})$ is a deterministic algorithm that takes as input the public key pk , an encoded file \mathbf{f}' , and a challenge $\mathbf{c} \in \mathbb{Z}_p^n$. It outputs a proof π .

$b := \text{Vrfy}(pk, st, \mathbf{c}, \pi)$: is a deterministic algorithm that takes as input the public key pk , the state st , a challenge $\mathbf{c} \in \mathbb{Z}_p^n$, and a proof π . It outputs a bit, where ‘1’ indicates acceptance and ‘0’ indicates rejection.

We require that for all $k \in \mathbb{N}$, all (pk, sk) output by $\text{Gen}(1^k)$, all $\mathbf{f} \in [B]^n$, all (\mathbf{f}', st) output by $\text{Encode}_{sk}(\mathbf{f})$, and all $\mathbf{c} \in \mathbb{Z}_p^n$, it holds that

$$\text{Vrfy}(pk, st, \mathbf{c}, \text{Prove}(pk, \mathbf{f}', \mathbf{c})) = 1.$$

Note that the above defines a *publicly-verifiable* PoS since the original secret key sk is not needed in order to perform verification.

Security of a PoS, roughly speaking, guarantees that if the verifier accepts then the prover indeed has (sufficient information to recover) the entire original file \mathbf{f} . As noted in [11][14][5], soundness can be formalized using the notion of a knowledge extractor [7][3]. As in [5], we phrase our definition using the paradigm of “witness-extended emulation” [12].

Definition 6 (Security for a publicly-verifiable PoS). Let $\Pi = (\text{Gen}, \text{Encode}, \text{Prove}, \text{Vrfy})$ be a publicly-verifiable PoS. Π is secure if there is an expected polynomial-time knowledge extractor \mathcal{K} such that, for any PPT adversary \mathcal{A} we have:

1. The distributions

$$\left\{ \begin{array}{l} (pk, sk) \leftarrow \text{Gen}(1^k); (\mathbf{f}, st_{\mathcal{A}}) \leftarrow \mathcal{A}^{\text{Encode}_{sk}(\cdot)}(pk); \\ (\mathbf{f}', st) \leftarrow \text{Encode}_{sk}(\mathbf{f}); \mathbf{c} \leftarrow \mathbb{Z}_p^n \end{array} : (\mathbf{c}, \mathcal{A}(st_{\mathcal{A}}, \mathbf{f}', st, \mathbf{c})) \right\}$$

and

$$\left\{ \begin{array}{l} (pk, sk) \leftarrow \text{Gen}(1^k); (\mathbf{f}, st_{\mathcal{A}}) \leftarrow \mathcal{A}^{\text{Encode}_{sk}(\cdot)}(pk); \\ (\mathbf{f}', st) \leftarrow \text{Encode}_{sk}(\mathbf{f}) \end{array} : \mathcal{K}_1^{\mathcal{A}(st_{\mathcal{A}}, \mathbf{f}', st, \cdot)}(pk, st) \right\}$$

are identical. (Above, \mathcal{K}_1 denotes the first output of \mathcal{K} .)

2. The following is negligible:

$$\Pr \left[\begin{array}{l} (pk, sk) \leftarrow \text{Gen}(1^k); \\ (\mathbf{f}, st_{\mathcal{A}}) \leftarrow \mathcal{A}^{\text{Encode}_{sk}(\cdot)}(pk); \\ (\mathbf{f}', st) \leftarrow \text{Encode}_{sk}(\mathbf{f}); \\ ((\mathbf{c}, \pi), \mathbf{f}^*) \leftarrow \mathcal{K}^{\mathcal{A}(st_{\mathcal{A}}, \mathbf{f}', st, \cdot)}(pk, st) \end{array} : \text{Vrfy}(pk, st, \mathbf{c}, \pi) = 1 \wedge \mathbf{f}^* \neq \mathbf{f} \right].$$

3 From Homomorphic Identification Protocols to HLAs

We now show how to transform any homomorphic identification protocol $\Sigma = (\text{Setup}, \text{Comm}, \text{Resp}, \text{Vrfy})$ into a public-key HLA. The basic idea is to use the file blocks f_1, \dots, f_n as the “challenges” in n parallel invocations of the identification protocol. Thus, a very basic PoS would be as follows:

- The client computes $(pk, sk) \leftarrow \text{Gen}(1^k)$.
- For each block f_i of the file, the client computes α_i, γ_i such that $(\alpha_i, f_i, \gamma_i)$ is an accepting transcript in the underlying identification scheme.
- The client sends to the server the file $\mathbf{f} = f_1 | \dots | f_n$ and the tags $\gamma_1, \dots, \gamma_n$; the client stores $\alpha_1, \dots, \alpha_n$ as its own local state.

To verify that the server stores the i th block of the file, the client requests the server to send (f_i, γ_i) ; the client can authenticate this response by checking that $(\alpha_i, f_i, \gamma_i)$ is an accepting transcript.

There are several drawbacks to the above approach. First, the client’s state is linear in the file size.² This is easy to remedy by having the client generate each α_i using a pseudorandom function (if private verifiability suffices) or a random oracle (if public verifiability is desired, as here). A more serious problem is that a server can easily “cheat” without being caught “too often” by throwing away blocks of the file. If the server deletes, say, 1 block from the file then it is only caught with probability $1/n$. This can be addressed, to some extent, by having the client request many blocks but then the communication complexity increases.

Instead, we rely on the *homomorphic* property of the identification scheme to “batch” the authentication of multiple blocks. Specifically, the client will send a random integer vector \mathbf{c} and the server will respond with $\mu' := \sum_i c_i f_i$ and $\gamma' := \text{Combine}_3(\mathbf{c}, \boldsymbol{\gamma})$; This response can be verified by checking whether

$$\text{Vrfy}(pk, \text{Combine}_1(\mathbf{c}, \boldsymbol{\alpha}), \mu', \gamma') \stackrel{?}{=} 1.$$

(See Figure 1) Although the client-to-server communication is large, the server-to-client communication is essentially independent of the file size (cf. footnote 1). We reduce the client-to-server communication when we construct a PoS in the next section.

Theorem 1. *If Σ is an unforgeable homomorphic identification protocol, then Λ as in Figure 1 is an unforgeable public-key HLA if H is modeled as a random oracle.*

Proof. Correctness is easy to verify, and so we consider security. Let \mathcal{A} be a PPT adversary attacking Λ . We construct an adversary \mathcal{A}' attacking Σ as follows:

1. \mathcal{A}' is given a public key pk , generates B and p in the obvious way, and runs $\mathcal{A}(pk, p, B)$.

² In some cases linear state may be acceptable, as long as the state is a constant fraction *shorter* than the file itself. When using certain homomorphic identification schemes, including the one discussed in Section 5, this indeed can be achieved.

Let $\Sigma = (\text{Setup}, \text{Comm}, \text{Resp}, \text{Vrfy})$ be a homomorphic identification protocol and let H be a function. Construct a public-key HLA $\Lambda = (\text{Gen}, \text{Tag}, \text{Auth}, \text{Vrfy})$ as follows:

- $\text{Gen}(1^k)$: Compute $(pk, sk) \leftarrow \Sigma.\text{Setup}(1^k)$. Let B be such that $[B]$ is in the challenge space of Σ , and choose a k -bit prime p . Output the public key (pk, p, B) and secret key sk .
- $\text{Tag}_{sk}(\mathbf{f})$, where $\mathbf{f} = f_1 | \dots | f_n$, and $f_i \in [B]$ for all i :
 1. Choose $st \leftarrow \{0, 1\}^k$.
 2. For $1 \leq i \leq n$:
 - a. Set $r_i := H(st; i)$ and $\alpha_i := \Sigma.\text{Comm}(pk; r_i)$.
 - b. Compute $\gamma_i := \Sigma.\text{Resp}(pk, sk, r_i, f_i)$.
 3. Output $\mathbf{t} := (\gamma_1, \dots, \gamma_n)$ and st .
- $\text{Auth}_{pk}(\mathbf{f}, \mathbf{t}, \mathbf{c})$: Compute and output $\tau \leftarrow \Sigma.\text{Combine}_3(\mathbf{c}, \mathbf{t})$.
- $\text{Vrfy}_{pk}(st, \mu, \mathbf{c}, \tau)$:
 1. for $1 \leq i \leq n$, set $r_i := H(st; i)$ and $\alpha_i := \Sigma.\text{Comm}(pk; r_i)$.
 2. Output $\Sigma.\text{Vrfy}(pk, \text{Combine}_1(\mathbf{c}, \boldsymbol{\alpha}), \mu, \tau)$.

Fig. 1. Transforming a homomorphic identification protocol into a HLA

2. When \mathcal{A} requests $\text{Tag}_{sk}(\mathbf{f})$ for $\mathbf{f} = f_1 | \dots | f_n$, then (for $i = 1$ to n) \mathcal{A}' queries f_i to its own oracle and receives in return (r_i, γ_i) . Then \mathcal{A}' chooses random $st \in \{0, 1\}^k$, sets answers to the random oracle appropriately, and gives $(\gamma_1, \dots, \gamma_n)$ and st to \mathcal{A} .
3. Eventually, \mathcal{A} outputs a file \mathbf{f} . Following this, \mathcal{A}' outputs the vector of n challenges $\mathbf{f} = f_1 | \dots | f_n$, and receives in return $(\mathbf{r}, \boldsymbol{\gamma})$. Then \mathcal{A}' chooses random $st \in \{0, 1\}^k$, sets ³ answers to the random oracle appropriately, and gives $(\boldsymbol{\gamma}, st)$ to \mathcal{A} .
4. When \mathcal{A} finally outputs \mathbf{c}, μ', τ' , then \mathcal{A}' outputs these same values.

It is easy to see that \mathcal{A} succeeds in attacking Λ exactly when \mathcal{A}' succeeds in attacking Σ .

4 From HLAs to Efficient Proofs of Storage

In this section we show how to use any HLA to construct a PoS having communication complexity independent of the file size. Our transformation is in the standard model.

It is immediate how an HLA can be used to construct a PoS with communication complexity linear in the file size: When storing a file \mathbf{f} , the client computes tags on all the file blocks and gives to the server the vector of tags \mathbf{t} (along with \mathbf{f} itself). To verify, the client chooses a random $\mathbf{c} \in \mathbb{Z}_p^n$ and sends it to the server; the server responds with $\sum_i c_i f_i$ and $\text{Auth}_{pk}(\mathbf{f}, \mathbf{t}, \mathbf{c})$ (which is authenticated by

³ We assume for simplicity that no $st \in \{0, 1\}^k$ is chosen twice throughout the experiment, since this occurs with only negligible probability.

the client in the obvious way). If authentication tags output by Auth have length $\mathcal{O}(k)$, then the server-to-client communication for an n -block file is bounded by

$$\mathcal{O}(k) + \log \left(\sum_i c_i f_i \right) \leq \mathcal{O}(k) + \log n \cdot p \cdot B = \mathcal{O}(k) + \log n.$$

For typical values of k, n , this means that the server-to-client communication is (essentially) independent of the file size.

To reduce the client-to-server communication, we use a pseudorandom function F : the client sends a key $K \in \{0, 1\}^k$, and the server then derives the challenge vector \mathbf{c} by setting $c_i := F_K(i)$ for all i . (See Figure 2.) This approach is, perhaps, quite “natural”⁴ but it turns out to be highly non-trivial to prove that it is sound. (This difficulty was mentioned in [14, 5].) The issue is that since the key K is public, we cannot reduce to the security of the pseudorandom function in the usual way. Instead we must use a more careful analysis.

Let $\Lambda = (\text{Gen}, \text{Tag}, \text{Auth}, \text{Vrfy})$ be a public-key HLA, and let F be a pseudorandom function. Construct a publicly-verifiable PoS $\Pi = (\text{Gen}, \text{Encode}, \text{Prove}, \text{Vrfy})$ as follows:

- $\text{Gen}(1^k)$: Compute and output $(pk, sk) \leftarrow \Lambda.\text{Gen}(1^k)$. Let p be the prime implicit in pk .
- $\text{Encode}_{sk}(\mathbf{f})$: Compute $(\mathbf{t}, st) \leftarrow \Lambda.\text{Tag}_{sk}(\mathbf{f})$, and output $\mathbf{f}' = (\mathbf{f}, \mathbf{t})$ and st .
- $\text{Prove}(pk, \mathbf{f}', K)$, where $K \in \{0, 1\}^k$:
 1. Parse \mathbf{f}' as (\mathbf{f}, \mathbf{t}) .
 2. For $1 \leq i \leq n$ let $c_i := F_K(i)$, where c_i is viewed as an element of \mathbb{Z}_p .
 3. Compute $\tau \leftarrow \Lambda.\text{Auth}_{pk}(\mathbf{f}, \mathbf{t}, \mathbf{c})$ and $\mu := \sum_i c_i f_i$.
 4. Output $\pi := (\mu, \tau)$.
- $\text{Vrfy}(pk, st, K, \pi)$:
 1. Parse π as (μ, τ) .
 2. For $1 \leq i \leq n$, let $c_i := F_K(i)$.
 3. Output $b := \Lambda.\text{Vrfy}_{pk}(st, \mu, \mathbf{c}, \tau)$.

Fig. 2. Transforming an HLA into a PoS

Theorem 2. *Let Λ be an unforgeable public-key HLA, and let F be a pseudorandom function secure against non-uniform polynomial-time adversaries. Then Π as in Figure 2 is a secure publicly-verifiable PoS.*

Proof. Correctness of the construction is easily verified, and so we turn to proving security. We describe a knowledge extractor \mathcal{K} that runs in expected polynomial-time and satisfies Definition 6. Recall that \mathcal{K} is given pk, st as input and has

⁴ A similar approach, based on pseudorandom generators, was proposed in [9] in the context of verifiable shuffles.

oracle access to $\mathcal{A}(st_{\mathcal{A}}, \mathbf{f}', st, \cdot)$, which we abbreviate as $\mathcal{A}(\cdot)$. Define $\mathbf{c}(K) = (F_K(1), \dots, F_K(n))$. The high-level structure of \mathcal{K} is as follows:

1. \mathcal{K} chooses random $K \leftarrow \{0, 1\}^k$ and runs $\mathcal{A}(K)$ to obtain a proof π . If $\text{Vrfy}(pk, st, K, \pi) = 0$ then \mathcal{K} outputs $((K, \pi), \perp)$ and stops. Otherwise, its first output will still be (K, π) but it attempts to recover the original file as described next.
2. \mathcal{K} repeatedly rewinds \mathcal{A} and sends it different challenges until \mathcal{A} responds correctly to a total of n challenges K_1, \dots, K_n such that $\mathbf{c}(K_1), \dots, \mathbf{c}(K_n)$ are linearly independent (over \mathbb{Q}). Given n successful responses to these n challenges, \mathcal{K} reconstructs a candidate file \mathbf{f} , and outputs it.

The above neglects some technical details that we now formalize. If $\mathcal{A}(K)$ outputs a proof $\pi = (\mu, \tau)$ for which $\text{Vrfy}_{pk}(st, \mu, \mathbf{c}(K), \tau) = 1$, then we say that K is a *good* challenge. \mathcal{K} implements step 2, above, as follows:

1. Initialize sets $\text{Good}_K := \text{Good}_{\mathbf{c}} := \emptyset$. Keep track of the total number of calls to \mathcal{A} , and halt execution with output *fail* if 2^k calls are made.
2. Estimate the probability \tilde{p}^* with which a random key K is good by running \mathcal{A} with a random challenge until some fixed polynomial number $q = q(k)$ successful verifications occur. By appropriate choice of q , it is possible to ensure that the estimate \tilde{p}^* is within a factor of 2 of the true probability with all but negligible probability 2^{-k^2} .
3. For $j = 1$ to n do:
 - Repeatedly sample K_j uniformly, querying \mathcal{A} on each one, until a good K_j with $\mathbf{c}(K_j) \notin \text{span}(\text{Good}_{\mathbf{c}})$ is found. If found, then add K_j to Good_K and add $\mathbf{c}_j = \mathbf{c}(K_j)$ to $\text{Good}_{\mathbf{c}}$, and go to the next value of j . If no such K_j is found in at most k^2/\tilde{p}^* tries, then output *fail* and halt.
4. Let $\text{Good}_K = \{K_1, \dots, K_n\}$ and $\text{Good}_{\mathbf{c}} = \{\mathbf{c}_1, \dots, \mathbf{c}_n\}$, where $\mathbf{c}_j = \mathbf{c}(K_j)$, and let $\pi_j = (\mu_j, \tau_j)$ be the output of $\mathcal{A}(K_j)$. Set up the system of linear equations $\{\sum_i c_{j,i} \cdot f_i = \mu_j\}_{1 \leq j \leq n}$ in the unknowns $\mathbf{f} = (f_1, \dots, f_n)$. Solve for \mathbf{f} (over the integers) and output it.

We refer to the above as the *extraction subroutine*.

To complete the proof, we need to show three things. First, that \mathcal{K} runs in expected polynomial time for any \mathcal{A} . Second, that if \mathcal{A} successfully convinces a verifier in the PoS protocol with sufficiently high probability, then the extraction procedure will successfully complete (specifically, step 3 will be successful) with overwhelming probability. Third, that with overwhelming probability the file \mathbf{f} output by the extraction procedure is indeed equal to the true file \mathbf{f} . The first and third of these items are essentially standard. The second step would be relatively straightforward if the challenge in the PoS protocol were a random vector \mathbf{c} ; what makes it more complicated is that the challenge is a PRF key K that is expanded to a vector $\mathbf{c} = \mathbf{c}(K)$.

Fixing $st_{\mathcal{A}}$, \mathbf{f}' , and st , we let p^* denote the probability that a random challenge K is good; i.e., this is the probability with which $\mathcal{A}(st_{\mathcal{A}}, \mathbf{f}', st, \cdot)$ responds correctly to the verifier's challenge (we assume $st_{\mathcal{A}}$ includes \mathcal{A} 's coins).

Claim. \mathcal{K} runs in expected polynomial time.

Proof. If $p^* = 0$ then it is clear that \mathcal{K} runs in expected polynomial time. So assume $p^* > 0$. We must then analyze the expected running time of the extraction procedure, following [8,12]. Steps 1 and 4 take strict polynomial time. The expected running time of step 2 is exactly (some polynomial times) $q(k)/p^*$. As for step 3, there are two cases: If $\tilde{p}^* \leq p^*/2$, then the only thing we can claim is that the running time is bounded by (some polynomial times) 2^k , due to the counter being maintained in step 1. But the probability that $\tilde{p}^* \leq p^*/2$ is at most 2^{-k^2} . On the other hand, if $\tilde{p}^* > p^*/2$ then the expected running time of step 4 is at most (some polynomial times) $n \cdot k^2/\tilde{p}^* < 2nk^2/p^*$.

\mathcal{K} only runs the extraction procedure with probability p^* . Thus, the overall expected running time of \mathcal{K} is upper-bounded by

$$p^* \cdot \left(\text{poly}(k) + \text{poly}(k) \cdot q(k)/p^* + \text{poly}(k) \cdot 2^k \cdot 2^{-k^2} + \text{poly}(k) \cdot 2nk^2/p^* \right),$$

which is polynomial.

Claim. There exists a negligible function $\epsilon(\cdot)$ such that if $p^* > \epsilon(k)$ then the probability (conditioned on the extraction procedure being run) that the extraction procedure outputs fail is negligible.

Observe this implies that

$$\Pr \left[\begin{array}{l} (pk, sk) \leftarrow \text{Gen}(1^k); \\ (\mathbf{f}, st_{\mathcal{A}}) \leftarrow \mathcal{A}^{\text{Encode}_{sk}(\cdot)}(pk); \\ (\mathbf{f}', st) \leftarrow \text{Encode}_{sk}(\mathbf{f}); \\ ((\mathbf{c}, \pi), \mathbf{f}^*) \leftarrow \mathcal{K}^{\mathcal{A}(st_{\mathcal{A}}, \mathbf{f}', st, \cdot)}(pk, st) \end{array} : \text{Vrfy}(pk, st, \mathbf{c}, \pi) = 1 \wedge \mathbf{f}^* = \text{fail} \right]$$

is negligible.

Proof. We view the $\mathbf{c}_j = \mathbf{c}(K_j)$ as vectors over \mathbb{Z}_p , and use the fact that integer vectors $\mathbf{c}_1, \dots, \mathbf{c}_\ell$, with entries in the range $\{0, \dots, p-1\}$, are linearly dependent over \mathbb{Q} only if they are linearly dependent over \mathbb{Z}_p ; thus, an upper bound on the probability of the latter implies an upper bound on the probability of the former.

Define

$$\epsilon'(k) = \max_L \{ \Pr[K \leftarrow \{0, 1\}^k : \mathbf{c}(K) \in L] \},$$

where the maximum is taken over all $(n-1)$ -dimensional subspaces $L \subset \mathbb{Z}_p^n$. It is not hard to see that if F is a non-uniformly secure PRF then $\epsilon'(k) - 1/p$ is negligible. Since $1/p$ is negligible, we see that ϵ' is negligible too. Take $\epsilon = 2\epsilon'$. We show that if $p^* > \epsilon$ then, conditioned on the extraction procedure being run, the probability that it outputs fail is negligible.

First, observe that the probability that \mathcal{K} times out by virtue of running for 2^k steps is negligible (this follows from the fact that the expected running time of \mathcal{K} is polynomial). Next, fix any j and consider step 3. The number of challenges that are good is exactly $p^* \cdot 2^k$, and the number of challenges K_j for which $c(K_j)$ lies in $\text{span}(\text{Good}_c)$ (which has dimension at most $n - 1$) is at most $e' \cdot 2^k < p^* \cdot 2^k / 2$. Thus, the probability that a random K_j is both good and does not lie in $\text{span}(\text{Good}_c)$ is at least $p^*/2$. If \tilde{p}^* is within a factor of 2 of p^* , which occurs with all but negligible probability, then \mathcal{K} finds such a K_j within k^2/\tilde{p}^* steps with all but negligible probability; a union bound over all values of $j \in [n]$ then shows that it fails in some iteration with only negligible probability. This completes the proof.

Finally, we show that the probability that the extraction procedure outputs an incorrect file is negligible. In conjunction with the previous claims, this completes the proof that \mathcal{K} satisfies Definition 6.

Claim. For any PPT adversary \mathcal{A} , the following is negligible:

$$\Pr \left[\begin{array}{l} (pk, sk) \leftarrow \text{Gen}(1^k); \\ (\mathbf{f}, st_{\mathcal{A}}) \leftarrow \mathcal{A}^{\text{Encode}_{sk}(\cdot)}(pk); \\ (\mathbf{f}', st) \leftarrow \text{Encode}_{sk}(\mathbf{f}); \\ ((c, \pi), \mathbf{f}^*) \leftarrow \mathcal{K}^{\mathcal{A}(st_{\mathcal{A}}, \mathbf{f}', st, \cdot)}(pk, st) \end{array} \quad ; \quad \begin{array}{l} \text{Vrfy}(pk, st, c, \pi) = 1 \\ \wedge \mathbf{f}^* \notin \{\text{fail}, \mathbf{f}\} \end{array} \right].$$

Proof. The event in question can only occur if, at the end of the extraction procedure, there exists $c \in \text{Good}_c$, with $c = c(K)$, for which $\mathcal{A}(K)$ outputs (μ, τ) such that $\text{Vrfy}(pk, st, K, (\mu, \tau)) = 1$ yet $\mu \neq \sum_i c_i f_i$. But this exactly means that \mathcal{A} has violated the assumed unforgeability of Λ . Since \mathcal{K} runs in expected polynomial-time, it follows by a standard argument that this occurs with only negligible probability.

This concludes the proof of Theorem 2.

5 A Concrete Instantiation Based on Factoring

In this section we describe a homomorphic variant of the identification protocol of Shoup [15], whose security is based on the hardness of factoring. Together with the transformations described in the previous sections, this yields a factoring-based PoS in the random oracle model.

Protocol Σ_{Shoup} , described in Figure 3, relies on a Blum modulus generator Gen_{Blum} that takes as input a security parameter 1^k and outputs a tuple (N, p, q) such that $N = p \cdot q$ where p and q are k -bit primes with $p = q = 3 \pmod 4$. We denote by \mathcal{QR}_N the set of quadratic residues modulo N , and by \mathcal{J}_N^{+1} the elements of \mathbb{Z}_N^* with Jacobi symbol $+1$. We use the following standard facts regarding Blum integers: (1) given $x \in \mathbb{Z}_N^*$ it can be efficiently decided whether $x \in \mathcal{J}_N^{+1}$; (2) if $x \in \mathcal{J}_N^{+1}$, then exactly one of x or $-x$ is in \mathcal{QR}_N ; (3) every $x \in \mathcal{QR}_N$ has four square roots, exactly one of which is itself in \mathcal{QR}_N .

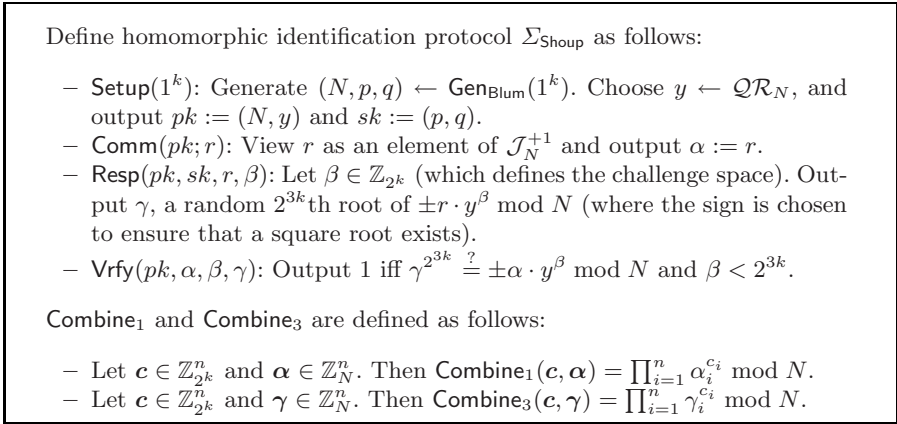


Fig. 3. A homomorphic identification protocol based on factoring

Correctness of Σ_{Shoup} as a stand-alone identification protocol is immediate. Let us verify that it is homomorphic. Fix public key (N, y) , challenge vector $\mathbf{c} \in \mathbb{Z}_{2^k}^n$, and $\{(\alpha_i, \beta_i, \gamma_i)\}_{1 \leq i \leq n}$ such that $\gamma_i^{2^{3k}} = \pm \alpha_i \cdot y^{\beta_i} \bmod N$ for all i . Then

$$\begin{aligned}
 \text{Combine}_3(\mathbf{c}, \boldsymbol{\gamma})^{2^{3k}} &= \left(\prod_{i=1}^n \gamma_i^{c_i} \right)^{2^{3k}} \bmod N \\
 &= \prod_{i=1}^n \left(\gamma_i^{2^{3k}} \right)^{c_i} \bmod N \\
 &= \prod_{i=1}^n \left(\pm \alpha_i \cdot y^{\beta_i} \right)^{c_i} \bmod N \\
 &= \pm \prod_{i=1}^n \alpha_i^{c_i} \cdot y^{\beta_i c_i} \bmod N \\
 &= \pm \text{Combine}_1(\mathbf{c}, \boldsymbol{\alpha}) \cdot y^{\sum_i c_i \beta_i} \bmod N,
 \end{aligned}$$

and furthermore $\sum_i c_i \beta_i < n \cdot 2^k \cdot 2^k < 2^{3k}$.

Theorem 3. Σ_{Shoup} is an unforgeable homomorphic identification protocol if the factoring assumption holds with respect to Gen_{Blum} .

Proof. The high-level ideas are similar to those in [15], though the proof here is a bit simpler. Given a PPT adversary \mathcal{A} attacking Σ_{Shoup} , we construct a PPT algorithm \mathcal{B} computing square roots modulo N output by Gen_{Blum} . This implies factorization of N in the standard way. Algorithm \mathcal{B} works as follows:

- \mathcal{B} is given a Blum modulus N and a random $y \in \mathcal{QR}_N$. It runs \mathcal{A} on the public key $pk = (N, y)$.
- When \mathcal{A} outputs $\beta' \in \mathbb{Z}_{2^k}$, then \mathcal{B} chooses random $\gamma \in \mathbb{Z}_N$ and $b \in \{0, 1\}$, and sets $r := \alpha := (-1)^b \cdot \gamma^{2^{3k}} / y^{\beta'} \bmod N$. It then gives (r, γ) to \mathcal{A} .

- When \mathcal{A} outputs an n -vector of challenges β , then for each i algorithm \mathcal{B} computes (r_i, γ_i) as in the previous step. It gives (\mathbf{r}, γ) to \mathcal{A} .
- If \mathcal{A} outputs $(\mathbf{c}, \mu', \gamma')$ with $\text{Vrfy}(pk, \text{Combine}_1(\mathbf{c}, \alpha), \mu', \gamma') = 1$ but $\mu' \neq \sum_i c_i \beta_i$, then \mathcal{B} computes a square root of y as described below.

Note that the simulation provided for \mathcal{A} by \mathcal{B} is perfect, and so \mathcal{A} succeeds in the above with the same probability with which it succeeds in attacking the real-world protocol Σ_{Shoup} .

To complete the proof, we describe the final step in more detail. Define

$$\alpha^* = \text{Combine}_1(\mathbf{c}, \alpha), \quad \gamma^* = \text{Combine}_3(\mathbf{c}, \gamma), \quad \mu = \sum_i c_i \beta_i.$$

If $\text{Vrfy}(pk, \alpha^*, \mu', \gamma') = 1$ but $\mu' \neq \mu$, then $(\gamma')^{2^{3k}} = \pm \alpha^* \cdot y^{\mu'} \pmod N$; furthermore, \mathcal{B} also knows that $(\gamma^*)^{2^{3k}} = \pm \alpha^* \cdot y^\mu \pmod N$. Assume without loss of generality that $\mu > \mu'$. Since $y \in \mathcal{QR}_N$ this implies

$$(\gamma'/\gamma^*)^{2^{3k}} = y^{\mu-\mu'} \pmod N \tag{1}$$

with $\mu, \mu' < 2^{3k}$ (and so $\mu - \mu' < 2^{3k}$). Write $\mu - \mu' = f \cdot 2^t$ for $t < 3k$ and f odd. Since squaring is a permutation of \mathcal{QR}_N , Equation (1) implies

$$(\gamma'/\gamma^*)^{2^{3k-t}} = y^f \pmod N.$$

Using the extended Euclidean algorithm, \mathcal{B} computes integers A, B such that $Af + B2^{3k-t} = 1$. Then

$$\left(\left((\gamma'/\gamma^*)^A y^B \right)^{2^{3k-t-1}} \right)^2 = \left((\gamma'/\gamma^*)^A y^B \right)^{2^{3k-t}} = y^{Af} y^{B2^{3k-t}} = y,$$

and so \mathcal{B} can compute a square root of y . Since \mathcal{B} computes a square root whenever \mathcal{A} succeeds, the success probability of \mathcal{A} must be negligible.

Acknowledgments. We are grateful to Gene Tsudik for his insightful comments and contributions during the early stages of this work.

References

1. Ateniese, G., Burns, R., Curtmola, R., Herring, J., Kissner, L., Peterson, Z., Song, D.: Provable data possession at untrusted stores. In: ACM Conference on Computer and Communications Security. ACM, New York (2007)
2. Ateniese, G., Di Pietro, R., Mancini, L.V., Tsudik, G.: Scalable and efficient provable data possession. In: Proc. 4th Intl. Conf. on Security and Privacy in Communication Networks (SecureComm 2008), pp. 1–10. ACM, New York (2008)
3. Bellare, M., Goldreich, O.: On defining proofs of knowledge. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 390–420. Springer, Heidelberg (1993)

4. Bowers, K., Juels, A., Oprea, A.: Proofs of retrievability: Theory and implementation. Technical Report 2008/175, Cryptology ePrint Archive (2008)
5. Dodis, Y., Vadhan, S., Wichs, D.: Proofs of retrievability via hardness amplification. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 109–127. Springer, Heidelberg (2009)
6. Erway, C., Papamanthou, C., Kupcu, A., Tamassia, R.: Dynamic provable data possession. In: ACM Conf. on Computer and Communications Security (to appear, 2009). Available as Cryptology ePrint Archive, Report 2008/432
7. Feige, U., Fiat, A., Shamir, A.: Zero knowledge proofs of identity. *J. Cryptology* 1(2), 77–94 (1988)
8. Goldreich, O., Kahan, A.: How to construct constant-round zero-knowledge proof systems for NP. *J. Cryptology* 9(3), 167–190 (1996)
9. Groth, J.: A verifiable secret shuffle of homomorphic encryptions. Technical Report 2005/246, IACR ePrint Cryptography Archive (2005)
10. Guillou, L., Quisquater, J.-J.: A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory. In: Günther, C.G. (ed.) EUROCRYPT 1988. LNCS, vol. 330, pp. 123–128. Springer, Heidelberg (1988)
11. Juels, A., Kaliski, B.: PORs: Proofs of retrievability for large files. In: ACM Conference on Computer and Communications Security. ACM, New York (2007)
12. Lindell, Y.: Parallel coin-tossing and constant-round secure two-party computation. *J. Cryptology* 16(3), 143–184 (2003)
13. Naor, M., Rothblum, G.: The complexity of online memory checking. In: IEEE Symposium on Foundations of Computer Science, pp. 573–584. IEEE Computer Society, Los Alamitos (2005)
14. Shacham, H., Waters, B.: Compact proofs of retrievability. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 90–107. Springer, Heidelberg (2008), Full version <http://eprint.iacr.org>
15. Shoup, V.: On the security of a practical identification scheme. *J. Cryptology* 12(4), 247–260 (1999)

Simple Adaptive Oblivious Transfer without Random Oracle

Kaoru Kurosawa and Ryo Nojima

¹ Ibaraki University, Japan

² NICT, Japan

Abstract. Adaptive oblivious transfer (OT) is a two-party protocol which simulates an ideal world such that the sender sends M_1, \dots, M_n to the trusted third party (TTP), and the receiver receives M_{σ_i} from TTP adaptively for $i = 1, 2, \dots, k$. This paper shows the first pairing-free *fully simulatable* adaptive OT. It is also the first *fully simulatable* scheme which does not rely on dynamic assumptions. Indeed our scheme holds under the DDH assumption.

Keywords: Adaptive OT, Fully Simulatable, DDH, Standard Model.

1 Introduction

In a non-adaptive (k, n) oblivious transfer (OT) scheme which is denoted by OT_k^n [6,11,14], a sender has n secret strings M_1, \dots, M_n , and a receiver has k secret choice indices $\sigma_1, \dots, \sigma_k \in \{1, \dots, n\}$. At the end of the protocol, the receiver learns $M_{\sigma_1}, \dots, M_{\sigma_k}$ (only), and the sender learns nothing on $\sigma_1, \dots, \sigma_k$. Efficient OT schemes are important because OT_1^4 is a key building block for secure multi-party computation [20,7,12].

In an adaptive (k, n) oblivious transfer protocol which is denoted by $OT_{k \times 1}^n$, the receiver chooses σ_i adaptively depending on $M_{\sigma_1}, \dots, M_{\sigma_{i-1}}$ [15]. In other words, $OT_{k \times 1}^n$ is a two-party protocol (S, R) which simulates an ideal world protocol (S', R') such that

1. the sender S' sends M_1, \dots, M_n to the trusted third party (TTP), and
2. the receiver R' receives M_{σ_i} from TTP adaptively for $i = 1, 2, \dots, k$, where the receiver chooses σ_i based on $M_{\sigma_1}, \dots, M_{\sigma_{i-1}}$.

Adaptive OT has wide applications such as oblivious database searches, secure multiparty computation and etc, too.

As a security notion of OT (for both non-adaptive and adaptive), half simulatability was considered until recently [15,16,11,18]. This definition requires

- (Sender’s privacy.) For any receiver R in the real world, there exists a receiver \hat{R} in the ideal world such that the outputs of R and \hat{R} are indistinguishable.
- (Receiver’s privacy.) For any input to the receiver, the view of the sender must be indistinguishable. (Note that the honest sender outputs nothing.)

However, Naor and Pinkas noticed that there can be a practical attack on a half simulatable adaptive OT [15].

To solve this problem, Camenisch, Neven and shelat formalized a notion of *full simulatability* [2]. In this definition, we consider a pair of outputs of the sender and the receiver. Although the honest sender outputs nothing, a malicious sender may output its view in the execution of the protocol. Full simulatability now requires that

- (Sender’s privacy) For any receiver \hat{R} in the real world, there exists a receiver \hat{R}' in the ideal world such that $(S'_{out}, \hat{R}'_{out})$ is indistinguishable from (S_{out}, \hat{R}_{out}) , where A_{out} denotes the output of A .
- (Receiver’s privacy) For any sender \hat{S} in the real world, there exists a sender \hat{S}' in the ideal world such that $(\hat{S}'_{out}, R'_{out})$ is indistinguishable from $(\hat{S}'_{out}, R_{out})$.

They then showed a fully simulatable adaptive OT in the random oracle model, and one in the standard model, respectively [2].

We focus on the standard model in this paper [1]. Then all fully simulatable adaptive OT known so far have been constructed based on pairing, and they rely on dynamic assumptions such as q -strong DH assumption. For example, Camenisch et al.’s $OT_{k \times 1}^n$ relies on q -strong DH assumption and q -PDDH assumption. Green and Hohenberger’s $OT_{k \times 1}^n$ relies on q -hidden LRSW assumption [9]. (This scheme achieves UC security.) Jarecki and Liu’s $OT_{k \times 1}^n$ relies on the decisional q -DHI assumption [10].

This paper shows the first pairing-free *fully simulatable* adaptive OT. It is also the first *fully simulatable* scheme which does not rely on dynamic assumptions. Indeed our scheme holds under the DDH assumption. While the previous schemes use a signature scheme as a building block [2], our scheme utilizes ElGamal encryption scheme. (Hence we do not need a pairing.)

Our scheme is conceptually very simple and efficient. The initialization phase and each transfer phase are constant round protocols. Thus the total round complexity is proportional to k .

Finally we extend our scheme to a fully simulatable non-adaptive OT which requires constant rounds. Green and Hohenberger showed a fully simulatable non-adaptive OT_k^n based on pairing under the decisional BDH assumption [8]. On the other hand, our OT_k^n is pairing-free and relies on the DDH assumption.

Lindell showed a fully simulatable OT_1^2 under DDH, Paillier’s decisional N th residuosity, and quadratic residuosity assumptions as well as under the assumption that homomorphic encryption exists [13]. (He claimed that they can be extended to OT_k^n .) Under the DDH assumption, our OT_1^2 is more efficient than the Lindell’s scheme [13].

¹ In the random oracle model, Ogata and Kurosawa showed an adaptive OT based on Chaum’s blind signature scheme [18]. Camenisch, Neven and shelat [2] proved that it is fully simulatable as well as they corrected a flaw of [18]. Green and Hohenberger showed a scheme under the decisional BDH assumption [8].

² Maybe because an adaptive OT shown by Ogata and Kurosawa [18] utilizes Chaum’s blind signature scheme.

Table 1. Fully simulatable Adaptive OT without RO

scheme	pairing	dynamic assumption	assumption
Caménisch et al. [2]	yes	yes	q -strong DH and q -PDDH
Green and Hohenberger [9]	yes	yes	q -hidden LRSW (UC secure)
Jarecki and Liu [10]	yes	yes	q -DHI
Proposed	no	no	DDH

2 Preliminaries

2.1 Notations

In this paper, we denote a security parameter by $\tau \in \mathbb{N}$. All the algorithms take τ as the first input and run in (expected) polynomial-time in τ . We denote probabilistic polynomial-time by PPT for short. We often do not write the security parameter explicitly.

2.2 Proof Systems

To design our scheme, we use several proof systems. We follow the definitions described in [4,5,2].

Let $R = \{(\alpha, \beta)\} \subseteq \{0, 1\}^* \times \{0, 1\}^*$ be a binary relation R such that $|\beta| \leq \text{poly}(\alpha)$ for all $(\alpha, \beta) \in R$, where poly is some polynomial. We only consider the relation R such that $(\alpha, \beta) \in R$ can be decided in polynomial in $|\alpha|$ for all (α, β) . We define $L_R = \{\alpha \mid \exists \beta \text{ such that } (\alpha, \beta) \in R\}$.

Proof of Membership (PoM): A pair of interacting algorithms (P, V) , called a *prover* and a *verifier*, is a *proof of membership* (PoM) for a relation R if the *completeness* and *soundness* are satisfied. Here, we say that (P, V) satisfies the completeness if for all $(\alpha, \beta) \in R$, the probability of $V(\alpha)$ accepting a conversation with $P(\alpha, \beta)$ is 1. Also we say that (P, V) satisfies the soundness if for all $\alpha \notin L_R$ and all $P^*(\alpha)$ (including cheating provers), the probability of $V(\alpha)$ accepting the conversation with P^* is negligible in $|\alpha|$. We say that this probability as *soundness error* of the proof system.

Proof of Knowledge (PoK): We say a pair of interacting algorithms (P, V) is PoK for a relation R with knowledge error $\kappa \in [0, 1]$ if it satisfies completeness described above and has an expected polynomial-time algorithm, called *knowledge extractor*, E . Here, the algorithm E is a knowledge extractor for a relation R if possibly cheating \hat{P} has probability ϵ of convincing V to accept α , then E , when given black-box access to \hat{P} , outputs a witness β for α with probability $\epsilon - \kappa$.

Witness Indistinguishability (WI): A proof system (P, V) is *perfect* WI if for every $(\alpha, \beta_1), (\alpha, \beta_2) \in R$, and any PPT cheating verifier, the output of $\hat{V}(\alpha)$ (including cheating verifier) after interacting with $P(\beta_1)$ and that of $\hat{V}(\alpha)$ after interacting with $P(\beta_2)$ are identically distributed.

Zero Knowledge (ZK): We say that a proof system (P, V) is *perfect ZK* if there exists an expected polynomial-time algorithm Sim , called a *simulator*, such that for any PPT cheating verifier \widehat{V} and any $(\alpha, \beta) \in R$, the outputs of $\widehat{V}(\alpha)$ after interacting with $P(\beta)$ and that of $\text{Sim}^{\widehat{V}(\alpha)}(\alpha)$ are identically distributed.

3 *k*-Out-of-*n* Oblivious Transfer

In this section, we present a UC-like definition of fully simulatable non-adaptive OT. Similarly, we present a UC-like definition of fully simulatable adaptive OT.

We consider a weak model of UC framework as follows.

- At the beginning of the game, an adversary A can corrupt either a sender S or a receiver R , but not both.
- A can send a message (which will be denoted by A_{out}) to an environment \mathcal{Z} after the end of the protocol. (A cannot communicate with \mathcal{Z} during the protocol execution.)

The ideal functionalities of OT_k^n and $\text{OT}_{k \times 1}^n$ will be shown below. For a protocol $\pi = (S, R)$, define $\text{Adv}(\mathcal{Z})$ as

$$\text{Adv}(\mathcal{Z}) = |\text{Pr}(\mathcal{Z} = 1 \text{ in the real world}) - \text{Pr}(\mathcal{Z} = 1 \text{ in the ideal world})|$$

3.1 Non-adaptive *k*-Out-of-*n* Oblivious Transfer

In the ideal world of OT_k^n , the ideal functionality \mathcal{F}_{non} , an ideal world adversary A' and an environment \mathcal{Z} behave as follows.

(Initialization phase:)

1. An environment \mathcal{Z} sends (M_1, \dots, M_n) to the dummy sender S' .
2. S' sends (M_1^*, \dots, M_n^*) to \mathcal{F}_{non} , where $(M_1^*, \dots, M_n^*) = (M_1, \dots, M_n)$ if S' is not corrupted.

(Transfer phase:)

1. \mathcal{Z} sends $(\sigma_1, \dots, \sigma_k)$ to the dummy receiver R' , where $1 \leq \sigma_i \leq n$.
2. R' sends $(\sigma_1^*, \dots, \sigma_k^*)$ to \mathcal{F}_{non} , where $(\sigma_1^*, \dots, \sigma_k^*) = (\sigma_1, \dots, \sigma_k)$ if R' is not corrupted.
3. \mathcal{F}_{non} sends **received** to an ideal process adversary A' .
4. A' sends $b = 1$ or 0 to \mathcal{F}_{non} , where $b = 1$ if S' is not corrupted.
5. \mathcal{F}_{non} sends Y to R' , where

$$Y = \begin{cases} (M_{\sigma_1}^*, \dots, M_{\sigma_k}^*) & \text{if } b = 1 \\ \perp & \text{if } b = 0 \end{cases}$$

6. R' sends Y to \mathcal{Z} .

After the end of the protocol, A' sends a message A'_{out} to \mathcal{Z} . Finally \mathcal{Z} outputs 1 or 0.

In the real world, a protocol (S, R) is executed without \mathcal{F}_{non} , where the environment \mathcal{Z} and a real world adversary A behave in the same way as above.

Definition 1. *We say that (S, R) is secure against the sender (receiver) corruption if for any real world adversary A who corrupts the sender S (the receiver R), there exists an ideal world adversary A' who corrupts the dummy sender S' (the dummy receiver R') such that for any environment \mathcal{Z} , $\text{Adv}(\mathcal{Z})$ is negligible.*

Definition 2. *We say that (S, R) is a fully simulatable OT_k^n if it is secure against the sender corruption and the receiver corruption.*

3.2 Adaptive k -Out-of- n Oblivious Transfer

In the ideal world of $OT_{k \times 1}^n$, the ideal functionality \mathcal{F}_{adapt} , an ideal world adversary A' and an environment \mathcal{Z} behave as follows.

(Initialization phase:)

1. An environment \mathcal{Z} sends (M_1, \dots, M_n) to the dummy sender S' .
2. S' sends (M_1^*, \dots, M_n^*) to \mathcal{F}_{adapt} , where $(M_1^*, \dots, M_n^*) = (M_1, \dots, M_n)$ if S' is not corrupted.

(Transfer phase:) For $i = 1, \dots, k$,

1. \mathcal{Z} sends σ_i to the dummy receiver R' , where $1 \leq \sigma_i \leq n$.
2. R' sends σ_i^* to \mathcal{F}_{adapt} , where $\sigma_i^* = \sigma_i$ if R' is not corrupted.
3. \mathcal{F}_{adapt} sends **received** to an ideal process adversary A' .
4. A' sends $b = 1$ or 0 to \mathcal{F}_{adapt} , where $b = 1$ if S' is not corrupted.
5. \mathcal{F}_{adapt} sends Y_i to R' , where

$$Y_i = \begin{cases} M_{\sigma_i}^* & \text{if } b = 1 \\ \perp & \text{if } b = 0 \end{cases}$$

6. R' sends Y_i to \mathcal{Z} .

After the end of the protocol, A' sends a message A'_{out} to \mathcal{Z} . Finally \mathcal{Z} outputs 1 or 0.

In the real world, a protocol (S, R) is executed without \mathcal{F}_{adapt} , where the environment \mathcal{Z} and a real world adversary A behave in the same way as above.

Definition 3. *We say that (S, R) is secure against the sender (receiver) corruption if for any real world adversary A who corrupts the sender S (the receiver R), there exists an ideal world adversary A' who corrupts the dummy sender S' (the dummy receiver R') such that for any environment \mathcal{Z} , $\text{Adv}(\mathcal{Z})$ is negligible.*

Definition 4. *We say that (S, R) is a fully simulatable $OT_{k \times 1}^n$ if it is secure against the sender corruption and the receiver corruption.*

3.3 Remarks

Our definition of fully simulatable adaptive OT is weaker than the UC security because our adversaries \mathcal{A} cannot communicate with \mathcal{Z} during the protocol execution. On the other hand, it is stronger than that of [2] which is not UC-like. In our definition, \mathcal{Z} chooses σ_i . Hence σ_i can depend on all of (M_1, \dots, M_n) . In the definition of [2], receiver chooses σ_i . Hence σ_i can depend on $(M_{\sigma_1}, \dots, M_{\sigma_{i-1}})$ only.

4 Our Fully Simulatable Adaptive OT

In this section, we show an adaptive $\text{OT}_{k \times 1}^n$ based on ElGamal encryption scheme, and prove its full simulatability under the DDH assumption.

Let \mathbb{G} be a multiplicative group of prime order q . Then the DDH assumption states that, for every PPT distinguisher \mathcal{D} ,

$$\epsilon_{\text{DDH}}(\mathcal{D}) = |\Pr(\mathcal{D}(g, g^\alpha, g^\beta, g^{\alpha\beta}) = 1) - \Pr(\mathcal{D}(g, g^\alpha, g^\beta, g^\gamma) = 1)|$$

is negligible, where the probability is taken over the random bits of \mathcal{D} , the random choice of the generator g , and the random choice of $\alpha, \beta, \gamma \in \mathbb{Z}_q$. We denote

$$\epsilon_{\text{DDH}} = \max\{\epsilon_{\text{DDH}}(\mathcal{D})\},$$

where the maximum is taken over all PPT distinguishers \mathcal{D} .

The initialization phase and each transfer phase are constant round protocols. Hence the total round complexity is proportional to k .

Initialization Phase

1. The sender chooses \mathbb{G}, g and $(x_1, \dots, x_n, r) \in (\mathbb{Z}_q)^{n+1}$ randomly, and computes $h = g^r$.
2. For $i = 1, \dots, n$, the sender computes

$$C_i = (A_i, B_i) = (g^{x_i}, M_i \cdot h^{x_i}),$$

where $M_1, \dots, M_n \in \mathbb{G}$.

3. The sender sends $(\mathbb{G}, h, C_1, \dots, C_n)$.
4. The sender proves by ZK-PoK that he knows r .

The protocol stops if the receiver rejects.

The j th Transfer Phase

1. The receiver chooses a choice index $1 \leq \sigma_j \leq n$ based on $M_{\sigma_1}, \dots, M_{\sigma_{j-1}}$.
2. The receiver chooses $u \in \mathbb{Z}_q$ randomly and computes $U = (A_{\sigma_j})^u$. He then sends U .
3. The receiver proves in WI-PoK that he knows u such that

$$U = A_1^u \vee \dots \vee U = A_n^u.$$

The protocol stops if the sender rejects.

4. The sender computes $V = U^r$ and sends V .
5. The sender proves that (g, h, U, V) in ZK-PoM that it is a DDH-tuple. The protocol stops if the receiver rejects.
6. The receiver obtains M_{σ_j} by computing $B_{\sigma_j}/V^{1/u}$.

Three ZK or WI proof systems in the scheme are constructed efficiently as follows.

- An efficient 4-round ZK-PoK exists which can be used in the initialization phase. It is obtained by applying the technique of [4] to Schnorr’s identification scheme [19].
- An efficient 3-round WI-PoK exists which can be used in the transfer phase. It is implemented by applying the or-composition technique [5] to [19].
- An efficient 4-round ZK-PoM exists which can be used in the transfer phase. It comes from the confirmation protocol of Chaum’s undeniable signature scheme (which is a ZK-PoM for the DDH-tuple [3]).

Theorem 1. *The above protocol is a fully-simulatable adaptive $OT_{k \times 1}^n$ under the DDH assumption.*

The proof is given in Section 6.

5 Extension to Fully Simulatable Non-adaptive OT

In this section, we extend our adaptive OT to a fully simulatable non-adaptive OT which requires constant rounds.

5.1 How to Prove Many DDH-Tuples

We show a 4-round ZK-PoM which proves that $(g, h, U_1, V_1), \dots, (g, h, U_k, V_k)$ are all DDH-tuples.

1. The receiver sends random (a_1, \dots, a_k) .
2. The sender proves that $(g, h, \prod_{i=1}^k U_i^{a_i}, \prod_{i=1}^k V_i^{a_i})$ is a DDH-tuple by using the confirmation protocol of [3].

The confirmation protocol of [3] is a 4-round ZK-PoM on a DDH-tuple. Hence the above protocol runs in 4-round. (Step 1 and the 1st round of the confirmation protocol are merged.)

Lemma 1. *Suppose that some (g, h, U_i, V_i) is not a DDH-tuples. Then $(g, h, \prod_{i=1}^k U_i^{a_i}, \prod_{i=1}^k V_i^{a_i})$ is a DDH-tuples with negligible probability.*

Proof. Assume that $U_i = g^{x_i}$ and $V_i = h^{y_i}$ for $i = 1, \dots, k$. Then

$$\prod_{i=1}^k U_i^{a_i} = g^{\sum_{i=1}^k a_i x_i}$$

$$\prod_{i=1}^k V_i^{a_i} = h^{\sum_{i=1}^k a_i y_i}$$

Suppose that (g, h, U_1, V_1) is not a DDH-tuples. That is, $x_1 \neq y_1$. Then for any values of a_2, \dots, a_k , there exists a unique a_1 such that

$$\sum_{i=1}^k a_i(x_i - y_i) = 0 \pmod q. \tag{1}$$

Hence the numbers of (a_1, \dots, a_k) which satisfies eq.(1) is equal to q^{k-1} . Therefore

$$\Pr(\text{eq. (1) holds}) = q^{k-1}/q^k = 1/q.$$

This means that $(g, h, \prod_{i=1}^k U_i^{a_i}, \prod_{i=1}^k V_i^{a_i})$ is a DDH-tuples with negligible probability. \square

Theorem 2. *The above protocol is a ZK-PoM on many DDH-tuples.*

Proof. The completeness is clear. The zero-knowledgeness follows from that of the confirmation protocol of [3]. The soundness follows from Lemma 1 and that of the confirmation protocol of [3]. \square

5.2 Constant Round OT_k^n

In this section, we modify our $\text{OT}_{k \times 1}^n$ to obtain a constant round OT_k^n as follows.

- At step 4 of the initialization phase, the sender sends $(\mathbb{G}, h, A_1, \dots, A_n)$.
- At the end of the transfer phase, the sender sends (B_1, \dots, B_n) .
- In the transfer phase, run step 3 in parallel (still it is a WI protocol).
At step 5, the sender proves that $(g, h, U_1, V_1), \dots, (g, h, U_k, V_k)$ are all DDH-tuples by using the ZK-PoM of Sec.5.1

Theorem 3. *The proposed OT_k^n is a constant round fully-simulatable OT_k^n under the DDH assumption.*

The proof is similar to that of Theorem 1.

6 Proof of Theorem 1

We first prove that the proposed scheme is secure against sender corruption. We next prove that it is secure against receiver corruption.

6.1 Security against Sender Corruption

Lemma 2. *The proposed scheme is secure against sender corruption.*

Proof. For every real-world adversary A who corrupts the sender, we construct an ideal-world adversary A' such that $\text{Adv}(\mathcal{Z})$ is negligible.

We will consider a sequence of games $\text{Game}_0, \text{Game}_1, \dots, \text{Game}_4$, where Game_0 is the real world experiment of Sec.3 and Game_4 is the ideal world experiment, respectively. Let

$$\Pr(\text{GAME}_i) = \Pr(\mathcal{Z} = 1 \text{ in Game}_i).$$

Game₀: This is the real world experiment such that the sender is controlled by an adversary A . Hence

$$\Pr(\text{GAME}_0) = \Pr(\mathcal{Z} = 1 \text{ in the real world}).$$

Game₁: This is the same as the previous game except for the following. In the initialization phase, if the receiver accepts the ZK-PoK, then he extracts r from A by running the knowledge extractor E_1 which is allowed to rewind A . This game outputs \perp if the extractor E_1 fails in extracting r . Unless this happens, these two games are identical. Therefore,

$$|\Pr(\text{GAME}_0) - \Pr(\text{GAME}_1)| \leq \kappa_1,$$

where κ_1 be the knowledge error of the extractor.

Game₂: This is the same as the previous game except for the following. In each transfer phase, if the receiver accepts the ZK-PoM which proves that (g, h, U, V) is a DDH-tuple, then he obtains M_{σ_i} by computing $B_{\sigma_i}/A_{\sigma_i}^r$. These two games are identical unless the above M_{σ_i} is different from $B_{\sigma_j}/V^{1/u}$. This happens if the receiver accepts the ZK-PoM even though (g, h, U, V) is not a DDH-tuple. Hence

$$|\Pr(\text{GAME}_1) - \Pr(\text{GAME}_2)| \leq k\kappa_3,$$

where κ_3 is the soundness error probability of ZK-PoM.

Game₃: This is the same as the previous game except for the following. In each transfer phase, the receiver computes U as $U = A_1^u$. (The receiver can still obtain M_{σ_i} as can be seen from **Game₂**.) Since our WI-PoK is perfect,

$$\Pr(\text{GAME}_2) = \Pr(\text{GAME}_3).$$

Game₄: This game is the ideal world experiment in which an ideal-world adversary A' plays the role of the receiver of **Game₃** and uses A as a blackbox. A' can do this because the receiver does not use $\sigma_1, \dots, \sigma_k$ in **Game₃**.

Finally A' outputs what A outputs. It is easy to see that **Game₃** and **Game₄** are identical from a view point of \mathcal{Z} . Hence

$$\Pr(\text{GAME}_3) = \Pr(\text{GAME}_4).$$

Further

$$\Pr(\text{GAME}_4) = \Pr(\mathcal{Z} = 1 \text{ in the ideal world}).$$

Now, we can summarize this lemma as follows:

$$\begin{aligned} \text{Adv}(\mathcal{Z}) &= |\Pr(\text{GAME}_4) - \Pr(\text{GAME}_0)| \\ &\leq \sum_{i=0}^3 |\Pr(\text{GAME}_{i+1}) - \Pr(\text{GAME}_i)| \\ &\leq \kappa_1 + k\kappa_3. \end{aligned}$$

□

6.2 Security against Receiver Corruption

Lemma 3. *The proposed scheme is secure against receiver corruption under the DDH assumption.*

Proof. For every real-world adversary A who corrupts the receiver, we construct an ideal-world adversary A' such that $\text{Adv}(\mathcal{Z})$ is negligible.

We will consider a sequence of games $\text{Game}_0, \text{Game}_1, \dots, \text{Game}_5$, where Game_0 is the real world experiment of Sec. 3 and Game_5 is the ideal world experiment.

Game_0 : This is the real world experiment such that the receiver is controlled by an adversary A . Hence

$$\Pr(\text{GAME}_0) = \Pr(\mathcal{Z} = 1 \text{ in the real world}).$$

Game_1 : This is the same as the previous game except for the following. In each transfer phase, instead of running the ZK-PoM which proves that (g, h, U, V) is a DDH-tuple, the sender runs the zero-knowledge simulator of the ZK-PoM which is allowed to rewind A . Since the ZK-PoM is perfect ZK, we have

$$\Pr(\text{GAME}_1) = \Pr(\text{GAME}_0).$$

Game_2 : This is the same as the previous game except for the following. In each transfer phase, if the sender accepts the WI-PoK, then she extracts u from A by running the knowledge extractor E_2 which is allowed to rewind A . This game outputs \perp if the extractor E_2 fails in extracting u . Unless this happens, these two games are identical. Therefore,

$$|\Pr(\text{GAME}_2) - \Pr(\text{GAME}_1)| \leq k\kappa_2,$$

where κ_2 is the knowledge error of the extractor.

Game_3 : This is the same as the previous game except for that the sender computes V as $V = (B_\sigma/M_\sigma)^u$ instead of $V = U^r$. It is clear that there is no essential difference between two games. Therefore,

$$\Pr(\text{GAME}_3) = \Pr(\text{GAME}_2).$$

Game_4 : This is the same as the previous game except for that the sender uses a random M'_i to compute each C_i in the initialization phase. The difference $|\Pr(\text{GAME}_4) - \Pr(\text{GAME}_3)|$ is still negligible by the semantic security of the ElGamal cryptosystem which is implied by the DDH assumption.

Claim. If the DDH problem is hard then $|\Pr(\text{GAME}_4) - \Pr(\text{GAME}_3)|$ is negligible. More concretely,

$$|\Pr(\text{GAME}_4) - \Pr(\text{GAME}_3)| \leq \epsilon_{\text{DDH}}. \quad (2)$$

The proof of this claim is given later.

Game₅: This game is the ideal world experiment in which an ideal-world adversary A' plays the role of the sender of **Game₄**, and uses A as a blackbox. A' can do this because the sender does not use M_1, \dots, M_n in **Game₄**.

Finally A' outputs what A outputs. It is easy to see that **Game₄** and **Game₅** are identical from a view point of \mathcal{Z} . Hence

$$\Pr(\overline{\text{GAME}}_4) = \Pr(\text{GAME}_5).$$

Further

$$\Pr(\text{GAME}_5) = \Pr(\mathcal{Z} = 1 \text{ in the ideal world}).$$

Now, we can summarize this lemma as follows:

$$\begin{aligned} \text{Adv}(\mathcal{Z}) &= |\Pr(\text{GAME}_5) - \Pr(\text{GAME}_0)| \\ &\leq \sum_{i=0}^4 |\Pr(\text{GAME}_{i+1}) - \Pr(\text{GAME}_i)| \\ &\leq k\kappa_2 + \epsilon_{\text{DDH}}. \end{aligned} \quad \square$$

To complete the proof, we must provide the proof of the claim. To do so, we need the following lemma³ which can be thought of as an “extended” version of the DDH assumption.

Lemma 4 (Lemma 4.2 in [17]). *If there exists a probabilistic algorithm D with running time t such that*

$$\left| \Pr(D(g, g^r, g^{x_1}, \dots, g^{x_n}, g^{rx_1}, \dots, g^{rx_n}) = 1) - \Pr(D(g, g^r, g^{x_1}, \dots, g^{x_n}, g^{z_1}, \dots, g^{z_n}) = 1) \right| \geq \epsilon$$

where the probability is taken over the random bits of D , the random choice of the generator g in \mathbb{G} , and the random choice of $x_1, \dots, x_n, r, z_1, \dots, z_n \in \mathbb{Z}_q$, then there exists a probabilistic algorithm with running time $n \cdot \text{poly}(\tau) + t$ that breaks the DDH assumption with probability $\geq \epsilon$ with some polynomial poly .

We now show a proof of the claim.

Proof (of the claim). Let **Game'₃** (**Game'₄**) be the same as **Game₃** (**Game₄**) except for the following. In the initialization phase, instead of running the ZK-PoK in which the sender proves that he knows r , the sender runs the zero-knowledge simulator of the ZK-PoK which is allowed to rewind A . Since the ZK-PoK is perfect ZK, it holds that

$$\begin{aligned} \Pr(\text{GAME}'_3) &= \Pr(\text{GAME}_3), \\ \Pr(\text{GAME}'_4) &= \Pr(\text{GAME}_4). \end{aligned}$$

³ Naor and Reingold proved it by using the random reducibility of the DDH-tuple.

We now construct a DDH distinguisher D in the sense of Lemma 4. The input to D is $(g, h, g^{x_1}, \dots, g^{x_n}, y_1, \dots, y_n)$, where $y_i = g^{r x_i}$ or g^{z_i} . Our D simulates \mathcal{Z} , A and the sender of Game'_3 or Game'_4 faithfully except for that in the initialization phase, D simulates the sender by using $(g, h, g^{x_1}, \dots, g^{x_n})$, and $h_i = y_i$ for each i . Finally D outputs 1 iff \mathcal{Z} outputs 1.

It is easy to see that D simulates Game'_3 if $y_i = g^{r x_i}$ for each i , and Game'_4 otherwise. Therefore

$$|\Pr(\text{GAME}'_4) - \Pr(\text{GAME}'_3)| \leq \epsilon_{\text{DDH}}. \quad (3)$$

Hence eq. (2) holds. \square

7 Fully Simulatable OT_1^2

We have constructed a fully-simulatable adaptive OT under the DDH assumption in the standard model. It is clear that we can obtain a fully-simulatable $(1, 2)$ -OT (OT_1^2) as a special case.

On the other hand, Lindell showed a fully simulatable OT_1^2 under DDH, Paillier's decisional N th residuosity, and quadratic residuosity assumptions as well as under the assumption that homomorphic encryption exists in the standard model [13].

Let's compare our scheme with Lindell's OT_1^2 which is based on the DDH assumption. His scheme builds on the OT_1^2 of [16] and uses a cut-and-choose technique. The computational cost and the communication cost are $O(\ell)$ times larger than those of our first scheme to achieve

$$\text{Adv}(\mathcal{Z}) \leq 2^{-\ell+2}.$$

Hence our scheme is more efficient.

References

1. Brassard, G., Crépeau, C., Robert, J.M.: All-or-Nothing Disclosure of Secrets. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 234–238. Springer, Heidelberg (1987)
2. Camenisch, J.L., Neven, G., Shelat, A.: Simulatable Adaptive Oblivious Transfer. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 573–590. Springer, Heidelberg (2007)
3. Chaum, D.: Zero-Knowledge Undeniable Signatures. In: Damgård, I.B. (ed.) EUROCRYPT 1990. LNCS, vol. 473, pp. 458–464. Springer, Heidelberg (1991)
4. Cramer, R., Damgård, I., MacKenzie, P.D.: Efficient Zero-Knowledge Proofs of Knowledge Without Intractability Assumptions. In: Imai, H., Zheng, Y. (eds.) PKC 2000. LNCS, vol. 1751, pp. 354–373. Springer, Heidelberg (2000)
5. Cramer, R., Damgård, I.B., 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)

6. Even, S., Goldreich, O., Lempel, A.: A Randomized Protocol for Signing Contracts. *Commun. ACM* 28(6), 637–647 (1985)
7. Goldreich, O., Micali, S., Wigderson, A.: How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority. In: *STOC 1987*, pp. 218–229 (1987)
8. 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)
9. Green, M., Hohenberger, S.: Universally Composable Adaptive Oblivious Transfer. In: Pieprzyk, J. (ed.) *ASIACRYPT 2008*. LNCS, vol. 5350, pp. 179–197. Springer, Heidelberg (2008)
10. 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)
11. Kalai, Y.T.: Smooth Projective Hashing and Two-Message Oblivious Transfer. In: Cramer, R. (ed.) *EUROCRYPT 2005*. LNCS, vol. 3494, pp. 78–95. Springer, Heidelberg (2005)
12. Kilian, J.: Founding Cryptography on Oblivious Transfer. In: *STOC 1988*, pp. 20–31 (1988)
13. Lindell, A.Y.: Efficient Fully-Simulatable Oblivious Transfer. In: Malkin, T.G. (ed.) *CT-RSA 2008*. LNCS, vol. 4964, pp. 52–70. Springer, Heidelberg (2008)
14. Naor, M., Pinkas, B.: Oblivious Transfer and Polynomial Evaluation. In: *STOC 1999*, pp. 245–254 (1999)
15. Naor, M., Pinkas, B.: Oblivious Transfer with Adaptive Queries. In: Wiener, M. (ed.) *CRYPTO 1999*. LNCS, vol. 1666, pp. 573–590. Springer, Heidelberg (1999)
16. Naor, M., Pinkas, B.: Efficient Oblivious Transfer Protocols. In: *SODA 2001*, pp. 448–457 (2001)
17. Naor, M., Reingold, O.: Number-Theoretic Constructions of Efficient Pseudorandom Functions. *J. ACM* 51(2), 231–262 (2004)
18. Ogata, W., Kurosawa, K.: Oblivious Keyword Search. *J. Complexity* 20(2-3), 356–371 (2004); *Cryptology ePrint Archive: Report 2002/182*
19. Schnorr, C.-P.: Efficient Signature Generation by Smart Cards. *J. Cryptology* 4(3), 161–174 (1991)
20. Yao, A.C.-C.: How to Generate and Exchange Secrets. In: *FOCS 1986*, pp. 162–167 (1986)

Improved Generic Algorithms for 3-Collisions

Antoine Joux¹ and Stefan Lucks²

¹ DGA and Université de Versailles Saint-Quentin-en-Yvelines
UVSQ PRISM, 45 avenue des États-Unis, F-78035, Versailles CEDEX, France
antoine.joux@m4x.org

² BAUHAUS-UNIVERSITÄT WEIMAR, 99423 Weimar, Germany
Stefan.Lucks@uni-weimar.de

Abstract. An r -collision for a function is a set of r distinct inputs with identical outputs. Actually finding r -collisions for a random map over a finite set of cardinality N requires at least about $N^{(r-1)/r}$ units of time on a sequential machine. For $r=2$, memoryless and well-parallelizable algorithms are known. The current paper describes memory-efficient and parallelizable algorithms for $r \geq 3$. The main results are: (1) A sequential algorithm for 3-collisions, roughly using memory N^α and time $N^{1-\alpha}$ for $\alpha \leq 1/3$. In particular, given $N^{1/3}$ units of storage, one can find 3-collisions in time $N^{2/3}$. (2) A parallelization of this algorithm using $N^{1/3}$ processors running in time $N^{1/3}$, where each single processor only needs a constant amount of memory. (3) A generalisation of this second approach to r -collisions for $r \geq 3$: given N^s parallel processors, with $s \leq (r-2)/r$, one can generate r -collisions roughly in time $N^{((r-1)/r)-s}$, using memory $N^{((r-2)/r)-s}$ on every processor.

Keywords: multicollision, random map, memory-efficient, parallel implementation, cryptanalysis.

1 Introduction

The problem of finding collisions and multicollisions in random mappings is of significant interest for cryptography, and mainly for cryptanalysis. It is well known that finding an r -collision for a random map over a finite set of cardinality N requires [\[1\]](#) more than $N^{(r-1)/r}$ map evaluations.

Multicollisions for hash functions. If the map under consideration is a hash function, or has been derived from a hash function, many researchers consider faster multicollisions as a certification hash function weakness. Accordingly, it was worrying for the research community to learn that multicollisions could be found much faster for a widely used class of hash functions: iterated hash functions [\[9\]](#). For n -bit hash functions from this class, one can generate 2^k -collisions in time

¹ An r -collision is a set of r different inputs x_1, \dots, x_r which all generate the same output $\text{map}(x_1) = \dots = \text{map}(x_r)$. For an r -collision, one needs to evaluate the map $(r!)^{1/r} \cdot N^{(r-1)/r}$ times [\[22\]](#). For small r , we can approximate this by $O(N^{(r-1)/r})$.

$k \cdot 2^{n/2}$, rather than the expected time $2^{n(2^k-1)/2^k}$. The basic observation is straightforward: given a sequence of k consecutive 2-collisions, it is possible with iterated hash functions to consider the 2^k different messages obtained by taking all possible choices of message block for each collision and obtain 2^k times the same output. These iterated multicollisions have been generalized later, to more complex types of iterated hash functions, for example, see [6,13,7]. It was also remarked that these iterated multicollisions were a rediscovery and generalization of an older attack of Coppersmith [2].

In particular, this type of multicollisions allowed a surprising attack on hash cascades, i.e., hash functions H , which are the concatenation of two hash functions G_1 and G_2 , i.e., $H(X) := (G_1(X), G_2(X))$. If, say, G_1 is an iterated hash function and vulnerable to the multicollision attack, and G_2 is any n -bit hash function, the adversary just needs to generate a $2^{n/2}$ -multicollision for G_1 . Thanks to the birthday paradox, among the $2^{n/2}$ messages colliding for G_1 , one expects to find a pair of messages colliding for G_2 with constant probability. As a consequence, a collision for the $2n$ -bit hash function H can be obtained with much less than 2^n hash evaluations.

Multicollisions for random maps. In contrast to [9], we consider generic attacks, and, accordingly, we model our functions as random maps. In that case, the number of $N^{(r-1)/r}$ is a lower bound on the sequential time required for finding a r -collision, and time-optimal algorithms are well-known. Furthermore, it is well-known how to find ordinary collisions (aka 2-collisions) with negligible memory (using Floyd, Brent or Nivasch [15] cycle finding algorithms), and also how to parallelize these algorithms using distinguished point methods [18,19,20,23,24,25,26,27].

In general, the issue of memory-efficient and parallelizable r -collision algorithms appears to be an unsolved question. Authors usually assume $N^{(r-1)/r}$ units of memory (i.e., the maximum any algorithm can use in the given amount of time) and neglect parallelization entirely. For recent examples of the application of multicollisions to cryptography, see, e.g., the cryptanalysis of the SHA-3 candidates Aurora-512 [3,21] and JH-512 [12,29]. We stress that [3,21,12,29] employ generic multicollisions as a part of their attacks, always assuming maximum memory and ignoring the issue of parallel implementations.

So the question is, *do authors need to be so pessimistic, or are there memory-efficient and parallelizable algorithms for r -collisions?* For small r , and mainly for $r = 3$, the current paper provides a clearly positive answer. As an application of our results, we will observe attacks on the SHA-3 candidate hash function Aurora-512. These attacks make heavy use of multicollisions on internal structures. Some attacks on other SHA-3 candidates don't benefit from our algorithms for different reasons. See section B of the appendix.

Notation. To avoid writing cumbersome logarithmic factors, we often express running times using the soft-Oh-notation. Namely, $\tilde{O}(g(n))$ is used as a shorthand for $O(g(n) \cdot \log(g(n))^k)$ for some fixed k .

2 Known Algorithm for 3-Collisions

While the number of values that needs to be computed before a 3-collision can be formed is often considered and analyzed, e.g. in [17, Appendix B] or [22], the known algorithmic method to find such a 3-collision is rarely considered in detail and is mostly folklore. In order to compare the new algorithms which we describe in sections 3 to 6 with existing algorithms, we thus give a precise description of the folklore algorithm, together with a larger variety of time/memory tradeoffs. Throughout this section, we fix two parameters α and β and consider 3-collisions for a function F defined on a set of cardinality N . The parameter α controls the amount of memory, limiting it to $\tilde{O}(N^\alpha)$. Similarly, β controls the running time, at $\tilde{O}(N^\beta)$. Of course, these parameters need to satisfy the relation $\alpha \leq \beta$.

We consider Algorithm 1. This algorithm is straightforward. First, it computes, stores and sorts N^α images of random points under F . For bookkeeping purposes, it also keeps track of the corresponding preimages. Second, it computes N^β additional images of random points and seek each in the precomputed table. Whenever a hit occurs, it is stored together with the initial preimage in the sorted table. The algorithm succeeds if one of the N^α original images is found twice more during the second phase and if the three corresponding preimages are distinct. In the formal description given as Algorithm 1, we added an optional step which packs colliding values generated during the first step into the same array element. If this optional step is omitted, then the early collisions are implicitly discarded. Indeed, in the second phase, we make sure that the search algorithm always returns the first position where a given value occurs among the known images $F(x)$. During the complexity analysis, we ignore the optional packing step since it runs in time N^α and can only improve the overall running time by making the algorithm stop earlier.

We now perform a rough heuristic analysis of Algorithm 1, where constants and logarithmic factors are ignored. On average, among the N^β images of the second phase, we expect that $N^{\alpha+\beta-1}$ values hit the sorted table of N^α elements. Due to the birthday paradox, after $N^{\alpha/2}$ hits, we expect a double hit to occur. At that point, the algorithm succeeds if the three known preimages corresponding to the double hit are distinct, which occurs with constant probability. For the algorithm to succeed, we need:

$$\alpha + \beta - 1 \geq \alpha/2,$$

as a consequence, to minimize the running time, we enforce the condition:

$$\alpha + 2\beta = 2. \tag{1}$$

For $\alpha = \beta$, we find $\alpha = \beta = 2/3$ and obtain the classical folklore result with time and memory $\tilde{O}(N^{2/3})$. Other tradeoffs are also possible. With constant memory, i.e. $\alpha = 0$, we find a running time $\tilde{O}(N)$. Another tradeoff with $\alpha = 1/2$ and $\beta = 3/4$ will be used as a point of comparison in section 3.

Algorithm 1. Folklore 3-collision finding algorithm

Require: Oracle access to F operating on $[0, N - 1]$ **Require:** Parameters: $\alpha \leq \beta$ satisfying condition \square Let $N_\alpha \leftarrow \lceil N^\alpha \rceil$ Let $N_\beta \leftarrow \lceil N^\beta \rceil$ Create arrays Img , Pr_1 and Pr_2 of N_α elements.**First step:****for** i from 1 to N_α **do**Let $a \leftarrow_R [0, N - 1]$ Let $\text{Img}[i] \leftarrow F(a)$ Let $\text{Pr}_1[i] \leftarrow a$ Let $\text{Pr}_2[i] \leftarrow \perp$ **end for****Sort** Img , applying the same permutation on elements of Pr_1 and Pr_2 **Optional step (packing of existing collisions):**Let $i \leftarrow 1$ **while** $i < N_\alpha$ **do**Let $j \leftarrow i + 1$ **while** $\text{Img}[i] == \text{Img}[j]$ **do****if** $\text{Pr}_1[i] \neq \text{Pr}_1[j]$ **then****if** $\text{Pr}_2[i] == \perp$ **then**Let $\text{Pr}_2[i] \leftarrow \text{Pr}_1[j]$ **else****if** $\text{Pr}_2[j] \neq \text{Pr}_1[j]$ **then**Output ‘3-Collision ($\text{Pr}_1[i], \text{Pr}_2[i], \text{Pr}_1[j]$) under F ’ and **Exit****end if****end if****end if**Let $j \leftarrow j + 1$ **end while**Let $i \leftarrow j$ **end while****Second step:****for** i from 1 to N_β **do**Let $a \leftarrow_R [0, N - 1]$ Let $b \leftarrow F(a)$ **if** b is in Img (first occurrence in position j) **then****if** $\text{Pr}_1[j] \neq a$ **then****if** $\text{Pr}_2[j] == \perp$ **then**Let $\text{Pr}_2[j] \leftarrow a$ **else****if** $\text{Pr}_2[j] \neq a$ **then**Output ‘3-Collision ($\text{Pr}_1[j], \text{Pr}_2[j], a$) under F ’ and **Exit****end if****end if****end if****end if****end for**

3 A New Algorithm for 3-Collisions

Now equipped with an analysis of Algorithm [1](#), we are ready to propose a new algorithm which offers different time-memory tradeoffs, which are better balanced for existing hardware. The basic idea is extremely simple: Instead of initializing an array with N^α images, we propose to initialize it with N^α collisions under F . To make this efficient in terms of memory use, each collision in the array is generated using a cycle finding algorithm on a (pseudo-)randomly permuted copy of F . Since each collision is found in time $N^{1/2}$ the total running time of this new first step is $N^{1/2+\alpha}$.

The second step is left unchanged, we simply create N^β images of random points until we hit one of the known collisions. Note that, thanks to the new first phase, it now suffices to land once on a known point to succeed. As a consequence, we can replace condition [1](#) by the weaker condition:

$$\alpha + \beta = 1. \tag{2}$$

Since the running time of the first step is $N^{1/2+\alpha}$, it would not make sense to have $\beta < 1/2 + \alpha$. Thus, we also enforce the condition $\alpha \leq 1/4$. Under this condition, the new algorithm runs in time $\tilde{O}(N^{1-\alpha})$ using $\tilde{O}(N^\alpha)$ bits of memory. In particular, we can find 3-collisions in time $\tilde{O}(N^{3/4})$ using $\tilde{O}(N^{1/4})$ bits of memory. This is a notable improvement over Algorithm [1](#) which requires $\tilde{O}(N^{1/2})$ bits of memory to achieve the same running time.

Note on the creation of the N^α initial collisions. One question that frequently arises when this algorithm is presented is: “Why is it necessary to randomize F with a pseudo-random permutation?”

Behind this question is the idea that changing the starting point of the cycle finding algorithm should suffice to obtain random collisions. However, this is not true. Indeed, the analysis of random mapping (for example, see [4](#)) shows that on average a constant fraction of points belong to a so-called “giant tree”. By definition, each starting point in the giant tree enters the main cycle in the same place. As a consequence, without randomization of F the corresponding collision would be generated over and over again and the 3-collision algorithm would not work.

4 Detailed Complexity Analysis of Algorithms [1](#) and [2](#)

In this section, we analyze in more details the complexity and success probability of algorithms [1](#) and [2](#), assuming that F is a random mapping. This detailed analysis particularly focuses on the following problematic issues which were initially neglected:

1. Among the N_α candidates stored in `Img` and its companion arrays, which fraction can non-trivially be completed into a 3-collision?
2. In the second step, when a value $F(a)$ hits the array `Img`, what is the probability of obtaining a real 3-collision and not simply replaying a known value of a ?

Algorithm 2. Improved 3-collision finding algorithm

Require: Oracle access to F operating on $[0, N - 1]$ **Require:** Family of pseudo-random permutation Π_K , indexed by K in \mathcal{K} **Require:** Parameters: $\alpha \leq \beta$ satisfying condition 2Let $N_\alpha \leftarrow \lceil N^\alpha \rceil$ Let $N_\beta \leftarrow \lceil N^\beta \rceil$ Create arrays Img , Pr_1 and Pr_2 of N_α elements.**First step:****for** i from 1 to N_α **do**Let $K \leftarrow_R \mathcal{K}$ Use cycle finding algorithm on $F \circ \Pi_K$ to produce collision $F \circ \Pi_K(a) = F \circ \Pi_K(b)$ Let $\text{Img}[i] \leftarrow F \circ \Pi_K(a)$ Let $\text{Pr}_1[i] \leftarrow \Pi_K(a)$ Let $\text{Pr}_2[i] \leftarrow \Pi_K(b)$ **end for****Sort** Img , applying the same permutation on elements of Pr_1 and Pr_2 **Optional step (packing of existing collisions):**Let $i \leftarrow 1$ **while** $i < N_\alpha$ **do**Let $j \leftarrow i + 1$ **while** $\text{Img}[i] == \text{Img}[j]$ **do****if** $\text{Pr}_1[i] \neq \text{Pr}_1[j]$ **then****if** $\text{Pr}_2[i] \neq \text{Pr}_1[j]$ **then**Output ‘3-Collision $(\text{Pr}_1[i], \text{Pr}_2[i], \text{Pr}_1[j])$ under F ’ and **Exit****end if****end if**Let $j \leftarrow j + 1$ **end while**Let $i \leftarrow j$ **end while****Second step:****for** i from 1 to N_β **do**Let $a \leftarrow_R [0, N - 1]$ Let $b \leftarrow F(a)$ **if** b is in Img (first occurrence in position j) **then****if** $\text{Pr}_1[j] \neq a$ **then****if** $\text{Pr}_2[j] == \perp$ **then**Let $\text{Pr}_2[j] \leftarrow a$ **else****if** $\text{Pr}_2[j] \neq a$ **then**Output ‘3-Collision $(\text{Pr}_1[j], \text{Pr}_2[j], a)$ under F ’ and **Exit****end if****end if****end if****end if****end for**

3. Which logarithmic factors are hidden in the \tilde{O} expression ?
4. In the first step of Algorithm 2, how can we make sure that we never encounter a bad configuration where the cycle finding algorithm runs for longer than $\tilde{O}(N^{1/2})$?

To answer the first question, remark that each candidate stored into Img is a random point that has at least one preimage for Algorithm 1 or at least two preimages for Algorithm 2. According to 4, we know that the expected fraction of points with exactly k distinct preimages is $e^{-1}/k!$. As a consequence, if we denote by P_k the fraction of points with at least k preimages, we find:

$$P_1 = \frac{e - 1}{e}, \quad P_2 = \frac{e - 2}{e} \quad \text{and} \quad P_3 = \frac{e - 5/2}{e}.$$

The expected fraction of elements from Img which can be correctly completed into a 3-collision is $P_3/P_1 \approx 0.127$ for Algorithm 1 and $P_3/P_2 \approx 0.304$ for Algorithm 2. To compensate the loss, the easiest is to make the stored set larger by a factor of 8 in the first case and 3 in the second.

We now turn to the second question. Of course, at this point, the candidates that cannot be correctly completed need to be ignored. Among the original set of N_α candidates, we now focus on the subset of candidates that can correctly be computed and let N'_α denote the size of this subset. Since in the second phase we are sampling points uniformly at random, the *a posteriori* probability of having chosen one of the two already known preimages is at most $2/k$, where k is the number of distinct preimages for this point. Since $k \geq 3$, the *a posteriori* probability of choosing a new preimage is, at least, $1/3$. Similarly, for Algorithm 1, the *a posteriori* probability of choosing a preimage distinct from the single originally known one is at least $2/3$. To offset this loss of probability, N_β should be multiplied by a constant factor of 3.

The logarithmic factors involved in the third question are easy to find, they simply come from the sort and binary search steps. Note that when $N^\alpha \cdot \log(N^\alpha) < N^\beta$ the sort operation costs less than the second step and can be ignored. Moreover, as soon as $\alpha < \beta$, this bound is asymptotically achieved when N tends to infinity. However, the binary search appears within the second step and a real penalty is paid.

If we are willing to spend some extra memory – blowing up the memory by a constant factor –, this cost can be eliminated using hashing techniques. To cover the case of $N^\alpha \cdot \log(N^\alpha) = N^\beta$, we need a data structure with constant-time insert and lookup operations. One such data structure is “cuckoo hashing”, where lookup operations need worst-case constant time, and insert operations need expected constant time – as long as less than half of the memory slots are used [16] 2. However, for typical applications, the cost of the binary search ought to remain small, compared to the cost of evaluating the function F . Thus, in practice, we expect only a tiny benefit from using hash tables.

² Furthermore, delete operations only need worst-case constant time, and recent improvements even enable update operations in worst-case constant time [1].

The simplest answer to the fourth question is to fix some upper bound on the allowed running time of each individual call to the collision through cycle finding algorithm. If the running time is exceeded, we abort and restart with a fresh permutation Π_K . With a time limit of the form $\lambda\sqrt{N}$ and a large enough value of λ , we make sure that each individual call to the cycle finding algorithm runs in time $O(N^{1/2})$ and the probability of success is a constant close to 1, say larger than $2/3$.

5 A Second Algorithm with More Tradeoff Options

The algorithm presented in section 3 only works for memory up to $N^{1/4}$. This limitation is due to the way the collisions are generated during the first step of Algorithm 2. In order to extend the range of possible tradeoffs beyond that point, it suffices to find a replacement for this first step. Indeed, the second step clearly works with a larger value of α , as long as we keep the relation $\alpha + \beta = 1$. Of course, since no 3-collision is expected before we have performed $N^{2/3}$ evaluations of F , the best we can hope for is an algorithm with running time $N^{2/3}$. Such an algorithm may succeed if we can precompute a table containing $N^{1/3}$ ordinary collisions.

In this section, we consider the problem of generating $N^{1/3}$ collisions in time bounded by $\tilde{O}(N^{2/3})$ using at most $\tilde{O}(N^{1/3})$ bits of memory. Surprisingly, a simple method inspired from Hellman's time-memory tradeoff [5] is able to solve this problem. More generally, for $\alpha \leq 1/3$, this method allows us to compute N^α collisions in time less than $\tilde{O}(N^{1-\alpha})$ using at most $\tilde{O}(N^\alpha)$ bits of memory. The idea is to first build N^α chains of length N^γ ; each chain starts from a random point and is computed by repeatedly applying F up to the N^γ -th iteration. The end-point of each chain is stored together with its corresponding start-point. Once the chains have been build, we sort them by end-point values. Then, restarting from N^α new random points, we once again compute chains of length N^γ , the difference is that we now test after each evaluation of F whether the current value is one of the known end-points. In that case, we know that the chain we are currently computing has merged with one chain from the precomputation step. Such a merge usually corresponds to a collision, the only exception occurs when the start-point of the current chain already belongs to a precomputed chain (a "Robin Hood" using the terminology of [27]). Then, backtracking to the beginning of both chains, we can easily construct the corresponding collision. A pseudo-code description of this alternative first step is given as Algorithm 3.

Note that, instead of building two sets of chains, it is also possible to build a single set and look for previously known end-points. This alternative approach is a bit trickier to implement but uses fewer evaluations of F . However, the overall cost of the algorithm remains within the same order.

Clearly, since each of the two sets of chains we are constructing contain $N^{\alpha+\gamma}$ points, the expected number of collisions is $O(N^{2\alpha+2\gamma-1})$. Remembering that we wish to construct N^α collisions, we need to let $\gamma = (1 - \alpha)/2$. The running time necessary to compute these collisions is $N^{\alpha+\gamma} = N^{(1+\alpha)/2}$. Note that, since

Algorithm 3. Alternative method for constructing N^α collisions**Require:** Oracle access to F operating on $[0, N - 1]$ **Require:** Parameter: $\alpha \leq 1/3$ Let $\gamma \leftarrow (1 - \alpha)/2$ Let $N_\alpha \leftarrow \lceil N^\alpha \rceil$ Let $N_\gamma \leftarrow \lceil N^\gamma \rceil$ Create arrays Start and End of N_α elements.Create arrays Img, Pr₁ and Pr₂ of N_α elements.**Construction of first set:****for** i from 1 to N_α **do**Let $a \leftarrow_R [0, N - 1]$ Let Start[i] $\leftarrow a$ **for** i from 1 to N_γ **do**Let $a \leftarrow F(a)$ **end for**Let End[i] $\leftarrow a$ **end for****Sort** End, applying the same permutation on elements of Start**Construction of second set and collisions:**Let $t \leftarrow 1$ **while** $t < N_\alpha$ **do**Let $a \leftarrow_R [0, N - 1]$ Let $b \leftarrow a$ **for** j from 1 to N_γ **do**Let $b \leftarrow F(b)$ **if** b is in End (first occurrence in position k) **then**Let $a' \leftarrow \text{Start}[k]$ **for** l from 1 to $N_\gamma - j$ **do**Let $a' \leftarrow F(a')$ **end for****if** $a \neq a'$ **then**

{Checks that a genuine merge between chains exists}

Let $b \leftarrow F(a)$ Let $b' \leftarrow F(a')$ **while** $b \neq b'$ **do**Let $a \leftarrow b$ Let $a' \leftarrow b'$ Let $b \leftarrow F(a)$ Let $b' \leftarrow F(a')$ **end while**Let Img[t] $\leftarrow b$ Let Pr₁[t] $\leftarrow a$ Let Pr₂[t] $\leftarrow a'$ Let $t \leftarrow t + 1$ **end if**Exit Loop on j **end if****end for****end while**Return arrays Img, Pr₁ and Pr₂ containing N_α collisions.

$\alpha \leq 1/3$, we have $(1 + \alpha)/2 \leq 1 - \alpha$. As a consequence, the running time of the complete algorithm is dominated by the running time $N^\beta = N^{1-\alpha}$ of the second step.

6 Parallelizable 3-Collision Search

Since the computation involved during a search for 3-collisions is massive, it is essential to study the possibility of parallelizing such a search. For ordinary collisions, parallelization is studied in details in [27] using ideas introduced in [18,19,20,23,24,25,26].

We first remark that the algorithms we have studied up to this point are badly suited to parallelization. Their main problem is that a large amount of memory needs to be replicated on every processor which is very impractical, especially when we want to use a large amount of low-end processors. We now propose an algorithm specifically suited to parallelization. For simplicity of exposition, we first assume that $N_p \approx N^{1/3}$ processors are available and aim at a running time $\tilde{O}(N^{1/3})$. Moreover, we would like each processor to use only a constant amount of memory. However, we assume that every processor can efficiently communicate with every other processor, as long as the amount of transmitted data remains small. It would be easy to adapt the approach to a network of small processors, with each processor connected to a central computer possessing $\tilde{O}(N^{1/3})$ bits of memory.

As for ordinary collisions, the key idea is to use *distinguished points*. By definition, a set of distinguished points is a set of points together with an efficient procedure for deciding membership. For example, the set of elements in $[0, M - 1]$ can be used as a set of distinguished points since membership can be tested using a single comparison. Moreover, with this choice, the fraction of distinguished points among the whole set is simply M/N . Here, since we wish to have chains of average length $N^{1/3}$, we choose for M an integer near $N^{2/3}$.

The distinguished point algorithm works in two steps. During the first step, each processor starts from a random start-point s and iteratively applies F until a distinguished point d is encountered. It then transmits a triple (s, d, L) , where L is the length of the path from s to d , to the processor whose number is $d \pmod{N_p}$. We abort any processor if it doesn't find a distinguished point within a reasonable amount of time, for example, following what [27] does for 2-collisions, we may abort after $20N/M$ steps. Once all the paths have been computed, we start the second step. Each processor looks at the triples it now holds. If a given value of d appears three or more times, the processor recomputes the corresponding chains, using the known length information to synchronize the chains. If three of the chains merge at a common position, a 3-collision is obtained.

Of course, even with less than $N^{1/3}$ processors, it is possible to do a partial parallelization. More precisely, given N^θ processors with $\theta \leq 1/3$, it is possible to find 3-collisions in time $\tilde{O}(N^{2/3-\theta})$. In that case, each processor needs a local memory of size $O(N^{1/3-\theta})$ to store all the triples it owns.

Algorithm 4. Parallelizable 3-collisions using distinguished points

Require: Oracle access to F operating on $[0, N - 1]$ **Require:** Number of processors $N_p \leq N^{1/3}$ **Require:** Identity of current processor: $\text{Id} \in [0, N_p - 1]$ Let $M \leftarrow \lceil N^{2/3} \rceil$ $\{M$ defines distinguished points $\}$ Let $L_{\max} = 20 \lceil N^{1/3} \rceil$ **Construction of triples:**Let $s \leftarrow_R [0, N - 1]$; $a \leftarrow s$; $L \leftarrow 0$ **while** $L < L_{\max}$ **do** Let $a \leftarrow F(a)$; $L \leftarrow L + 1$ **if** $a < M$ **then** Send triple $T \leftarrow (s, a, L)$ to processor $a \pmod{N_p}$ and **Exit Loop** **end if****end while****Acquisition of triples:**Store received triples (s, d, L) in local arrays A, D, \mathcal{L} numbered from 1 to K **Sort** D , applying the same permutation on elements of A and \mathcal{L} **Processing of triples:**Let $i \leftarrow 1$ **while** $i \leq K$ **do** Let $j \leftarrow i + 1$ **while** $j \leq K$ and $D[j] = D[i]$ **do** Let $j \leftarrow j + 1$ **end while** **if** $j \geq i + 3$ **then** Let $L \leftarrow \max(\mathcal{L}[i], \dots, \mathcal{L}[j - 1])$ **for** ℓ from L downto 0 **do** **for** k from i to $j - 1$ **do** **if** $\mathcal{L}[k] \geq \ell$ **then** Let $D[k] \leftarrow A[k]$; $A[k] \leftarrow F(A[k])$ $\{D[k]$ overwritten to keep previous value of $A[k]\}$ **end if** **end for** Check for 3 equal values in $A[i \dots j - 1]$ with differing values of D If found, Output the 3-collision and **Exit** **end for** **end if** Let $i \leftarrow j$ **end while**

7 Extension to r -Collisions, for $r > 3$

For r -collisions, recall that we need to evaluate F on approximately $r^{1/r} N^{(r-1)/r}$ points before hoping for a collision. When considering that r is a fixed value,

$r^{1/r}$ is a constant and vanishes within the \tilde{O} notation. With this new context, Algorithm 4 is quite easy to generalize. Here, the important parameter is to create shorter chains and compute more of them. The reason for shorter chains is that (as in Hellman’s Algorithm 5), we need to make sure that there are not too many collisions between one chain and all the others. Otherwise, the algorithm spends too much time recomputing the same evaluations of the random map, which is clearly a bad idea. To avoid this, we construct chains which are short enough to make sure that the average number of (initial 3) collisions between an individual chain and all the other chains is a constant. Since the total number of elements in all the other chains is essentially $N^{(r-1)/r}$, the length of chains should remain below $N^{1/r}$.

To achieve maximal parallelization when searching for an r -collision, $N_p \approx N^{(r-2)/r}$ processors are required. The integer M that defines distinguished points should be near $N^{(r-1)/r}$. Each processor first builds a chain of average length $N^{1/r}$ (as before we abort after $20 N/M$ steps), described by a triple (s, d, L) . Each chain is sent to the processor whose number is $d \pmod{N_p}$. During the second step, any processor that holds a value of d that appears in r or more triples recomputes the corresponding chains. If r chains merge at the same position, a r -collision is obtained.

Given N^θ processors with $\theta \leq (r - 2)/r$, it is possible to find r -collisions in time $\tilde{O}(N^{(r-1)/r-\theta})$. In that case, each processor needs a local memory of size $O(N^{(r-2)/r-\theta})$.

With a single processor, the required amount of memory is $O(N^{(r-2)/r})$. Thus, as r grows, the advantage of the single processor approach on the folklore algorithm (which requires $O(N^{(r-1)/r})$ memory) becomes smaller and smaller. As a consequence, for larger values of r , it is essential to rely on parallelization.

8 Conclusion

In this paper, we revisited the problem of constructing multicollisions on random mappings and showed that it can be done using less memory than required by the folklore algorithm. For 3-collisions, the sequential running remains at $\tilde{O}(N^{2/3})$ but the amount of memory can be reduced from $O(N^{2/3})$ to $O(N^{1/3})$. A remaining open problem is to determine whether this amount of memory can further be reduced.

Furthermore, finding 3-collisions can be very efficiently parallelized. Given $N^{1/3}$ parallel processors, each equipped with constant memory, the problem can be solved in time $\tilde{O}(N^{1/3})$. More generally for $r \geq 3$, we show how to generate r -collisions on N^θ processors, each with local memory $O(N^{(r-2)/r-\theta})$, in time $\tilde{O}(N^{(r-1)/r-\theta})$. It is interesting to note that the cost of the parallelizable approach in the full-cost model [28] decreases as θ grows.

³ Of course, once a collision occurs, all the values that follow are colliding. However, we do not count these follow-up collisions.

References

1. Arbitman, Y., Naor, M., Segev, G.: De-amortized cuckoo hashing: Provable worst-case performance and experimental results. In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Niko-letsea, S. (eds.) ICALP 2009. LNCS, vol. 5556, pp. 411–422. Springer, Heidelberg (2009)
2. Coppersmith, D.: Another birthday attack. In: Williams, H.C. (ed.) CRYPTO 1985. LNCS, vol. 218, pp. 14–17. Springer, Heidelberg (1986)
3. Ferguson, N., Lucks, S.: Attacks on AURORA-512 and the double-mix Merkle-Damgård transform. Cryptology ePrint Archive, Report 2009/113 (2009)
4. Flajolet, P., Odlyzko, A.M.: Random mapping statistics. In: Quisquater, J.-J., Vandewalle, J. (eds.) EUROCRYPT 1989. LNCS, vol. 434, pp. 329–354. Springer, Heidelberg (1990)
5. Hellman, M.E.: A cryptanalytic time-memory trade-off. IEEE Transactions on Information Theory 26(4), 401–406 (1980)
6. Hoch, J.J., Shamir, A.: Breaking the ICE - finding multicollisions in iterated concatenated and expanded (ICE) hash functions. In: Robshaw, M.J.B. (ed.) FSE 2006. LNCS, vol. 4047, pp. 179–194. Springer, Heidelberg (2006)
7. Hoch, J.J., Shamir, A.: On the strength of the concatenated hash combiner when all the hash functions are weak. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfssdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 616–630. Springer, Heidelberg (2008)
8. Iwata, T., Shibutani, K., Shirai, T., Moriai, S., Akishita, T.: AURORA: a cryptographic hash algorithm family. Submission to NIST’s SHA-3 competition (2008)
9. Joux, A.: Multicollisions in iterated hash functions. application to cascaded constructions. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 306–316. Springer, Heidelberg (2004)
10. Mendel, F.: Preimage attack on Blender,
<http://ehash.iaik.tugraz.at/wiki/Blender>
11. Mendel, F., Rechberger, C., Schläffer, M.: Cryptanalysis of twister. In: Proceedings of ACNS. Springer, Heidelberg (to appear),
<http://ehash.iaik.tugraz.at/wiki/Twister>
12. Mendel, F., Thomsen, S.S.: An observation on JH-512,
<http://ehash.iaik.tugraz.at/wiki/JH>
13. Nandi, M., Stinson, D.R.: Multicollision attacks on some generalized sequential hash functions. IEEE Transactions on Information Theory 53(2), 759–767 (2007)
14. Newbold, C.: Observations and attacks on the SHA-3 candidate Blender,
<http://ehash.iaik.tugraz.at/wiki/Blender>
15. Nivasch, G.: Cycle detection using a stack. Information Processing Letter 90(3), 135–140 (2004)
16. Pagh, R., Rodler, F.F.: Cuckoo hashing. J. Algorithms 51(2), 122–144 (2004)
17. Preneel, B.: Analysis and Design of Cryptographic Hash Functions. PhD thesis, KU Leuven (1993)
18. Quisquater, J.-J., Delescaille, J.-P.: Other cycling tests for DES. In: Pomerance, C. (ed.) CRYPTO 1987. LNCS, vol. 293, pp. 255–256. Springer, Heidelberg (1988)
19. Quisquater, J.-J., Delescaille, J.-P.: How easy is collision search? Application to DES. In: Quisquater, J.-J., Vandewalle, J. (eds.) EUROCRYPT 1989. LNCS, vol. 434, pp. 429–434. Springer, Heidelberg (1990)
20. Quisquater, J.-J., Delescaille, J.-P.: How easy is collision search. New results and applications to DES. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 408–413. Springer, Heidelberg (1990)

21. Sasaki, Y.: A collision attack on AURORA-512. Cryptology ePrint Archive, Report 2009/106 (2009)
22. Suzuki, K., Tonien, D., Kurosawa, K., Toyota, K.: Birthday paradox for multi-collisions. In: Rhee, M.S., Lee, B. (eds.) ICISC 2006. LNCS, vol. 4296, pp. 29–40. Springer, Heidelberg (2006)
23. van Oorschot, P.C., Wiener, M.: A known-plaintext attack on two-key triple encryption. In: Damgård, I.B. (ed.) EUROCRYPT 1990. LNCS, vol. 473, pp. 318–325. Springer, Heidelberg (1991)
24. van Oorschot, P.C., Wiener, M.J.: Parallel collision search with application to hash functions and discrete logarithms. In: ACM CCS 1994, Fairfax, Virginia, USA, pp. 210–218. ACM Press, New York (1994)
25. van Oorschot, P.C., Wiener, M.: Improving implementable meet-in-the-middle attacks by orders of magnitude. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 229–236. Springer, Heidelberg (1996)
26. van Oorschot, P.C., Wiener, M.: On diffie-hellman key agreement with short exponents. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 332–343. Springer, Heidelberg (1996)
27. van Oorschot, P.C., Wiene, M.J.: Parallel collision search with cryptanalytic applications. *Journal of Cryptology* 12(1), 1–28 (1999)
28. Wiener, M.J.: The full cost of cryptanalytic attacks. *Journal of Cryptology* 17(2), 105–124 (2004)
29. Wu, H.: The complexity of Mendel and Thomsen’s preimage attack on JH-512, <http://ehash.iaik.tugraz.at/wiki/JH>

A Practical Implementation

Since we only performed a heuristic analysis of our algorithms, in order to show that they are really effective, we decided to illustrate our 3-collision techniques with a practical example. For this purpose, we construct a random function by Xoring two copies of the DES algorithm (with two different keys). More precisely, we let:

$$F(x) = \text{DES}_{K_1}(x) \oplus \text{DES}_{K_2}(x),$$

where⁴ $K_1 = (3322110077665544)_{16}$ and $K_2 = (3b2a19087f6e5d4c)_{16}$. Since x is on 64 bits, the time and memory requirements of the folklore algorithm are around 2^{43} . Where current computers are concerned, performing 2^{43} operations is easily feasible. However, storing 2^{43} values of x requires 2^{46} bytes, i.e. 64 Terabytes. As a consequence, finding 3-collisions on F with the basic parameters of the folklore algorithm is probably beyond feasibility. Using a different time-memory trade-off, restricting the storage to 2^{32} values would raise the time requirement to 2^{48} operations. This is within the range of currently accessible computations. However, since the algorithm is not parallelizable, it would require a high-end computer.

⁴ This keys might seem weird, but they should not have any special properties. In truth, we intended to choose $K_1 = (0011223344556677)_{16}$ and $K_2 = (08192a3b4c5d6e7f)_{16}$, i.e., $(8899aabbccddeeff)_{16}$ with high bits stripped. Unfortunately, the first-named author made a classical endianness mistake while implementing the algorithm.

With the new algorithms presented in this paper, it becomes possible to compute triple collisions much more efficiently on the function F . For our implementation, we chose $M = 2^{44}$ to define the distinguished points, which yielded chains of expected length 2^{20} . The abort length was set at 8 times the expected length, rather than the factor 20 given in Algorithm 4. For computing the chains, we used a mix of 32 Intel Xeon processors at 2.8 GHz and 8 Nvidia CUDA cards (Tesla type). We collected a total of 35 447 322 chains and obtained 3 078 699 groups of three or more chains yielding the same distinguished endpoints. The largest group contained 36 chains, which shows that it would have been preferable to use slightly shorter chains. On processors only, this first phase would have taken about 94 CPU-days to run. On a single CUDA card, it would have taken 11.5 days.

For simplicity of implementation, the second phase of the algorithm was only performed on Intel processors and not on CUDA cards. It took less than 18 CPU-days to test all groups and it yielded the following triple-collisions:

$$\begin{aligned} F(d332b9ba5e5a7d4e) &= F(51b8095db532afcc) = F(b084dc15dce042ab), \\ F(ca76ff906d6587cf) &= F(e1f7f59a5757d01b) = F(0285f58147e863c2), \\ F(c3783ef30c8bcc3d) &= F(65f14d412fd91173) = F(1042d827e5078000). \end{aligned}$$

We would like to thank CEA/DAM⁵ for kindly providing the necessary computing time on its Tesla servers.

B Applications

B.1 Collisions for the Hash Function AURORA-512

AURORA is a family of cryptographic hash functions submitted to the NIST SHA-3 hash function competition [8]. Like the other members of the AURORA family, AURORA-512 employs different internal compression functions, each mapping a 256-bit chaining value and a 512-bit message block to generate a new 256-bit chaining value. AURORA-512 is the high-end member of that family, maintaining an internal state of 512 bit. As required by the NIST, the authors of AURORA-512 explicitly claim “collision resistance of approximately 512/2 bits” for AURORA-512. In other words, collision attacks must not significantly improve over the generic birthday attack, which takes roughly the time of 2^{256} hash operations.

Internally, AURORA-512 works almost like the cascade of two iterated hash functions, except for one important extra operation:

$$\text{MF} : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}^n \times \{0, 1\}^n.$$

See Algorithm 5 for a simplified description of AURORA-512.

Every eighth iteration, MF is called to mix the two half-states. This seems to defend against the cascade-attack from [9]: Between two MF-operations, one

⁵ Commissariat à l’énergie atomique, Direction des applications militaires.

Algorithm 5. AURORA-512: Hashing 8 message blocks.

Require: Input Chaining Values (Left, Right) $\in (\{0, 1\}^{256})^2$
for i from 0 to 7 **do**
 Left \leftarrow Compress(Left, Message_Block(i))
 Right \leftarrow Compress(Right, Message_Block(i))
end for
(Left, Right) \leftarrow MF(Left, Right)

can generate local collisions in each iteration in one of either the left string, or the right string. Thus, the adversary can get a local 2^8 -collision. But to apply the attack from [9], one would rather need a 2^{128} -collision, so the attack fails.

Assume, for a moment, that the adversary has generated a 2^7 -collision on Left in the first 7 iterations of the loop. For the right string, we have 2^7 different values $\text{Right}_1, \text{Right}_2, \dots, \text{Right}_{128}$. If two of them collide, a collision for AURORA-512 has been found. For a fixed Message_Block(7), the chance of a collision, i.e. of $j \neq k$ with

$$\begin{aligned} &\text{Compress}(\text{Right}_j, \text{Message_Block}(7)) \\ &= \\ &\text{Compress}(\text{Right}_k, \text{Message_Block}(7)) \end{aligned}$$

is about $2^7 \cdot (2^7 - 1) \cdot 2^{-1}/2^{256}$. By trying out $2^{256-(6+7)}$ different values for Message_Block(7), we expect to find a collision. Note that this means to make 2^7 calls to the function Compress. Hence, this attack takes the time of about $2^{256-(6+7)+7} = 2^{250}$ compression function calls, plus the time to generate the 2^7 -collision at the beginning. This is essentially the memoryless variant of the attack from [3], except that the authors of [3] actually generate a 2^8 -collision on Left, by exploiting the previous eight-tuple of message blocks. The attack is memoryless, since the adversary only needs to generate 2-collisions on Left, and the claimed time is 2^{249} .

In [3], Ferguson and Lucks further propose an attack which uses local r -collision, instead of local 2-collisions. A similar attack has been proposed independently [21]. Using eight local r -collisions allows to speed-up the attack to roughly $2^{256}/r^7$ compression function calls (plus the time to generate the required r -collisions). [3] suggest $r = 9$ (beyond that, computing the r -collisions becomes too costly) and claim time $2^{234.5}$, including the time to generate ten local 9-collisions. The price for the speed-up is utilizing a huge amount of memory, however.

Our memory-efficient 3-collision allows a different time-memory tradeoff. The time is $2^{256}/3^7 \approx 2^{245}$. Recall $N = 2^{256}$, and set $\alpha := 1/16$, $\beta := 15/16$ in Algorithm 2. In that case one local 3-collision requires time 2^{240} , which we neglect. The memory requirements are down to 2^{16} , i.e., almost negligible.

It is also possible to use more general r -collisions to further improve this attack. For example, we can use 4-collisions obtained using the algorithm of section 7. To simplify the comparison with previous attacks, we assume a single processor, i.e. set $\theta = 0$, however, with more processors, we would obtain an

even better attack. With this choice, a 4-collision on 256-bits is obtained in time 2^{192} using a memory of size 2^{128} . The corresponding speedup is 4^7 . Similarly, 8-collisions on 256 bits are obtained in time 2^{224} each, using 2^{192} units of memory. The speed-up is 8^7 . Other trade-offs are possible.

The results on collision attacks for AURORA-512 can be summarised as follows:

r (arity)	time [compr. fn. calls]	memory	reference
9	$2^{234.5}$	$2^{229.6}$	[3]
8	2^{236}	2^{236}	[21]
2	2^{249}	—	[3]
3	2^{245}	2^{16}	(this paper)
4	2^{242}	2^{128}	(this paper)
8	2^{235}	2^{192}	(this paper)

B.2 Attacks on Other Hash Functions

Several attacks on several other SHA-3 candidates make heavy use of multicollisions, and it appears a natural idea to plug in our algorithms for reducing the memory consumption of these attacks. We actually tried to do so, but only succeeded for Aurora-512. In the current section, we will explain why we failed for other obvious candidates.

Several attacks, such as the attacks on Blender [14,10] and on Twister [11], employ multicollisions, but it turns out that these can actually be generated by Joux-style iterated 2-collisions, which is very memory-efficient – and also faster than our general multicollision algorithms, anyway.

An obvious candidate to employ our algorithms to improve given cryptanalytic attacks is a preimage attack on JH-512 [12]. Like Aurora, JH is a family of hash functions submitted to the SHA-3 competition. The high-end 512-bit variant is denoted as JH-512. Internally, JH-512 is a wide-pipe hash function with an internal state of 1024 bit, and it employs an invertible compression function. [12] propose a meet-in-the-middle attack which requires “ $2^{510.3}$ compression function evaluations *and a similar amount of memory*” (our emphasis). The authors of [12] stress: “We do not claim that our attack breaks JH-512 (due to the high memory requirements).” The author of JH-512 provides a more detailed analysis of this attack, claiming “ $2^{510.6}$ [units of] memory”. A main phase of the attack is generating several 51-collisions on one half of the chaining values (i.e., on 512 bits). By applying our algorithms to this task, it is possible to reduce the memory required for this phase to $2^{(512/51) \cdot 49}$ units of memory.

But another phase of the attack from [12] is to apply the inverse of the compression function to generate 2^{509} internal target values. The attack successfully generates a message which hashes to a given preimage, if the first part of the message hashes to any of these 2^{509} target values. Finally, the overall amount of storage for the attack is dominated by storing these 2^{509} values, regardless of improving memory-efficiency of the multicollision phase.

A Modular Design for Hash Functions: Towards Making the Mix-Compress-Mix Approach Practical

Anja Lehmann¹ and Stefano Tessaro²

¹ Darmstadt University of Technology, Germany
alehmann@cdc.informatik.tu-darmstadt.de

² Department of Computer Science, ETH Zurich, 8092 Zurich, Switzerland
tessaros@inf.ethz.ch

Abstract. The design of cryptographic hash functions is a very complex and failure-prone process. For this reason, this paper puts forward a completely *modular* and *fault-tolerant* approach to the construction of a full-fledged hash function from an underlying simpler hash function H and a further primitive F (such as a block cipher), with the property that collision resistance of the construction only relies on H , whereas indifferenziability from a random oracle follows from F being ideal. In particular, the failure of one of the two components must not affect the security property implied by the other component.

The *Mix-Compress-Mix* (MCM) approach by Ristenpart and Shrimpton (ASIACRYPT 2007) envelops the hash function H between two *injective* mixing steps, and can be interpreted as a first attempt at such a design. However, the proposed instantiation of the mixing steps, based on block ciphers, makes the resulting hash function impractical: First, it cannot be evaluated online, and second, it produces larger hash values than H , while only inheriting the collision-resistance guarantees for the shorter output. Additionally, it relies on a *trapdoor* one-way permutation, which seriously compromises the use of the resulting hash function for random oracle instantiation in certain scenarios.

This paper presents the first *efficient* modular hash function with online evaluation and short output length. The core of our approach are novel block-cipher based designs for the mixing steps of the MCM approach which rely on significantly weaker assumptions: The first mixing step is realized *without* any computational assumptions (besides the underlying cipher being ideal), whereas the second mixing step only requires a one-way permutation *without* a trapdoor, which we prove to be the minimal assumption for the construction of injective random oracles.

1 Introduction

MULTI-PROPERTY HASH FUNCTIONS. Cryptographic hash functions play a central role in efficient schemes for several cryptographic tasks, such as message authentication, public-key encryption, digital signatures, key derivation, and many others. Yet the huge variety of contexts in which hash functions are deployed makes

the security requirements on them very diverse: While some schemes only assume relatively simple properties such as *one-wayness* or different forms of *collision resistance*, other schemes, including practical ones such as OAEP [4,15] and PSS [5], are only proven secure under the assumption that the underlying hash function is a *random oracle* [3], i.e., a truly random function which can be evaluated by the adversary. On the one hand, while a number of *provably-secure* collision-resistant hash functions, such as VSH [9] or SWIFFT [18], have been designed, they are not appropriate candidates for random oracle instantiation. On the other hand, well-known theoretical limitations [8,19] only permit constructions of hash functions for random oracle instantiation from *idealized* primitives [10], such as a *fixed-input-length* random oracle or an *ideal cipher*,¹ but (as first pointed out in [2]) these constructions may lose any security guarantees as soon as the adversary gets to exploit non-ideal properties of the underlying primitive.²

While one could in principle always employ a suitable hash function tailored at the individual security property needed by one particular cryptographic scheme at hand, common practices such as code re-use and the development of standards call for the design of a *single* hash function satisfying as many properties as possible. This point of view has also been adopted by NIST's on-going SHA-3 competition [17], and motivated a series of works [20] shifting the design problem of multi-property hash functions to the task of constructing good multi-property *compression* functions. A further line of research has been devoted to robust multi-property *combiners* [13], which merge two hash functions such that the resulting function satisfies each of the properties possessed by at least one of the two starting functions. While these works simplify the design task, building multi-property hash functions from *single*-property primitives remains far from being simple, and is the main topic of this paper.

STATEMENT OF THE MAIN PROBLEM. This paper presents a *modular* design for hash functions that are *collision resistant* in the *standard* model and can, simultaneously, be used for random oracle instantiation in the *ideal* model. We consider a setting where both a hash function H as well as some other (potentially ideal) primitive F (such as a block cipher) are given (a similar setup was previously considered by Ristenpart and Shrimpton [23]): We aim at devising a construction $C^{H,F}$ which is collision resistant as long as H is collision resistant,³ and which behaves as a random oracle (with respect to the notion of *indifferentiability* [19,10]) whenever F is ideal. For this approach to be practically appealing, the construction must preserve the good properties of H : For instance, it must allow for online processing of data (which is crucial for large

¹ An ideal cipher $\mathbf{E} : \{0, 1\}^k \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ associates an (invertible) random permutation $\mathbf{E}(k, \cdot)$ with each key k .

² Of course, a real block cipher *cannot* be ideal. (Likewise, a hash function cannot be a random oracle either.) Yet modeling it as ideal captures the adversary's inability of exploiting any structure, and a security proof in this model implies in particular the inexistence of any generic attacks treating the block cipher as a black box.

³ In particular, we require the existence of a standard-model reduction.

inputs or in streaming applications) whenever H can be evaluated online.⁴ Also, the construction should not increase the size of the hashes of H .

In particular, we advocate a safe and modular design paradigm where each of both properties should ideally rely only on *one* of both component primitives, whereas the other primitive may be arbitrarily insecure, except for (possibly) satisfying some minimal structural requirement (that can be ensured by design), such as F being a permutation or H being sufficiently regular. This differs from the point of view taken in [23], where H is *guaranteed* to be collision resistant and is extended by means of an ideal primitive F into a random oracle, while preserving the collision-resistance guarantees of H : We believe that practical considerations, especially efficiency, may in fact motivate the use of hash functions with no provable security guarantees. Thus, it is desirable that even the ability of finding collisions for H does not impact the indistinguishability of the construction, as long as F is still ideal. Either way, both points of view are related: Any solution satisfying our stronger requirements (including the one we propose in this paper) also fits within the framework of [23], while the solution proposed in [23] also satisfies stronger requirements, as discussed below.

We also remark that using the multi-property combiner of [14] one can combine a random oracle (built from F) and H into a hash function that provably observes both properties. However, as combiners inherently do not exploit the knowledge of which one of both functions has a certain property, the resulting construction is rather inefficient, e.g., it doubles the output length.

THE MCM APPROACH. Given a hash function H as above, the so-called *mix-compress-mix* (MCM) approach, introduced by Ristenpart and Shrimpton [23], considers the construction

$$\text{MCM}^{M_1, M_2, H}(x) := M_2(H(M_1(x))),$$

where M_1 and M_2 are arbitrary-input-length injective maps (the so-called *mixing stages*) with *stretch* τ_1 and τ_2 , respectively, i.e., such that M_i outputs a string of length $|x| + \tau_i$ on input $x \in \{0, 1\}^*$. The injectivity of the mixing stages ensures that MCM preserves the collision resistance of H in the standard model. Additionally, it was shown in [23] that MCM is indistinguishable from a random oracle if M_1 and M_2 are random *injective* oracles (i.e., M_i returns a random $(|x| + \tau_i)$ -bit string for each input $x \in \{0, 1\}^*$ that differs from all previously returned values with the same length) *and* H is collision resistant and sufficiently regular. Dodis et al. [12] subsequently interpreted this result as the combination of two facts: (i) The mapping $x \mapsto H(M_1(x))$ is *preimage aware*⁵ under the same

⁴ Most hash functions rely on some iterated (and thus inherently online) design, such as Merkle-Damgård [11][21], or sponges [6].

⁵ Informally, a construction \mathbf{C}^F based on an ideal primitive \mathbf{F} is *preimage aware* if there exists an algorithm – called the *preimage extractor* – which given the input-output history of \mathbf{F} and an output y , either aborts or returns x such that $\mathbf{C}^F(x) = y$, and after such query no adversary can find an input x' such that $\mathbf{C}^E(x') = y$ (and $x' \neq x$ in case the extraction query did not abort).

assumptions, and (ii) Post-processing the output of a preimage-aware function with a (possibly injective) random oracle yields a full-fledged random oracle. A concrete instantiation of injective random oracles – called the TE-construction – relying on an ideal cipher and a *trapdoor* one-way permutation has also been proposed in [23]. To date, this was the only known such construction.

Interestingly, we observe that the MCM approach provides a modular design approach for hash functions as advocated above, since the indistinguishability result can be made *independent* of the collision resistance of H . (This was unnoticed in [23], and is briefly discussed in the full version of this paper.) However, its deployment is subject to a number of practical and theoretical drawbacks, whose solution was stated as an open problem in [23]: First, *every* construction of injective random oracles (and in particular the TE-construction) cannot be *online*, as, roughly speaking, each output bit needs to be influenced by *all* of the input in order to exhibit random behavior. Additionally, the fact that the TE-construction is length-increasing has a serious impact on the resulting hash size: In particular, the stretch τ_i typically equals the bit length of a sufficiently secure RSA modulus, i.e., $\tau_i \geq 2048$ bits for reasonable security. Finally, the use of a *trapdoor* one-way permutation within the TE-construction is rather undesirable: In contrast to (non trapdoor) one-way permutations, the assumption is very strong, e.g., it implies public-key encryption in the random oracle model [4]. Also, as pointed out in [23], the compositional guarantees of protocols using the MCM approach (with the TE-construction) to instantiate a random oracle are affected, as properties such as deniability may be lost (cf. e.g. the works by Pass [22] and by Canetti et al. [7]).

These observations give rise to a number of challenging open questions. Can we instantiate the *first* mixing stage of MCM with a weaker primitive which allows for online processing? Can we instantiate the *second* mixing stage (where online processing is not an issue) as an injective RO with limited stretch (possibly even with no stretch at all)? And finally, can we weaken the underlying assumption, eliminating the need of the trapdoor, or possibly even entirely removing the underlying assumption?

CONTRIBUTIONS AND ROADMAP OF THIS PAPER. In this paper, we present the first efficient modular construction of a hash function in the sense described above. Our solution relies on the MCM approach, and in particular we address and solve all of the aforementioned open questions, and hence make a substantial step towards making the MCM approach practical.

First Mixing Stage. In Section 3, we present a novel mode of operation for a block cipher $E : \{0, 1\}^{2n} \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ implementing an arbitrary-input-length injective map – called *iterated mix* (IM) – that permits *online* processing of its inputs, making only one call to E per n -bit message block, and has only stretch $n/2$. Our first main theorem shows that the construction $\text{IMC}^{E,H}(M) := H(\text{IM}^E(M))$ applying H to the output of IM is preimage aware if E is an ideal cipher and, additionally, the hash function H satisfies a rather weak regularity requirement (which is somewhat incomparable to the one used in [23], albeit equally natural): Namely, given a random n -bit string m and some arbitrary

string S , the value $H(S\|m)$ has (min-)entropy not much lower than n (if n is smaller than H 's hash size), or not much lower than the hash size otherwise. In fact, even completely insecure hash functions can have this property, and it is also natural to assume that it is satisfied by any reasonably built hash function. We also present a variant of the IM-construction which requires a block cipher with single-block key length n at the price of making two block-cipher calls per message block.

We stress that (contrary to the TE-construction) our result does not rely on *any* computational assumptions: In particular, the IM-construction relies on invertible primitives, and is itself *efficiently* invertible. Thus, IM does not implement a random injective oracle.

Second Mixing Stage. With the goal of making the MCM approach preserve the hash size of the underlying hash function in mind, the second part of this paper (Section 4) addresses the question of building *length-preserving* injective random oracles. (We call this a (non-invertible) *random permutation oracle* (RPO).) We show that for any three permutations E, E', π from n bits to n bits, the permutation

$$\text{NIRP}^{E, E', \pi}(x) := E'(\pi(E(x)))$$

is indistinguishable from a RPO if both E and E' are (fixed-key) ideal ciphers, and π is a one-way permutation, without a trapdoor.

In practice, E, E' are instantiated by a block cipher with two distinct fixed keys. This limits us to n being a valid block size (e.g. $n = 128$ bits), which can be smaller than the usual hash size (e.g. $h = 256$). This motivates the question of extending the input/output size of random permutation oracles: In Section 4.2, we present constructions (which are reminiscent of the Shrimpton-Stam compression function [25]) for extending every n to n bits RPO into a $\gamma \cdot n$ bits to $\gamma \cdot n$ bits RPO for any fixed $\gamma > 1$.

In the full version we further show that in order to construct injective ROs the assumption of a one-way permutation cannot be weakened to a one-way function (at least under black-box security reductions).

Putting Pieces Together. Finally, instantiating MCM with IM and NIRP (or its extension through our extender) as its first and second mixing stage, respectively, leads to the first construction of a hash function with the following properties:

- (i) Its collision resistance can be reduced in the standard model to the collision resistance of the underlying hash function.
- (ii) It is indistinguishable from a random oracle in the ideal cipher model (with a one-way permutation), as long as the underlying hash function is sufficiently regular.
- (iii) It can be evaluated online as long as the underlying hash function can be evaluated in an online fashion.
- (iv) It has hash size equal to the one of the underlying hash function.
- (v) It can be used to instantiate a random oracle in *all* computationally secure schemes in the random oracle model, with no composability limitations.

2 Preliminaries

NOTATIONAL PRELIMINARIES. Throughout this paper, $\{0, 1\}^n$ denotes the set of strings s of length $|s| = n$, whereas $(\{0, 1\}^n)^*$ and $(\{0, 1\}^n)^+$ are the sets of strings consisting of n -bit blocks with and without the empty string, respectively. The notation $s||s'$ stands for the concatenation of the strings s and s' . Also, we use $\text{INJ}(m, n)$ to denote the set of *injective* functions $f : \{0, 1\}^m \rightarrow \{0, 1\}^n$ (in particular, $\text{INJ}(n, n)$ is the set of permutations from n bits to n bits). Further, it is convenient to define $\text{BC}(\kappa, n)$ as the set of *block ciphers*, i.e., of *keyed* functions $E : \{0, 1\}^\kappa \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ such that each key $k \in \{0, 1\}^\kappa$ defines a permutation $E_k(\cdot) := E(k, \cdot) \in \text{INJ}(n, n)$ (and denote as $E^{-1}(k, \cdot)$ the corresponding *inverse*).

Algorithms are in general randomized, and throughout this paper we fix a RAM model of computation for these algorithms. We use the notation $\mathcal{A}^{\mathcal{O}}(r)$ to denote the (oracle) algorithm $\mathcal{A}^{(\cdot)}$ which runs on input r with access to the oracle \mathcal{O} . In particular, an algorithm $\mathcal{A}^{(\cdot)}$ is said to have *running time* t (also denoted as $\text{time}(\mathcal{A}) = t$) if the sum of its description length and the worst-case number of steps it takes (counting oracle queries as single steps), taken over all randomness values, all inputs and all compatible oracles, is at most t . If the algorithm takes inputs of arbitrary length, then $\text{time}(\mathcal{A}, \ell)$ refines the above notion to only take the maximum over inputs of length at most ℓ .

Finally, the shorthand $x \stackrel{\$}{\leftarrow} S$ stands for the action of drawing a fresh random element x uniformly from the set S , whereas $x \stackrel{\$}{\leftarrow} \mathcal{A}^{\mathcal{O}}(r)$ denotes the process of sampling x by letting \mathcal{A} interact with \mathcal{O} on input r (and probabilities are taken over the random coins of \mathcal{A} and \mathcal{O}).

ONE-WAY FUNCTIONS AND PERMUTATIONS. We define the *one-way advantage* of an adversary \mathcal{A} against a function $f : \{0, 1\}^m \rightarrow \{0, 1\}^n$ as

$$\mathbf{Adv}_f^{\text{owf}}(\mathcal{A}) = \mathbb{P}[x \stackrel{\$}{\leftarrow} \{0, 1\}^m, x' \stackrel{\$}{\leftarrow} \mathcal{A}(f(x)) : f(x) = f(x')].$$

For the special case of a permutation $\pi : \{0, 1\}^n \rightarrow \{0, 1\}^n$, it is convenient to use the shorthand $\mathbf{Adv}_\pi^{\text{owp}}(\mathcal{A}) = \mathbb{P}[x \stackrel{\$}{\leftarrow} \{0, 1\}^n, x' \stackrel{\$}{\leftarrow} \mathcal{A}(\pi(x)) : x = x']$ for the *one-way permutation advantage*.

IDEALIZED PRIMITIVES. We consider a number of (more or less) standard *idealized primitives* throughout this paper, which are always denoted by bold-face letters. For a set X , a *random oracle (RO)* $\mathbf{R} : X \rightarrow \{0, 1\}^n$ is a system associating a random n -bit string $\mathbf{R}(x)$ with each input x . If $X = \{0, 1\}^m$, then \mathbf{R} is called a *fixed-input-length RO (FIL-RO)*, whereas it is a *variable-input-length RO (VIL-RO)* if $X = \{0, 1\}^*$. An *ideal cipher (IC)* $\mathbf{E} : \{0, 1\}^\kappa \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ is a block cipher E chosen uniformly from the set $\text{BC}(\kappa, n)$, and allows both *forward queries* $\mathbf{E}(k, x)$ as well as *backward queries* $\mathbf{E}^{-1}(k, y)$. If $\kappa = 0$, then we omit the first input and we call this a *fixed-key ideal cipher*. Note that for an IC \mathbf{E} and distinct fixed key values k_0, k_1, \dots , $\mathbf{E}(k_0, \cdot), \mathbf{E}(k_1, \cdot), \dots$ are independent fixed-key ICs. In contrast, a (*fixed-input-length*) *random injective oracle (FIL-RIO)* $\mathbf{I} : \{0, 1\}^m \rightarrow \{0, 1\}^n$ implements a uniformly chosen function from

INJ(m, n). In the special case $m = n$ we call this a *random permutation oracle (RPO)* $\mathbf{P} : \{0, 1\}^n \rightarrow \{0, 1\}^n$.

We stress that the substantial difference between a fixed-key IC and a RPO is that the former allows for inversion queries, whereas the latter does not (and is in particular hard to invert).

INDIFFERENTIABILITY. The notion of indifferenciability was introduced by Maurer et al. [19] to generalize indistinguishability to constructions $\mathbf{C}^{\mathbf{F}} : X \rightarrow \{0, 1\}^n$ using a *public* (idealized) primitive \mathbf{F} (e.g., an IC, a FIL-RO, or a combination of these), i.e., that can be accessed by the adversary. Roughly speaking, $\mathbf{C}^{\mathbf{F}}$ is *indifferenciability* from an ideal primitive \mathbf{F}' if there exists a simulator $\mathcal{S}^{\mathbf{F}'}$ accessing \mathbf{F}' such that $(\mathbf{C}^{\mathbf{F}}, \mathbf{F})$ and $(\mathbf{F}', \mathcal{S}^{\mathbf{F}'})$ are indistinguishable. In particular, we will be concerned with the cases where \mathbf{F}' is either a RO or a RIO/RPO, and we define the *RO-indifferenciability advantage* of the distinguisher \mathcal{D} against the construction $\mathbf{C}^{\mathbf{F}}$ and simulator \mathcal{S} as the quantity

$$\mathbf{Adv}_{\mathbf{C}^{\mathbf{F}}, \mathcal{S}}^{\text{ind-ro}}(\mathcal{D}) = \left| \mathbb{P} \left[\mathcal{D}^{\mathbf{C}^{\mathbf{F}}, \mathbf{F}} = 1 \right] - \mathbb{P} \left[\mathcal{D}^{\mathbf{R}, \mathbf{S}^{\mathbf{R}}} = 1 \right] \right|,$$

where $\mathbf{R} : X \rightarrow \{0, 1\}^n$ is a RO with the same input and output sets as \mathbf{C} . The *IRO-indifferenciability advantage* $\mathbf{Adv}_{\mathbf{C}^{\mathbf{F}}, \mathcal{S}}^{\text{ind-iro}}$ is defined analogously by using a RIO \mathbf{I} instead of \mathbf{R} . We stress that both quantities are related by a simple birthday-like argument, i.e., $\mathbf{Adv}_{\mathbf{C}^{\mathbf{F}}, \mathcal{S}}^{\text{ind-iro}}(\mathcal{D}) \leq \mathbf{Adv}_{\mathbf{C}^{\mathbf{F}}, \mathcal{S}}^{\text{ind-ro}}(\mathcal{D}) + \frac{1}{2} \cdot (q + q_{\mathcal{S}})^2 \cdot 2^{-n}$, where q is the number of query \mathcal{D} makes to its first oracle, whereas $q_{\mathcal{S}}$ is the overall number of queries \mathcal{S} makes when answering \mathcal{D} 's queries. Note that indifferenciability ensures *composability*, i.e., if a cryptographic scheme is secure using an ideal primitive \mathbf{F}' accessible by the adversary, then it remains secure when replacing \mathbf{F}' with a construction $\mathbf{C}^{\mathbf{F}}$ which is indifferenciability from \mathbf{F}' and letting the adversary access \mathbf{F} . See [19, 10] for a formal treatment in the information-theoretic and computational models.

COLLISION-RESISTANCE. Let $H : K \times \{0, 1\}^* \rightarrow \{0, 1\}^h$ be a (keyed) hash function with key generator \mathcal{K} . The *collision-finding advantage* of an adversary \mathcal{A} is

$$\mathbf{Adv}_H^{\text{cr}}(\mathcal{A}) := \mathbb{P} \left[k \xleftarrow{\$} \mathcal{K}, (M, M') \xleftarrow{\$} \mathcal{A}(k) : M \neq M' \wedge H_k(M) = H_k(M') \right]$$

The notion naturally extends to keyless hash functions (which can be considered in the same spirit proposed in [24]) and to constructions from some ideal primitive \mathbf{F} (where \mathcal{A} is additionally given access to \mathbf{F}).

THE MCM-CONSTRUCTION. For a hash function $H : \{0, 1\}^* \rightarrow \{0, 1\}^h$, and injective maps $M_1 : \{0, 1\}^* \rightarrow \{0, 1\}^*$, $M_2 \in \text{INJ}(h', n)$, where $n \geq h' \geq h$, the *MCM-construction* implements a map $\{0, 1\}^* \rightarrow \{0, 1\}^n$ as

$$\text{MCM}^{M_1, H, M_2}(M) := M_2(H(M_1(M)) \parallel 0^{h'-h}).$$

We also define $\text{MCM}_k^{M_1, H, M_2} := \text{MCM}^{M_1, H_k, M_2}$ for all $k \in K$ if the hash function H is keyed (with key space K). Also, the definition does not allow M_1, M_2 to be

keyed (in contrast to [23]). This is because we will present *keyless* instantiations of M_1, M_2 . Note that we assume M_2 to be fixed input length without loss of generality. The following simple result was shown in [23], and holds both for keyed as well as for keyless hash functions.

Lemma 1. *For all collision-finding adversaries \mathcal{A} outputting a pair of messages each of length at most ℓ , there exists a collision-finding adversary \mathcal{B} such that $\text{Adv}_{\text{MCM}^{M_1, H, M_2}}^{\text{cr}}(\mathcal{A}) = \text{Adv}_H^{\text{cr}}(\mathcal{B})$, where $\text{time}(\mathcal{B}) = \text{time}(\mathcal{A}) + \mathcal{O}(2(\ell + \text{time}(M_1, \ell)))$.*

PREIMAGE AWARENESS. We briefly review the notion of preimage awareness [12] for a hash function $H^F : \{0, 1\}^* \rightarrow \{0, 1\}^h$ built from an idealized primitive F . A *preimage extractor* \mathcal{E} is a (deterministic) algorithm taking a history α of input-output pairs of F and a value $y \in \{0, 1\}^h$ such that $\mathcal{E}(\alpha, y)$ returns a value $x \in \{0, 1\}^* \cup \{\perp\}$. We consider a random experiment (called the *pra-game*) involving an adversary \mathcal{A} which can query *both* F and $\mathcal{E}(\alpha, \cdot)$ (where α is the current history containing the interaction with F so far, i.e., the adversary cannot change the first argument), and where a set Q contains all \mathcal{E} -queries y of \mathcal{A} and an associative array V stores as $V[y] \in \{0, 1\}^* \cup \{\perp\}$ (for all $y \in Q$) the answer of the query y to \mathcal{E} . The *pra-advantage of the adversary \mathcal{A} with preimage extractor \mathcal{E} , and primitive F* is the quantity

$$\text{Adv}_{H, F, \mathcal{E}}^{\text{pra}}(\mathcal{A}) := \mathbb{P}[(M, y) \stackrel{\$}{\leftarrow} \mathcal{A}^{\mathcal{E}(\alpha, \cdot), F} : y \in Q \wedge H^F(M) = y \wedge V[y] \neq M].$$

It turns out that preimage aware functions are good domain extenders for FIL-ROs: More concretely, with H as above, consider the construction $C^{F, R'} : M \mapsto R'(H^F(M))$ for a FIL-RO $R' : \{0, 1\}^h \rightarrow \{0, 1\}^n$. Then, the following result was proved in [12].

Lemma 2 (PRA + FIL-RO = VIL-RO [12]). *There exists a simulator \mathcal{S} such that for all distinguishers \mathcal{D} making q queries to $C^{F, R'}$ of length at most ℓ , q_1 queries to F and q_2 queries to R' , there exists an adversary \mathcal{A} with*

$$\text{Adv}_{C^{F, R'}, \mathcal{S}}^{\text{ind-ro}}(\mathcal{D}) \leq \text{Adv}_{H, F, \mathcal{E}}^{\text{pra}}(\mathcal{A}).$$

The simulator \mathcal{S} runs in time $\mathcal{O}(q_1 + q_2 \cdot \text{time}(\mathcal{E}))$ and makes q_2 queries, whereas \mathcal{A} runs in time $\text{time}(\mathcal{D}) + \mathcal{O}(q \cdot \text{time}(H, \ell) + q_0 + q_1)$ and makes $q \cdot q_{H, \ell} + q_1$ F -queries and q_2 extraction queries, where $q_{H, \ell}$ is the maximal number of oracle queries made by H to process an input of length at most ℓ .

3 An On-Line Mixing Stage: The IMC-Construction

3.1 Description

THE IM-CONSTRUCTION. The *iterated mix construction* (or IM-construction for short), depicted in Figure 1, relies on a block cipher $E : \{0, 1\}^{2n} \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ and an injective mapping $\text{PAD} : \{0, 1\}^* \rightarrow \{0, 1\}^{n/2} \times (\{0, 1\}^n)^*$ which

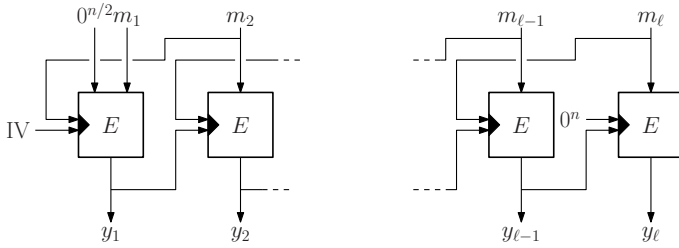


Fig. 1. The IM-construction with block cipher $E : \{0, 1\}^{2n} \times \{0, 1\}^n \rightarrow \{0, 1\}^n$

pads every string so that it consists of one $n/2$ -bit block, followed by as many n -bit blocks as necessary.⁶ On input $M \in \{0, 1\}^*$, it first obtains $\text{PAD}(M) = m_1 \| \dots \| m_\ell$, and computes the output $y_1 \| \dots \| y_\ell$ iteratively such that $y_1 := E(\text{IV} \| m_2, 0^{n/2} \| m_1)$ (where IV is an n -bit fixed *initialization value*) and $y_i := E(y_{i-1} \| m_{i+1}, m_i)$ for all $i = 1, \dots, \ell$, where $m_{\ell+1} := 0^n$.

In contrast to the TE-construction of [23], the IM-construction is *iterated* and allows for (essentially) online processing, with the minimal restriction that only the first $i - 1$ output blocks y_1, \dots, y_{i-1} can be computed from the first i message blocks m_1, \dots, m_i . This one-block-lookahead evaluation strategy only marginally impacts the efficiency of the construction, and is crucial in order to ensure the desired security requirements.

INJECTIVITY OF THE IM-CONSTRUCTION. It is not difficult to see that the construction is injective: Given an output $y_1 \| \dots \| y_\ell$ (for some ℓ) we can iteratively *efficiently* reconstruct the padding $m_1 \| \dots \| m_\ell$ of the input M by computing $m_i := E^{-1}(y_{i-1} \| m_{i+1}, m_i)$ for all $i = \ell, \ell - 1, \dots, 2$, with $m_{\ell+1} = 0^n$, and finally $0^{n/2} \| m_1 := E^{-1}(\text{IV} \| m_2, y_1)$. Thus, IM *cannot* be a VIL-RIO, and not even one way, even though it is surprisingly still strong enough to instantiate the first mixing step of the MCM approach, as we show below.

THE IMC-CONSTRUCTION. It is convenient to define the combination of the IM-construction and a hash function H as the *iterated mix-compress construction* (or IMC-construction, for short), which, on input a string $M \in \{0, 1\}^*$, outputs $\text{IMC}^{E,H}(M) := H(\text{IM}^E(M))$. If H is keyed, then we similarly define the keyed function $\text{IMC}_k^{E,H}(M) := \text{IMC}^{E,H_k}(M)$. Note that if H can be evaluated online, then this is the case for the IMC-construction as well.

SHORTER KEY SIZE. The use of a block cipher with key length equal twice the block length is acceptable in practice.⁷ Still, in order to ensure compatibility with a larger number of block ciphers, we propose an alternative construction (called the DM-IM-construction) which relies on a block cipher $E : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}^n$, at the cost of making *two* calls per processed message block. The underlying idea consists of producing an n -bit key value at each round by using

⁶ This can be done in the canonical way by appending the bit 1 followed by as many 0 bits as necessary in order to fulfill the length requirement.

⁷ For instance, AES supports key size 256 bits with block length $n = 128$ bits.

the Davies-Meyer construction on y_{i-1} and m_{i+1} : More precisely, we compute $y_1 := E(E(m_2, IV) \oplus IV, 0^{n/2} || m_1)$ and $y_i := E(E(m_{i+1}, y_{i-1}) \oplus y_{i-1}, m_i)$ for all $i = 2, \dots, \ell$. As above, for a hash function H , we define $\text{DM-IMC}^{E,H}(M) := H(\text{DM-IM}^E(M))$. (And analogously for the keyed case.)

3.2 Preimage Awareness

The purpose of this section is to prove that, for an ideal cipher $\mathbf{E} : \{0, 1\}^{2n} \times \{0, 1\}^n \rightarrow \{0, 1\}^n$, the construction $\text{IMC}^{\mathbf{E},H}$ is preimage aware, provided H satisfies very weak randomness-preserving properties that we discuss first.

HASH FUNCTION BALANCE. The IMC-construction does not exhibit any useful properties if H can be arbitrary (consider e.g. the case where H is constant). It is nevertheless reasonable to assume H to satisfy minimal structural properties which could be (and generally are) ensured by design. In particular, we require H to preserve some of the randomness of a uniformly chosen input m of a given length n (where n is e.g. the block length of the cipher used in the IM-construction), and this should hold even if m is appended to some other fixed input string M .

Definition 1. An (unkeyed) hash function $H : \{0, 1\}^* \rightarrow \{0, 1\}^h$ is (ϵ, n) -prefix-balanced if for all messages $M \in (\{0, 1\}^n)^*$ and hash function outputs $y \in \{0, 1\}^h$ we have $\mathbb{P}[m \stackrel{\$}{\leftarrow} \{0, 1\}^n : H(M||m) = y] \leq \epsilon$.

The notion extends naturally to a *keyed* hash function $H : \{0, 1\}^\kappa \times \{0, 1\}^* \rightarrow \{0, 1\}^h$: We say that it is (ϵ, n) -prefix balanced if for all keys k the function H_k is $(\epsilon(k), n)$ -prefix balanced, and $\sum_k \mathbb{P}(k) \cdot \epsilon(k) \leq \epsilon$, where $\mathbb{P}(k)$ is the probability that the key generator samples the key k . We remark that the best ϵ one can hope for is $\epsilon = 2^{-n}$ as long as $n \leq h$ holds, whereas $\epsilon \geq 2^{-h}$ for $n \geq h$. Note that our notion is somewhat incomparable to the one of [23], where on the one hand balancedness under variable input lengths is considered (rather than for some fixed length n , as in our case), but, on the other hand, the property is not required under prepending of fixed prefixes: Still we find this extension to be natural in a hashing scenario. It is important to realize that prefix balancedness does not imply any useful security properties for H : The function $H : (\{0, 1\}^n)^+ \rightarrow \{0, 1\}^n$ such that $H(M||m) := m$ for all n -bit strings m and all M with length multiple of n is $(n, 2^{-n})$ -prefix-balanced, despite finding collisions or preimages in this function being trivial.

MAIN THEOREM. The following theorem is the main result of the first part of this paper: It provides a *concrete* characterization of the security of the IMC-construction in the ideal-cipher model. We stress that the result only relies on E being an ideal cipher, and H being sufficiently balanced, but no computational assumption is made, i.e., the result holds with respect to computationally unbounded adversaries.

Theorem 1 (Preimage Awareness of IMC). Let $\mathbf{E} : \{0, 1\}^{2n} \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ be an ideal cipher and let $H : \{0, 1\}^* \rightarrow \{0, 1\}^h$ be an (ϵ, n) -prefix-balanced

hash function. There exists a preimage extractor \mathcal{E} (given in the proof) such that, for all adversaries \mathcal{A} issuing at most q queries to \mathbf{E} and $q_{\mathcal{E}}$ queries to \mathcal{E} , we have

$$\mathbf{Adv}_{\text{IMC}^{\mathbf{E},H},\mathbf{E},\mathcal{E}}^{\text{pra}}(\mathcal{A}) \leq 3 \cdot q(q+1) \cdot 2^{-(n+1)} + q \cdot 2^{-n/2} + q(q+2q_{\mathcal{E}}) \cdot \frac{\epsilon}{2}.$$

Furthermore, \mathcal{E} answers an extraction query in time $\mathcal{O}(|\alpha| \cdot \log |\alpha|)$.

The result extends naturally to a keyed hash function by just averaging the bound over all choices of the key. The security of IMC is bounded by (roughly) $\min\{2^{n/2}, \sqrt{\epsilon}\}$, and is not worse than the one in the TE-construction (which additionally relies on the security of the underlying trapdoor one-way permutation). Note that Theorem 1 is concerned with the entire IMC-construction: An interesting (and seemingly challenging) open question consists of distilling the (minimal) properties needed by IM to yield preimage awareness for IMC.

The remainder of this section is devoted to the proof outline of Theorem 1. Technical details are postponed to the full version, as well as a discussion on how to obtain similar bounds for DM-IMC.

INTERACTION GRAPHS. An interaction with the ideal cipher \mathbf{E} can be described in terms of the *history* α , consisting of triples (k, x, y) , where $k \in \{0, 1\}^{2n}$, and $x, y \in \{0, 1\}^n$. Both a forward query $\mathbf{E}(k, x)$ with output y and a backward query $\mathbf{E}^{-1}(k, y)$ with output x result in a triple (k, x, y) being added to α .⁸ However, it is far more convenient to describe α in terms of a directed (edge labeled) graph $G = G(\alpha) = (V, E)$ with vertex set $V := \{0, 1\}^n$ and edge set $E \subseteq V \times V$ such that $(y, y') \in E$ with labels $\text{label}(y, y') = m$ and $\text{next}(y, y') = m'$ if (i) $(y \| m', m, y') \in \alpha$ with $y \neq \text{IV}$ or (ii) $(y \| m', 0^{n/2} \| m, y') \in \alpha$ if $y = \text{IV}$. A (directed) path $\text{IV} = y_0 \rightarrow y_1 \rightarrow \dots \rightarrow y_{\ell}$ in G is called *valid* if for all $i = 1, \dots, \ell - 1$ we have $\text{label}(y_i, y_{i+1}) = \text{next}(y_{i-1}, y_i)$. It is additionally called *complete* if $\text{next}(y_{\ell-1}, y_{\ell}) = 0^n$. The *value* of a complete valid path is defined as $H(y_1 \| \dots \| y_{\ell})$, and its *preimage* is the string M which is padded to $\text{label}(y_0, y_1) \| \dots \| \text{label}(y_{\ell-1}, y_{\ell})$.

THE PREIMAGE EXTRACTOR \mathcal{E} . On input a history α and a (potential) output $z \in \{0, 1\}^h$ of IMC, the preimage extractor \mathcal{E} first computes the subgraph G' of $G(\alpha)$ induced by the vertices which are reachable through a valid path. If G' is not a directed tree, then \mathcal{E} aborts and outputs \perp . Otherwise, if G' contains one single valid complete path with value z and preimage M , it outputs M . In any other case, it outputs \perp .

It is not hard to see that \mathcal{E} can be implemented with running time $\mathcal{O}(|\alpha| \cdot \log |\alpha|)$ (i.e., where $|\alpha|$ approximately equals the number of edges in the graph $G(\alpha)$) due to the fact that \mathcal{E} aborts if G' is not a tree: Otherwise, the number of possible valid paths may be very high, even exponential.⁹

⁸ The actual history used in the definition of preimage awareness indeed contains more information, such as whether the triple is added by a forward or by a backward query, but this is irrelevant in the following.

⁹ One may argue that we are taking a rather conservative approach: Even if the graph were not a tree, it would most likely have a limited number of valid paths. Still, this considerably simplifies the security analysis with no noticeable loss in the obtained bounds.

PROOF INTUITION. Assume without loss of generality that the adversary \mathcal{A} never repeats a query twice¹⁰ and that whenever it terminates in the pre-game outputting a pair (M, z) , it has made all queries necessary to evaluate the IMC-construction on input M (with output z). In other words, the interaction graph $G(\alpha)$ of the final history α contains a valid complete path with preimage M and value z . But because the query z was previously issued to \mathcal{E} , if \mathcal{A} wins the game, one of the following has to occur: (i) The subgraph of the valid paths is *not* a directed tree, (ii) No valid path with value z existed when the \mathcal{E} -query z was issued, but such a path was created afterwards, or (iii) There exist at least two valid paths with value z . We show that these events are unlikely.

A key step is proving that, with very high probability, valid paths are constructed only by means of *forward* queries: A construction of a valid path by backward queries may be successful either because we can “connect” the path with an already existing one (built by forward queries), or because we construct the entire path backwards. However, both cases turn out to be unlikely: In the former case, a fresh backward query outputs a random m (under the permutation property), and this can only be the next-label for an already existing edge with low probability. (This motivates the one-block-lookahead strategy in IM.) In the latter case, it is very unlikely to have all of the first $n/2$ bits returned by the last evaluation query being equal to 0. (This motivates the padding in the first block.) However, if a path is generated only by forward queries, we can ensure that the value of a valid path is always sufficiently random due to the prefix-balancedness of H . We refer the reader to the full version for a formalization of this argument.

This highlights a very intriguing property of the IM-construction: Although it can be efficiently inverted on any *valid* output, it is very unlikely that we can come up with such a valid output without first evaluating the construction. (In particular, this prevents that even a known collision for H will lead to a valid collision for the IMC-construction.)

4 A Length-Preserving Mixing Stage: Random Permutation Oracles

Post-processing the output of the IMC-construction with a random injective oracle yields a full-fledged random oracle (by Theorem 1 and Lemma 2), whose collision resistance can be reduced to the one of the underlying function H in the standard model by Lemma 1. The use of the TE-construction [23] for this task is subject to two main drawbacks: It requires a *trapdoor* one-way permutation and also enlarges the output of the compressing stage. (The lack of online evaluation capabilities is not a restriction, as we have to process only inputs of *fixed* length equal the output length of the underlying hash function.) In this section, we solve both issues. We present a block-cipher based construction of a *fixed* input-length *length-preserving* RIO, i.e., a (non-invertible) random permutation oracle

¹⁰ In particular, if \mathcal{A} asks a forward query $\mathbf{E}(k, x)$ which is answered by y , the matching backward query $\mathbf{E}^{-1}(k, y)$ is never issued. (And vice versa.)

(RPO), that only relies on a one-way permutation *without* a trapdoor. In the full version, we show that this assumption is somewhat minimal, as RPOs/RPOs cannot be built from an ideal primitive and a one-way *function*.

Additionally, in order to reduce the dependence between the underlying block- and hash sizes, we present domain/range extenders for RPOs.

4.1 Making Block-Ciphers Non-invertible: The NIRP-Construction

DESCRIPTION. The NIRP-construction combines a permutation $\pi : \{0, 1\}^n \rightarrow \{0, 1\}^n$ and two (fixed-key) ciphers $E_1, E_2 : \{0, 1\}^n \rightarrow \{0, 1\}^n$ in a “sandwich-like” manner. More precisely, for any input $m \in \{0, 1\}^n$ the NIRP-construction is defined such that $\text{NIRP}^{E_1, E_2, \pi}(m) := E_2(\pi(E_1(m)))$. (Also cf. Figure 2) Obviously, $\text{NIRP}^{E_1, E_2, \pi}$ is a permutation.

SECURITY OF NIRP. We show that the NIRP-construction is indistinguishable from a (non-invertible) random permutation oracle if instantiated with two *ideal* single-key¹¹ block ciphers $\mathbf{E}_1, \mathbf{E}_2$ and a one-way permutation π (*without* a trapdoor). The result is summarized by the following theorem.

Theorem 2. *Let $\mathbf{E}_1, \mathbf{E}_2 : \{0, 1\}^n \rightarrow \{0, 1\}^n$ be two independent fixed-key ideal ciphers and let $\pi : \{0, 1\}^n \rightarrow \{0, 1\}^n$ be a permutation. There exists a simulator \mathcal{S} (given in the proof) such that for all distinguisher \mathcal{D} issuing at most q queries to the NIRP-construction, and at most q_a, q_b, q_c, q_d queries to $\mathbf{E}_1, \mathbf{E}_1^{-1}, \mathbf{E}_2, \mathbf{E}_2^{-1}$, respectively, there exists an owp-adversary \mathcal{A} with*

$$\text{Adv}_{\text{NIRP}^{\mathbf{E}_1, \mathbf{E}_2, \pi, \mathcal{S}}}^{\text{ind-rio}}(\mathcal{D}) \leq 2 \cdot q_c(2q + q_a) \cdot 2^{-n} + q_d \cdot \text{Adv}_{\pi}^{\text{owp}}(\mathcal{A}).$$

The simulator \mathcal{S} runs in time $\mathcal{O}(q_a + q_b + q_c + q_d + (2q_a + q_b + 2q_d) \cdot \text{time}(\pi))$ and makes $q_a + 2q_b + 2q_c$ queries to its oracle, whereas the adversary \mathcal{A} runs in time $\text{time}(\mathcal{A}) \leq \text{time}(\mathcal{D}) + \text{time}(\mathcal{S})$.

OUTLINE OF THE PROOF. The first part of the indistinguishability proof describes the simulator $\mathcal{S}^{\mathbf{P}}$ that mimics the ideal ciphers $\mathbf{E}_1, \mathbf{E}_2$ (with their inverses $\mathbf{E}_1^{-1}, \mathbf{E}_2^{-1}$) given access to a RPO $\mathbf{P} : \{0, 1\}^n \rightarrow \{0, 1\}^n$. Moreover we use the notation $\mathcal{S}^{\mathbf{P}} = (\mathcal{S}_{\mathbf{E}_1}, \mathcal{S}_{\mathbf{E}_2}, \mathcal{S}_{\mathbf{E}_1^{-1}}, \mathcal{S}_{\mathbf{E}_2^{-1}})$ to make the four sub-oracles of the simulator (answering the different query types) explicit. The second part (which is postponed to the full version) upper bounds \mathcal{D} ’s advantage $\text{Adv}_{\text{NIRP}^{\mathbf{E}_1, \mathbf{E}_2, \pi, \mathcal{S}}}^{\text{ind-rio}}(\mathcal{D})$ in distinguishing the ideal setting (with a simulator) and the real setting.

THE SIMULATOR. The global state of the simulator $\mathcal{S}^{\mathbf{P}}$ consists of a table \mathcal{T} (which is initially empty) of tuples of the form (a, b, c, d) consistent with evaluations of the NIRP-construction as in Figure 2, that is, where a, b are simulated input-output values of the first cipher \mathbf{E}_1 , i.e., $\mathbf{E}_1(a) = b$ (which can be generated both by forward queries to \mathbf{E}_1 and by backward queries to \mathbf{E}_1^{-1}) and analogously

¹¹ Recall that in the ideal cipher model, it is easy to derive two such ciphers from a single ideal cipher $\mathbf{E} : \{0, 1\}^k \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ as $\mathbf{E}_1 := \mathbf{E}(k_1, \cdot)$ and $\mathbf{E}_2 := \mathbf{E}(k_2, \cdot)$ for two arbitrary distinct keys $k_1 \neq k_2$.

c, d play the same role for the second block cipher \mathbf{E}_2 . Furthermore, the invariant $c = \pi(b)$ and $\mathbf{P}(a) = d$ holds. It is also convenient to define $A \subseteq \{0, 1\}^n$ as the set of values $a \in \{0, 1\}^n$ such that $(a, b, c, d) \in \mathcal{T}$ for some b, c, d . Analogously, we define the sets B, C , and D .

To achieve *perfect* simulation given oracle access to \mathbf{P} , upon a new query to one of its four sub-oracles the simulator defines a new tuple (a, b, c, d) in \mathcal{T} , with the input of the query placed at the appropriate position (as long as no such tuple already exists, in which case the corresponding output value is returned), and such that all remaining components are set to independent random values *conditioned* on these individual values appearing in no other tuple, on $d = \mathbf{P}(a)$, and on $c = \pi(b)$. This is easily achievable with access to π^{-1} and \mathbf{P}^{-1} : For example, on input a (to $\mathcal{S}_{\mathbf{E}_1}$), we choose a random $b \stackrel{\$}{\leftarrow} \{0, 1\}^n \setminus B$ (i.e., different from all b' appearing in some other tuple), and set $c := \pi(b)$ and $d := \mathbf{P}(a)$. (This is done analogously on input b .) On the other hand, on input c , we compute $b := \pi^{-1}(c)$, a random $a \stackrel{\$}{\leftarrow} \{0, 1\}^n \setminus A$ (i.e., different from all previous a'), and then set $d := \mathbf{P}(a)$. Finally, on input d , we set $a := \mathbf{P}^{-1}(d)$ and subsequently generate a random $b \leftarrow \{0, 1\}^n \setminus B$ and set $c := \pi(b)$.

However, in our setting we have to dispense with π^{-1} and \mathbf{P}^{-1} . In particular, this means that in the latter two cases the simulator cannot set the values b and a , respectively, but rather sets these components to a *dummy value* \perp , and completes these tuples with the actual values if they eventually appear as inputs of \mathbf{E}_1 or \mathbf{E}_1^{-1} queries. Also note that the simulator must not generate random values a and b that collide with a dummy value in order to ensure the permutation property. This can be efficiently avoided by simply testing that $\mathbf{P}(a) \neq d$ (and $\pi(b) \neq c$) for all d 's in tuples of the form (\perp, b, c, d) (all c 's in tuples of the form (a, \perp, c, d)), and whenever the test fails, we replace the dummy value by the actual value, and draw a new a (or b). There are only two remaining cases where the simulator fails to answer queries (and aborts):

- (i) A query a is made and a tuple (a, \perp, c, d) exists: In this case the simulator *must* return $\pi^{-1}(c)$, but this requires inverting π , which is generally not feasible. (Call this event **Abort**₁.)
- (ii) A query b is made and a tuple (\perp, b, c, d) exists: In this case, the simulator *must* return $\mathbf{P}^{-1}(d)$, but cannot invert \mathbf{P} . (Call this event **Abort**₂.)

By the above discussion, perfect simulation is achieved until one of these events occurs: A game-based argument yields $\mathbf{Adv}_{\text{NIRP}_{\mathbf{E}_1, \mathbf{E}_2, \pi, \mathcal{S}}}^{\text{ind-rio}}(\mathcal{D}) \leq \mathbf{P}[\text{Abort}_1] + \mathbf{P}[\text{Abort}_2]$. In the full version we give a complete pseudo-code description of the simulator and show that the probabilities of both events are very small.

NIRP = MCM WITH INVERTIBLE MIXING STEPS? Our NIRP-construction somehow reflects the MCM design with a permutation, instead of a hash function, and this may suggest that the MCM approach works for invertible mixing steps as well. Yet, we remark that the proof cannot be adapted to the case where the first mixing stage processes inputs of variable input-length: The problem is that in the simulation of queries to \mathbf{E}_2 and \mathbf{E}_2^{-1} we need to choose

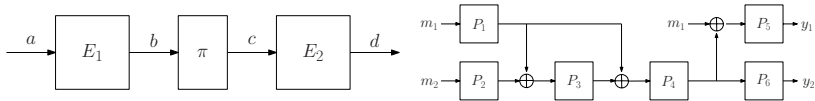


Fig. 2. Left: The NIRP-construction for underlying fixed-key block ciphers E_1, E_2 , with (a, b, c, d) corresponding to the notation used in the simulator of Theorem 2. Right: The ESS-construction for underlying permutations $P_1, \dots, P_6 : \{0, 1\}^n \rightarrow \{0, 1\}^n$.

a pair $a, \mathbf{P}(a)$ and $b, \pi(b)$ respectively, and at a later time possibly learn the missing dummy values b and c when they are queried. But in order for this to succeed, we need the length of a and b to be compatible with the one of such later query, which is of course impossible in the variable-input-length case.

4.2 Extension of Random Permutation Oracles

The use of the NIRP-construction to post-process the output of a hash function H requires a block cipher with block size at least as large as its hash size, i.e., typically at least 160 bits. While block ciphers with large block size exist^[12] ciphers such as AES support only rather small block lengths, such as 128 bits. This motivates the following natural question: Given a RPO $\mathbf{P} : \{0, 1\}^n \rightarrow \{0, 1\}^n$, can we devise a construction $\mathbf{C}^{\mathbf{P}} : \{0, 1\}^m \rightarrow \{0, 1\}^m$ for $m > n$ which implements a permutation and is indistinguishable from a RPO? Note that this calls for simultaneous domain *and* range extension of \mathbf{P} , while we additionally want to ensure injectivity of the resulting construction. The problem is similar in spirit to the one considered in the private-key setting by Halevi and Rogaway [16], even though the peculiarities of the public setting make constructions far more challenging^[13]

THE ESS-CONSTRUCTION. We present a construction – called ESS – for the case $m = 2n$ that relies on six permutations $P_1, \dots, P_6 : \{0, 1\}^n \rightarrow \{0, 1\}^n$ and is reminiscent of the compression function $\text{SS}^{P_1, P_2, P_3} : \{0, 1\}^{2n} \rightarrow \{0, 1\}^n$ by Shrimpton and Stam [25] such that $\text{SS}^{P_1, P_2, P_3}(m_1 || m_2) := P_3(P_1(m_1) \oplus P_2(m_2)) \oplus P_1(m_1)$: It adds three extra calls (as depicted in Figure 2) to ensure both indistinguishability of the $2n$ -bit output, as well as invertibility. It is indeed not hard to verify that ESS implements a permutation: Given output $y_1 || y_2$, the first input-half m_1 is retrieved by computing $z := P_6^{-1}(y_2)$, $m_1 := z \oplus P_5^{-1}(y_1)$, and finally we compute $m_2 := P_2^{-1}(P_1(m_1) \oplus P_3^{-1}(P_1(m_1) \oplus P_4^{-1}(z)))$. (Of course, the inverses P_i^{-1} are not efficiently computable in general, but they are well-defined.)

¹² Interestingly, such block ciphers are exactly the ones used within hash functions, e.g., to instantiate the Davies-Mayer construction.

¹³ In particular, each such extender implies the construction of a compression function $\{0, 1\}^m \rightarrow \{0, 1\}^\ell$ for all $\ell < m$ from length-preserving random oracles which is indistinguishable from a random oracle from m bits to ℓ bits, a problem which has recently received much interest (cf. e.g. [20, 25]). On top of this, injectivity is an extra design challenge.

INDIFFERENTIABILITY OF ESS. The following theorem shows that whenever the underlying permutations are independent RPOs, the ESS-construction is indifferentiable from a RPO up to the birthday barrier.

Theorem 3. *Let $\mathbf{P}_1, \dots, \mathbf{P}_6 : \{0, 1\}^n \rightarrow \{0, 1\}^n$ be independent RPOs. There exists a simulator \mathcal{S} such that for all distinguishers \mathcal{D} making at most q queries to the ESS-construction and to each of the underlying RPOs, we have*

$$\text{Adv}_{\text{ESS}^{\mathbf{P}_1, \dots, \mathbf{P}_6}, \mathcal{S}}^{\text{ind-rio}}(\mathcal{D}) \leq [q^2 \cdot (4n^2 + n + 28) + q \cdot (3n + 13)] \cdot 2^{-n}.$$

The simulator \mathcal{S} runs in time $\mathcal{O}(q^2)$ and makes q queries.

ARBITRARY EXTENSION. A generalization of ESS – called MD-ESS – to construct a RPO $\{0, 1\}^{i \cdot n} \rightarrow \{0, 1\}^{i \cdot n}$ for $i > 2$ using $4 + i$ independent RPOs from n bits to n bits and making $4i + 1$ RPO evaluations in total can be obtained as follows: Let $\text{MD-SS}^{P_1, P_2, P_3} : \{0, 1\}^{n \cdot i} \rightarrow \{0, 1\}^n$ be the (plain) Merkle-Damgård iteration (with no strengthening) that on input $M = m_1 \parallel \dots \parallel m_i$ computes $v_j := \text{SS}^{P_1, P_2, P_3}(v_{j-1} \parallel m_j)$ for $j = 1, \dots, i$ (with v_0 being the IV), and outputs v_i . Then, on input $M = m_1 \parallel \dots \parallel m_i \in \{0, 1\}^{n \cdot i}$, MD-ESS first computes $y := P_4(\text{MD-SS}^{P_1, P_2, P_3}(M))$, and finally outputs

$$(P_{4+1}(y) \oplus m_1) \parallel \dots \parallel (P_{4+i-1}(y) \oplus m_{i-1}) \parallel P_{4+i}(y).$$

To verify that MD-ESS implements a permutation, we remark that its output uniquely determines y and m_1, \dots, m_{i-1} , whereas m_i is determined by the chaining value v_{i-1} and $P_4^{-1}(y)$ as in the ESS-construction. Its security is shown in the full version. There, we also show that $P_{4+1}, \dots, P_{4+i-1}$ (but not P_{4+i}) can be replaced by (invertible) single-key (ideal) ciphers. Also, it can easily be modified to support inputs with lengths $n' \geq n$ which are not multiples of n .

5 Conclusions

In this paper, we have shown the first modular and fault-tolerant hash function construction which achieves both collision resistance in the standard model and indifferenciability in the ideal model. In particular, this was achieved by building appropriate mixing steps IM and NIRP that are compatible with the MCM-construction and preserve the practical features of the inner compressing part, i.e., the hash function H . By Lemma 1, the construction $\text{MCM}^{\text{IM}, H, \text{NIRP}}$ (where possibly NIRP is replaced by its extension through one of the constructions presented in Section 4.2) inherits the collision resistance of H , as IM and NIRP are injective functions. In the ideal setting, we have shown that the combination of IM and H is preimage aware as long as H is sufficiently balanced (Theorem 1), and that NIRP is indifferenciability from a random permutation oracle (Theorem 2). Thus, by applying Lemma 2, we conclude that $\text{MCM}^{\text{IM}, H, \text{NIRP}}$ is indifferenciability from a variable-input-length random oracle.

While the IM-construction is very practical, the implementation of the NIRP-construction, despite its efficiency, is conditioned on the existence of a one-way

permutation with input length equal the one of existing block ciphers. Indeed, sufficiently-secure candidate one-to-one functions exist for similar input parameters (e.g., the discrete logarithm problem in properly chosen elliptic curves of prime order $q \approx 2^n$ can in general not be solved better than with running time roughly $\mathcal{O}(2^{n/2})$, i.e., the security of our constructions), but the fact that the block cipher expects n -bit inputs makes their use difficult¹⁴. However, we stress that such data-type conversion problems are common in practical constructions. For instance, when using an RSA-based trapdoor one-way permutation, the output of the TE-construction [23] must be (injectively) transformed into a string, and the result may be far from being random (attempting to extract random bits would destroy the injectivity property). It is our strong belief that these results should foster further research in designing good candidates for such central cryptographic primitives working at the bit level.

Acknowledgments. We thank Marc Fischlin, Ueli Maurer, Thomas Ristenpart, and the anonymous reviewers for valuable comments. Anja Lehmann is supported by the Emmy Noether Program Fi 940/2-1 of the German Research Foundation (DFG). Stefano Tessaro is supported by the Swiss National Science Foundation (SNF), project no. 200020-113700/1. The work described in this paper has been supported in part by the European Commission through the ICT program under contract ICT-2007-216676 ECRYPT II.

References

1. Andreeva, E., Neven, G., Preneel, B., Shrimpton, T.: Seven-property-preserving iterated hashing: ROX. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 130–146. Springer, Heidelberg (2007)
2. Bellare, M., Ristenpart, T.: Multi-property preserving hash domain extensions and the EMD transform. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 299–314. Springer, Heidelberg (2006)
3. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: ACM CCS 1993. ACM Press, New York (1993)
4. 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)
5. 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)
6. Bertoni, G., Daemen, J., Peeters, M., Assche, G.V.: On the indistinguishability of the sponge construction. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 181–197. Springer, Heidelberg (2008)
7. Canetti, R., Dodis, Y., Pass, R., Walfish, S.: Universally composable security with global setup. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 61–85. Springer, Heidelberg (2007)

¹⁴ A natural candidate operating on n -bit strings arises from exponentiation in the multiplicative group of the extension field $GF(2^n)$: However, due to the existence of non-generic attacks, n has to be chosen large enough, i.e., around the same size as a reasonably secure RSA-modulo.

8. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited. In: STOC 1998, pp. 209–218. ACM Press, New York (1998)
9. Contini, S., Lenstra, A.K., Steinfeld, R.: VSH, an efficient and provable collision-resistant hash function. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 165–182. Springer, Heidelberg (2006)
10. Coron, J.-S., Dodis, Y., Malinaud, C., Puniya, P.: Merkle-Damgård revisited: How to construct a hash function. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 430–448. Springer, Heidelberg (2005)
11. Damgård, I.: A design principle for hash functions. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 416–427. Springer, Heidelberg (1990)
12. Dodis, Y., Ristenpart, T., Shrimpton, T.: Salvaging merkle-damgård for practical applications. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 371–388. Springer, Heidelberg (2009)
13. Fischlin, M., Lehmann, A.: Multi-property preserving combiners for hash functions. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 375–392. Springer, Heidelberg (2008)
14. Fischlin, M., Lehmann, A., Pietrzak, K.: Robust multi-property combiners for hash functions revisited. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfssdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 655–666. Springer, Heidelberg (2008)
15. 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)
16. Halevi, S., Rogaway, P.: A tweakable enciphering mode. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 482–499. Springer, Heidelberg (2003)
17. NIST SHA-3 Competition, <http://csrc.nist.gov/groups/ST/hash/sha-3/>
18. Lyubashevsky, V., Micciancio, D., Peikert, C., Rosen, A.: SWIFFT: A modest proposal for FFT hashing. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 54–72. Springer, Heidelberg (2008)
19. 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)
20. Maurer, U., Tessaro, S.: Domain extension of public random functions: Beyond the birthday barrier. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 187–204. Springer, Heidelberg (2007)
21. Merkle, R.: One way hash functions and DES. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 428–446. Springer, Heidelberg (1990)
22. Pass, R.: On deniability in the common reference string and random oracle model. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 316–337. Springer, Heidelberg (2003)
23. Ristenpart, T., Shrimpton, T.: How to build a hash function from any collision-resistant function. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 147–163. Springer, Heidelberg (2007)
24. Rogaway, P.: Formalizing human ignorance. In: Nguyen, P.Q. (ed.) VIETCRYPT 2006. LNCS, vol. 4341, pp. 211–228. Springer, Heidelberg (2006)
25. Shrimpton, T., Stam, M.: Building a collision-resistant compression function from non-compressing primitives. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfssdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 643–654. Springer, Heidelberg (2008)

How to Confirm Cryptosystems Security: The Original Merkle-Damgård Is Still Alive!

Yusuke Naito¹, Kazuki Yoneyama², Lei Wang³, and Kazuo Ohta³

¹ Mitsubishi Electric Corporation

² NTT Corporation

³ The University of Electro-Communications

Abstract. At Crypto 2005, Coron et al. showed that Merkle-Damgård hash function (MDHF) with a fixed input length random oracle is not indifferentiable from a random oracle RO due to the extension attack. Namely MDHF does not behave like RO. This result implies that there exists some cryptosystem secure in the RO model but insecure under MDHF. However, this does not imply that no cryptosystem is secure under MDHF. This fact motivates us to establish a criteria methodology for confirming cryptosystems security under MDHF.

In this paper, we confirm cryptosystems security by using the following approach:

1. Find a variant, \widetilde{RO} , of RO which leaks the information needed to realize the extension attack.
2. Prove that MDHF is indifferentiable from \widetilde{RO} .
3. Prove cryptosystems security in the \widetilde{RO} model.

From the indifferentiability framework, a cryptosystem secure in the \widetilde{RO} model is also secure under MDHF. Thus we concentrate on finding \widetilde{RO} , which is weaker than RO.

We propose the Traceable Random Oracle (TRO) which leaks enough information to permit the extension attack. By using TRO, we can *easily* confirm the security of OAEP and variants of OAEP. However, there are several practical cryptosystems whose security cannot be confirmed by TRO (e.g. RSA-KEM). This is because TRO leaks information that is irrelevant to the extension attack. Therefore, we propose another \widetilde{RO} , the Extension Attack Simulatable Random Oracle, ERO, that leaks *just* the information needed for the extension attack. Fortunately, ERO is *necessary and sufficient* to confirm the security of cryptosystems under MDHF. This means that the security of *any* cryptosystem under MDHF is *equivalent* to that under the ERO model. We prove that RSA-KEM is secure in the ERO model.

Keywords: Indifferentiability, Merkle-Damgård hash function, Variants of Random Oracle, Cryptosystems Security.

1 Introduction

Indifferentiability Framework. Maurer et al. [9] introduced the indifferentiable framework as a notion stronger than indistinguishability. This framework

deals with the security of two systems $\mathcal{C}(\mathcal{V})$ and $\mathcal{C}(\mathcal{U})$: for cryptosystem \mathcal{C} , $\mathcal{C}(\mathcal{V})$ retains at least the same level of provable security of $\mathcal{C}(\mathcal{U})$ if primitive \mathcal{V} is indistinguishable from primitive \mathcal{U} , denoted by $\mathcal{V} \sqsubset \mathcal{U}$. This definition will allow us to use construction \mathcal{V} instead of \mathcal{U} in any cryptosystem \mathcal{C} and retain the same level of provable security due to the indistinguishability framework of Maurer et al. [9]. We denote “ $\mathcal{C}(\mathcal{V})$ is at least as secure as $\mathcal{C}(\mathcal{U})$ ” by $\mathcal{C}(\mathcal{V}) \succ \mathcal{C}(\mathcal{U})$. More strictly, $\mathcal{V} \sqsubset \mathcal{U} \Leftrightarrow \mathcal{C}(\mathcal{V}) \succ \mathcal{C}(\mathcal{U})$ holds. This result implies that if cryptosystem \mathcal{C} is secure in the \mathcal{U} model and $\mathcal{V} \sqsubset \mathcal{U}$ holds, \mathcal{C} is secure in the \mathcal{V} model, and if $\mathcal{U} \not\sqsubset \mathcal{V}$ holds, there is some cryptosystem that is secure in the \mathcal{U} model but insecure in the \mathcal{V} model.

Indistinguishability and the MD Construction. While many cryptosystems have been proven to be secure in the random oracle (RO) model [3] (e.g. FDH [3], OAEP [4], RSA-KEM [11], Prefix-MAC [12] and so on), where RO is modeled as a monolithic entity (i.e. a black box working in domain $\{0, 1\}^*$), in practice most instantiations that use a hash function are usually constructed by iterating a fixed input length primitive (e.g. a compression function). There are many architectures based on iterated hash functions. The most well-known one is the Merkle-Damgård (MD) construction [6, 10]. A hash function with MD construction iterates underlying compression function $f : \{0, 1\}^n \times \{0, 1\}^t \rightarrow \{0, 1\}^n$ as follows.

```

MDf(m1, ..., ml) (|mi| = t, i = 1, ..., l):
  let y0 = IV be some n bit fixed value.
  for i = 1 to l do yi = f(yi-1, mi)
  return yl

```

There is a significant gap between RO and hash functions, since hash functions are constructed from a small primitive f while RO is a monolithic random function.

Coron et al. [5] made important observations on the cryptosystems that use the indistinguishability framework. They introduced the new iterated hash function property of indistinguishability from RO. In this framework, the underlying primitive, G , is a fixed input length random oracle (denoted here as FILRO or h) or an ideal block cipher. We say that hash function H^G is indistinguishable from RO if there exists simulator S such that no distinguisher can distinguish H^G from RO (S mimics G). The distinguisher can access RO/H^G and S/G ; S can access RO . A hash function that satisfies this property, H^G , behaves like RO. Therefore, replacing the RO of any cryptosystem by H^G does not destroy its security.

Coron et al. analyzed the indistinguishability from RO for several specific constructions. For example, they have shown that MD^h is not indistinguishable from RO due to the extension attack which uses the following property: The output value $z' = MD^h(M||m)$ can be calculated by $c = h(z, m)$ where $z = MD^h(M)$, so $z' = c$. On the other hand, no S can return the output value $z' = RO(M||m)$ from query (z, m) where $z = RO(M)$, since no S knows z' from z and m , and z' is chosen at random. Therefore, no S can simulate the extension attack. This result implies that MD^h does not behave like RO and there exists some cryptosystem

that is secure in the RO model but insecure under MD^h due to the indistinguishability framework. Their solution was to propose several constructions such as Prefix-Free MD, chop MD, NMAC and HMAC. Hash functions with these constructions are, under h , indistinguishable from RO. It seems impossible to prove that the important *original* MD cryptosystem is secure.

MD Construction Dead? The MD construction is among the most important foundations of modern cryptosystems [2,5,8]. There are two main reasons:

1. MD construction is employed by many popular hash functions such as SHA-1 and SHA-256, and
2. MD construction is more efficient than other iterated hash functions such as Prefix-Free MD, and chop MD.

Since $MD^h \not\sqsubseteq RO$ holds, there is some cryptosystem \mathcal{C}^* that is secure in the RO model but insecure under MD^h . Thus the important question is “can we confirm that a given cryptosystem is secure in the RO model and secure under MD^h ?” There might be several cryptosystems that remain secure when RO is replaced by MD^h . If we can confirm this for many cryptosystems that are widely used, the *original* MD construction remains alive in the indistinguishability framework!

Our Contribution. Since $MD^h \not\sqsubseteq RO$ holds, we modify RO such that MD^h is indistinguishable from the modified RO. Then we analyze cryptosystems security within the modified RO model. Concretely, we adopt the following approach.

1. Find a variant \widetilde{RO} of RO that leaks enough information such that S can simulate the extension attack.
2. Prove that $MD^h \sqsubseteq \widetilde{RO}$ holds.
3. Prove the cryptosystem’s security in the \widetilde{RO} model.

Secure cryptosystems in the \widetilde{RO} model are also secure under MD^h due to the indistinguishability framework. Therefore, we concentrate on proposing \widetilde{RO} that can support many applications.

First we propose *Traceable Random Oracle* TRO as \widetilde{RO} .

Traceable Random Oracle. Our proposal of TRO is motivated by the following points:

- Applications of TRO hide the outputs of hash functions from adversaries. One example is OAEP encryption: Adversaries cannot know the outputs of the hash functions that are used for calculating a cipher text, since these values are hidden by a random value or a trapdoor one-way permutation.
- TRO leaks useful information such that S can run the extension attack.

By considering the above points, it is convenient for S to obtain useful information from value z which is the output of $RO(M)$. Thus we define TRO that leaks input M on query z such that $RO(M) = z$. Since S can obtain value M such that $z = RO(M)$, S can know value $z' = RO(M||m)$ by using TRO. Therefore, S can run the extension attack. We will prove that $MD^h \sqsubseteq TRO$ holds (Corollary 2).

Since the hash function outputs for OAEP and variants of OAEP (e.g. OAEP+) are hidden, adversaries cannot use TRO effectively. So we can easily confirm that these cryptosystems are secure in the TRO model.

Limitation of TRO. Though TRO can easily confirm the security of many cryptosystems under MD^h , there are several cryptosystems whose security we cannot confirm by TRO. For example, RSA-KEM is insecure in the TRO model (Theorem 7). It is possible that there are cryptosystems that are secure under MD^h because TRO leaks information beyond that needed to simulate the extension attack. The essential information to simulate the extension attack is just $z' = RO(M||m)$, but TRO leaks M , which is not essential.

Our response is to propose *Extension Attack Simulatable Random Oracle* ERO as \widetilde{RO} .

Extension Attack Simulatable Random Oracle. We define ERO that leaks just z' ($= RO(M||m)$). By using ERO, S can run the extension attack, since S can know z' . We will prove that $MD^h \sqsubset ERO$ holds (Theorem 5). We will also prove that RSA-KEM is secure in the ERO model (Theorem 8). Therefore, we can confirm RSA-KEM security under MD^h by using ERO. Fortunately, MD^h is equivalent to ERO, since $ERO \sqsubset MD^h$ holds (Theorem 6). Namely, any cryptosystem that is secure under MD^h is equally secure in the ERO model and vice versa. Therefore, ERO is necessary and sufficient to confirm the security of cryptosystems under MD^h . When we analyze a cryptosystem under MD^h , all that is needed is to prove cryptosystems security in the ERO model.

TRO v.s. ERO. Since TRO leaks more information than ERO, we will prove $ERO \sqsubset TRO$. Since ERO has wider applicability, we recommend that ERO be used for cryptosystems whose security cannot be proven in the TRO model.

ERO v.s. RO. Since ERO leaks several bits of information in permitting the simulation of the extension attack, $RO \sqsubset ERO$ and $ERO \not\sqsubset RO$ explicitly hold. As evidence of the separation between RO and ERO, we pick up prefix MAC [12] which is secure in the RO model, and prove that prefix MAC is insecure in the ERO model (Theorem 4). Since ERO is equivalent to MD^h , prefix MAC is also insecure in the MD^h model.

Leaky Random Oracle. Leaky random oracle LRO was proposed by Yoneyama et al. [13] but with a different motivation. LRO has a function that leaks all query-response pairs of RO. In this paper, we will prove that $TRO \sqsubset LRO$ and $LRO \not\sqsubset TRO$ hold. Therefore, all cryptosystems secure in the LRO model are also secure in the TRO model and there is some cryptosystem that is insecure in the LRO model but secure in the TRO model. Since FDH is secure in LRO model [13], FDH is secure under MD^h . Since OAEP is insecure in the LRO model [13] and secure in the TRO model, OAEP is evidence of the separation between LRO and TRO.

Remarks. First we compare LRO, TRO and ERO from the viewpoint of security proofs of cryptosystems. LRO, TRO, and ERO consist of RO and the additional

oracle (denote LO, TO and EO respectively). Since LO leaks more information to adversaries than TO, adversaries that are given LRO have more flexible strategies than adversaries given TRO. That is, security proofs in the LRO model are more complex than those in the TRO model. The same is true for TRO and ERO.

Finally, for the security proof of cryptosystem $\mathcal{C}(\text{MD}^h)$ we compare the direct proof in MD^h with the proof via ERO. Since MD^h has the MD structure, we must consider this structure in the direct proof. On the other hand, since ERO does not have this structure, we does not need to consider it. For example we must consider the events of inner collisions for MD^h in the direct proof. However this is not necessary for the proof in the ERO model. Moreover, since we can reuse existing proofs for the simulation of RO in the security proof in the ERO model, we only consider the simulation of EO in the security proof. Therefore, the security proof in the ERO model is easier than the direct proof in MD^h . Since $\text{ERO} = \text{MD}^h$ holds, we can confirm a cryptosystems security under MD^h by proving its security in ERO, an easier task than a direct proof.

Related Works. Recently, Dodis et al. independently proposed a methodology to salvage the original and modified MD constructions in many applications [7]. They found two properties: one is preimage awareness (PrA), and the other is public-use random oracle (pub-RO). pub-RO is the same as LRO. The approach of pub-RO is almost same as our approach of LRO. Dodis et al. pointed out that the security of cryptosystems that satisfy the following property can be easily proven in the pub-RO model: all inputs of hash functions are public to the adversaries. Therefore, PSS and the Fiat-Shamir signature scheme, and other, are easily proven to be secure in the pub-RO model by using existing proofs in the RO model. Since $\text{LRO}(\text{pub-RO}) \not\sqsubseteq \text{TRO}$ and $\text{TRO} \sqsubseteq \text{LRO}(\text{pub-RO})$ hold, TRO and ERO have more applications than $\text{LRO}(\text{pub-RO})$ (e.g. OAEP is secure in the TRO model but insecure in the pub-RO model). The approach of PrA is interesting in that this approach can treat the case where the compression function f requirement is relaxed from FILRO to property PrA. It seems, however, that this approach is not effective in saving the original MD construction, since this approach *modifies* MD construction by processing the output of the MD construction by FILRO.

Cryptosystems Security under the Merkle-Damgård Hash Function. PSS, Fiat-Shamir, and so on are secure under MD^h thanks to pub-RO [7], OAEP and variants of OAEP are secure under MD^h thanks to TRO, and RSA-KEM is secure under MD^h thanks to ERO. Since many cryptosystems are secure under MD^h , the original Merkle-Damgård construction is still alive!

2 Preliminaries

2.1 Merkle-Damgård Construction

We first give a short description of the Merkle-Damgård (MD) construction. Function $\text{MD}^f : \{0, 1\}^* \rightarrow \{0, 1\}^n$ is built by iterating compression function $f : \{0, 1\}^n \times \{0, 1\}^t \rightarrow \{0, 1\}^n$ as follows.

- $MD^f(M)$:
 1. calculate $M' = pad(M)$ where pad is a padding function such that $pad : \{0, 1\}^* \rightarrow (\{0, 1\}^t)^*$.
 2. calculate $c_i = f(c_{i-1}, m_i)$ for $i = 1, \dots, l$ where for $i = 1, \dots, l$, $|m_i| = t$, $M' = m_1 || \dots || m_l$ and c_0 is an initial value (s.t. $|c_0| = n$).
 3. return c_n

In this paper we ignore the above padding function, this does not degrade generality, so hereafter we discuss $MD^f : (\{0, 1\}^t)^* \rightarrow \{0, 1\}^n$. We use random oracle compression function h as f where $h : \{0, 1\}^n \times \{0, 1\}^t \rightarrow \{0, 1\}^n$. Thus we discuss below hash function MD^h with MD construction using h .

2.2 Random Oracle

RO : $\{0, 1\}^* \rightarrow \{0, 1\}^n$ can be realized as follows. RO has initially the empty hash list \mathcal{L}_{RO} . On query M , if $\exists(M, z) \in \mathcal{L}_{RO}$, it returns z . Otherwise, it chooses $z \in \{0, 1\}^n$ at random, adds (M, z) to the \mathcal{L}_{RO} , hereafter denoted by $\mathcal{L}_{RO} \leftarrow (M, z)$, and returns z .

2.3 Leaky Random Oracle

LRO was proposed by Yoneyama et al. [13]. LRO can be realized as follows. LRO consists of RO and LO. On a leak query to LO, LO outputs the entire contents of \mathcal{L}_{RO} . We can define S that can simulate the extension attack by using LRO, since S can know M from z by using LO and can know z' by posing $M || m$ to RO.

2.4 Indifferentiability

The indifferentiability framework generalizes the fundamental concept of the indistinguishability of two cryptosystems $\mathcal{C}(\mathcal{U})$ and $\mathcal{C}(\mathcal{V})$ where $\mathcal{C}(\mathcal{U})$ is the cryptosystem \mathcal{C} that invokes the underlying primitive \mathcal{U} and $\mathcal{C}(\mathcal{V})$ is the cryptosystem \mathcal{C} that invokes the underlying primitive \mathcal{V} . \mathcal{U} and \mathcal{V} have two interfaces: public and private interfaces. Adversaries can only access the public interfaces and honest parties (e.g. the cryptosystem \mathcal{C}) can access only the private interface.

We denote the private interface of the system \mathcal{W} by \mathcal{W}^{priv} and the public interface of the system \mathcal{W} by \mathcal{W}^{pub} . The definition of indifferentiability is as follows.

Definition 1. \mathcal{V} is indifferentiable from \mathcal{U} , denote $\mathcal{V} \sqsubset \mathcal{U}$, if for any distinguisher D with binary output (0 or 1) there is a polynomial time simulator S such that $|Pr[D^{\mathcal{V}^{priv}, \mathcal{V}^{pub}} \Rightarrow 1] - Pr[D^{\mathcal{U}^{priv}, S(\mathcal{U}^{pub})} \Rightarrow 1]| < \epsilon$. Simulator S has oracle access to \mathcal{U}^{pub} and runs in time at most t_S . Distinguisher D runs in time at most t_D and makes at most q queries. ϵ is negligible in security parameter k .

This definition will allow us to use construction \mathcal{V} instead of \mathcal{U} in any cryptosystem \mathcal{C} and retain the same level of provable security due to the indifferentiability theory of Maurer et al. [9]. We denote “ $\mathcal{C}(\mathcal{V})$ is at least as secure as $\mathcal{C}(\mathcal{U})$ ” by $\mathcal{C}(\mathcal{V}) \succ \mathcal{C}(\mathcal{U})$. Namely, $\mathcal{C}(\mathcal{V}) \succ \mathcal{C}(\mathcal{U})$ denotes the case that if $\mathcal{C}(\mathcal{U})$ is secure, then $\mathcal{C}(\mathcal{V})$ is secure. More strictly, $\mathcal{V} \sqsubset \mathcal{U} \Leftrightarrow \mathcal{C}(\mathcal{V}) \succ \mathcal{C}(\mathcal{U})$ holds.

2.5 Extension Attack

Coron et al. showed that MD^h is not indifferentiable from RO due to the extension attack. The extension attack targets MD^h where we can calculate a new hash value from some hash value. Namely $z' = \text{MD}^h(M||m)$ can be calculated from only z and m by $z' = h(z, m)$ where $z = \text{MD}^h(M)$. Note that z' can be calculated without using M . The differentiable attack with extension attack is as follows. Let \mathcal{O}_a be MD^h or RO and let \mathcal{O}_b be h or S . First, a distinguisher poses M to \mathcal{O}_a and gets z from \mathcal{O}_a . Second, he poses (z, m) to \mathcal{O}_b and gets c from \mathcal{O}_b . Finally, he poses $M||m$ to \mathcal{O}_a and gets z' from \mathcal{O}_a .

If $\mathcal{O}_a = \text{MD}^h$ and $\mathcal{O}_b = h$, then $z' = c$, while, if $\mathcal{O}_a = \text{RO}$ and $\mathcal{O}_b = S$, then $z' \neq c$. This is because no simulator can obtain the output value of $\text{RO}(M||m)$ from just (z, m) and the output value of $\text{RO}(M||m)$ is independently and randomly defined from c . Therefore, $\text{MD}^h \not\sqsubseteq \text{RO}$ holds.

3 Variants of Random Oracles

In this section, we will introduce several variants of random oracles in order for S to simulate the extension attack described above, and then show the relationships among these oracles within the indifferentiability framework.

3.1 Definition of Variants of Random Oracles

Traceable Random Oracle: TRO consists of RO and TO. On trace query z ,

1. If there exist pairs such that $(M_i, z) \in \mathcal{L}_{\text{RO}}$ ($i = 1, \dots, n$), it returns (M_1, \dots, M_n) .
2. Otherwise, it returns \perp .

We can define S that can simulate the extension attack by using TRO, since S can know M from z by using TO and can know z' by posing $M||m$ to RO.

Extension Attack Simulatable Random Oracle: TRO leaks too much information to simulate the extension attack. So we define ERO such that S is given just the important information. The important information is value z' such that $z' = \text{RO}(M||m)$. Therefore, we define ERO as follows. ERO consists of RO and EO. EO has initially the empty list \mathcal{L}_{EO} and can look into \mathcal{L}_{RO} . On simulation query (m, z) to EO where $|m| = t$,

1. If $(m, z, z') \in \mathcal{L}_{\text{EO}}$, it returns z' .
2. Else if $z = IV$, EO poses query m to RO, receives z' , $\mathcal{L}_{\text{EO}} \leftarrow (m, z, z')$, and returns z' .
3. Else if there exists only one pair $(M, z) \in \mathcal{L}_{\text{RO}}$, EO poses query $M||m$ to RO, receives z' , $\mathcal{L}_{\text{EO}} \leftarrow (m, z, z')$, and returns z' .
4. Else EO chooses $z' \in \{0, 1\}^n$ at random, $\mathcal{L}_{\text{EO}} \leftarrow (m, z, z')$ and returns z' .

We can construct S that can simulate the extension attack by using ERO, since S can obtain z' from (m, z) where $z' = \text{RO}(M||m)$ by using EO.

3.2 Relationships among LRO, TRO, ERO, and RO Models within the Indifferentiability Framework

LRO leaks more information of \mathcal{L}_{RO} than TRO, and TRO leaks more information of \mathcal{L}_{RO} than ERO. Therefore, it seems reasonable to suppose that anything secure in the LRO model is also secure in the TRO model, anything secure in the TRO model is also secure in the ERO model, and any cryptosystem secure in the ERO model is also secure in the RO model. We prove the validity of these suppositions by using the indifferentiability framework.

First we clarify the relationship between TRO and LRO.

Theorem 1. $TRO \sqsubset LRO$ and $LRO \not\sqsubset TRO$.

Proof. We construct S which simulates TO by using LRO as follows. Given query z , S poses a leak query to LO and receives the entire information of \mathcal{L}_{RO} . If there exists pairs such that $(M_i, z) \in \mathcal{L}_{RO}$ ($i = 1, \dots, n$), it returns (M_1, \dots, M_n) . Otherwise it returns \perp .

It is easy to see that $|Pr[D^{RO, TO} \Rightarrow 1] - Pr[D^{RO, S(LRO)} \Rightarrow 1]| = 0$, since the output from each step of S is equal to that from each step of TO.

LRO $\not\sqsubset$ TRO is trivial, since no S cannot acquire all values in \mathcal{L}_{RO} by using TRO. \square

Since $TRO \sqsubset LRO$, any cryptosystem secure in the LRO model is also secure in the TRO model by the indifferentiability framework. Since $LRO \not\sqsubset TRO$, there exists some cryptosystem that is secure in the TRO model but insecure in the LRO model. For example, Yoneyama et al. proved that OAEP is insecure in the LRO model [13]. Since OAEP is secure in the TRO model, OAEP is evidence of the separation between LRO and TRO.

Next we will clarify the relationship between ERO and TRO.

Theorem 2. $ERO \sqsubset TRO$ and $TRO \not\sqsubset ERO$.

Proof. We construct S which simulates EO by using TRO as follows. S initially has the empty list \mathcal{L}_S . On query (m, z) , if $\exists(m, z, z') \in \mathcal{L}_S$, it returns z' . Otherwise S poses query z to TO, and receives string X . If X consists of one value, it poses query $X||m$ to RO, receives z' , $\mathcal{L}_S \leftarrow (m, z, z')$ and returns z' . Otherwise, it chooses $z' \in \{0, 1\}^n$ at random, $\mathcal{L}_S \leftarrow (m, z, z')$ and returns z' .

It is easy to see that $|Pr[D^{RO, EO} \Rightarrow 1] - Pr[D^{RO, S(TRO)} \Rightarrow 1]| = 0$, since the output from each step of S is equal to that from each step of EO.

TRO $\not\sqsubset$ ERO is trivial, since no S cannot decide whether there exists (M, z) in \mathcal{L}_{RO} or not by using ERO. \square

Since $ERO \sqsubset TRO$, any cryptosystem secure in the TRO model is also secure in the ERO model in the indifferentiability framework. Since $TRO \not\sqsubset ERO$, there exists some cryptosystem that is secure in the ERO model but insecure in the TRO model. We will prove that RSA-KEM is secure in the ERO model but insecure in the TRO model in Section 5. Therefore, RSA-KEM is evidence of the separation between TRO and ERO.

Finally we will clarify the relationship between RO and ERO.

Theorem 3. $RO \sqsubset ERO$ and $ERO \not\sqsubset RO$.

This proof of theorem 3 is trivial because ERO consists of RO and the additional oracle EO which leaks some information of \mathcal{L}_{RO} . Since $RO \sqsubset ERO$, any cryptosystem secure in the ERO model is also secure in the RO model by the indifferntiability framework. Since $ERO \not\sqsubset RO$, there exists some cryptosystem which is secure in the RO model but insecure in the ERO model. We can show simple evidence of the separation between ERO and RO as follows: We consider the following Prefix-MAC protocol which is unforgeable in the RO model. Note that the concept of unforgeability with regard to MAC schemes is defined in [11].

Prefix MAC [12]: Alice and Bob share one secret key, K , as an authentication key. Before sending message M to Bob, Alice sends $K||M$ to RO H to obtain a MAC value denoted as y . Finally, Alice sends (M, y) to Bob. When Bob obtains (M, y) , he sends $K||M$ to H to obtain another MAC value y' . If y' is equal to y , then Bob is convinced that message M is from Alice. Otherwise, Bob will reject message M .

We will show that Prefix MAC fails to satisfy unforgeability for MAC schemes in the ERO model.

Theorem 4 (Insecurity of Prefix MAC in the ERO model). *Prefix MAC does not satisfy unforgeability for MAC schemes where H is modeled as ERO.*

Proof. A forgery procedure is as follows: forger \mathcal{F} obtains a valid pair of (M, h) from MAC, where $h = H(K||M)$. \mathcal{F} sends (h, m) to EO, and obtains $h' = H(K||M||m)$. Since $M||m$ is not queried to MAC, \mathcal{F} succeeds in Existential forgery of known message attack (EF-KMA) attack using ERO H . \square

Therefore, Prefix-MAC is secure in the RO model but insecure in the ERO model. Consequently, Prefix-MAC is evidence of the separation between ERO and RO.

From the above discussions, the following corollary is obtained.

Corollary 1. $RO \sqsubset ERO \sqsubset TRO \sqsubset LRO$, and $LRO \not\sqsubset TRO \not\sqsubset ERO \not\sqsubset RO$.

4 Relationship between MD^h and ERO in the Indifferntiability Framework

In this section we prove that $MD^h \sqsubset ERO$ and $ERO \sqsubset MD^h$ hold as follows. In theorem 5, we use statements σ_H and q_h instead of the total number of queries q . σ_H is the total number of message blocks for RO/MD^h and q_h is the total number of queries to S/h

Theorem 5. $MD^h \sqsubset ERO$, for any t_D , with $t_S = O(q_h^2)$ and $\epsilon \leq \frac{4(\sigma_H+q_h)^2+2(\sigma_H+q_h)}{2^n}$.

This proof is given in subsection 4.1.

In theorem 6, we use statements σ_H and q_{EO} instead of the total number of queries q . σ_H is the total number of message blocks for RO/MD^h and q_{EO} is the total number of queries to EO/S

Theorem 6. $\text{ERO} \sqsubset \text{MD}^h$, for any t_D , with $t_S = O(q_{\text{EO}})$ and $\epsilon \leq \frac{2(\sigma_H + q_{\text{EO}})^2 + (\sigma_H + q_{\text{EO}})}{2^n}$.

This proof is given in subsection 4.2.

From Theorem 5 and Theorem 6, ERO is equivalent to MD^h in the indistinguishability framework. From Corollary 1, Theorem 5 and Theorem 6, the following corollary is obtained.

Corollary 2. $\text{RO} \sqsubset \text{MD}^h = \text{ERO} \sqsubset \text{TRO} \sqsubset \text{LRO}$, and $\text{LRO} \not\sqsubset \text{TRO} \not\sqsubset \text{ERO} = \text{MD}^h \not\sqsubset \text{RO}$

4.1 Proof of Theorem 5

First we define simulator S as follows. S has a list \mathcal{T} which is initially empty. We define chain triples as follows.

Definition 2 (Chain Triples). Triples $(x_1, m_1, y_1), \dots, (x_i, m_i, y_i)$ are chain triples if $x_1 = IV$ and $y_j = x_{j+1}$ ($j = 1, \dots, j - 1$) holds.

Simulator S: On a query (x, m) ,

1. If $\exists(x, m, y) \in \mathcal{T}$, it outputs y .
2. Else if chain triples $\exists(x_1, m_1, y_1), \dots, (x_i, m_i, y_i) \in \mathcal{T}$ such that $x = y_i, y \leftarrow \text{RO}(m_1 || \dots || m_i || m)$.
3. Else, $y \leftarrow \text{EO}(m, x)$.
4. $\mathcal{T} \leftarrow (x, m, y)$.
5. S returns y .

Since S needs to search pairs in \mathcal{T} , this requires at most $O(q_h^2)$ time.

We need to prove that S cannot tell apart two scenarios, ERO and MD^h . In one scenario D has oracle access to RO and S while in the other D has access to MD^h and h . The proof involves a hybrid argument starting in the ERO scenario, and ending in the MD^h scenario through a sequence of mutually indistinguishable hybrid games.

We give six events that allow D to distinguish MD^h from ERO. These events arise from the fact that MD^h has the MD construction but ERO does not. We explain these events as follows. Details of these events are given in Game 3.

First we discuss distinguishing events that occur due to differences among RO and MD^h . RO and MD^h return a random value unless collision occurs. Therefore, distinguishing events occur when collision occurs. When a collision of MD^h occurs, one of following events occurs due to the MD construction: an output of h is equal to IV (event E1) or a collision of h occurs (event E2). On the other hand, since RO is a monolithic function, these events don't occur. Therefore, these events are distinguishing events between MD^h and ERO.

Second, we discuss distinguishing events that occur due to differences among S and h . Since for h there is the relation that $h(x, m) = \text{RO}(M || m)$ where $\text{MD}^h(M) = x$, S must simulate the relation such that $S(x, m) = \text{RO}(M || m)$ where $\text{RO}(M) = x$. On query (x, m) to S, if only one pair exists $(M, x) \in \mathcal{L}_{\text{RO}}$

such that $x \neq IV$ holds, S can know $MD^h(M||m)$ by using EO . Therefore, S can simulate the relation. If such a pair does not exist ($(M, x) \notin \mathcal{L}_{RO}$), since S cannot know M , S cannot know the value of $RO(M||m)$. Therefore, S cannot simulate the relation (event $E3$ and event $E5$). If two or more such pairs exist ($(M, x), (M', x), \dots \in \mathcal{L}_{RO}$), S must simulate the relation such that $RO(M||m) = RO(M'||m) = \dots$. However, since S cannot control the outputs of RO , it cannot simulate the relation (event $E4$).

On the other hand, if $\exists(M, x) \in \mathcal{L}_{RO}$ such that $x = IV$, S must simulate the relation such that $RO(m) = RO(M||m)$. However, since S cannot control the outputs of RO , it cannot simulate the relation (event $E6$).

In following game transforms, since the MD construction is considered in Game 3 for the first time, we discuss these events in the transform from Game 2 to Game 3. In this discussion, we show that if distinguishing events don't occur, Game 3 is identical to Game 2, and the probability that one of the events will occur is negligible.

Game 1: This is the random oracle model, where D has oracle access to RO and S . Let $G1$ denote the event that D outputs 1 after interacting with RO and S . Thus $Pr[G1] = Pr[D^{RO, S(ERO)} = 1]$.

Game 2: In this game, we give the distinguisher oracle access to a dummy relay algorithm R_0 instead of direct oracle access to RO . R_0 is given oracle access to RO . On query M to R_0 , it queries M to RO and returns $RO(M)$. Let $G2$ denote the event that D outputs 1 in Game 2. Since the view of D remains unchanged in this game, $Pr[G2] = Pr[G1]$.

Game 3: In this game, we modify the relay algorithm R_0 into R_1 as follows. For hash oracle query M , R_1 applies the MD construction to M by querying S . R_1 is essentially the same as MD^h except that R_1 is based on S instead of the fixed input length random oracle h .

We show that Game 3 is identical with Game 2 unless the following bad events occur. In response to query (x, m) , S chooses response $y \in \{0, 1\}^n$:

- $E1$: It is the case that $y = IV$.
- $E2$: There is a triple $(x', m', y') \in \mathcal{T}$, with $(x', m') \neq (x, m)$, such that $y' = y$.
- $E3$: There is a triple $(x', m', y') \in \mathcal{T}$, with $(x', m') \neq (x, m)$, such that $x' = y$ and (x', m', y') is defined except for step 3 of EO .

and in a response to a query M to RO , RO returns z :

- $E4$: There is a pair $(M', z') \in \mathcal{L}_{RO}$, with $M \neq M'$ such that $z = z'$.
- $E5$: There is a triple $(x', m', y') \in \mathcal{T}$ such that $z = x'$.
- $E6$: $z = IV$.

We demonstrate that Game 3 is identical with Game 2 unless bad events occur and the probability that bad events occur is negligible. Before we demonstrate these facts, we give an useful property as follows.

Lemma 1. *For any chain triples $(x_1, m_1, y_1), \dots, (x_i, m_i, y_i)$ in \mathcal{T} , $y_i = \text{RO}(m_1 || \dots || m_i)$ holds unless bad events occur.*

Proof. To contrary, assume that $y_i \neq \text{RO}(m_1 || \dots || m_i)$. Since y_i is defined in step 2 of S (case A), step 2 of EO (case B), step 3 of EO (case C), or step 4 of EO (case D), we show that when y_i is defined in each step, bad events occur.

First, we discuss the case A. In this case, we divided two case: When (x_i, m_i, y_i) is stored, another chain triples $(x'_1, m'_1, y'_1), \dots, (x'_t, m'_t, y'_t)$ are already stored in \mathcal{T} such that $y_t = y_{i-1}$ (case A-1) and chain triples are not stored in \mathcal{T} (case A-2). The case A-1 is equal to collision of MD^S. Therefore a collision of S occurs or an output of S is equal to IV in this case. Therefore event E1 or E2 occurs. In the case A-2, since $y_i = \text{RO}(m_1 || \dots || m_i)$ holds from the definition of S, this is contrary to the assumption.

We discuss the case B. In this case, we divided two cases: $i = 1$ (case B-1) and $i \neq 1$ (case B-2). In the case B-1, $y_1 = \text{RO}(m_1)$ holds due to the definition of S. This is contrary to the assumption. In the case B-2, since $x_i = IV$, $y_{i-1} = IV$ holds. Therefore event E1 or E6 occurs.

We discuss the case C. In this case, (M, x_i) is already in \mathcal{L}_{RO} , when y_i is defined. We consider two cases: $M = m_1 || \dots || m_{i-1}$ (case C-1) and $M \neq m_1 || \dots || m_{i-1}$ (case C-2). In the case C-1, $y_i = \text{RO}(m_1 || \dots || m_i)$ holds and this is contrary to the assumption. In the case C-2, we consider two case: y_{i-1} is chosen at random by EO (case C-2-1) and y_{i-1} is defined by RO (case C-2-2). For the case C-2-1, from the definition of S, when $(x_{i-1}, m_{-i}, y_{i-1})$ is stored in \mathcal{T} , some triple (x_j, m_j, y_j) is not in \mathcal{T} . Assume that j is the maximum number. Therefore y_{j+1}, \dots, y_{i-1} are defined at random by EO and independent from RO. $(x_{j+1}, m_{j+1}, y_{j+1})$ is stored in \mathcal{T} before (x_j, m_j, y_j) is stored in \mathcal{T} . If y_j is defined at random by EO and independent from RO, event E3 occurs. If y_j is defined by RO ($y_i = \text{RO}(m_1 || \dots || m_j)$), event E5 occurs. The case C-2-2 is equal to event E4.

Finally we discuss the case D. From the same discussion of the case C-2-1, bad event E3 or E5 occurs. □

For the view of D for R_0 and R_1 , from Lemma 1, for any M , $R_1(M) = \text{RO}(M)$ holds unless bad events occur. Therefore the view of D for R_0 is equal to that for R_1 . For consistency in Game 2, from the definition of S and Lemma 1, for any chain triples $(x_1, m_1, y_1), \dots, (x_i, m_i, y_i) \in \mathcal{T}$, $y_i = \text{RO}(m_1 || \dots || m_i) = R_0(m_1 || \dots || m_i)$ holds unless bad events occur. Therefore, the answers given by S are consistent with those given by R_0 . For consistency in Game 3, from the definition of S, the definition of R_1 and Lemma 1, for any chain triples $(x_1, m_1, y_1), \dots, (x_i, m_i, y_i) \in \mathcal{T}$, $y_i = R_1(m_1 || \dots || m_i) = \text{RO}(m_1 || \dots || m_i)$ holds unless bad events occur. Therefore, the answers given by S are consistent with those given by R_1 . Therefore, Game 3 is identical with Game 2 unless bad events occur.

Next we examine the probability that bad events occur as follows.

Lemma 2. $Pr[E1 \vee E2 \vee E3 \vee E4 \vee E5 \vee E6] \leq \frac{2q_1^2 + q_2^2 + q_1 q_2 + q_1 + q_2}{2^n}$ where q_1 is the maximum number of invoking the simulator and q_2 is the maximum number of invoking RO.

Proof. We will examine each of the three events and bound their probability. Since outputs of S are chosen at random, $Pr[E1] \leq \frac{q_1}{2^n}$. Since $E2$ is the event where a collision occurs, $Pr[E2] \leq 1 - \frac{2^n-1}{2^n} \dots \frac{2^n-q_1+1}{2^n} \leq \frac{q_1^2}{2^n}$. Since y is chosen at random, the probability that event $E3 \leq \frac{q_1^2}{2^n}$. Since $E4$ is the event that a RO collision occurs, $Pr[E4] \leq \frac{q_2^2}{2^n}$. Since $E5$ is the event that a random value is equal to some fixed value, $Pr[E5] \leq \frac{q_1 q_2}{2^n}$. Since $E6$ is the event that a random value is equal to IV , $Pr[E6] \leq \frac{q_2}{2^n}$. Therefore $Pr[E1 \vee E2 \vee E3 \vee E4 \vee E5 \vee E6] \leq Pr[E1] + Pr[E2] + Pr[E3] + Pr[E4] + Pr[E5] + Pr[E6] \leq \frac{2q_1^2+q_2^2+q_1q_2+q_1+q_2}{2^n}$. \square

Let $G3$ denote the event that the distinguisher D outputs 1 in Game 3, $B2$ be the event wherein $E1 \vee E2 \vee E3 \vee E4 \vee E5 \vee E6$ occurs in Game 2 and $B3$ be the event wherein $E1 \vee E2 \vee E3 \vee E4 \vee E5 \vee E6$ occurs in Game 3. From Lemma 2, the probability that bad events occur in Game 2 is less than $\frac{\sigma_H^2+3q_h^2+3q_j\sigma_H+2q_h+\sigma_H}{2^n}$ and the probability that bad events occur in Game 3 is less than $\frac{4(\sigma_H+q_h)^2+2(\sigma_H+q_h)}{2^n}$. Therefore $|Pr[G3]-Pr[G2]| = |Pr[G3 \wedge B3]+Pr[G3 \wedge \neg B3]-Pr[G2 \wedge B2]-Pr[G2 \wedge \neg B2]| \leq |Pr[G3|B3] \times Pr[B3] - Pr[G2|B2] \times Pr[B2]| \leq \max\{Pr[B2], Pr[B3]\} = \frac{4(\sigma_H+q_h)^2+2(\sigma_H+q_h)}{2^n}$.

Game 4: In this Game, we modify simulator S to S_1 . RO is removed from simulator S_1 as follows.

Simulator S_1 : On query (x, m) ,

1. If $\exists(x, m, y) \in \mathcal{T}$, it responds with y .
2. Else S_1 chooses $y \leftarrow \{0, 1\}^n$ at random.
3. $\mathcal{T} \leftarrow (x, m, y)$.
4. S_1 responds with y .

The output of S is chosen at random or chosen by RO. Therefore, for any fresh query to S , the response is chosen at random. Since RO is invoked only by S , no D can access RO. Namely, no D distinguish S_1 from S , though RO is removed in S_1 , so Game 4 is identical to Game 3. Let $G4$ denote the event that distinguisher D outputs 1 in Game 4. $Pr[G4] = Pr[G3]$ holds.

Game 5. This is the final game of our argument. Here we finally replace S_1 with the fixed input length random oracle h . Let $G5$ denote the event that distinguisher D outputs 1 in Game 5. Since for a new query S_1 responds with a random value and for a repeated query S_1 responds a repeated value, Game 5 is identical to Game 4. Therefore, we can deduce that $Pr[G5] = Pr[G4]$.

Now we can complete the proof of Theorem by combining Games 1 to 5, and observing that Game 1 is the same as ERO scenario while Game 5 is same as MD^h scenario. Hence we can deduce that $\epsilon \leq \frac{4(\sigma_H+q_h)^2+2(\sigma_H+q_h)}{2^n}$. \square

4.2 Proof of Theorem 6

We define simulator S that simulates EO. S has initially empty list \mathcal{L}_S . On query (m, z) , S is defined as follows: $z' \leftarrow h(z, m)$, and it returns z' . The simulator's running time requires at most $O(q_{EO})$ time.

We need to prove that S cannot tell apart two scenarios, MD^h and ERO scenarios, one where D has oracle access to MD^h and S and the other where D has access to RO and EO. The proof involves a hybrid argument starting in the MD^h scenario, and ending in the ERO scenario through a sequence of mutually indistinguishable hybrid games.

Game 1: This is the MD^h scenario, where D has oracle access to MD^h and $S(h)$. Let $G1$ denote the event that D outputs 1 after interacting with MD^h and $S(h)$. Thus $Pr[G1] = Pr[D^{MD^h, S(h)} = 1]$.

Game 2: In this game, we change the underlying primitive of MD from h to S . Thus D interacts with MD^S and $S(h)$. For any query to S , S poses it to h and returns the value received from h . Let $G2$ denote the event that D outputs 1 in Game 2. Since the view of D remains unchanged in this game, so $Pr[G2] = Pr[G1]$.

Game 3: In this game, we remove S and h and insert EO and RO. In this game, D interacts with MD^{EO} and EO and does not access to RO. Since for a fresh query EO returns a fresh random value and for a repeated query EO returns the corresponding value, Game 3 is identical with Game 2. Let $G3$ denote the event that D outputs 1 in Game 3. Since the view of D remains unchanged in this game, so $Pr[G3] = Pr[G2]$.

Game 4. This is the final game of our argument. In this game, we remove MD^{EO} and D interacts with RO and EO. We show that Game 4 is identical with Game 3 unless following bad events occur and probability that bad events occur is negligible.

Bad events are as follows. On query (m, x) , EO returns y :

- Bad1: $y = IV$.

On query M , RO returns z :

- Bad2: There is a pair (M', z') in \mathcal{L}_{EO} , with $M \neq M'$, such that $z = z'$.
- Bad3: There is a triple (m, x, y) in \mathcal{L}_{EO} such that $z = x$.

We demonstrate that Game 4 is identical with Game 3 unless bad events occur and the probability that bad events occur is negligible. Before we demonstrate these facts, we give an useful property as follows.

Lemma 3. *For any chain triples $(x_1, m_1, y_1), \dots, (x_i, m_i, y_i)$ in \mathcal{L}_{EO} , $y_i = RO(m_1 || \dots || m_i)$ holds unless bad events occur.*

Due to lack of space, we omit this proof. We will show this in the full version.

For the view of D for MD^{EO} and RO, from Lemma 3, the view of D for MD^{EO} is equal to that for RO. For consistency in Game 3, from the definition of MD and Lemma 3, for any chain triples $(m_1, x_1, y_1), \dots, (m_i, x_i, y_i) \in \mathcal{L}_{EO}$, $y_i = RO(m_1 || \dots || m_i) = MD^{EO}(m_1 || \dots || m_i)$ holds unless bad events occur. Therefore,

the answers given by S are consistent with those given by MD^{EO} . For consistency in Game 4, from Lemma 3 for any chain triples $(x_1, m_1, y_1), \dots, (x_i, m_i, y_i) \in \mathcal{L}_{EO}$, $y_i = RO(m_1 || \dots || m_i)$ holds unless bad events occur. Therefore, the answers given by S are consistent with those given by RO . Therefore, Game 4 is identical with Game 3 unless bad events occur.

Next we examine the probability that bad events occur as follows.

Lemma 4. $Pr[\text{Bad1} \vee \text{Bad2} \vee \text{Bad3}] \leq \frac{q_1 + q_2^2 + q_1 q_2}{2^n}$ where q_1 is the maximum number of invoking EO and q_2 is the maximum number of invoking RO .

Due to lack of space we omit this proof.

Let $G4$ denote the event that the distinguisher D outputs 1 in Game 4, $B3$ be the event that $\text{Bad1} \vee \text{Bad2} \vee \text{Bad3}$ occurs in Game 3 and $B4$ be the event that $\text{Bad1} \vee \text{Bad2} \vee \text{Bad3}$ occurs in Game 4. Therefore $|Pr[G4] - Pr[G3]| \leq \max\{Pr[B3], Pr[B4]\} = \frac{2(\sigma_H + q_{EO})^2 + (\sigma_H + q_{EO})}{2^n}$. □

4.3 MGF1 Transform

In the above discussions, we ignored range extension algorithms such as $MGF1$ which is an instantiated hash function of $OAEP$. When we consider these algorithms, we need to modify TRO and ERO . Due to the lack of space, we only modify TRO for $MGF1$ as follows and will discuss ERO in the full paper.

Let $H : \{0, 1\}^* \rightarrow \{0, 1\}^n$ be some hash function and $MGF1 : \{0, 1\}^* \rightarrow \{0, 1\}^{jn}$ be $H(M || [1]) || H(M || [2]) || \dots || H(M || [j])$ where M is the input of the hash function and $[s]$ is the encoding value of s . We confirm the security of cryptosystems that use $MGF1$ transform with MD^h by the following approach. Let $MGF1 : \{0, 1\}^* \rightarrow \{0, 1\}^{jn}$.

- Propose the modification of TRO (denote TRO' that consists of random oracle $RO' : \{0, 1\}^* \rightarrow \{0, 1\}^{jn}$ and TO of RO') such that $MGF1(TRO) \sqsubset TRO'$.
- Prove cryptosystems security in TRO' model.

If we can find above TRO' , since $MD^h \sqsubset TRO$, cryptosystems that are secure in TRO' model are secure under MD^h .

TRO' is as follows. TRO' consists of random oracle $RO' : \{0, 1\}^* \rightarrow \{0, 1\}^{jn}$ and TO' , a variant of TO . Let $z[s]$ be the s -th block of z . On trace query (j, w) to TO' ,

- If there exist pairs such that $(M, z) \in \mathcal{L}_{RO}$ such that $z[j] = w$, TO' returns all such pairs.
- Otherwise, TO' returns \perp .

When H is a random oracle, we can see $H(* || [1]), \dots, H(* || [j])$ as independent random oracles RO_1, \dots, RO_j . In order to prove $MGF1(TRO) \sqsubset TRO'$, we need to find a simulator that simulates each TO of RO_1, \dots, RO_j . The simulator of TO of RO_s can be easily shown by using queries $(s, *)$ to TO' . Therefore, we can prove $MGF1(TRO) \sqsubset TRO'$.

Cryptosystems that are secure in the TRO model are also secure in the TRO' model by discussions similar to those for the cases of TRO. Note that security bound of these cryptosystems is dependent on n , not jn .

The same discussion can be applied to KDF3 which is an instantiated hash function of RSA-KEM [11].

5 Security Analysis of RSA-KEM in TRO and ERO Models

The RSA-based key encapsulation mechanism (RSA-KEM) scheme [11] is a secure KEM scheme in the RO model. In this section, we consider the security of RSA-KEM in the TRO and ERO models.

The notation of the scheme follows that in [11]. The security of RSA-KEM in the RO model is proved as follows;

Lemma 5 (Security of RSA-KEM in the RO model [11]). *If the RSA problem is hard, then RSA-KEM satisfies IND-CCA for KEM where KDF is modeled as RO.*

5.1 Insecurity of RSA-KEM in TRO Model

Though RSA-KEM is secure in the RO model, it is insecure in the TRO model. More specifically, we can show that RSA-KEM does not even satisfy IND-CPA for KEM in the TRO model. Note that IND-CPA means IND-CCA without $\mathcal{D}\mathcal{O}$.

Theorem 7 (Insecurity of RSA-KEM in the TRO model). *Even if the RSA problem is hard, RSA-KEM does not satisfy IND-CPA for KEM where KDF is modeled as TRO.*

Proof. We construct an adversary, \mathcal{A} , which successfully plays the IND-CPA by using TRO KDF. The construction of \mathcal{A} is as follows;

Input : (n, e) as the public key

Output : b' as the guessed bit

Step 1 : Return *state* and receive (K_b^*, C_0^*) as the challenge. Pose the trace query K_b^* to *KDF*, and obtain $\{r\}$.

Step 2 : For all r in $\{r\}$, check whether $r^e \stackrel{?}{\equiv} C_0^* \pmod{n}$. If there is r^* that satisfies the relation, output $b' = 0$. Otherwise, output $b' = 1$.

We estimate the success probability of \mathcal{A} . When challenge ciphertext C_0^* is generated, r^* such that $K_0^* = KDF(r^*)$ is certainly posed to *KDF* because C_0^* is generated following the protocol description. Thus, \mathcal{L}_{KDF} contains (r^*, C_0^*, K_0^*) . If (r^*, C_0^*, K_b^*) is not in \mathcal{L}_{KDF} , then $b = 1$. Therefore, \mathcal{A} can successfully play the IND-CPA game. \square

5.2 Security of RSA-KEM in ERO Model

We can also prove the security of RSA-KEM in the ERO model as well as in the RO model.

Theorem 8 (Security of RSA-KEM in the ERO model). *If the RSA problem is (t', ϵ') -hard, then RSA-KEM satisfies (t, ϵ) -IND-CCA for KEM as follows: $t' = t + (q_{RKDF} + q_{EKDF}) \cdot \text{expo}$, $\epsilon' \geq \epsilon - \frac{qd}{n}$, where KDF is modeled as ERO, q_{RKDF} is the number of hash queries posed to the RO of KDF, q_{EKDF} is the number of extension attack queries posed to the EO of KDF, q_D is the number of queries posed to the decryption oracle \mathcal{DO} and expo is the running time of exponentiation modulo n .*

The proof will be described in the full paper.

Acknowledgements. We would like to thank the anonymous referees for their many useful comments.

References

1. An, J.H., Bellare, M.: Constructing vil-macs from fil-macs: Message authentication under weakened assumptions. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 252–269. Springer, Heidelberg (1999)
2. Bellare, M., Ristenpart, T.: Multi-property-preserving hash domain extension and the EMD transform. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 299–314. Springer, Heidelberg (2006)
3. 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)
4. Bellare, M., Rogaway, P.: Optimal asymmetric encryption. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 92–111. Springer, Heidelberg (1995)
5. Coron, J.-S., Dodis, Y., Malinaud, C., Puniya, P.: Merkle-damgård revisited: How to construct a hash function. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 430–448. Springer, Heidelberg (2005)
6. Damgård, I.: A design principle for hash functions. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 416–427. Springer, Heidelberg (1990)
7. Dodis, Y., Ristenpart, T., Shrimpton, T.: Salvaging merkle-damgård for practical applications. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 371–388. Springer, Heidelberg (2009)
8. Hirose, S., Park, J.H., Yun, A.: A Simple Variant of the Merkle-Damgård Scheme with a Permutation. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 113–129. Springer, Heidelberg (2007)
9. 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)
10. Merkle, R.C.: One way hash functions and des. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 428–446. Springer, Heidelberg (1990)
11. Shoup, V.: A proposal for an iso standard for public key encryption, version 2.1 (2001)
12. Tsudik, G.: Message authentication with one-way hash functions. In: INFOCOM, pp. 2055–2059 (1992)
13. Yoneyama, K., Miyagawa, S., Ohta, K.: Leaky random oracle (extended abstract). In: Baek, J., Bao, F., Chen, K., Lai, X. (eds.) ProvSec 2008. LNCS, vol. 5324, pp. 226–240. Springer, Heidelberg (2008)

On the Analysis of Cryptographic Assumptions in the Generic Ring Model*

Tibor Jager and Jörg Schwenk

Horst Görtz Institute for IT Security
Ruhr-University Bochum, Germany

Abstract. At *Eurocrypt 2009* Aggarwal and Maurer proved that breaking RSA is equivalent to factoring in the *generic ring model*. This model captures algorithms that may exploit the full algebraic structure of the ring of integers modulo n , but no properties of the given representation of ring elements. This interesting result raises the question how to interpret proofs in the generic ring model. For instance, one may be tempted to deduce that a proof in the generic model gives some evidence that solving the considered problem is also hard in a general model of computation. But is this reasonable?

We prove that computing the *Jacobi symbol* is equivalent to factoring in the generic ring model. Since there are simple and efficient non-generic algorithms computing the Jacobi symbol, we show that the generic model cannot give any evidence towards the hardness of a computational problem. Despite this negative result, we also argue why proofs in the generic ring model are still interesting, and show that solving the *quadratic residuosity* and *subgroup decision* problems is generically equivalent to factoring.

1 Introduction

The security of asymmetric cryptographic systems relies on assumptions that certain computational problems, mostly from number theory and algebra, are intractable. Since proving useful lower complexity bounds in a general model of computation seems to be impossible with currently available techniques, these assumptions have been analyzed in restricted models, see [22,17,8,1], for instance. A natural and very general class of algorithms is considered in the *generic ring model*. This model captures all algorithms solving problems defined over an algebraic ring without exploiting specific properties of a given representation of ring elements. Such algorithms work in a similar way for arbitrary representations of ring elements, thus are *generic*.

Considering fundamental cryptographic problems in the generic model is motivated by the following ideas. First, showing that a cryptographic assumption

* This is an extended abstract, the full version is available on eprint [13]. Supported by the European Community (FP7/2007-2013), grant ICT-2007-216646 - European Network of Excellence in Cryptology II (ECRYPT II).

holds with respect to a restricted but meaningful class of algorithms might indicate that the idea of basing the security of cryptosystems on this assumption is not totally flawed, and may therefore be seen as evidence that the assumption is also valid in a general model of computation. Second, showing that a large class of algorithms is not able to solve a computational problem efficiently is an important insight for the search for cryptanalytic algorithms, and can be used to deduce the optimality of certain classes of algorithms. Moreover, the generic model is a valuable tool to study the relationship among computational problems, such as the equivalence of the discrete logarithm and the Diffie-Hellman problem, as done in [6,18,19,16,2], for instance.

In this paper we prove a general theorem which states that solving certain subset membership problems in the ring \mathbb{Z}_n is equivalent to factoring n . This main theorem allows us to provide an example for a computational problem with high cryptographic relevance which is easy to solve in general, but equivalent to factoring in the generic model. Concretely, we show that computing the *Jacobi symbol* is equivalent to factoring in the generic ring model.

For many common idealized models in cryptography it has been shown that a cryptographic reduction in the ideal model need not guarantee security in the “real world”. Well-known examples are, for instance, the random oracle model [9], the ideal cipher model [3], and the generic *group* model [12,11]. All these results have in common that they used somewhat contrived constructions that deviate from standard cryptographic practice [4]. In contrast, our result on the generic equivalence of computing the Jacobi symbol and factoring is an example for a truly practical computational problem that is provably hard in the generic model, but easy to solve in general. This is an important aspect for interpreting results in the generic ring model, like [7,8,15,2,1]. Thus a proof in the generic model is unfortunately not even an indicator that the considered problem is indeed useful for cryptographic applications.

This negative result does not affect the other mentioned motivations for the analysis of computational problems in the generic ring model. A lower bound in this model allows to deduce the optimality of certain classes of algorithms, and gives insight into the relationship between cryptographic problems, which is also of interest. Motivated by this fact, we also show that solving the *quadratic residuosity* and *subgroup decision* problems is generically equivalent to factoring. For the latter problem we show that the equivalence holds even in presence of a Diffie-Hellman oracle. Thus, a Diffie-Hellman oracle does not help in solving the subgroup decision problem.

By taking a closer look at the construction of the simulator used in the proof of our main theorem, we furthermore deduce that for a certain class of computational problems there exists an efficient generic ring algorithm if and only if there is an efficient straight line program solving the problem.

¹ An exception is the result of [20], showing a (non-generic) attack on a scheme with provable security in the generic model. However, [14] note that this stems not from a weakness in the generic model, but from an incorrect security proof.

1.1 Related Work

Previous work considering fundamental cryptographic assumptions in the generic model considered primarily discrete logarithm-based problems and the RSA problem. Starting with Shoup’s seminal paper [22], it was proven that solving the discrete logarithm problem, the Diffie-Hellman problem, and related problems [18,17,21] is hard with respect to generic *group* algorithms. Damgård and Koprowski showed the generic intractability of root extraction in groups of hidden order [10].

Brown [8] reduced the problem of factoring integers to solving the *low-exponent* RSA problem with *straight line programs*, which are a subclass of generic ring algorithms. Leander and Rupp [15] augmented this result to generic ring algorithms, where the considered algorithms may only perform the operations addition, subtraction and multiplication modulo n , but not multiplicative inversion operations. Recently, Aggarwal and Maurer [1] extended this result from low-exponent RSA to full RSA and to generic ring algorithms that may also compute multiplicative inverses. Boneh and Venkatesan [7] have shown that there is no straight line program reducing integer factorization to the low-exponent RSA problem, unless factoring integers is easy.

The notion of generic ring algorithms has also been applied to study the relationship between the discrete logarithm and the Diffie-Hellman problem and the existence of ring-homomorphic encryption schemes [6,16,2].

2 Preliminaries

2.1 Notation

For a set A and a probability distribution \mathcal{D} on A , we denote with $a \stackrel{\mathcal{D}}{\leftarrow} A$ the action of sampling an element a from A according to distribution \mathcal{D} . We denote with U the uniform distribution. When sampling k elements $a_1, \dots, a_k \stackrel{\mathcal{D}}{\leftarrow} A$, we assume that all elements are chosen independently.

Throughout the paper we let n be the product of at least two different primes, and denote with $n = \prod_{i=1}^k p_i^{e_i}$ the prime factor decomposition of n such that $\gcd(p_i^{e_i}, p_j^{e_j}) = 1$ for $i \neq j$.

Let $P = (S_1, \dots, S_m)$ be a finite sequence. Then $|P|$ denotes the length of P , i.e. $|P| = m$. For $k \leq m$ we denote with P_k the subsequence (S_1, \dots, S_k) of P . For a sequences P with we write $P_k \sqsubseteq P$ to denote that P_k is a subsequence of P such that P_k consists of the *first* $|P_k|$ elements of P .

2.2 Uniform Closure

By the Chinese Remainder Theorem, for $n = \prod_{i=1}^k p_i^{e_i}$ the ring \mathbb{Z}_n is isomorphic to the direct product of rings $\mathbb{Z}_{p_1^{e_1}} \times \dots \times \mathbb{Z}_{p_k^{e_k}}$. Let ϕ be the isomorphism $\mathbb{Z}_{p_1^{e_1}} \times \dots \times \mathbb{Z}_{p_k^{e_k}} \rightarrow \mathbb{Z}_n$, and for $\mathcal{C} \subseteq \mathbb{Z}_n$ let $\mathcal{C}_i := \{y \bmod p_i^{e_i} \mid y \in \mathcal{C}\}$ for $1 \leq i \leq k$.

Definition 1 (Uniform Closure). We say that $\mathcal{U}[\mathcal{C}] \subseteq \mathbb{Z}_n$ is the uniform closure of $\mathcal{C} \subseteq \mathbb{Z}_n$, if

$$\mathcal{U}[\mathcal{C}] = \{y \in \mathbb{Z}_n \mid y = \phi(y_1 \dots, y_k), y_i \in \mathcal{C}_i \text{ for } 1 \leq i \leq k\}.$$

In particular note that $\mathcal{C} \subseteq \mathcal{U}[\mathcal{C}]$, but not necessarily $\mathcal{U}[\mathcal{C}] \subseteq \mathcal{C}$. The following lemma follows directly from the above definition.

Lemma 1. Sampling $y \stackrel{U}{\leftarrow} \mathcal{U}[\mathcal{C}]$ uniformly random from $\mathcal{U}[\mathcal{C}]$ is equivalent to sampling y_i uniformly and independently from \mathcal{C}_i for $1 \leq i \leq k$ and setting $y = \phi(y_1, \dots, y_k)$.

2.3 Straight Line Programs

A straight line program over a ring R is a generic ring algorithm performing a fixed sequence of ring operations, without branching, that outputs an element of R . Thus straight line programs are a subclass of generic ring algorithms. The following definition is a simple extension of [S, Definition 1] to straight line programs that may also compute multiplicative inverses.

Definition 2 (Straight Line Programs). A straight line program P of length m over \mathbb{Z}_n is a sequence of tuples

$$P = ((i_1, j_1, \circ_1), \dots, (i_m, j_m, \circ_m))$$

where $-1 \leq i_k, j_k < k$ and $\circ_i \in \{+, -, \cdot, /\}$ for $i \in \{1, \dots, m\}$. The output $P(x)$ of straight line program P on input $x \in \mathbb{Z}_n$ is computed as follows.

1. Initialize $L_{-1} := 1 \in \mathbb{Z}_n$ and $L_0 := x$.
2. For k from 1 to m do:
 - if $\circ_k = /$ and $L_{j_k} \notin \mathbb{Z}_n^*$ then return \perp ,
 - else set $L_k := L_{i_k} \circ L_{j_k}$.
3. Return $P(x) = L_m$.

We say that each triple $(i, j, \circ) \in P$ is a SLP-step.

For notational convenience, for a given straight line program P we will denote with P_k the straight line program given by the sequence of the first k elements of P , with the additional convention that $P_{-1}(x) = 1$ and $P_0(x) = x$ for all $x \in \mathbb{Z}_n$.

2.4 Generic Ring Algorithms

Similar to straight line programs, generic ring algorithms perform a sequence of ring operations on the input values $1, x \in \mathbb{Z}_n$. However, while straight line programs perform the same fixed sequence on ring operations to any input value, generic ring algorithms can decide adaptively which ring operation is performed next. The decision is made either based on equality checks, or on coin tosses. Moreover, the output of generic ring algorithms is not restricted to ring elements.

We formalize the notion of generic ring algorithms in terms of a game between an algorithm \mathcal{A} and a black-box \mathcal{O} , the *generic ring oracle*. The generic ring oracle receives as input a secret value $x \in \mathbb{Z}_n$. It maintains a sequence P , which is set to the empty sequence at the beginning of the game, and implements two internal subroutines $\text{test}()$ and $\text{equal}()$.

- The $\text{test}()$ -procedure takes a tuple $(j, \circ) \in \{-1, \dots, |P|\} \times \{+, -, \cdot, /\}$ as input. The procedure returns **false** if $\circ = /$ and $P_j(x) \notin \mathbb{Z}_n^*$, and **true** otherwise.
- The $\text{equal}()$ -procedure takes a tuple $(i, j) \in \{-1, \dots, |P|\} \times \{-1, \dots, |P|\}$ as input. The procedure returns **true** if $P_i(x) \equiv P_j(x) \pmod n$ and **false** otherwise.

In order to perform computations, the algorithm submits SLP-steps to \mathcal{O} . Whenever the algorithm submits (i, j, \circ) with $\circ \in \{+, -, \cdot, /\}$, the oracle runs $\text{test}(j, \circ)$. If $\text{test}(j, \circ) = \text{false}$, the oracle returns the error symbol \perp . Otherwise (i, j, \circ) is appended to P . Moreover, the algorithm can query the oracle to check for equality of computed ring elements by submitting a query (i, j, \circ) such that $\circ \in \{=\}$. In this case the oracle returns $\text{equal}(i, j)$. We measure the complexity of \mathcal{A} by the number of oracle queries.

2.5 Some Lemmas on Straight Line Programs over \mathbb{Z}_n

In the following we will state a few lemmas on straight line programs over \mathbb{Z}_n that will be useful for the proof of our main theorem.

Lemma 2. *Suppose there exists a straight line program P such that for $x, x' \in \mathbb{Z}_n$ holds that $P(x') \neq \perp$ and $P(x) = \perp$. Then there exists $P_j \subseteq P$ such that $P_j(x') \in \mathbb{Z}_n^*$ and $P_j(x) \notin \mathbb{Z}_n^*$.*

Proof. $P(x) = \perp$ means that there exists an SLP-step $(i, j, \circ) \in P$ such that $\circ = /$ and $L_j = P_j(x) \notin \mathbb{Z}_n^*$. However, $P(x')$ does not evaluate to \perp , thus it must hold that $P_j(x') \in \mathbb{Z}_n^*$.

The following lemma provides a lower bound on the probability of factoring n by evaluating a certain straight line program P with $y \xleftarrow{\mathcal{U}} \mathcal{U}[\mathcal{C}]$ and computing $\text{gcd}(n, P(y))$, relative to the probability that $P(x') \notin \mathbb{Z}_n^*$ and $P(x) \in \mathbb{Z}_n^*$ for randomly chosen $x, x' \xleftarrow{\mathcal{U}} \mathcal{C}$.

Lemma 3. *For any straight line program P and $\mathcal{C} \subseteq \mathbb{Z}_n$ holds that*

$$\begin{aligned} & \Pr \left[P(x') \notin \mathbb{Z}_n^* \text{ and } P(x) \in \mathbb{Z}_n^* \mid x, x' \xleftarrow{\mathcal{U}} \mathcal{C} \right] \\ & \leq \left(\frac{|\mathcal{U}[\mathcal{C}]|}{|\mathcal{C}|} \right)^2 \Pr \left[\text{gcd}(n, P(y)) \notin \{1, n\} \mid y \xleftarrow{\mathcal{U}} \mathcal{U}[\mathcal{C}] \right]. \end{aligned}$$

Similar to the above, the following lemma provides a lower bound on the probability of factoring n by computing $\text{gcd}(n, P(y) - Q(y))$ with $y \xleftarrow{\mathcal{U}} \mathcal{U}[\mathcal{C}]$ for two given straight line programs P and Q , relative to the probability $\Pr[(P(x) \equiv_n Q(x) \text{ and } P(x') \not\equiv_n Q(x')) \mid x, x' \xleftarrow{\mathcal{U}} \mathcal{C}]$.

Lemma 4. *For any pair (P, Q) of straight line programs and $\mathcal{C} \subseteq \mathbb{Z}_n$ holds that*

$$\begin{aligned} & \Pr \left[P(x) \equiv_n Q(x) \text{ and } P(x') \not\equiv_n Q(x') \mid x, x' \stackrel{\mathcal{U}}{\leftarrow} \mathcal{C} \right] \\ & \leq \left(\frac{|\mathcal{U}[\mathcal{C}]|}{|\mathcal{C}|} \right)^2 \Pr \left[\gcd(n, P(y) - Q(y)) \notin \{1, n\} \mid y \stackrel{\mathcal{U}}{\leftarrow} \mathcal{U}[\mathcal{C}] \right]. \end{aligned}$$

The proofs of Lemma 3 and 4 are based on the Chinese Remainder Theorem. Full proofs are given in Appendix C and D of the full version [13]. We also discuss the intuition behind these lemmas in Appendix E of [13].

3 Subset Membership Problems in Generic Rings

Definition 3 (Subset Membership Problem). *Let $\mathcal{C} \subseteq \mathbb{Z}_n$ and $\mathcal{V} \subseteq \mathbb{Z}_n$ be subsets of \mathbb{Z}_n such that $\mathcal{V} \subseteq \mathcal{C} \subseteq \mathbb{Z}_n$. The subset membership problem defined by $(\mathcal{C}, \mathcal{V})$ is: given $x \stackrel{\mathcal{U}}{\leftarrow} \mathcal{C}$, decide whether $x \in \mathcal{V}$.*

Whenever considering a subset membership problem in the following we assume that $|\mathcal{V}| > 1$.

Let $(\mathcal{C}, \mathcal{V})$ be subsets of \mathbb{Z}_n defining a subset membership problem. We formalize the notion of subset membership problems in the generic ring model in terms of a game between an algorithm \mathcal{A} and a generic ring oracle \mathcal{O}_{smp} . Oracle \mathcal{O}_{smp} is defined exactly like the generic ring oracle described in Section 2.4, except that \mathcal{O}_{smp} receives a uniformly random element $x \stackrel{\mathcal{U}}{\leftarrow} \mathcal{C}$ as input. We say that \mathcal{A} wins the game, if $x \in \mathcal{V}$ and $\mathcal{A}^{\mathcal{O}_{\text{smp}}}(n) = 1$, or $x \notin \mathcal{V}$ and $\mathcal{A}^{\mathcal{O}_{\text{smp}}}(n) = 0$.

Note that any algorithm for a given subset membership problem $(\mathcal{C}, \mathcal{V})$ has at least the trivial success probability $\Pi(\mathcal{C}, \mathcal{V}) := \max\{|\mathcal{V}|/|\mathcal{C}|, 1 - |\mathcal{V}|/|\mathcal{C}|\}$ by guessing, due to the fact that x is sampled uniformly from \mathcal{C} . For an algorithm solving the subset membership problem given by $(\mathcal{C}, \mathcal{V})$ with success probability $\Pr[\mathcal{S}]$, we denote with

$$\text{Adv}_{(\mathcal{C}, \mathcal{V})}(\mathcal{A}^{\mathcal{O}_{\text{smp}}}(n)) := |\Pr[\mathcal{S}] - \Pi(\mathcal{C}, \mathcal{V})|$$

the *advantage* of \mathcal{A} .

Theorem 1. *For any generic ring algorithm \mathcal{A} solving a given subset membership problem $(\mathcal{C}, \mathcal{V})$ over \mathbb{Z}_n with advantage $\text{Adv}_{(\mathcal{C}, \mathcal{V})}(\mathcal{A}^{\mathcal{O}_{\text{smp}}}(n))$ by performing m queries to \mathcal{O}_{smp} , there exists an algorithm \mathcal{B} that outputs a factor of n with success probability at least*

$$\frac{\text{Adv}_{(\mathcal{C}, \mathcal{V})}(\mathcal{A}^{\mathcal{O}_{\text{smp}}}(n))}{2m(m^2 + 5m + 3)} \cdot \left(\frac{|\mathcal{C}|}{|\mathcal{U}[\mathcal{C}]|} \right)^2$$

by running \mathcal{A} once and performing $O(m^3)$ additional operations in \mathbb{Z}_n , m gcd-computations on $\lceil \log_2 n \rceil$ -bit numbers, and sampling m random elements from $\mathcal{U}[\mathcal{C}]$.

Proof Outline. We replace \mathcal{O}_{smp} with a simulator \mathcal{O}_{sim} . Let \mathcal{S}_{sim} denote the event that \mathcal{A} is successful when interacting with the simulator, and let \mathcal{F} denote the event that \mathcal{O}_{sim} answers a query of \mathcal{A} different from how \mathcal{O}_{smp} would have answered. Then \mathcal{O}_{smp} and \mathcal{O}_{sim} are indistinguishable unless \mathcal{F} occurs. Therefore the success probability $\Pr[\mathcal{S}]$ of \mathcal{A} in the simulation game is upper bound by $\Pr[\mathcal{S}_{\text{sim}}] + \Pr[\mathcal{F}]$. We derive a bound on $\Pr[\mathcal{S}_{\text{sim}}]$ and describe a factoring algorithm whose success probability is lower bound by $\Pr[\mathcal{F}]$.

3.1 Introducing a Simulation Oracle

We replace oracle \mathcal{O}_{smp} with a simulator \mathcal{O}_{sim} . \mathcal{O}_{sim} receives $x \stackrel{U}{\leftarrow} \mathcal{C}$ as input, but never uses this value throughout the game. Instead, all computations are performed *independent* of the challenge value x . Note that the original oracle \mathcal{O}_{smp} uses x only inside the `test()` and `equal()` procedures. Let us therefore consider an oracle \mathcal{O}_{sim} which is defined exactly like \mathcal{O}_{smp} , but replaces the procedures `test()` and `equal()` with procedures `testsim()` and `equalsim()`.

- The `testsim()`-procedure samples $x_r \stackrel{U}{\leftarrow} \mathcal{C}$ and returns `false` if $\circ = /$ and $P_j(x_r) \notin \mathbb{Z}_n^*$, and `true` otherwise (even if $P_j(x_r) = \perp$).
- The `equalsim()`-procedure samples $x_r \stackrel{U}{\leftarrow} \mathcal{C}$ and returns `true` if $P_i(x_r) \equiv P_j(x_r) \pmod n$ and `false` otherwise (even if $P_i(x_r) = \perp$ or $P_j(x_r) = \perp$).

Note that the simulator samples m random values $x_r, r \in \{1, \dots, m\}$. Also note that all computations of \mathcal{A} are independent of the challenge value x when interacting with \mathcal{O}_{sim} . Hence, any algorithm \mathcal{A} has at most trivial success probability in the simulation game, and therefore

$$\Pr[\mathcal{S}_{\text{sim}}] \leq \Pi(\mathcal{C}, \mathcal{V}).$$

3.2 Bounding the Probability of Simulation Failure

We say that a *simulation failure*, denoted \mathcal{F} , occurs if \mathcal{O}_{sim} does not simulate \mathcal{O}_{smp} perfectly. Observe that an interaction of \mathcal{A} with \mathcal{O}_{sim} is perfectly indistinguishable from an interaction with \mathcal{O}_{smp} , unless at least one of the following events occurs.

1. The `testsim()`-procedure fails to simulate `test()` perfectly. This means that `testsim()` returns `false` on a procedure call where `test()` would have returned `true`, or `testsim()` returns `true` where `test()` would have returned `false`. Let $\mathcal{F}_{\text{test}}$ denote the event that this happens on at least one call of `testsim()`.
2. The `equalsim()`-procedure fails to simulate `equal()` perfectly. This means that `equalsim()` has returned `true` where `equal()` would have returned `false`, or `equalsim()` has returned `false` where `equal()` would have returned `true`. Let $\mathcal{F}_{\text{equal}}$ denote the event that this happens at at least one call of `equalsim()`.

Since \mathcal{F} implies that at least one of the events $\mathcal{F}_{\text{test}}$ and $\mathcal{F}_{\text{equal}}$ has occurred, it holds that

$$\Pr[\mathcal{F}] \leq \Pr[\mathcal{F}_{\text{test}}] + \Pr[\mathcal{F}_{\text{equal}}].$$

In the following we will bound $\Pr[\mathcal{F}_{\text{test}}]$ and $\Pr[\mathcal{F}_{\text{equal}}]$ separately.

Bounding the Probability of $\mathcal{F}_{\text{test}}$. The `testsim()`-procedure fails to simulate `test()` only if either `testsim()` has returned false where `test()` would have returned true, or `testsim()` has returned true where `test()` would have returned false. A necessary condition² for this is that there exists $P_j \sqsubseteq P$ and $x_r \in \{x_1, \dots, x_m\}$ such that

$$(P_j(x) \in \mathbb{Z}_n^* \text{ and } P_j(x_r) \notin \mathbb{Z}_n^*) \text{ or } (P_j(x) = \perp \text{ and } P_j(x_r) \notin \mathbb{Z}_n^*),$$

or

$$(P_j(x_r) \in \mathbb{Z}_n^* \text{ and } P_j(x) \notin \mathbb{Z}_n^*) \text{ or } (P_j(x_r) = \perp \text{ and } P_j(x) \notin \mathbb{Z}_n^*).$$

We can simplify this condition a little by applying Lemma 2. The existence of $P_j \sqsubseteq P$ and x_r such that $(P_j(x_r) = \perp \text{ and } P_j(x) \notin \mathbb{Z}_n^*)$ implies the existence of $P_k \sqsubseteq P$ such that $k < j$ and $(P_k(x_r) \notin \mathbb{Z}_n^* \text{ and } P_k(x) \in \mathbb{Z}_n^*)$. An analogous argument holds for the case $(P_j(x) = \perp \text{ and } P_j(x_r) \notin \mathbb{Z}_n^*)$. Hence, `testsim()`-procedure fails to simulate `test()` only if there exists $P_j \sqsubseteq P$ such that

$$(P_j(x) \in \mathbb{Z}_n^* \text{ and } P_j(x_r) \notin \mathbb{Z}_n^*) \text{ or } (P_j(x_r) \in \mathbb{Z}_n^* \text{ and } P_j(x) \notin \mathbb{Z}_n^*).$$

Proposition 1

$$\Pr[\mathcal{F}_{\text{test}}] \leq 2m(m+2) \max_{0 \leq j \leq m} \left\{ \Pr \left[P_j(x) \notin \mathbb{Z}_n^* \text{ and } P_j(x') \in \mathbb{Z}_n^* \mid x, x' \stackrel{U}{\leftarrow} \mathcal{C} \right] \right\}$$

We sketch the proof of Proposition 1 in Appendix B. A full proof is given in Appendix F of the full version.

Bounding the Probability of $\mathcal{F}_{\text{equal}}$. The `equalsim()`-procedure fails to simulate `equal()` only if either `equalsim()` has returned false where `equal()` would have returned true, or `equalsim()` has returned true where `equal()` would have returned false. A necessary³ condition for this is that there exist $P_i, P_j \sqsubseteq P$ and $x_r \in \{x_1, \dots, x_m\}$ such that

$$\begin{aligned} & (P_i(x) \equiv_n P_j(x) \text{ and } P_i(x_r) \not\equiv_n P_j(x_r)) \\ \text{or } & (P_i(x) \equiv_n P_j(x) \text{ and } (P_i(x_r) = \perp \text{ or } P_j(x_r) = \perp)) \\ \text{or } & (P_i(x_r) \equiv_n P_j(x_r) \text{ and } P_i(x) \not\equiv_n P_j(x)) \\ \text{or } & (P_i(x_r) \equiv_n P_j(x_r) \text{ and } (P_i(x) = \perp \text{ or } P_j(x) = \perp)). \end{aligned}$$

Again we can apply Lemma 2 to simplify this a little: the existence of $P_j \in P$ and x_r such that $(P_j(x_r) = \perp \text{ and } P_j(x) \neq \perp)$ implies the existence of $P_k \in P$ such that $(P_k(x_r) \notin \mathbb{Z}_n^* \text{ and } P_k(x) \in \mathbb{Z}_n^*)$. Analogous arguments hold for the

² The condition is not sufficient, because algorithm \mathcal{A} need not have queried a division by P_j in its r -th query.

³ The condition is not sufficient, because algorithm \mathcal{A} need not have queried $(i, j, =)$ in its r -th query.

other cases where one straight line program evaluates to \perp . Hence, `equal()`-procedure fails to simulate `equal()` only if there exist $P_i, P_j \sqsubseteq P$ or $P_k \sqsubseteq P$ such that

$$\begin{aligned} & (P_i(x) \equiv_n P_j(x) \text{ and } P_i(x_r) \not\equiv_n P_j(x_r)) \\ \text{or } & (P_i(x_r) \equiv_n P_j(x_r) \text{ and } P_i(x) \not\equiv_n P_j(x)) \\ \text{or } & (P_k(x_r) \notin \mathbb{Z}_n^* \text{ and } P_k(x) \in \mathbb{Z}_n^*) \\ \text{or } & (P_k(x) \notin \mathbb{Z}_n^* \text{ and } P_k(x_r) \in \mathbb{Z}_n^*). \end{aligned}$$

Proposition 2

$$\Pr[\mathcal{F}_{\text{equal}}] \leq 2m(m^2 + 3m + 1)\Phi + 2m(m + 1)\Psi,$$

where

$$\begin{aligned} \Phi &= \max_{-1 \leq i < j \leq m} \left\{ \Pr \left[P_i(x) \equiv_n P_j(x) \text{ and } P_i(x') \not\equiv_n P_j(x') \mid x, x' \stackrel{U}{\leftarrow} \mathcal{C} \right] \right\} \\ \Psi &= \max_{0 \leq k \leq m} \left\{ \Pr \left[P_k(x) \notin \mathbb{Z}_n^* \text{ and } P_k(x') \in \mathbb{Z}_n^* \mid x, x' \stackrel{U}{\leftarrow} \mathcal{C} \right] \right\}. \end{aligned}$$

The proof of Proposition 2, which is based on the same ideas as the proof of Proposition 1, is given in Appendix G of the full version.

Bounding the Probability of \mathcal{F} . Summing up, we obtain that the total probability of \mathcal{F} is at most

$$\begin{aligned} \Pr[\mathcal{F}] &\leq \Pr[\mathcal{F}_{\text{test}}] + \Pr[\mathcal{F}_{\text{equal}}] \\ &\leq 2m(m^2 + 3m + 1)\Phi + 4m(m + 1)\Psi. \end{aligned}$$

where Φ and Ψ are defined as above.

3.3 Bounding the Success Probability

Since all computations of \mathcal{A} are independent of the challenge value x in the simulation game, any algorithm has only the trivial success probability when interacting with the simulator. Thus the success probability of any algorithm when interacting with the original oracle is bound by

$$\Pi(\mathcal{C}, \mathcal{V}) + \text{Adv}_{(\mathcal{C}, \mathcal{V})}(\mathcal{A}^{\mathcal{O}_{\text{smp}}}) = \Pr[\mathcal{S}] \leq \Pr[\mathcal{S}_{\text{sim}}] + \Pr[\mathcal{F}] \leq \Pi(\mathcal{C}, \mathcal{V}) + \Pr[\mathcal{F}],$$

which implies

$$\text{Adv}_{(\mathcal{C}, \mathcal{V})}(\mathcal{A}^{\mathcal{O}_{\text{smp}}}) \leq \Pr[\mathcal{F}].$$

3.4 The Factoring Algorithm

Consider a factoring algorithm \mathcal{B} running \mathcal{A} , recording the sequence of queries \mathcal{A} issues, and proceeding as follows.

- Whenever the algorithm submits (i, j, \circ) with $\circ \in \{+, -, \cdot, /\}$ in its r -th query, the algorithm samples $y \stackrel{\mathcal{U}}{\leftarrow} \mathcal{U}[\mathcal{C}]$ and computes $\gcd(P_k(y), n)$ for $0 \leq k \leq r$.
- Whenever the algorithm submits (i, j, \circ) with $\circ \in \{=\}$ in its r -th query, the algorithm samples $y \stackrel{\mathcal{U}}{\leftarrow} \mathcal{U}[\mathcal{C}]$ and computes $\gcd(P_i(y) - P_j(y), n)$ for $-1 \leq i < j \leq r$.

Running time. By assumption, \mathcal{A} submits m queries. Thus, the algorithm evaluates $O(m^2)$ straight line programs. Each query can be evaluated by performing at most m steps, which yields $O(m^3)$ operations in \mathbb{Z}_n . Moreover, the algorithm samples m random values y from $\mathcal{U}[\mathcal{C}]$ and performs m gcd-computations on $\lceil \log_2 n \rceil$ -bit numbers.

Success probability. \mathcal{B} evaluates any straight line program P_k with a uniformly random element y of $\mathcal{U}[\mathcal{C}]$. In particular, \mathcal{B} computes $\gcd(P_k(y), n)$ for $y \stackrel{\mathcal{U}}{\leftarrow} \mathcal{U}[\mathcal{C}]$ and the straight line program $P_k \sqsubseteq P$ satisfying

$$\begin{aligned} & \Pr \left[P_k(x) \notin \mathbb{Z}_n^* \text{ and } P_k(x') \in \mathbb{Z}_n^* \mid x, x' \stackrel{\mathcal{U}}{\leftarrow} \mathcal{C} \right] \\ &= \max_{0 \leq k \leq m} \left\{ \Pr \left[P_k(x) \notin \mathbb{Z}_n^* \text{ and } P_k(x') \in \mathbb{Z}_n^* \mid x, x' \stackrel{\mathcal{U}}{\leftarrow} \mathcal{C} \right] \right\}. \end{aligned}$$

Let $\gamma_1 := \max_{0 \leq k \leq m} \{ \Pr [P_k(x) \notin \mathbb{Z}_n^* \text{ and } P_k(x') \in \mathbb{Z}_n^* \mid x, x' \stackrel{\mathcal{U}}{\leftarrow} \mathcal{C}] \}$, then by Lemma 3 algorithm \mathcal{B} finds a factor in this step with probability at least $\gamma_1 \left(\frac{|\mathcal{C}|}{|\mathcal{U}[\mathcal{C}]|} \right)^2$.

Moreover, \mathcal{B} evaluates any pair P_i, P_j of straight line programs in P with a uniformly random element $y \stackrel{\mathcal{U}}{\leftarrow} \mathcal{U}[\mathcal{C}]$. So in particular \mathcal{B} evaluates $\gcd(P_i(y) - P_j(y), n)$ with $y \stackrel{\mathcal{U}}{\leftarrow} \mathcal{U}[\mathcal{C}]$ for the pair of straight line programs $P_i, P_j \sqsubseteq P$ satisfying

$$\begin{aligned} & \Pr \left[P_i(x) \equiv_n P_j(x) \text{ and } P_i(x') \not\equiv_n P_j(x') \mid x, x' \stackrel{\mathcal{U}}{\leftarrow} \mathcal{C} \right] \\ &= \max_{-1 \leq i < j \leq m} \left\{ \Pr \left[P_i(x) \equiv_n P_j(x) \text{ and } P_i(x') \not\equiv_n P_j(x') \mid x, x' \stackrel{\mathcal{U}}{\leftarrow} \mathcal{C} \right] \right\}. \end{aligned}$$

Let $\gamma_2 := \max_{-1 \leq i < j \leq m} \{ \Pr [P_i(x) \equiv_n P_j(x) \text{ and } P_i(x') \not\equiv_n P_j(x') \mid x, x' \stackrel{\mathcal{U}}{\leftarrow} \mathcal{C}] \}$, then by Lemma 4 algorithm \mathcal{B} succeeds in this step with probability at least $\gamma_2 \left(\frac{|\mathcal{C}|}{|\mathcal{U}[\mathcal{C}]|} \right)^2$. So, for $\gamma := \max\{\gamma_1, \gamma_2\}$, the total success probability of algorithm \mathcal{B} is at least

$$\gamma \left(\frac{|\mathcal{C}|}{|\mathcal{U}[\mathcal{C}]|} \right)^2.$$

Relating the success probability of \mathcal{B} to the advantage of \mathcal{A} . Using the above definitions of γ_1, γ_2 , and γ , the fact that $\text{Adv}_{(\mathcal{C}, \mathcal{V})}(\mathcal{A}^{\text{O}_{\text{smp}}}(n)) \leq \Pr[\mathcal{F}]$, and the derived bound on $\Pr[\mathcal{F}]$, we can obtain a lower bound on γ by

$$\begin{aligned} \text{Adv}_{(\mathcal{C}, \mathcal{V})}(\mathcal{A}^{\mathcal{O}_{\text{smp}}}(n)) &\leq \Pr[\mathcal{F}] \leq 4m(m+1)\gamma_1 + 2m(m^2 + 3m + 1)\gamma_2 \\ &\leq 2m(m^2 + 5m + 3)\gamma, \end{aligned}$$

which implies the inequality

$$\gamma \geq \frac{\text{Adv}_{(\mathcal{C}, \mathcal{V})}(\mathcal{A}^{\mathcal{O}_{\text{smp}}}(n))}{2m(m^2 + 5m + 3)}.$$

Therefore the success probability of \mathcal{B} is at least

$$\frac{\text{Adv}_{(\mathcal{C}, \mathcal{V})}(\mathcal{A}^{\mathcal{O}_{\text{smp}}}(n))}{2m(m^2 + 5m + 3)} \cdot \left(\frac{|\mathcal{C}|}{|\mathcal{U}[\mathcal{C}]|}\right)^2.$$

4 Computing the Jacobi Symbol with Generic Ring Algorithms

Let us denote with $\text{QR}_n \subseteq \mathbb{Z}_n$ the set of *quadratic residues* modulo n , i.e.

$$\text{QR}_n := \{x \in \mathbb{Z}_n^* \mid x \equiv y^2 \pmod n, y \in \mathbb{Z}_n^*\}.$$

Let $(x \mid n)$ denote the *Jacobi symbol* [23, p.287] and let $J_n := \{x \in \mathbb{Z}_n \mid (x \mid n) = 1\}$ be the set of elements of \mathbb{Z}_n having Jacobi symbol 1. Recall that $\text{QR}_n \subseteq J_n$, and therefore given $x \in \mathbb{Z}_n \setminus J_n$ it is easy to decide that x is not a quadratic residue by computing the Jacobi symbol.

There exist simple efficient algorithms computing the Jacobi symbol in \mathbb{Z}_n without factoring n . These algorithms are not generic, cf. [23, p.288].

Theorem 2. *Suppose there exist a generic ring algorithm \mathcal{A} solving the subset membership problem given by $(\mathcal{C}, \mathcal{V})$ with $\mathcal{C} = \mathbb{Z}_n^*$ and $\mathcal{V} = J_n$ with advantage $\text{Adv}_{(\mathcal{C}, \mathcal{V})}(\mathcal{A}^{\mathcal{O}_{\text{smp}}}(n))$ by performing m ring operations. Then there exists an algorithm \mathcal{B} finding a factor of n with probability at least*

$$\frac{\text{Adv}_{(\mathcal{C}, \mathcal{V})}(\mathcal{A}^{\mathcal{O}_{\text{smp}}}(n))}{2m(m^2 + 5m + 3)}$$

by running \mathcal{A} once and performing $O(m^3)$ additional operations in \mathbb{Z}_n , m gcd-computations on $\lceil \log_2 n \rceil$ -bit numbers, and sampling m random elements from \mathbb{Z}_n^* .

Proof. The theorem follows by applying Theorem 1 and the fact that $\mathcal{U}[\mathbb{Z}_n^*] = \mathbb{Z}_n^*$, since

$$\left(\frac{|\mathcal{C}|}{|\mathcal{U}[\mathcal{C}]|}\right)^2 = \left(\frac{|\mathbb{Z}_n^*|}{|\mathbb{Z}_n^*|}\right)^2 = 1$$

5 The Generic Quadratic Residuosity Problem and Factoring

Definition 4 (Quadratic Residuosity Problem). *The quadratic residuosity problem is the subset membership problem given by $\mathcal{C} = J_n$ and $\mathcal{V} = \text{QR}_n$.*

Given the factorization of n , solving the quadratic residuosity problem in \mathbb{Z}_n is easy, also for generic ring algorithms. Thus, in order to show the equivalence of generic quadratic residuosity and factoring, we have to prove the following theorem.

Theorem 3. *Suppose there exist a generic ring algorithm \mathcal{A} that solves the quadratic residuosity problem in \mathbb{Z}_n with advantage $\text{Adv}_{(\mathcal{C}, \mathcal{V})}(\mathcal{A}^{\mathcal{O}_{\text{smp}}}(n))$ by performing m ring operations. Then there exists an algorithm \mathcal{B} finding a factor of n with probability at least*

$$\frac{\text{Adv}_{(\mathcal{C}, \mathcal{V})}(\mathcal{A}^{\mathcal{O}_{\text{smp}}}(n))}{8m(m^2 + 5m + 3)}$$

by running \mathcal{A} once and performing $O(m^3)$ additional operations in \mathbb{Z}_n , m gcd-computations on $\lceil \log_2 n \rceil$ -bit numbers, and sampling m random elements from \mathbb{Z}_n^* .

Proof. The cardinality $|J_n|$ of the set of elements having Jacobi symbol 1 depends on whether n is a square in \mathbb{N} .

$$|J_n| = \begin{cases} \phi(n)/2, & \text{if } n \text{ is not a square in } \mathbb{N}, \\ \phi(n), & \text{if } n \text{ is a square in } \mathbb{N}, \end{cases}$$

where $\phi(\cdot)$ is the Euler totient function [23, p.24]. Note also that $\mathcal{U}[J_n] = \mathcal{U}[\mathcal{C}] = \mathbb{Z}_n^*$. Therefore it holds that $|J_n| = |\mathcal{C}| \geq \phi(n)/2$ and $|\mathcal{U}[\mathcal{C}]| = |\mathbb{Z}_n^*| = \phi(n)$. Thus we can apply Theorem 1, using that

$$\left(\frac{|\mathcal{C}|}{|\mathcal{U}[\mathcal{C}]|}\right)^2 = \left(\frac{|J_n|}{|\mathbb{Z}_n^*|}\right)^2 \geq \left(\frac{\phi(n)/2}{\phi(n)}\right)^2 = \frac{1}{4}.$$

6 The Generic Subgroup Decision Problem and Factoring

Let $n = pq$ and let \mathbb{G} be a cyclic group of order n . Then there exists a subgroup $\mathbb{G}_p \subseteq \mathbb{G}$ of order p .

Definition 5 (Subgroup Decision Problem). *The subgroup decision problem is the subset membership problem $(\mathcal{C}, \mathcal{V})$ with $\mathcal{C} = \mathbb{G}$ and $\mathcal{V} = \mathbb{G}_p$.*

Recall that any cyclic group of order n is isomorphic to the additive group of integers $(\mathbb{Z}_n, +)$. Now, since we are going to consider *generic* algorithms, we may assume that the algorithm operates on the group $\mathbb{G} = (\mathbb{Z}_n, +)$, of course without exploiting any property of this representation [4]. Assuming an oracle *DH* solving

⁴ One may equivalently assume that the *generic group oracle* uses the group $(\mathbb{Z}_n, +)$ for the *internal* representation of group elements.

the Diffie-Hellman problem in \mathbb{G} , we observe that this operation corresponds to the *multiplication* in \mathbb{Z}_n . Hence, the group \mathbb{G} together with oracle DH exhibits the same algebraic structure as the *ring* \mathbb{Z}_n .

By the Chinese Remainder Theorem, the ring \mathbb{Z}_n is isomorphic to the direct product $\mathbb{Z}_p \times \mathbb{Z}_q$. Let $\phi : \mathbb{Z}_p \times \mathbb{Z}_q \rightarrow \mathbb{Z}_n$ denote this isomorphism. The subgroup \mathbb{G}_p of \mathbb{G} with order p consists of the elements $\mathbb{G}_p = \{\phi(x_p, 0) \mid x_p \in \mathbb{Z}_p\}$. So for generic ring algorithms the subgroup decision problem can be stated as: given $x \in \mathbb{Z}_n$, decide whether $x \equiv 0 \pmod q$.

In order to model the generic subgroup decision problem, consider an oracle \mathcal{O}_{sdp} which is defined exactly like the generic ring oracle described in Section 2.4, except that it does not provide the operation $/$. \mathcal{O}_{sdp} receives an element $x \in \mathbb{Z}_n$ as input, where x is constructed as follows: sample $(x_p, x_q) \xleftarrow{U} \mathbb{Z}_p \times \mathbb{Z}_q$ and bit $b \xleftarrow{U} \{0, 1\}$ uniformly random, and let $x := \phi(x_p, bx_q)$. An algorithm can query the oracle for the (inverse) group operation by submitting a query (i, j, \circ) with $\circ \in \{+, -\}$. The Diffie-Hellman oracle is queried by submitting (i, j, \circ) with $\circ \in \{\cdot\}$.

We say that the algorithm wins the game, if $x \in \mathbb{G}_p$ and $\mathcal{A}^{\mathcal{O}_{\text{sdp}}}(n) = 1$, or $x \notin \mathbb{G}_p$ and $\mathcal{A}^{\mathcal{O}_{\text{sdp}}}(n) = 0$. We define the *advantage* of an algorithm \mathcal{A} solving the subgroup decision problem with probability $\Pr[\mathcal{S}]$ as

$$\text{Adv}(\mathcal{A}^{\mathcal{O}_{\text{sdp}}}(n)) := \left| \Pr[\mathcal{S}] - \left(\frac{1}{2} + \frac{1}{q} \right) \right|.$$

Remark 1. If we would also allow to query the oracle for divisions (which correspond to an “inverse Diffie-Hellman oracle” in the above setting), then there would be a simple algorithm determining whether $x \in \mathbb{G}_p$ by returning **true** iff division by x fails. Interestingly, we will show that there is *no* generic algorithm making similar use of a *standard* Diffie-Hellman oracle, unless factoring n is easy. Therefore a further consequence of the theorem presented in the following section is that a standard Diffie-Hellman oracle does not imply a inverse Diffie-Hellman oracle in general, unless factoring is easy.

Remark 2. The subgroup decision problem was introduced in [5] for groups with bilinear pairing. Essentially such a pairing can be added to the generic model by allowing the algorithm to perform a single multiplication operation when evaluating the bilinear pairing map [5] as done in [4]. By providing a Diffie-Hellman oracle, we do not restrict the algorithm to a fixed number of multiplications. Hence, our proof includes the problem stated in [5] as a special case.

6.1 Generic Equivalence to Factoring

In the sequel we show that solving the subgroup decision problem in groups of order n is as hard as factoring n , even if the algorithm has access to an oracle solving the Diffie-Hellman problem.

⁵ Plus some minor technical details to distinguish between different groups.

Theorem 4. *Suppose there exist a generic ring algorithm \mathcal{A} solving the subgroup membership problem in \mathbb{G} with advantage $\text{Adv}(\mathcal{A}^{\mathcal{O}_{\text{sdp}}}(n))$ by making m queries to an oracle performing the (inverse) group operation and solving the Diffie-Hellman problem. Then there exists an algorithm \mathcal{B} finding a factor of n with probability at least $\text{Adv}(\mathcal{A}^{\mathcal{O}_{\text{sdp}}}(n))$ by running \mathcal{A} once and performing $O(m^3)$ additional operations in \mathbb{Z}_n and m gcd-computations on $\lceil \log_2 n \rceil$ -bit numbers.*

Proof. Let us consider an interaction of \mathcal{A} with an oracle \mathcal{O}_p which is defined as follows. \mathcal{O}_p works similar to \mathcal{O}_{sdp} , but performs all computations in \mathbb{Z}_p . That is, the $\text{equal}()$ -procedure returns **true** on input (i, j) iff $P_i(x) \equiv P_j(x) \pmod p$. Note that now all computations are performed in the \mathbb{Z}_p -component of the decomposition $\mathbb{Z}_p \times \mathbb{Z}_q$ of \mathbb{Z}_n , hence the algorithm receives no information on whether $x \equiv 0 \pmod q$. Thus in the simulation game any algorithm has only trivial success probability $\Pr[\mathcal{S}_{\text{sim}}] = 1/2 + 1/q$.

Now consider an interaction of \mathcal{A} with oracle \mathcal{O}_{sdp} . Either this interaction is indistinguishable from an oracle \mathcal{O}_p , in which case the algorithm has only trivial success probability, or there exist $P_i, P_j \subseteq P$ with such that $P_i(x) \equiv P_j(x) \pmod p$, but $P_i(x) \not\equiv P_j(x) \pmod n$. In this case a factor of n is found by computing $\text{gcd}(P_i(x) - P_j(x), n)$. Note that

$$\begin{aligned} \frac{1}{2} + \text{Adv}_{(\mathcal{C}, \mathcal{V})}(\mathcal{A}^{\mathcal{O}_{\text{sdp}}}(n)) &\leq \Pr[\mathcal{S}_{\text{sim}}] + \Pr[\mathcal{F}] \\ \iff \text{Adv}_{(\mathcal{C}, \mathcal{V})}(\mathcal{A}^{\mathcal{O}_{\text{sdp}}}(n)) &\leq \Pr[\mathcal{F}] \end{aligned}$$

Thus, n is factored this way by running \mathcal{A} , recording P and computing

$$\text{gcd}(P_i(x) - P_j(x), n)$$

for all $-1 \leq i < j \leq m$ with probability at least $\text{Adv}_{(\mathcal{C}, \mathcal{V})}(\mathcal{A}^{\mathcal{O}_{\text{sdp}}}(n))$.

The above proof generalizes from $n = pq$ to $n = \prod_{i=1}^k p_i^{e_i}$ for all subgroups with prime-power order $p_i^{e_i}$ in a straightforward manner.

7 Analyzing Search Problems in the Generic Ring Model

In Section 3 we have constructed a simulator for a generic ring oracle for the ring \mathbb{Z}_n . When interacting with the simulator, all computations are independent of the secret challenge value x . Therefore we have been able to conclude that any generic algorithm has only the trivial probability of success in solving certain decisional problems (namely the considered subset membership problems) when interacting with the simulator. Moreover, we have shown that any algorithm that can distinguish between simulator and original oracle can be turned into a factoring algorithm with (asymptotically) the same running time.

In contrast to *decisional* problems, where the algorithm outputs a bit, our construction of the simulator can also be applied to prove the generic hardness of *search* problems where the algorithm outputs a ring element or integer. Let

us sketch two possibilities. The first one is to formulate a suitable subset membership problem which reduces to the considered search problem and then apply Theorem 1. Another possibility is to use our construction of the simulator to bound the probability of a simulation failure relative to factoring. In order to bound the success probability in the simulation game, it remains to show that there exists no *straight line program* solving the considered problem efficiently under the factoring assumption.

Acknowledgements. We would like to thank Andy Rupp and Sven Schäge for helpful discussions, and Yvo Desmedt and the program committee members for valuable suggestions.

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 (2009)
2. Altmann, K., Jager, T., Rupp, A.: On black-box ring extraction and integer factorization. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfssdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 437–448. Springer, Heidelberg (2008)
3. Black, J.A.: The ideal-cipher model, revisited: An uninstantiable blockcipher-based hash function. In: Robshaw, M.J.B. (ed.) FSE 2006. LNCS, vol. 4047, pp. 328–340. Springer, Heidelberg (2006)
4. Boneh, D., Boyen, X.: Short signatures without random oracles and the SDH assumption in bilinear groups. *J. Cryptology* 21(2), 149–177 (2008)
5. 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)
6. Boneh, D., Lipton, R.J.: Algorithms for black-box fields and their application to cryptography. In: Kobitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 283–297. Springer, Heidelberg (1996)
7. Boneh, D., Venkatesan, R.: Breaking RSA may not be equivalent to factoring. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 59–71. Springer, Heidelberg (1998)
8. Brown, D.R.L.: Breaking RSA may be as difficult as factoring. *Cryptology ePrint Archive, Report 2005/380* (2005), <http://eprint.iacr.org/>
9. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited. *J. ACM* 51(4), 557–594 (2004)
10. Damgård, I.B., Koprowski, M.: Generic lower bounds for root extraction and signature schemes in general groups. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 256–271. Springer, Heidelberg (2002)
11. 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)
12. 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)

13. Jager, T., Schwenk, J.: On the analysis of cryptographic assumptions in the generic ring model, full version. Cryptology ePrint Archive (2009), <http://eprint.iacr.org/>
14. Kobitz, N., Menezes, A.J.: Another look at generic groups, pp. 13–28 (2006)
15. 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)
16. Maurer, U.M., Raub, D.: Black-box extension fields and the inexistence of field-homomorphic one-way permutations. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 427–443. Springer, Heidelberg (2007)
17. Maurer, U.M.: Abstract models of computation in cryptography. In: Smart, N.P. (ed.) Cryptography and Coding 2005. LNCS, vol. 3796, pp. 1–12. Springer, Heidelberg (2005)
18. Maurer, U.M., Wolf, S.: Lower bounds on generic algorithms in groups. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 72–84. Springer, Heidelberg (1998)
19. Maurer, U.M., Wolf, S.: The relationship between breaking the Diffie-Hellman protocol and computing discrete logarithms. SIAM J. Comput. 28(5), 1689–1721 (1999)
20. Nguyễn, P.Q., Shparlinski, I.E.: On the insecurity of a server-aided RSA protocol. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 21–35. Springer, Heidelberg (2001)
21. Rupp, A., Leander, G., Bangarter, E., Dent, A.W., Sadeghi, A.-R.: 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)
22. 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)
23. Shoup, V.: A Computational Introduction to Number Theory and Algebra. Cambridge University Press, Cambridge (2005)

A Proof Sketch for Lemma 3

For notational convenience, let us define $\Gamma(P) := \Pr[P(x') \notin \mathbb{Z}_n^*$ and $P(x) \in \mathbb{Z}_n^* \mid x, x' \stackrel{U}{\leftarrow} \mathcal{C}]$ and $\Lambda(P) := \Pr[\gcd(n, P(y)) \notin \{1, n\} \mid y \stackrel{U}{\leftarrow} \mathcal{U}[\mathcal{C}]]$. Thus, in order to prove Lemma 3 we have to show that the inequality

$$\left(\frac{|\mathcal{U}[\mathcal{C}]|}{|\mathcal{C}|} \right)^2 \Lambda(P) \geq \Gamma(P) \quad (1)$$

holds. To this end, we will define an auxiliary function $\nu_i(P)$. Then we express $\Gamma(P)$ and $\Lambda(P)$ in terms of $\nu_i(P)$. More precisely, we will upper bound $\Gamma(P)$ by an expression in $\nu_i(P)$ and lower bound $\Lambda(P)$ by an expression in $\nu_i(P)$. The resulting inequality is proven easily by complete induction.

Defining an auxiliary function. Recall that we denote with $n = \prod_{i=1}^k p_i^{e_i}$ the prime factor decomposition of n . Let

$$\nu_i(P) := \Pr \left[P(x) \equiv 0 \pmod{p_i} \mid x \stackrel{\mathcal{U}}{\leftarrow} \mathcal{C} \right]$$

be the probability that $P(x) \equiv 0 \pmod{p_i}$ for some prime p_i dividing n and $x \stackrel{\mathcal{U}}{\leftarrow} \mathcal{C}$. Recall that $\phi : \mathbb{Z}_{p_1^{e_1}} \times \cdots \times \mathbb{Z}_{p_k^{e_k}} \rightarrow \mathbb{Z}_n$ is a ring isomorphism, and P performs only ring operations in \mathbb{Z}_n . Therefore P *implicitly* performs all operations on each component $\mathbb{Z}_{p_i^{e_i}}$ separately (and *independently*). Moreover, sampling $x \stackrel{\mathcal{U}}{\leftarrow} \mathcal{C}$ is equivalent to sample $\phi(x_1, \dots, x_k)$ with x_i chosen *independently* and *uniform* from \mathcal{C}_i for $1 \leq i \leq k$ (cf. Lemma [□](#)). Thus we can express the probability that $P(x) \in \mathbb{Z}_n^*$ for $x \stackrel{\mathcal{U}}{\leftarrow} \mathcal{C}$ as

$$\Pr \left[P(x) \in \mathbb{Z}_n^* \mid x \stackrel{\mathcal{U}}{\leftarrow} \mathcal{C} \right] = \prod_{i=1}^k (1 - \nu_i(P)).$$

Bounding $\Gamma(P)$ in terms of $\nu_i(P)$. For independently sampled x, x' , we have

$$\begin{aligned} \Gamma(P) &= \Pr \left[P(x') \notin \mathbb{Z}_n^* \text{ and } P(x) \in \mathbb{Z}_n^* \mid x, x' \stackrel{\mathcal{U}}{\leftarrow} \mathcal{C} \right] \\ &= \Pr \left[P(x) \notin \mathbb{Z}_n^* \mid x \stackrel{\mathcal{U}}{\leftarrow} \mathcal{C} \right] \cdot \Pr \left[P(x) \in \mathbb{Z}_n^* \mid x \stackrel{\mathcal{U}}{\leftarrow} \mathcal{C} \right] \end{aligned}$$

Note that, since $\mathcal{C} \subseteq \mathcal{U}[\mathcal{C}]$, it holds that

$$\Pr \left[P(x) \in \mathbb{Z}_n^* \mid x \stackrel{\mathcal{U}}{\leftarrow} \mathcal{C} \right] \leq \Pr \left[P(y) \in \mathbb{Z}_n^* \mid y \stackrel{\mathcal{U}}{\leftarrow} \mathcal{U}[\mathcal{C}] \right] \frac{|\mathcal{U}[\mathcal{C}]|}{|\mathcal{C}|}$$

and similarly

$$\Pr \left[P(x) \notin \mathbb{Z}_n^* \mid x \stackrel{\mathcal{U}}{\leftarrow} \mathcal{C} \right] \leq \left(1 - \Pr \left[P(y) \in \mathbb{Z}_n^* \mid y \stackrel{\mathcal{U}}{\leftarrow} \mathcal{U}[\mathcal{C}] \right] \right) \frac{|\mathcal{U}[\mathcal{C}]|}{|\mathcal{C}|}.$$

Therefore we can conclude that

$$\begin{aligned} \Gamma(P) &\leq \Pr \left[P(y) \in \mathbb{Z}_n^* \mid y \stackrel{\mathcal{U}}{\leftarrow} \mathcal{U}[\mathcal{C}] \right] \left(1 - \Pr \left[P(y) \in \mathbb{Z}_n^* \mid y \stackrel{\mathcal{U}}{\leftarrow} \mathcal{U}[\mathcal{C}] \right] \right) \left(\frac{|\mathcal{U}[\mathcal{C}]|}{|\mathcal{C}|} \right)^2 \\ &= \prod_{i=1}^k (1 - \nu_i(P)) \left(1 - \prod_{i=1}^k (1 - \nu_i(P)) \right) \left(\frac{|\mathcal{U}[\mathcal{C}]|}{|\mathcal{C}|} \right)^2. \end{aligned} \tag{2}$$

Bounding $\Lambda(P)$ in terms of $\nu_i(P)$. We can find a factor of n by computing $\gcd(n, P(y))$, if $P(y) \equiv 0 \pmod{p_i}$ for at least one prime p_i dividing n , and $P(y) \not\equiv 0 \pmod{n}$. Using similar arguments as above, we can therefore express $\Lambda(P)$ in terms of $\nu_i(P)$ as

$$\begin{aligned} \Lambda(P) &= \Pr \left[\gcd(n, P(y)) \notin \{1, n\} \mid y \stackrel{U}{\leftarrow} \mathcal{C} \right] \\ &= 1 - \prod_{i=1}^k \nu_i(P) - \prod_{i=1}^k (1 - \nu_i(P)). \end{aligned} \tag{3}$$

Putting things together. Combining (2) and (3), we see that (1) holds if

$$\left(1 - \prod_{i=1}^k (1 - \nu_i(P)) \right)^2 \geq \prod_{i=1}^k \nu_i(P)$$

holds, which is shown easily by complete induction on $k \geq 2$.

B Proof Sketch for Proposition 1

If there exists P_j such that $(P_j(x) = \perp \text{ and } P_j(x_r) \neq \perp)$, then this implies that there exists $P_k \sqsubseteq P$ with $k < j$ such that $(P_j(x_r) \notin \mathbb{Z}_n^* \text{ and } P_j(x) \in \mathbb{Z}_n^*)$ by Lemma 2. Hence, in order to bound the probability of $\mathcal{F}_{\text{test}}$, it suffices to consider the probability that there exists a straight line program $P_j \sqsubseteq P$ such that

$$(P_j(x_r) \notin \mathbb{Z}_n^* \text{ and } P_j(x) \in \mathbb{Z}_n^*) \text{ or } (P_j(x) \notin \mathbb{Z}_n^* \text{ and } P_j(x_r) \in \mathbb{Z}_n^*) \tag{4}$$

for $x, x_1, \dots, x_m \stackrel{U}{\leftarrow} \mathcal{C}$.

By (essentially) applying the union bound we can see that for *fixed* P_j this probability is bounded by

$$2m \Pr \left[P_j(x) \notin \mathbb{Z}_n^* \text{ and } P_j(x') \in \mathbb{Z}_n^* \mid x, x' \stackrel{U}{\leftarrow} \mathcal{C} \right].$$

Using this, we obtain the following bound on the probability that there exists *any* $P_j \sqsubseteq P$ satisfying (4).

$$\begin{aligned} \Pr[\mathcal{F}_{\text{test}}] &\leq 2m \sum_{j=0}^m \Pr \left[P_j(x) \notin \mathbb{Z}_n^* \text{ and } P_j(x') \in \mathbb{Z}_n^* \mid x, x' \stackrel{U}{\leftarrow} \mathcal{C} \right] \\ &\leq 2m(m+1) \max_{0 \leq j \leq m} \left\{ \Pr \left[P_j(x) \notin \mathbb{Z}_n^* \text{ and } P_j(x') \in \mathbb{Z}_n^* \mid x, x' \stackrel{U}{\leftarrow} \mathcal{C} \right] \right\} \end{aligned}$$

Zero Knowledge in the Random Oracle Model, Revisited

Hoeteck Wee*

Queens College, CUNY
hoeteck@cs.qc.cuny.edu

Abstract. We revisit previous formulations of zero knowledge in the random oracle model due to Bellare and Rogaway (CCS '93) and Pass (Crypto '03), and present a hierarchy for zero knowledge that includes both of these formulations. The hierarchy relates to the programmability of the random oracle, previously studied by Nielsen (Crypto '02).

- We establish a subtle separation between the Bellare-Rogaway formulation and a weaker formulation, which yields a finer distinction than the separation in Nielsen's work.
- We show that zero-knowledge according to each of these formulations is not preserved under sequential composition. We introduce stronger definitions wherein the adversary may receive auxiliary input that depends on the random oracle (as in Unruh (Crypto '07)) and establish closure under sequential composition for these definitions. We also present round-optimal protocols for NP satisfying the stronger requirements.
- Motivated by our study of zero knowledge, we introduce a new definition of proof of knowledge in the random oracle model that accounts for oracle-dependent auxiliary input. We show that two rounds of interaction are necessary and sufficient to achieve zero-knowledge proofs of knowledge according to this new definition, whereas one round of interaction is sufficient in previous definitions.
- Extending our work on zero knowledge, we present a hierarchy for circuit obfuscation in the random oracle model, the weakest being that achieved in the work of Lynn, Prabhakaran and Sahai (Eurocrypt '04). We show that the stronger notions capture precisely the class of circuits that is efficiently and exactly learnable under membership queries.

Keywords: zero-knowledge, random oracle model, sequential composition, obfuscation.

1 Introduction

The random oracle (RO) model, introduced by Fiat and Shamir [10] and refined by Bellare and Rogaway [3], was proposed as a framework for designing and analyzing cryptographic schemes that offers a trade-off between provable security and practical efficiency. In this model, every party has oracle access to a truly random function. With this additional functionality, many cryptographic problems admit more efficient

* Work done while visiting Tsinghua University, Beijing, China.

solutions than in the standard model, along with considerably simpler proofs of security [34,24,11]. In practice, the idealized random function is instantiated using a “good” cryptographic hash function, like SHA-1 or a variation thereof. There are also cryptographic problems for which we have partial solutions in the random oracle model but not in the standard model, most notably that of circuit obfuscation [119,20]. In both cases, proofs in the random oracle model do not guarantee security or feasibility in the standard model (and in fact, there has been substantial evidence to the contrary [6,15,2]); nonetheless, the model provides a useful idealized test-bed for analyzing cryptographic schemes.

As a first step towards establishing security, it is necessary to define security in the random oracle model. A naive extension of a definition in the standard model may affect the semantics of the underlying notion of security. Consider the case of *zero-knowledge proofs*, namely proofs that yield no knowledge beyond the validity of the assertion proved [17]. Formally, an interactive protocol is zero-knowledge if there exists a simulator that can simulate the behavior of every, possibly cheating, verifier without access to the prover, such that its output is indistinguishable from the output of the verifier after having interacted with the honest prover. In the standard model, a zero-knowledge proof is necessarily *deniable*, in that the protocol’s transcript does not constitute any evidence of the interaction, since any party could have generated the transcript by himself. However, the Bellare-Rogaway formulation of zero-knowledge in the random oracle model does not imply deniability, since the simulator can *choose* the random oracle [22,21]. In particular, the formulation allows for (non-trivial) one-round zero-knowledge proof systems, and the transcript of such a protocol constitutes evidence of participating in the protocol, contradicting deniability.

In this work, we revisit two aspects of formulating zero-knowledge in the random oracle model. The first relates to defining security in the random oracle model and in particular, what it means to choose the random oracle, an issue first addressed by Nielsen [21]. The second relates to a different aspect of zero-knowledge proofs, namely, we want the zero-knowledge guarantee to hold even if the verifier may have some additional a priori information about the input. The need to account for such auxiliary input, which arises in typical applications such as sequential repetitions of a protocol, was articulated in the work of Goldreich and Oren [14] and again in that of Unruh [25]. While the Bellare-Rogaway formulation of zero-knowledge does take into account auxiliary input, it does not allow for dependencies between the auxiliary input and the random oracle, which arise for instance, when the auxiliary input is a transcript of a previous interaction using the oracle.

1.1 Programmability in the Random Oracle Model

There are two reasons why, in the simulation-based paradigm, it is easier to achieve security in the random oracle model:

- the simulator can see the queries parties make to the random oracle;
- the simulator can choose the answers to these queries.

The second is what we refer to as *programming* the random oracle, and may be qualified in several different ways. Suppose our goal is to simulate a transcript $\text{RO}(s)$, namely

the evaluation of the random oracle RO at some value s . Our intuition about the random oracle as a truly random function indicates that picking a truly random string τ should suffice (essentially choosing the evaluation of RO at s to be τ), and indeed, no distinguisher - even computationally unbounded ones - can distinguish a truly random string from $\text{RO}(s)$, provided the distinguisher does not get access to RO. On the other hand, if we give the distinguisher access to RO, then the only “good” simulation of the transcript is $\text{RO}(s)$, and the simulation must query RO at s . This is because the distinguisher may have s hardwired into it, then queries RO at s and checks whether the answer matches the transcript. In this setting, the simulator does not get to choose the answers to oracle queries. To distinguish between these two notions of security, we will refer to the former as the *fully programmable* random oracle (FPRO) model, and the latter as the *non-programmable* random oracle (NPRO) model (as coined by Nielsen [21]).

In the case where we allow the simulator to choose the answers to oracle queries, we may still impose an additional requirement, namely that the simulator must output its choices of these query/answer pairs. In the above example, whether the simulator chooses the output of RO at s to be some random string τ , its output will include the transcript τ , along with the list (s, τ) , corresponding to the query s and answer τ . This is in fact the notion of programmability raised by Bellare and Rogaway [3] for zero-knowledge, and we will refer to this as the *explicitly programmable* random oracle (EPRO) model. We defer a precise definition to the body of the paper, but note at this point that security in the non-programmable random oracle model (strongest security guarantee) implies security in the explicitly programmable random oracle model, which in turn implies security in the fully programmable random oracle model (weakest security guarantee).

1.2 Our Contributions and Techniques

Hierarchy for zero knowledge. We begin with a simple and unified framework for defining zero knowledge in the three variants of the random oracle model, and then present a (perhaps surprising) separation for zero knowledge in the fully programmable and explicitly programmable random oracle models. This yields a finer separation than that in Nielsen’s work [21], and complements Pass’s separation for zero knowledge in the explicitly programmable and non-programmable random oracle models.

Auxiliary input and sequential composition. Following the work of Goldreich et al. [14,13] for zero-knowledge in the standard model, we use closure under sequential composition as a yardstick for evaluating formulations of zero-knowledge. We show that zero-knowledge in all three variants of the random oracle model are not closed under sequential composition¹. This motivates a new formulation of zero-knowledge in the random oracle which allows for auxiliary inputs that depend on the oracle, as

¹ That this may be the case has been previously noted (e.g. [22]), but to our knowledge, there has been no formal (published) proof.

was done in [25] for one-way functions, encryption and other primitives.² We show that for efficient-prover protocols, zero-knowledge with oracle-dependent auxiliary input in the explicit-programmable and non-programmable random oracle models are preserved under a polynomial number of sequential repetitions. We also present round-optimal protocols for NP satisfying the new formulations of zero-knowledge.

Proofs of knowledge. Our constructions demonstrating that previous formulations of zero knowledge are not closed under sequential composition implicitly rely on a non-interactive zero-knowledge “proofs of knowledge” in the random oracle model. Specifically, non-interactive protocols are necessarily malleable (without unique identifiers), and the cheating verifier can generate a convincing proof of knowledge by copying one sent by the prover in a previous iteration of the protocol. This motivates a new formulation of proof of knowledge in the random oracle model that takes into account oracle-dependent auxiliary input. We show that two rounds of interaction are necessary and sufficient to achieve zero-knowledge proofs of knowledge according to this new definition.

Circuit obfuscation. We extend our framework for programmability to circuit obfuscation³ in the random oracle model [19,11], and note that the obfuscator constructions of Lynn et al. [19] achieve security in the fully programmable random oracle model. Next, we show circuit obfuscation in the explicit-programmable random oracle model can only be realized for classes of circuits that are efficiently and exactly learnable under membership queries, and for these classes, obfuscation may be (trivially) realized in the plain model, so the characterization is exact. We find it surprising that we can have non-trivial constructions in the explicitly programmable model for zero knowledge but not for circuit obfuscation.

1.3 Discussion

Formulating zero-knowledge. A general framework for defining security in the random oracle model was presented by Nielsen [21], based on augmenting the universally composable (UC) framework [5] with a random oracle functionality. This guarantees composability. As pointed out by Pass [22], deniability is not guaranteed in this framework. Nielsen also defined security with a non-programmable random oracle,

² For the primitives considered in [25], the random oracle is typically only used in the proof of security. Specifically, Unruh [25] does not explicitly address primitives with a simulation-based notion of security, which is the focus of this work and where the random oracle is also exploited in constructing a simulator. On the other hand, Unruh considers a stronger notion of oracle-dependent auxiliary input, where a polynomial bound is imposed only on the output length of the machine generating the auxiliary input and not its query complexity.

³ We use the term obfuscation to refer to the stronger notion of obfuscation against general adversaries, instead of obfuscation against predicate adversaries [11,26]. In the standard model, only classes of circuits that are efficiently and exactly learnable under membership queries are obfuscatable against general adversaries [26]. The result also extends to the fully-programmable RO model.

where the environment in the UC framework is also given access to the random oracle. This offers deniability, but may no longer guarantee universal composability.

Instead of adopting Nielsen’s formulation, we consider a minimal framework (based on [3,14]) for which we can provide the weaker guarantee of sequential composition. The simplicity allows us to focus on how the random oracle is incorporated differently in each of [21,3,22]. In addition, it offers several conceptual advantages: it offers modularity (which allows us to decouple the zero-knowledge property from the proof-of-knowledge property and other properties implied by UC zero-knowledge, and for impossibility results, these distinctions are particularly important) and reinforces the theme of this work, that of understanding how semantics can change between the standard model and the random oracle model. Furthermore, our framework is simple enough to be applied to circuit obfuscation, for which we have very few non-trivial positive results, let alone constructions that compose arbitrarily (which probably only exist for trivially obfuscatable families of circuits).

Sequential composition not the end-goal. We recall the arguments used in [14] to motivate the study of auxiliary-input zero-knowledge: first, it fully captures the intuitive meaning of the concept of zero-knowledge; and second, this stronger requirement is necessary when a zero-knowledge protocol is used as a sub-protocol within larger cryptographic protocols⁴. It is for these same reasons that we pursue a formulation of zero-knowledge in the random oracle model that incorporates auxiliary input (refer to [25] for additional arguments). Indeed, we regard our sequential composition lemma as evidence that we have properly accounted for auxiliary input in our formulation and not a goal in and of itself. Similarly, constructing protocols for NP that remain zero-knowledge under sequential composition should not be an objective in itself⁵. Neither should a generic method for transforming protocols that are zero-knowledge into another that remain zero-knowledge under sequential composition.

On “explicit programmability”. From previous work [3,22,19,26], we know that allowing the simulator to program the random oracle is necessary and sufficient for one-round zero-knowledge protocols for NP and obfuscating point functions in the random oracle model. However, while explicit programmability is sufficient for zero-knowledge, we show that full programmability is necessary for the latter. This means that the reason we are able to realize non-trivial circuit obfuscation in the random oracle model comes not only from programming the random oracle, but also from not having to specify explicitly how we program the random oracle.

The issue of explicit programmability also arises in the study of sequential composition. To obtain zero-knowledge that is closed under a polynomial number of sequential

⁴ One may ask, why not aim for universal composability then? This is addressed in the previous paragraph, and as with previous work in the standard setting, we feel that zero-knowledge w.r.t. auxiliary input is indeed the right compromise.

⁵ All “natural” zero-knowledge protocols for NP in the RO model (in the [3] sense) remain zero-knowledge under sequential, even concurrent, composition, but this does not obliterate the need for the “right” definition. After all, when auxiliary-input zero-knowledge was introduced, all known zero-knowledge protocols were black-box and therefore remained zero-knowledge under sequential composition.

compositions, it appears that explicit programmability is necessary, in addition to properly accounting for auxiliary input.

Self-composition for circuit obfuscation. Lynn et al. introduce self-composition for circuit obfuscation [19], a notion of composition analogous to sequential composition. In addition, they give an obfuscator for point functions in the random oracle model that is not 2-self-composing. This is because the construction is not a valid obfuscator w.r.t. dependent auxiliary input. To obtain polynomial self-composition for obfuscation using techniques in this work, we will need a definition that incorporates both oracle-dependent auxiliary input and explicit programmability.

2 Preliminaries

A negligible function is a function of the form $n^{-\omega(1)}$, and is denoted $\text{neg}(n)$. We use PPT as an abbreviation for a probabilistic (strict) polynomial-time Turing machine. We also consider the nonuniform and oracle analogues, which we denote by nonuniform PPT and oracle PPT respectively. In probability expressions that involve a probabilistic computation, the probability is also taken over the internal coin tosses of the underlying computation. We refer the reader to [12] for definitions of interactive proof systems, zero-knowledge, proofs of knowledge and witness-indistinguishability (WI) in the standard model. For a relation $R \subseteq \{0, 1\}^* \times \{0, 1\}^*$, the *language associated with* R is $L_R = \{x : \exists y (x, y) \in R\}$.

3 Zero Knowledge in the Random Oracle Model

In this section, we present our hierarchy of formulations for zero knowledge in the RO model, along with those that account for oracle-dependent auxiliary input. We begin with several formalisms we will use in defining zero knowledge:

- We use RO to denote the random oracle and ε to denote an oracle that returns the empty string on all inputs.
- Given a function $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$ and a list $\ell \subseteq \{0, 1\}^* \times \{0, 1\}^*$, we use $f[\ell]$ to denote a function that agrees with f everywhere except on inputs specified by the set ℓ . Specifically,

$$f[\ell](x) = \begin{cases} y & \text{if } \exists! y \text{ such that } (x, y) \in \ell \\ f(x) & \text{otherwise} \end{cases}$$

Informally, we refer to $f[\ell]$ as the function obtained by programming f on the inputs in ℓ . In the definition of zero-knowledge, the simulator generates a pair (τ, ℓ) : the simulator programs the random oracle on the inputs in ℓ , and τ corresponds to the view of the cheating verifier while interacting with the prover using the oracle $\text{RO}[\ell]$.

- We allow the auxiliary input to be generated by a nonuniform oracle PPT Z (the nonuniformity allows for auxiliary information that may depend on the input instance) which we refer as the *auxiliary input machine*. We will give Z oracle access to either ε or RO, depending on whether we allow the auxiliary input to depend on the random oracle.

Definition 1 (zero-knowledge [3,22]). Let (P, V) be an interactive protocol for a language $L = L_R$. Let V^*, S, Z, D be oracle PPTs. Given $(x, w) \in R$, we define $\text{diff}_{V^*, S, Z, D}^{\mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_3}(x, w)$ to be the quantity

$$\begin{aligned} & \Pr_{\text{RO}}[z \leftarrow Z^{\mathcal{O}_1}(1^{|x|}); \tau \leftarrow (P^{\text{RO}}(w), V^{*\text{RO}}(z))(x); D^{\mathcal{O}_2}(x, \tau, z) = 1] \\ & - \Pr_{\text{RO}}[z \leftarrow Z^{\mathcal{O}_1}(1^{|x|}); (\tau, \ell) \leftarrow S^{\text{RO}}(x, z); D^{\mathcal{O}_3}(x, \tau, z) = 1] \end{aligned}$$

We say that (P, V) is zero-knowledge in the fully programmable random oracle model (FPRO) if for every oracle PPT V^* , there exists an oracle PPT S such that for all $(x, w) \in R$ and for all nonuniform oracle PPTs Z and D , $\text{diff}_{V^*, S, Z, D}^{\varepsilon, \varepsilon, \varepsilon}(x, w)$ is negligible (as a function of $|x|$). In addition, we obtain zero-knowledge in the:

- explicitly programmable RO (EPRO) model if $\mathcal{O}_1 = \varepsilon, \mathcal{O}_2 = \text{RO}, \mathcal{O}_3 = \text{RO}[\ell]$
- non-programmable RO (NPRO) model if $\mathcal{O}_1 = \varepsilon, \mathcal{O}_2 = \text{RO}, \mathcal{O}_3 = \text{RO}$
- FPRO model w.r.t dependent auxiliary input if $\mathcal{O}_1 = \text{RO}, \mathcal{O}_2 = \varepsilon, \mathcal{O}_3 = \varepsilon$
- EPRO model w.r.t dependent auxiliary input if $\mathcal{O}_1 = \text{RO}, \mathcal{O}_2 = \text{RO}, \mathcal{O}_3 = \text{RO}[\ell]$
- NPRO model w.r.t. dependent auxiliary input if $\mathcal{O}_1 = \text{RO}, \mathcal{O}_2 = \text{RO}, \mathcal{O}_3 = \text{RO}$

For simplicity, we will also refer to the respective notions of zero-knowledge as FPRO zero-knowledge, EPRO zero-knowledge, NPRO zero-knowledge, auxiliary-input FPRO zero-knowledge, auxiliary-input EPRO zero-knowledge, and auxiliary-input NPRO zero-knowledge.

Zero-knowledge in the FPRO model. This definition captures the weakest requirement, in that the simulator may choose the random oracle in the simulated transcript, as long as it “looks” random. We point out that we require simulating the output of the cheating verifier, but not the random oracle used in the simulated transcript. This is equivalent to a definition wherein the simulator S is given access to RO. Since the distinguisher does not have access to RO, the simulator can simply generate a random oracle by itself, so giving the simulator access to RO does not give the simulator any extra power. Note that this definition also constitutes a relaxation of the UC framework augmented with a random oracle functionality (namely, that obtained by replacing the interactive environment with a non-interactive distinguisher) [215].

Zero-knowledge in the EPRO model. The main qualitative difference between FPRO zero-knowledge and EPRO zero-knowledge⁶ is that the simulator is required to completely specify a simulated random oracle (namely $\text{RO}[\ell]$) in the latter, which

⁶ Indeed, making this distinction in the UC framework would require clumsy modifications.

the distinguisher is given access to \mathcal{Z} . We require that S specifies ℓ explicitly, which implies a polynomial bound on the size of ℓ . On the other hand, the oracle $\text{RO}[\ell]$ is specified implicitly. EPRO zero-knowledge is equivalent to the Bellare-Rogaway formulation, except the latter does not give the simulator oracle access to RO . As with zero-knowledge in the FPRO model, this does not make any qualitative difference as the simulator can simply generate random answers to the RO queries and add these query-answer pairs to the list ℓ .

Zero-knowledge in the NPRO model. Here, the simulator is not allowed to choose the random oracle in the simulated transcript. This implies deniability, and is equivalent to Pass's formulation [22]. It is a special case of EPRO zero-knowledge with $\ell = \emptyset$. For efficient-prover protocols, the NPRO zero-knowledge requirement is equivalent to requiring that the following quantity be negligible [22][8]:

$$\mathbb{E}_{\text{RO}} \left[\left| \Pr[z \leftarrow Z^{\text{O}_1}(1^{|x|}); D^{\text{RO}}(x, (P^{\text{RO}}(w), V^{*\text{RO}}(z))(x), z) = 1] \right. \right. \\ \left. \left. - \Pr[z \leftarrow Z^{\text{O}_1}(1^{|x|}); D^{\text{RO}}(x, S^{\text{RO}}(x, z), z) = 1] \right| \right]$$

This is also true w.r.t. dependent auxiliary input.

Incorporating dependent auxiliary input. Incorporating dependent auxiliary input provides some guarantee of “independence” between the queries made to the random oracle in the protocol and prior queries, even though we do not know what the prior queries are. To achieve this definition, we construct simulators that program the random oracle on inputs that have not been previously queried by Z (here, we exploit the polynomial bound on the query complexity of Z). Unlike the case without auxiliary input, it is essential that we provide the simulator for zero-knowledge and EPRO zero-knowledge with oracle access to RO so that the simulator may generate transcripts that are consistent with the output from Z .

Verifier's view. A common convention in defining zero-knowledge in the standard model is to use $(P^{\text{RO}}(w), V^{*\text{RO}}(z))(x)$ to denote the view of the verifier V^* , which consists of the protocol's transcript and the verifier's random tape, instead of the output of the verifier. This is because we may incorporate the computation of the output from the view into the distinguisher. This argument does not necessarily apply to definitions in the RO model. In this case, the distinguisher does not have access to RO and may not be able to compute the output from the view.⁸ Therefore, we reserve $(P^{\text{RO}}(w), V^{*\text{RO}}(z))(x)$ to denote the output of the verifier.

A note on black-box simulation. As with previous works on zero knowledge in the RO model, we will establish the zero-knowledge property via black-box simulation,

⁷ For some secret value s and a random RO , we may easily simulate a view of $\text{RO}(s)$ with a random string. However, in order to simulate a view of $\text{RO}(s)$ along with an oracle that is consistent with this view, we will need to either query RO at s or program RO at s ; either operation requires “knowing” s .

⁸ Simply requiring that the verifier's query/answer pairs be included in its view may not be sufficient as we may also need the prover's query/answer pairs.

except we will allow the simulator to see the oracle queries made by the cheating verifier. This is consistent with the definition of zero knowledge because the simulator can execute the code of the cheating verifier and observe the oracle queries made during the executions. This is a crucial advantage over mere black-box simulation of the cheating verifier in the standard model. On the other hand, we do not allow the simulator to see the oracle queries made by Z . Consider a typical application, namely that of sequential repetitions of the protocol. Here, the auxiliary input is a transcript from previous executions of the protocol and may therefore depend on the oracle RO . The cheating verifier receives the transcript, but does not gain access to the private coin tosses used to generate the transcript. The distinction arises from the fact that we allow the simulator to depend on the cheating verifier but not on Z .

4 Zero-Knowledge Protocols and Separations

Several constructions of zero-knowledge protocols for NP in the RO model were given in [3,22,11]. It is straight-forward to verify that the zero-knowledge protocol in [3] is also auxiliary-input EPRO zero-knowledge. In an unpublished work [23], Pass determined the round-complexity of auxiliary-input NPRO zero-knowledge protocols for NP. We summarize these results below:

Theorem 1 (protocols [3,22,23]). *Assuming the existence of one-way functions, there exist:*

- a one-round proof of knowledge protocol for NP that is auxiliary-input EPRO zero-knowledge (moreover, we may assume that the knowledge extractor is straight-line and runs in strict polynomial time [22,11]);
- a two-round protocol for NP that is NPRO zero-knowledge; and
- a 3-round protocol for NP that is auxiliary-input NPRO zero-knowledge.

Furthermore, each of these protocols has perfect completeness, negligible soundness, and an efficient prover.

Theorem 2 (triviality [22,23]). *Only languages in BPP have a one-round NPRO zero-knowledge protocol or a 2-round auxiliary-input NPRO zero-knowledge protocol.*

We outline the proofs in [23]. The 3-round auxiliary-input NPRO zero-knowledge protocol for NP is based on the 2-round NPRO zero-knowledge protocol in [22] except we have the prover pick a random prefix α in the first round, and prepend α to all prover's and verifier's queries to the random oracle. The proof of Theorem 2 follows essentially from the fact that the proofs of the analogous statements in the standard model [14] relativizes.

Next, we state our first result, separating FPRO and EPRO zero-knowledge.

Theorem 3. *Assuming the existence of one-way permutations, there exists a protocol that is auxiliary-input FPRO zero-knowledge but not EPRO zero-knowledge.*

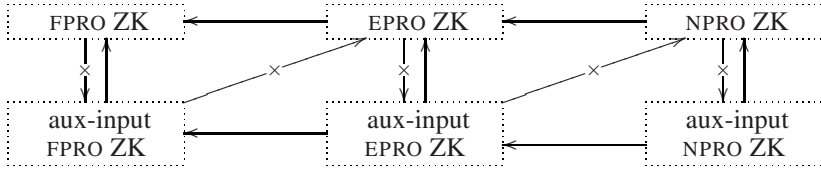


Fig. 1. Relations between different variants of zero knowledge in the RO model, assuming the existence of one-way permutations. An arrow is an implication, and a crossed arrow indicate separation. We stress that the relations refer to protocols satisfying the respective notion of zero-knowledge.

Proof. Let π be a one-way permutation, and consider the following protocol for the relation $R = \{(x, w) \mid x = \pi(w)\}$, where $L_R = \{0, 1\}^*$ (note that soundness holds vacuously):

Common input: An instance $x \in \{0, 1\}^n$.
Prover’s private input: A witness $w \in \{0, 1\}^n$.
 $P \rightarrow V$: Sends $\alpha \xleftarrow{R} \{0, 1\}^n$.
 $V \rightarrow P$: Sends $\tau \xleftarrow{R} \{0, 1\}^n$.
 $P \rightarrow V$: If $\tau = \text{RO}(\alpha \circ w)$, send w ; else, send $\text{RO}(\alpha \circ w)$.
 verification: V always accepts.

To see that this protocol is auxiliary-input FPRO zero-knowledge⁹, fix a cheating verifier V^* (along with its random tape and an auxiliary input z from Z), pick a random α , and simulate the execution of V^* , forwarding the oracle queries made by V^* to RO, until we obtain its first message τ . During the simulation, we also check if any of V^* ’s queries matches $\alpha \circ w$ (which we can check efficiently given x). If so, we would have recovered w , and may successfully compute the output of V^* . If we do not manage to recover w , we simulate the prover’s response with a random string $\tau' \in \{0, 1\}^n$ and continue to simulate the execution of V^* , forwarding all oracle queries to RO, unless the query matches $\alpha \circ w$, in which case we respond with τ' . This is ok because with probability $1 - \text{neg}(n)$ over α , none of the queries made by Z has prefix α . This completes the description of the zero-knowledge simulator.

Suppose on the contrary that the protocol is EPRO zero-knowledge, and consider the simulator S that outputs the view of the honest verifier. Fix $x \in L$, and consider a distinguisher with $w = \pi^{-1}(x)$ hardwired into it. Then, S must output a transcript that contains $\text{RO}[\ell](\alpha \circ w)$ with probability $1 - \text{neg}(n)$. For the latter, S must with high probability, either query RO at $\alpha \circ w$ or output a list ℓ that contains the string $\alpha \circ w$. In both cases, we may derive a PPT that on input x , outputs $\pi^{-1}(x)$ with high probability, which contradicts π being one-way. \square

⁹ Informally, the prover uses $(\alpha, \text{RO}(\alpha \circ w))$ to check whether the verifier already “knows” w .

5 Sequential Composition Fails without Dependent Auxiliary Input

In this section, we present zero-knowledge protocols which are no longer zero-knowledge when executed twice sequentially. The protocols are similar in spirit to that in [9], a zero-knowledge protocol in the standard model that is no longer zero-knowledge when executed twice in parallel. The protocols exploit zero-knowledge proofs of knowledge (which may be realized non-interactively in the random oracle model), using these proofs as the auxiliary input which a cheating verifier could use to “gain knowledge”. Specifically, the prover will send the verifier a zero-knowledge proof of knowledge of the witness, and the cheating verifier will copy this proof to “claim” knowledge of the witness. The apparent contradiction arises from a problem in the definition of proofs of knowledge in the random oracle model, an issue we will address in Section 7.

Theorem 4. *Assuming the existence of one-way functions, FPRO zero knowledge, EPRO zero knowledge, and NPRO zero knowledge are not closed under sequential composition.*

Proof (sketch). We begin by constructing an EPRO zero-knowledge protocol that is no longer zero-knowledge when composed twice. The protocol is for the language L corresponding to the relation $R = \{(x, w) \mid x = f(w)\}$, where f is a one-way function, and we use as an underlying protocol a one-round EPRO zero-knowledge proof of knowledge protocol (from Theorem 1).

Common input: An instance $x \in \{0, 1\}^n$.

Prover’s private input: A witness $w \in \{0, 1\}^n$.

$V \rightarrow P$: Send a random string τ .

$P \rightarrow V$: If τ is an EPRO zero-knowledge proof of knowledge that $x \in L$, send w ; else, send an EPRO zero-knowledge proof of knowledge that $x \in L$.

verification: V accepts if it receives either w such that $f(w) = x$ or an accepting proof of knowledge that $x \in L$.

To prove EPRO zero-knowledge, the simulator runs the cheating verifier to obtain the first message τ . If τ is an accepting proof of knowledge for $x \in L$, the simulator runs the knowledge extractor to obtain a valid witness w . Otherwise, the simulator runs the zero-knowledge simulator for the underlying zero-knowledge protocol to generate the second-round message. We would actually require that the underlying zero-knowledge protocol be auxiliary-input EPRO zero-knowledge, which is ok.

To see that this protocol is not zero-knowledge when composed twice, consider the cheating verifier V^* that sends a random string in the first execution, and sends the prover’s response as its first message in the second execution. For all $x \in L$, the transcript between the honest prover and V^* (for two sequential repetitions) will contain

$f^{-1}(x)$ with probability 1. That f is one-way implies that there is no PPT simulator for two sequential repetitions of this protocol.

A similar modification to Pass’s 2-round NPRO zero-knowledge protocol for NP yields a 2-round NPRO zero-knowledge protocol that is no longer NPRO zero-knowledge when composed twice. \square

Remark 1. Our counter-example are efficient-prover protocols (looking ahead, our sequential composition theorem only holds for efficient-prover protocols). This means that a cheating verifier (which is allowed to be nonuniform) can in fact simulate an interaction between the honest prover and the honest verifier. This is different from the counter-example in [13] wherein the cheating verifier cannot simulate such an interaction. There, the honest prover is allowed nonuniformity, whereas the cheating verifier is not, and the counter-example exploits the fact that the honest prover is “more powerful” than the class of cheating verifiers in an essential manner. This distinction was previously made in [9].

6 Sequential Composition with Dependent Auxiliary Input

Next, we prove a sequential composition lemma for auxiliary-input EPRO zero-knowledge and auxiliary-input NPRO zero-knowledge, which confirms that these are in some sense indeed the “right” definitions.

Theorem 5 (sequential composition). *Let (P, V) be an efficient-prover protocol in the RO model. Let $Q(\cdot)$ be a polynomial, and let (P_Q, V_Q) be an interactive protocol that on common input $x \in \{0, 1\}^n$, proceeds in $Q(n)$ phases, each of them consisting of executing the interactive protocol (P, V) on common input x (with independent coin tosses for P). If (P, V) is auxiliary-input EPRO (resp. NPRO) zero-knowledge, then (P_Q, V_Q) is also auxiliary-input EPRO (resp. NPRO) zero-knowledge.*

Proof (sketch). We begin with the proof for EPRO zero-knowledge. The high-level structure is similar to that in [14] for establishing a sequential composition lemma for zero-knowledge proofs in the standard model. We start by partitioning the cheating verifier V_Q^* for (P_Q, V_Q) into $Q(n)$ phases, each of which is the execution of a verifier V^* for a stand-alone protocol (P, V) . V^* takes as input the common input x and an auxiliary string encoding the state¹⁰ for V_Q^* at the end of some phase i (the string also encodes i) of an interaction with P_Q , and upon interacting with P produces as output another string encoding the state for V^* at the end of phase $i + 1$. The zero-knowledge property of (P, V) then guarantees a simulator S for V^* .

We generalize the earlier notation for programming a function by recursively defining $\text{RO}[\ell_1, \dots, \ell_{i+1}]$ as $(\text{RO}[\ell_1, \dots, \ell_i])[\ell_{i+1}]$. We may now specify the simulator for V_Q^* as follows: on input (x, z) ,

¹⁰ For simplicity, we may think of the string as encoding the transcript for the first i phases of interaction with P_Q along with the random tape and auxiliary input for V_Q^* .

- Set $\tau_0 = z$
- Run the simulator S for Q phases using the simulated oracle generated in the previous phase; that is, for $i = 1, 2, \dots, Q$,

$$S^{\text{RO}[\ell_1, \dots, \ell_{i-1}]}(x, \tau_{i-1}) \rightarrow (\tau_i, \ell_i)$$

- Output $(\ell_1 \cup \dots \cup \ell_Q, \tau_Q)$ ¹¹

We define $Q(n) + 1$ hybrids, H_0, \dots, H_Q . The j 'th hybrid is defined as the output of the following experiment:

- Run $Z^{\text{RO}}(1^n) \rightarrow z, \tau_0 = z$.
- Let the honest prover interact with the cheating verifier for j phases; that is, for $i = 1, 2, \dots, j$,

$$(P^{\text{RO}}, V^{*\text{RO}}(\tau_{i-1}))(x) \rightarrow \tau_i$$

- For the remaining $Q - j$ phases, run the simulator S using the simulated oracle generated in the previous phase; that is, for $i = j + 1, \dots, Q$,

$$S^{\text{RO}[\ell_j, \dots, \ell_{i-1}]}(x, \tau_{i-1}) \rightarrow (\tau_i, \ell_i)$$

- H_j is $(\text{RO}[\ell_{j+1}, \dots, \ell_Q], \tau_Q, z)$.

Note that H_0 and H_Q correspond to simulated transcript and the actual transcript respectively. We need to show that H_0 and H_Q are computationally indistinguishable. Suppose on the contrary that this is not the case. Therefore, we have a nonuniform oracle PPT D that distinguishes two consecutive hybrid distributions H_j and H_{j+1} . We define an auxiliary input machine Z_j that computes the interaction between P and V^* for the first j phases:

- Run $Z^{\text{RO}}(1^n) \rightarrow z, \tau_0 = z$.
- For $i = 1, 2, \dots, j$, $(P^{\text{RO}}, V^{*\text{RO}}(\tau_{i-1}))(x) \rightarrow \tau_i$
- Output (z, τ_j) .

This allows us to rewrite H_j and H_{j+1} as follows:

- | | |
|--|--|
| <ul style="list-style-type: none"> - Run $Z_j^{\text{RO}}(1^n) \rightarrow (z, \tau_j)$. - $S^{\text{RO}}(x, \tau_j) \rightarrow (\tau_{j+1}, \ell_{j+1})$ - for $i = j + 2, \dots, Q$,
 $(S^{\text{RO}[\ell_{j+1}, \dots, \ell_{i-1}]}(\tau_{i-1}))(x) \rightarrow (\tau_i, \ell_i)$ - Output $(\text{RO}[\ell_{j+1}, \dots, \ell_Q], \tau_Q, z)$. | <ul style="list-style-type: none"> - Run $Z_j^{\text{RO}}(1^n) \rightarrow (z, \tau_j)$. - $(P^{\text{RO}}, V^{*\text{RO}}(\tau_j))(x) \rightarrow \tau_{j+1}$ - for $i = j + 2, \dots, Q$,
 $(S^{\text{RO}[\ell_{j+2}, \dots, \ell_{i-1}]}(\tau_{i-1}))(x) \rightarrow (\tau_i, \ell_i)$ - Output $(\text{RO}[\ell_{j+2}, \dots, \ell_Q], \tau_Q, z)$. |
|--|--|

¹¹ We abuse \cup slightly here; we want $\ell_1 \cup \dots \cup \ell_Q$ to denote the set satisfying $\text{RO}[\ell_1 \cup \dots \cup \ell_Q] = \text{RO}[\ell_1, \dots, \ell_Q]$.

This is sufficient to contradict zero-knowledge of (P, V) for the $j + 1$ 'th phase^[12]. The result for NPRO zero-knowledge follows as a special case corresponding to $\ell_1 = \dots = \ell_Q = \emptyset$. \square

Remark 2. One might expect a priori that the zero knowledge is preserved under a polynomial number of repetitions once we take into account oracle-dependent auxiliary information. However, we are only able to establish such a statement in the EPRO and NPRO models. Technically, the proof breaks down for FPRO zero-knowledge because the simulator is not required to specify the simulated random oracle. In particular, this shows that sequential composition is more subtle than merely accounting for auxiliary input. A natural question that arises is whether prepending a random prefix to all oracle queries allows us to transform any protocol that is FPRO zero-knowledge into one that remains zero-knowledge under a polynomial number of sequential compositions. We note that using a random prefix only guarantees “independence” of the prover’s messages across different iterations; a cheating verifier is not limited to queries with the given prefix^[13].

7 Proofs of Knowledge with Dependent Auxiliary Input

Several constructions of zero-knowledge protocols begin with the verifier sending a proof of knowledge, for instance, that used in our counter-example, and the NPRO zero-knowledge protocol in [22]. If we allow the cheating verifier to receive an auxiliary input that depends on the random oracle, we would need to also extend the definition of proof of knowledge to incorporate auxiliary inputs that depend on the random oracle.

Definition 2. *Let (P, V) be an interactive protocol for a language $L = L_R$. We say that (P, V) is a proof of knowledge w.r.t. dependent auxiliary input in the RO model (or auxiliary-input proof of knowledge) if for every oracle PPT P^* , there exists an oracle PPT E such that for all nonuniform oracle PPT Z and for all x :*

$$\begin{aligned} & \Pr_{\text{RO}}[Z^{\text{RO}}(1^{|x|}) \rightarrow z; E^{\text{RO}}(x, z) \rightarrow w; (x, w) \in R] \\ & > \Pr_{\text{RO}}[Z^{\text{RO}}(1^{|x|}) \rightarrow z; (P^{*\text{RO}}(z), V^{\text{RO}}(x) \text{ accepts})] - \text{neg}(|x|) \end{aligned}$$

¹² Unlike in the standard model, we cannot use an averaging argument to fix the output (z, τ_j) from Z_j . This is because the output depends on RO. We may eliminate the efficient-prover constraint in the lemma by allowing the auxiliary input machine Z in the definition of zero-knowledge to be unbounded, but we do not know how to achieve zero-knowledge without a bound on the query complexity of Z .

¹³ Specifically, consider the trivial protocol for the language $\{0, 1\}^*$ wherein the prover sends nothing and the (honest) verifier always accepts. Note that using a random prefix does not affect this protocol in any way. Now, consider a cheating verifier that after each iteration outputs $\text{RO}(0^n)$. The zero-knowledge simulator for a single iteration (without dependent auxiliary input) may simply output a random string, but simply concatenating the output of this simulator for a polynomial number of times does not yield a correct simulation of the view of the cheating verifier for a polynomial number of iterations. This highlights the difference between simulating the transcript vs the output of the verifier, and the difficulty in ensuring “independence” of the random oracles amongst different iterations.

The next result implies a separation between zero-knowledge proofs of knowledge and zero-knowledge auxiliary-input proofs of knowledge. Specifically, we rule out non-interactive protocols that are simultaneously zero-knowledge and a proof of knowledge (in the above sense); otherwise, we can simply apply the knowledge extractor to the simulated proof to obtain a candidate witness. Some care is needed in arguing that the knowledge extractor works on the simulated proof, which uses a simulated random oracle and not the actual one. The reason why this approach only works for the new definition of proofs of knowledge is that we allow a cheating prover to receive oracle-dependent auxiliary input. In particular, the cheating prover may receive a convincing proof as auxiliary input, and the knowledge extractor can neither rewind the auxiliary input machine nor observe the oracle queries it makes. The proof is deferred to the full version.

Theorem 6. *Assuming the existence of one-way functions, there is a 2-round public-coin argument system for NP that is auxiliary-input EPRO zero-knowledge, and also an auxiliary-input proof of knowledge. On the other hand, only languages in BPP have a non-interactive argument system that is EPRO zero-knowledge and an auxiliary-input proof of knowledge.*

8 Circuit Obfuscation in the Random Oracle Model

Let \mathcal{O} be a probabilistic polynomial-time algorithm and let \mathcal{C} be a family of circuits. Let A, S, Z, D be oracle PPTs. Given $C \in \mathcal{C}$, we define $\text{diff}_{A,S,Z,D}^{\mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_3}(C)$ to be the quantity

$$\begin{aligned} & \Pr_{\text{RO}}[z \leftarrow Z^{\mathcal{O}_1}(1^{|C|}); \tau \leftarrow A^{\text{RO}}(\mathcal{O}^{\text{RO}}(C)); D^{\mathcal{O}_2}(\tau, z) = 1] \\ & - \Pr_{\text{RO}}[z \leftarrow Z^{\mathcal{O}_1}(1^{|C|}); (\tau, \ell) \leftarrow S^{\text{RO}, C}(z); D^{\mathcal{O}_3}(\tau, z) = 1] \end{aligned}$$

Definition 3 (circuit obfuscation [19,116]). *A probabilistic polynomial-time algorithm \mathcal{O} is an obfuscator for the family of circuits $\mathcal{C} = \cup_n \mathcal{C}_n$ in the FPRO model (where \mathcal{C}_n is the subset of circuits in \mathcal{C} that take inputs of length n) if the following three conditions hold:*

- (approximate functionality) *There exists a negligible function α such that for all n , for all $C \in \mathcal{C}_n$, with probability $1 - \alpha(n)$ over the internal coin tosses of the obfuscator and over RO, $\mathcal{O}^{\text{RO}}(C)$ describes a circuit with RO-gates that computes the same function as C .*
- (polynomial slowdown) *There is a polynomial p such that for every circuit $C \in \mathcal{C}$, $|\mathcal{O}(C)| \leq p(|C|)$.*
- (virtual black-box property) *For every oracle PPT A , there exists an oracle PPT S such that for all $C \in \mathcal{C}$ and for all nonuniform oracle PPTs Z and D , $\text{diff}_{A,S,Z,D}^{\varepsilon_1, \varepsilon_2, \varepsilon_3}(C)$ is negligible (as a function of $|C|$).*

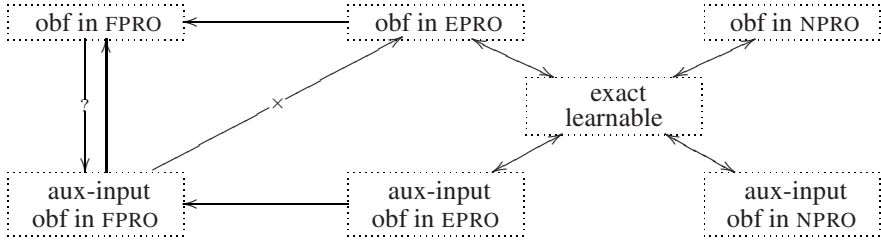


Fig. 2. Relations between different variants of obfuscation in the RO model. An arrow is an implication, a double-tailed arrow is an equivalence, and a crossed arrow indicate separation. We stress that the relations refer to families of circuits that are obfuscatable according to the respective notions.

We say that \mathcal{C} is FPRO obfuscatable if there exists an obfuscator for \mathcal{C} . In addition, we obtain:

- EPRO obfuscatable if $O_1 = \varepsilon, O_2 = RO, O_3 = RO[\ell]$
- NPRO obfuscatable if $O_1 = \varepsilon, O_2 = RO, O_3 = RO$
- auxiliary-input FPRO obfuscatable if $O_1 = RO, O_2 = \varepsilon, O_3 = \varepsilon$
- auxiliary-input EPRO obfuscatable if $O_1 = RO, O_2 = RO, O_3 = RO[\ell]$
- auxiliary-input NPRO obfuscatable if $O_1 = RO, O_2 = RO, O_3 = RO$

A point function I_w is a boolean function that evaluates to 1 on input w and 0 everywhere else. As observed in [19], to obfuscate I_w in the RO model, we may simply pick a random $\alpha \in \{0, 1\}^{|w|}$ and store $\alpha, RO(\alpha \circ w)$ in the obfuscated circuit, which on input x , outputs 1 iff $RO(\alpha \circ x) = RO(\alpha \circ w)$.

Theorem 7 (obfuscating point functions [19]). *There exists an auxiliary-input FPRO obfuscator for the class of point functions.*

One may expect some modification of the previous construction to yield an EPRO obfuscator for point functions, but this turns out to be impossible. The next result follows from a similar characterization in [26] for NPRO obfuscation:

Theorem 8 (triviality). *A family of circuits $\mathcal{C} = \cup_n \mathcal{C}_n$ is EPRO obfuscatable iff \mathcal{C} is efficiently and exactly learnable using membership queries.*

Proof (sketch). Suppose \mathcal{C} is efficiently and exactly learnable using membership queries. Consider an obfuscator that simply takes the input circuit C and outputs the circuit produced by the learning algorithm given oracle access to \mathcal{C} ; the simulator does essentially the same thing.

The learning algorithm for an EPRO obfuscatable family of circuits \mathcal{C} is very simple. To evaluate $C \in \mathcal{C}$ on input x (given oracle access to \mathcal{C} and input x), run the simulator S for the trivial adversary A that merely outputs the obfuscated circuit to obtain (τ, ℓ) ,

answering queries to RO with random coin tosses, and then evaluate τ on the input x using the simulated oracle $\text{RO}[\ell]$. This may be modified via standard techniques (specifically, we will need to amplify the soundness error via repetition and then take a union bound over all inputs) to yield a learning algorithm that on oracle access to C output w.h.p. a (standard) circuit that agrees with C on all inputs. \square

Acknowledgements. I am very grateful towards Rafael Pass for insightful exchanges on the subject and for allowing me to include his observations on deniable zero-knowledge [23], and the anonymous referees for meticulous and thought-provoking feedback. I also thank Andy Yao for hosting my visit at Tsinghua University in the 2005/2006 academic year (thereby funding this research) and for helpful discussions.

References

1. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S.P., Yang, K.: On the (Im)possibility of obfuscating programs. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 1–18. Springer, Heidelberg (2001)
2. 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)
3. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: ACM CCS, pp. 62–73 (1993)
4. Bellare, M., Rogaway, P.: Optimal asymmetric encryption. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 92–111. Springer, Heidelberg (1995)
5. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: FOCS, pp. 136–145 (2001)
6. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited. JACM 51(4), 557–594 (2004)
7. Canetti, R., Rabin, T.: Universal composition with joint state. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 265–281. Springer, Heidelberg (2003)
8. Feige, U., Lapidot, D., Shamir, A.: Multiple noninteractive zero knowledge proofs under general assumptions. SICOMP 29(1), 1–28 (1999)
9. Feige, U., Shamir, A.: Witness indistinguishable and witness hiding protocols. In: STOC, pp. 416–426 (1990)
10. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987)
11. Fischlin, M.: Communication-efficient non-interactive proofs of knowledge with online extractors. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 152–168. Springer, Heidelberg (2005)
12. Goldreich, O.: Foundations of Cryptography: Basic Tools. Cambridge University Press, Cambridge (2001)
13. Goldreich, O., Krawczyk, H.: On the composition of zero-knowledge proof systems. SIAM J. Comput. 25(1), 169–192 (1996)
14. Goldreich, O., Oren, Y.: Definitions and properties of zero-knowledge proof systems. J. Cryptology 7(1), 1–32 (1994)

15. Goldwasser, S., Kalai, Y.T.: On the (in)security of the Fiat-Shamir paradigm. In: FOCS, pp. 102–111 (2003)
16. Goldwasser, S., Kalai, Y.T.: On the impossibility of obfuscation with auxiliary input. In: FOCS, pp. 553–562 (2005)
17. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. *SIAM J. Comput.* 18(1), 186–208 (1989)
18. Hofheinz, D., Müller-Quade, J.: Universally composable commitments using random oracles. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 58–76. Springer, Heidelberg (2004)
19. Lynn, B.Y.S., Prabhakaran, M., Sahai, A.: Positive results and techniques for obfuscation. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 20–39. Springer, Heidelberg (2004)
20. Narayanan, A., Shmatikov, V.: Obfuscated databases and group privacy. In: ACM CCS, pp. 102–111 (2005)
21. 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)
22. Pass, R.: On deniability in the common reference string and random oracle model. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 316–337. Springer, Heidelberg (2003)
23. Pass, R.: Private communication (2005)
24. Schnorr, C.-P.: Efficient signature generation by smart cards. *J. Cryptology* 4(3), 161–174 (1991)
25. Unruh, D.: Random oracles and auxiliary input. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 205–223. Springer, Heidelberg (2007)
26. Wee, H.: On obfuscating point functions. In: STOC, pp. 523–532 (2005)

A Framework for Universally Composable Non-committing Blind Signatures

Masayuki Abe and Miyako Ohkubo

Information Sharing Platform Laboratories
NTT Corporation, Japan
{abe.masayuki,ookubo.miyako}@lab.ntt.co.jp

Abstract. A universally composable (UC) blind signature functionality demands users to commit to the message to be blindly signed. It is thereby impossible to realize in the plain model. We show that even non-committing variants of UC blind signature functionality remain not realizable in the plain model. We then characterize adaptively secure UC non-committing blind signatures in the common reference string model by presenting equivalent stand-alone security notions. We also present a generic construction based on conceptually simple Fischlin’s blind signature scheme.

1 Introduction

BACKGROUND. Since the introduction of blind signatures [9] vast number of papers are devoted to efficient constructions, security analysis, and extensions. Major applications include untraceable payment systems [9] and anonymous voting [10,13]. The standard notions of security for blind signature schemes in the stand-alone setting are *blindness* and *unforgeability* [9,22,18]. Universal composability (UC) framework [3] offers security in more general setting where other arbitrary protocols are running concurrently. It asserts that the properties provided by an idealized functionality retain even under general composition. A blind signature functionality is first suggested by Canetti in [4] and formally defined by Fischlin in [11] with a round-optimal realization in the common reference string (CRS) model. Kiayias and Zhou study adaptive security in [19].

In known blind signature functionalities, e.g., [11,19], a user *commits* to a message to request a signature. Then a signature is issued by the functionality *remotely* from the view of the signer. In [11], Fischlin pointed out that a UC blind signature protocol that realizes such a functionality implies a UC commitment protocol in the static corruption model and thus impossible to realize in the plain model [7]. A more formal argument is given by Lindell in [20,21]. A common idea for these arguments is that the existence of a simulator implies extraction of the input message and hence contradicts to the blindness.

Is there a hope to circumvent the above impossibility if the functionality is relaxed by giving up the commitment property? In some applications such as a simple e-cash or a coupon system, every message can be a random string that

the users do not need to know or fix in advance. Such applications only concern blindness and unforgeability. In [2], Buan, Gjøsteen, and Kråkmo presented a *non-committing* blind signature functionality where corrupt users no longer deposit messages. Thus there is no need to extract the messages for simulation. It was shown that such a non-committing blind signature functionality is realizable in the plain model and the presented security is equivalent to the unforgeability and weak blindness defined by Juels, Luby and Ostrovsky in [18].

OUR CONTRIBUTION. Somewhat contradictory, we show that universally composable non-committing blind signatures are still impossible in the plain model. Our proof shows that if the functionality provides blindness the presence of a simulator contradicts to the unforgeability in the plain model. Importantly, the positive result in [2] stands only in a restricted corruption model where the signer can be corrupted only after the key generation process. As stated in the paper, such a restriction is too strong that it is equivalent to incorporating a trusted party in the protocol. Our result holds for the most general corruption model. It is also pretty robust in the sense that it applies to wide variety of blind signature functionalities that formulate blindness in a reasonable way like all existing functionalities do.

Despite the negative result, non-committing blind signatures remain an interesting cryptographic object to study. The less demanding functionality would allow simple protocol designs in advanced models. This paper presents a thorough characterization of a non-committing blind signature functionality that is secure against adaptive adversaries without secure erasures. We prove that the properties captured by the functionality is equivalent to a pair of stand-alone security notions in the common reference string model, which are the standard unforgeability and a new strong notion of blindness which we call *equivocal simulation blindness*. We then decompose the equivocal simulation blindness to more handy notions called *session equivocality* and *signature equivocality* in a specific setting. We also show a generic construction. The simplicity of our framework can be highlighted when compared to the result on the adaptive security for the committing blind signatures [19].

Due to lack of space, most proofs are moved to the full version [1], which also includes results in the static corruption model.

2 Notations

All algorithms in this paper run in polynomial-time in the security parameter λ . By $y \leftarrow A(x; r)$ we mean that algorithm A is invoked with input x and uniformly chosen randomness r , and outputs something labeled as y . Randomness r may be omitted. By $(a, b) \leftarrow \langle A(x), B(y) \rangle$ we denote an execution of interactive Turing machines A and B on input x and y and with output a and b , respectively. When only one side of the output is of concern, we write $a \leftarrow \langle A(x), B(y) \rangle_L$ for the left side and $b \leftarrow \langle A(x), B(y) \rangle_R$ for the right side. We write $a[\omega] \leftarrow A$ when A has some extra output ω . The meaning of ω depends on the context and will be noted whenever this notation is used. For notations and notions related to the UC framework we refer to [6].

3 Blind Signature Schemes

3.1 Syntax and Standard Security Notions

A blind signature scheme BS in the common reference string model consists of five algorithms $BS = BS.\{\text{CrS}, \text{Key}, \text{User}, \text{Signer}, \text{Vrf}\}$. $BS.\text{CrS}$ is a common reference string generator. $BS.\text{Key}$ is a key generator. $BS.\text{User}$ is an interactive signature request algorithm and $BS.\text{Signer}$ is a signing algorithm. Interaction between $BS.\text{User}$ and $BS.\text{Signer}$ forms a signature generation protocol. $BS.\text{Vrf}$ is a signature verification algorithm. A blind signature scheme must provide *completeness* and *consistency*. Roughly, completeness is that for any (m, σ) made faithfully through $BS.\text{CrS}$, $BS.\text{Key}$, $BS.\text{User}$, and $BS.\text{Signer}$, verification algorithm $BS.\text{Vrf}$ outputs 1. Consistency is that $BS.\text{Vrf}$ outputs the same value for the same input (even for keys generated by an adversary). We refer to [14] for details and discussions on these properties. Two standard security notions are unforgeability and blindness as shown below.

Definition 1 (Unforgeability: UF). A blind signature scheme BS is unforgeable if $\text{Succ}_{F^*}^{uf}(\lambda) = \Pr[\text{Forge}_{F^*}^{BS}(\lambda) = 1]$ is negligible in λ for any algorithm F^* where $\text{Forge}_{F^*}^{BS}$ is the experiment shown below. F^* can access to the oracle arbitrary number of times concurrently.

Experiment $\text{Forge}_{F^*}^{BS}(\lambda)$:

$\Sigma \leftarrow BS.\text{CrS}(1^\lambda)$

$(vk, sk) \leftarrow BS.\text{Key}(\Sigma)$

$((m_1, \sigma_1), \dots, (m_{k+1}, \sigma_{k+1})) \leftarrow F^*(\langle \cdot, BS.\text{Signer}(\Sigma, sk) \rangle)(\Sigma, vk)$

Return 1 iff

completed $\leftarrow \langle \cdot, BS.\text{Signer}(\Sigma, sk) \rangle_R$ happens at most k times, and

$m_i \neq m_j$ for all $1 \leq i < j \leq k + 1$, and

$BS.\text{Vrf}(\Sigma, vk, m_i, \sigma_i) = 1$ for all $1 \leq i \leq k + 1$.

Strong unforgeability (sUF) is defined in the same way but requiring $(m_i, \sigma_i) \neq (m_j, \sigma_j)$ instead of $m_i \neq m_j$. This paper focuses on the above relatively weaker notion as it suffices for major applications.

Definition 2 (Blindness: BL). A blind signature scheme BS is blind if $\text{Adv}_{B^*}^{bl}(\lambda) = |\Pr[\text{Blind}_{B^*}^{BS}(\lambda, 0) = 1] - \Pr[\text{Blind}_{B^*}^{BS}(\lambda, 1) = 1]|$ is negligible in λ for any algorithm B^* where $\text{Blind}_{B^*}^{BS}$ is the experiment shown below.

$\text{Blind}_{B^*}^{BS}(\lambda, b)$:

$\Sigma \leftarrow BS.\text{CrS}(1^\lambda)$

$(vk, m_0, m_1) \leftarrow B^*(\Sigma)$

$\sigma_b \leftarrow \langle BS.\text{User}(\Sigma, vk, m_b), B^* \rangle_L$

$\sigma_{1-b} \leftarrow \langle BS.\text{User}(\Sigma, vk, m_{1-b}), B^* \rangle_L$

If $\sigma_0 = \perp$ or $\sigma_1 = \perp$ then set $\sigma_0 = \sigma_1 = \perp$.

Return $\tilde{b} \leftarrow B^*(\sigma_1, \sigma_0)$

For ease of notation, we represent algorithm B^* as stateful so that it implicitly takes over its internal state from the previous invocation every time it is invoked by the experiment. Only new inputs are explicitly presented in the description. This convention is applied to all algorithms denoted with asterisk (*) throughout the paper.

As observed in [17], the above definition captures the case where the adversary attempts to get useful information by aborting the sessions. [12] extends the notion in such a way that, when adversary B^* is given (\perp, \perp) at the end, it is given an extra piece of information that tells which session (the first or second or both) actually yields \perp in the user side. The results in this paper also holds with respect to the stronger notion of blindness.

4 UC Non-committing Blind Signatures

4.1 Functionality \mathcal{F}_{ncb}

Figure 1 illustrates our non-committing blind signature functionality \mathcal{F}_{ncb} . In the figure, v is a *deterministic* signature verification algorithm. Π is a description of a stateless signing algorithm. See [6] for remarks on running arbitrary algorithms in a functionality. As well as the ordinary signature functionality in [5] we formulate \mathcal{F}_{ncb} not to provide any security properties if an unregistered verification key is given as input to the signature generation and verification phases. See the discussion about the key management below.

The idea of using counters to enforce the unforgeability is the same as that in [2]. Due to the difference of the timing that the counters are increased, our formulation can live with the general communication model thoroughly controlled by the adversary while the one in [2] needs authenticated communication in its realization. Note that the bare signature functionality in [5] can be realized without authenticated channel because there is no link between the public-key and the identity of the signer and it is not a matter who issues a signature as long as the signature is valid.

NON-COMMITTING PROPERTY. Observe that input message m from a corrupt user is sent nowhere nor stored in the functionality. Thus \mathcal{S} working on behalf of a corrupt user can complete the signature generation process whatever m is. This formulation results in avoiding the need of extracting the message from the corrupt users.

UNFORGEABILITY. This property holds only while signer P_s is honest. Counter C_{cmpl} counts the number of completed signature generations in the signer's side while counter C_{valid} counts the number of valid signatures on distinct messages received by honest users with legitimate verification. The verification process accepts signatures on new messages only if $C_{\text{cmpl}} > C_{\text{valid}}$. From the specification, it is clear that $C_{\text{cmpl}} \geq C_{\text{valid}}$ always holds as long as the signer is honest. Thus unforgeability is guaranteed in the absolute sense. To capture weak unforgeability, C_{valid} is incremented only for unique messages in the signature generation

Key Generation : Given $(\text{KeyGen}, \text{sid})$ from a party P_s , verify that $\text{sid} = (P_s, \text{sid}')$ for some sid' . If not, then ignore. Else, forward $(\text{KeyGen}, \text{sid})$ to simulator \mathcal{S} . Then, on receiving $(\text{Generated}, \text{sid}, v, \Pi)$ from \mathcal{S} , send $(\text{Generated}, \text{sid}, v)$ to P_s and record (P_s, v, Π) . Let $C_{\text{cpl}} = C_{\text{valid}} = 0$, and Γ be empty. This phase must be completed only once and before other phases.

Signature Generation : On receiving $(\text{Request}, \text{sid}, \text{ssid}, v', m)$ for some m from P_u , send $(\text{Request}, \text{sid}, \text{ssid}, v')$ to \mathcal{S} and do the following.

- i. On receiving $(\text{Signed}, \text{sid}, \text{ssid})$ from \mathcal{S} , forward it to P_s . Set $C_{\text{cpl}} \leftarrow C_{\text{cpl}} + 1$.
- ii. On receiving $(\text{Received}, \text{sid}, \text{ssid})$ from \mathcal{S} , do as follows:
 - If P_u is honest and $v' = v$, then do as follows. If $(m, *, 1) \notin \Gamma$, set $C_{\text{valid}} \leftarrow C_{\text{valid}} + 1$. Compute $\sigma \leftarrow \Pi(m)$ and record $(m, \sigma, 1)$ to Γ . If $(m, \sigma, 0) \in \Gamma$, send an error message to signer P_s and halt. Send $(\text{Received}, \text{sid}, \text{ssid}, \sigma)$ to P_u .
 - Else if P_u is corrupt or $v' \neq v$, ask \mathcal{S} and forward P_u whatever received from \mathcal{S} .

Signature Verification : On receiving $(\text{Verify}, \text{sid}, \text{ssid}, v', m, \sigma)$ from some party P_v , set $\varphi = v'(m, \sigma)$ and do as follows.

1. If $v' \neq v$, set $f = \varphi$.
2. Else if $(m, \sigma, f') \in \Gamma$ for any f' , then set $f = f'$.
3. Else if P_s is corrupt or $(m, *, 1) \in \Gamma$, then set $f = \varphi$ and record (m, σ, f) to Γ .
4. Otherwise:
 - (a) If $C_{\text{cpl}} > C_{\text{valid}}$, then set $f = \varphi$ and $C_{\text{valid}} \leftarrow C_{\text{valid}} + f$.
 - (b) Otherwise, set $f = 0$.
 Then record (m, σ, f) to Γ .

Output $(\text{Verified}, \text{sid}, \text{ssid}, f)$ to P_v .

Player Corruption : On receiving corruption to P_u , send all inputs and outputs exchanged with P_u to simulator \mathcal{S} . Also send all randomness used in the evaluations of Π with respect to P_u .

Fig. 1. Non-committing blind signature functionality \mathcal{F}_{ncb}

process (see step (ii)). Strong unforgeability can be captured by removing conditions “if $(m, *, 1) \notin \Gamma$ ” and “or $(m, *, 1) \in \Gamma$ ” from the signature generation and verification phases respectively.

COMPLETENESS AND CONSISTENCY. If the signer and a user are not corrupted and the registered key is given as input to the signature generation phase, $(m, \sigma, 1)$ is recorded. The verification phase for such faithfully generated (m, σ) and registered v finds that record and always outputs $f = 1$. Thus completeness is captured. Consistency holds for free since algorithm v is deterministic. Limiting v to be deterministic loses generality but makes the exposition considerably simpler. For issues with respect to probabilistic verification algorithms see [5,6,14].

BLINDNESS. Important observations are; 1. Π is fixed *before* any sub-session for signature generation starts, 2. Π takes nothing but message m as input, and 3. Message m and $\Pi(m)$ are never sent to \mathcal{S} or P_s during the signature

generation phase. This formulation thereby assures that remotely computed σ is independent of the signature generation viewed by the signer. Such a mechanism, which we call *remote signing*, is suggested in [4] and employed by all known blind signature functionalities.

ON KEY MANAGEMENT. The “bare” signature functionality in [46] is formulated in such a way that it stores a single public-key in every session and the security properties are guaranteed only for the registered public-key. The functionality enjoys concise presentation and high modularity. We take over his approach to define \mathcal{F}_{ncb} . Namely, if unregistered v' is given as input to the signature generation or verification phase, \mathcal{F}_{ncb} behaves just as \mathcal{S} intended. So even though a user is honest, no security is guaranteed in such a case. (Recall that the environment can pass arbitrary v' to an honest user.) Accordingly, upper-level protocols that uses \mathcal{F}_{ncb} must be responsible to provide registered v to the honest users.

An alternative formulation would be to let \mathcal{F}_{ncb} to explicitly reject unregistered v' . It however results in incorporating a mechanism for distributing the correct public-key *within the blind signature protocol*. For instance, the protocol realizing \mathcal{F}_{ncb} may be constructed in \mathcal{F}_{ca} -hybrid model where \mathcal{F}_{ca} is the certificate-authority functionality [5] that serves *only* for the blind signature protocol. Though this kind of issue can be handled by the theorem of universal composition with joint state [8], we prefer \mathcal{F}_{ncb} to be basic for the sake of higher modularity.

In the literates, [11,2] implicitly follow the same approach as ours. They however define their functionality only for the case of receiving the registered public-key as input to the signature generation phase. It results in simpler presentation but eventually the details need to be provided with care. [19] shows more extended functionality such that it handles several public-keys under the same session-id and guarantees blindness for every set of signatures issued with the same public-key. This approach however suffers high complexity in its presentation.

VARIATIONS. \mathcal{F}_{ncb} in Fig. 11 notifies only the end of the signature generation process to the environment. It can be extended so that the environment can give the signer explicit approval or denial for starting the process by adding another round of interaction among \mathcal{S} , \mathcal{F}_{ncb} , and P_s . It is also possible to let the environment know about the abnormal termination of the protocol in the same way. These modifications do not affect to the results in this paper since they can be incorporated only by modifying the protocol wrapper in Section 5.2 accordingly.

4.2 Impossibility in the Plain Model

This section shows that \mathcal{F}_{ncb} cannot be realized without accessing to extra ideal functionalities or assuming some help from incorruptible parties. To make the statement meaningful, we consider non-trivial protocols where honest parties running the protocol with right inputs terminate and output something with noticeable probability.

Theorem 1. *There exists no non-trivial protocol that securely realizes \mathcal{F}_{ncb} in the plain model.*

Proof. We use \mathcal{S} to extract the remote signing function Π and use it to break the unforgeability in the real protocol. Recall that a forgery could never happen in the ideal model. Thus \mathcal{Z} can distinguish the ideal process and a real protocol execution by observing a successful forgery.

Suppose that there exists a non-trivial protocol π that realizes \mathcal{F}_{ncb} in the plain model. Recall that \mathcal{F}_{ncb} is invoked when it receives $(\text{KeyGen}, \text{sid})$ from a signer. It then outputs $(\text{Generated}, \text{sid}, v)$ to the signer. Protocol π works in the same way since it realizes \mathcal{F}_{ncb} . Let π_{KG} denote such a part of π that receives $(\text{KeyGen}, \text{sid})$ as input and outputs $(\text{Generated}, \text{sid}, v)$.

Consider a particular \mathcal{A}^* and \mathcal{Z}^* that behave in $\text{EXEC}_{\pi, \mathcal{A}^*, \mathcal{Z}^*}$ as follows. \mathcal{Z}^* first asks \mathcal{A}^* to corrupt the signer. \mathcal{Z}^* then runs π_{KG} with input $(\text{KeyGen}, \text{sid})$ and obtains $(\text{Generated}, \text{sid}, v)$. (Here, without loss of generality, we assume that π_{KG} can be run solely by the signer up to the moment $(\text{Generated}, \text{sid}, v)$ is output. See the discussion after the proof for generalization.) \mathcal{Z}^* then sends $(\text{KeyGen}, \text{sid})$ and v to \mathcal{A}^* and receives $(\text{Generated}, \text{sid}, v)$ from \mathcal{A}^* working on behalf of the corrupt signer. \mathcal{Z}^* then asks a signature on a message m by sending $(\text{Request}, \text{sid}, \text{ssid}, v, m)$ to an honest user. If \mathcal{A}^* is to join π on behalf of the signer to generate a signature, \mathcal{Z}^* takes over the role and completes the protocol by faithfully following π . The user eventually outputs $(\text{Received}, \text{sid}, \text{ssid}, \sigma)$. Finally \mathcal{Z}^* sends $(\text{Verify}, \text{sid}, \text{ssid}, v, m, \sigma)$ to a user and receives $(\text{Verified}, \text{sid}, \text{ssid}, f)$ as a result of verification. Observe that, even though the signer is corrupted, \mathcal{Z}^* simulates an honest signer by following π . Furthermore, due to the completeness and terminating property of π , \mathcal{Z}^* can complete signature generation with noticeable probability. If \mathcal{Z}^* completes, $f = 1$ appears at the end. Since π realizes \mathcal{F}_{ncb} , there exists a simulator \mathcal{S}^* for such \mathcal{A}^* and \mathcal{Z}^* . To successfully simulate \mathcal{A}^* , simulator \mathcal{S}^* has to send Π to \mathcal{F}_{ncb} before \mathcal{Z}^* sends $(\text{Request}, \text{sid}, \text{ssid}, v, m)$ to an honest user. Furthermore, with noticeable probability, $\Pi(m)$ must yield a valid signature accepted by protocol π .

Now we construct \mathcal{Z} that distinguishes $\text{EXEC}_{\pi, \mathcal{A}, \mathcal{Z}}$ and $\text{IDEAL}_{\mathcal{F}_{\text{ncb}}, \mathcal{S}, \mathcal{Z}}$ by using above \mathcal{S}^* as a subroutine. \mathcal{Z} first sends $(\text{KeyGen}, \text{sid})$ to the honest signer and receives $(\text{Generated}, \text{sid}, v)$. Then \mathcal{Z} starts simulating \mathcal{Z}^* . It asks \mathcal{S}^* to corrupt the simulated signer. Then it sends $(\text{KeyGen}, \text{sid})$ and v to \mathcal{S}^* and receives $(\text{Generated}, \text{sid}, v, \Pi)$ from \mathcal{S}^* on behalf of \mathcal{F}_{ncb} . Now \mathcal{Z} computes $\sigma \leftarrow \Pi(m)$ for some m . It then sends $(\text{Verify}, \text{sid}, \text{ssid}, v, m, \sigma)$ to a verifier and receives $(\text{Verified}, \text{sid}, \text{ssid}, f)$. The output of \mathcal{Z} is f .

Let us evaluate \mathcal{Z} . Suppose that \mathcal{Z} is in $\text{EXEC}_{\pi, \mathcal{A}, \mathcal{Z}}$. \mathcal{Z} simulates \mathcal{Z}^* perfectly for \mathcal{S}^* . In particular v in this case is generated honestly by π just as \mathcal{Z}^* does. So \mathcal{S}^* outputs $(\text{Generated}, \text{sid}, v, \Pi)$ as expected. Then with noticeable probability such Π yields σ that passes the verification protocol of π . Thus $f = 1$ happens with noticeable probability in this case. Next suppose that \mathcal{Z} is in $\text{IDEAL}_{\mathcal{F}_{\text{ncb}}, \mathcal{S}, \mathcal{Z}}$. In this case, v is generated by \mathcal{S} . If it is distinguishable from the one observed in $\text{EXEC}_{\pi, \mathcal{A}, \mathcal{Z}}$, \mathcal{Z} distinguishes $\text{EXEC}_{\pi, \mathcal{A}, \mathcal{Z}}$ and $\text{IDEAL}_{\mathcal{F}_{\text{ncb}}, \mathcal{S}, \mathcal{Z}}$ on that basis. If it is indistinguishable, \mathcal{S}^* outputs $(\text{Generated}, \text{sid}, v, \Pi)$ as well

as in the previous case. Since no signature generation process is completed in $\text{IDEAL}_{\mathcal{F}_{\text{ncb}}, \mathcal{S}, \mathcal{Z}}$ and \mathcal{F}_{ncb} provides absolute unforgeability, \mathcal{F}_{ncb} rejects σ generated by \mathcal{H} . Thus $f = 0$ for this case. Accordingly \mathcal{Z} distinguishes $\text{EXEC}_{\pi, \mathcal{A}, \mathcal{Z}}$ and $\text{IDEAL}_{\mathcal{F}_{\text{ncb}}, \mathcal{S}, \mathcal{Z}}$ with noticeable probability. \blacksquare

An essential point is that \mathcal{F}_{ncb} demands \mathcal{S} to extract \mathcal{H} even from a corrupt signer for the sake of blindness. But the successful extraction of \mathcal{H} contradicts to the unforgeability. The situation is very similar to the case of UC commitments [7] where the message from a corrupt committer must be extracted for the sake of binding property, and the successful extraction contradicts to the hiding property.

The proof does not go through if protocol π involves incorruptible trusted parties or any extra ideal functionalities. The point is that \mathcal{Z}^* should be able to run π_{KG} by itself so that the distribution of v is solely under the control. This allows \mathcal{Z} to simulate \mathcal{Z}^* simply by sending v generated outside of \mathcal{Z} . If π_{KG} involves parties other than the signer, \mathcal{Z}^* corrupts them before they send off any message and simulate them honestly by following π_{KG} . When \mathcal{Z} simulates \mathcal{Z}^* , these corrupted parties are simulated by following the behavior of the real uncorrupted players \mathcal{Z} is working with.

5 Characterization

5.1 Blindness Based on Simulatability

The following new notion called *simulation blindness* assures that the signature generation protocol can be executed without knowing the message. Similarly, the resulting signature can be generated without involving any information from the protocol run. To capture adaptive security, we require *state reconstruction property*. We use the term *equivocal* when a notion involves state reconstruction property.

Definition 3 (Equivocal Simulation Blindness: EqSimBLND). A blind signature scheme BS is equivocal simulation blind if there exists a set of algorithms $SIM = SIM.\{Crs, User, Sig, State\}$ such that $SIM.User$ and $SIM.State$ can be stateful and $SIM.Sig$ must be stateless, and advantage $\text{Adv}_{D^*}^{\text{eqsib}}(\lambda) = |\Pr[\text{EqSimBL}_{D^*}^{BS}(\lambda, 0) = 1] - \Pr[\text{EqSimBL}_{D^*}^{BS}(\lambda, 1) = 1]|$ is negligible in λ for any D^* , where $\text{EqSimBL}_{D^*}^{BS}(\lambda, b)$ is the following experiment. Oracles are accessible in arbitrary manner.

$$\begin{array}{l|l}
 \text{EqSimBL}_{D^*}^{BS}(\lambda, 1) : & \\
 \Sigma \leftarrow BS.Crs(1^\lambda) & \mathcal{O}_1(\Sigma, vk, m) \\
 vk \leftarrow D^*(\Sigma) & \sigma \leftarrow \langle BS.User(\Sigma, vk, m; r), D^* \rangle_L \\
 \tilde{b} \leftarrow D^{*\mathcal{O}_1(\Sigma, vk, \cdot)} & \text{Output } (\sigma, r) \\
 \text{Return } \tilde{b} &
 \end{array}$$

$\mathbf{EqSimBL}_{D^*}^{BS}(\lambda, 0) :$ $(\Sigma, t) \leftarrow \mathit{SIM.Crs}(1^\lambda)$ $vk \leftarrow D^*(\Sigma)$ $\tilde{b} \leftarrow D^*\mathcal{O}_0(\Sigma, vk, \cdot, t)$ $\text{Return } \tilde{b}$	$\mathcal{O}_0(\Sigma, vk, m, t)$ $\delta[\omega_u] \leftarrow \langle \mathit{SIM.User}(\Sigma, vk, t), D^* \rangle_L$ $\sigma[\omega_s] \leftarrow \mathit{SIM.Sig}(\Sigma, vk, m, t)$ $r \leftarrow \mathit{SIM.State}(\omega_u, \omega_s)$ $\text{If } \delta = 0, \text{ then set } \sigma = \perp.$ $\text{Output } (\sigma, r)$
---	--

Denoted by ω_u and ω_s are the state information of $\mathit{SIM.User}$ and $\mathit{SIM.Sig}$, respectively.

Note that $\mathit{SIM.State}$ is supposed to simulate the randomness even for the case where the interaction between $\mathit{SIM.User}$ and D^* is terminated abnormally. $\mathit{SIM.State}$ can see how the interaction is terminated by seeing the state information ω_u .

It would be more useful if we could present separate notions of simulatability for simulating the view of sessions by $\mathit{SIM.User}$ and the signatures by $\mathit{SIM.Sig}$. We call the notions *session equivocality* and *signature equivocality*. It is however not a proper way in general. Since $\mathit{SIM.User}$ and $\mathit{SIM.Sig}$ uses the same trapdoor as input and they may give negative influence each other when they are used at the same time. We thus consider a special case where trapdoors are separated like (t_1, t_2) , and $\mathit{SIM.User}$ (and $\mathit{SIM.Sig}$) can be run only with t_1 (and t_2 , respectively). With respect to the separate trapdoor generator we define two notions of simulatability.

Definition 4 (Separable Trapdoor Generator). *SIM.Crs is a separable trapdoor generator if it outputs $(\Sigma, (t_1, t_2))$ such that Σ is indistinguishable from those generated by $\mathit{BS.Crs}$ with negligible advantage, say $\mathbf{Adv}_{C^*}^{crs}$, for any algorithm C^* .*

Definition 5 (Signature Equivocality: SigEq). *A blind signature scheme BS is signature equivocal if there exists algorithms $\mathit{SIM.Sig}$ and $\mathit{SIM.SigState}$ such that advantage function $\mathbf{Adv}_{A^*}^{\text{sigEq}}(\lambda) = |\Pr[\mathbf{SigEQ}_{A^*}^{BS}(\lambda, 0) = 1] - \Pr[\mathbf{SigEQ}_{A^*}^{BS}(\lambda, 1) = 1]|$ is negligible in security parameter λ for any A^* , where $\mathbf{SigEQ}_{A^*}^{BS}(\lambda, b)$ is the following experiment.*

$\mathbf{SigEQ}_{A^*}^{BS}(\lambda, b) :$ $(\Sigma, (t_1, t_2)) \leftarrow \mathit{SIM.Crs}(1^\lambda)$ $vk \leftarrow A^*(\Sigma)$ $\tilde{b} \leftarrow A^*\mathcal{O}_b(\Sigma, vk, \cdot, t_1)$ $\text{Return } \tilde{b}$	$\mathcal{O}_1(\Sigma, vk, m, t_1)$ $\sigma \leftarrow \langle \mathit{BS.User}(\Sigma, vk, m; r_1 r_2), A^* \rangle_L$ $\text{Output } (\sigma, r_1 r_2)$ $\mathcal{O}_0(\Sigma, vk, m, t_1)$ $\sigma[\theta] \leftarrow \langle \mathit{BS.User}(\Sigma, vk, m; r_1 r_2), A^* \rangle_L$ $\sigma'[\omega_s] \leftarrow \mathit{SIM.Sig}(\Sigma, vk, m, t_1)$ $r'_1 \leftarrow \mathit{SIM.SigState}(\theta, \omega_s)$ $\text{If } \sigma = \perp, \text{ then } \sigma' = \perp, r'_1 = r_1.$ $\text{Output } (\sigma', r'_1 r_2)$
--	--

Symbol θ is the transcript observed by $\mathit{BS.User}$, and ω_s is a state information of $\mathit{SIM.Sig}$.

Definition 6 (Session Equivocality: SesEq). A blind signature scheme BS is session equivocal if there exists algorithms $SIM.User$ and $SIM.SesState$ such that advantage function $\text{Adv}_{E^*}^{\text{seseq}}(\lambda) = |\Pr[\text{SesEQ}_{E^*}^{BS}(\lambda, 0) = 1] - \Pr[\text{SesEQ}_{E^*}^{BS}(\lambda, 1) = 1]|$ is negligible in λ for any algorithm E^* , where experiment $\text{SesEQ}_{E^*}^{BS}$ is the following.

$\begin{aligned} & \text{SesEQ}_{E^*}^{BS}(\lambda, b) : \\ & (\Sigma, (t_1, t_2)) \leftarrow SIM.Crs(1^\lambda) \\ & vk \leftarrow E^*(\Sigma, t_1) \\ & \tilde{b} \leftarrow E^{*\mathcal{O}_b(\Sigma, vk, \cdot, t_2)} \\ & \text{Return } \tilde{b} \end{aligned}$	$\begin{aligned} & \mathcal{O}_1(\Sigma, vk, m, t_2) : \\ & \langle BS.User(\Sigma, vk, m; r_1 r_2), E^* \rangle \\ & \text{Return } r_2 \\ \\ & \mathcal{O}_0(\Sigma, vk, m, t_2) : \\ & \delta[\omega_u] \leftarrow \langle SIM.User(\Sigma, vk, t_2), E^* \rangle_L \\ & r_2 \leftarrow SIM.SesState(\omega_u, m) \\ & \text{Return } r_2 \end{aligned}$
--	--

Oracle \mathcal{O}_b receives a message m from E^* and interacts with E^* . Symbol ω_u is the state information of $SIM.User$.

In Definition 5 it is assumed that randomness r used in $BS.User$ can be separated into two parts r_1 and r_2 . An intuition is that r_2 is used while interacting with the signer and r_1 is used after receiving the final message from the signer for computing the output signature. This treatment does not lose generality as one can set either part as empty. Regarding Definition 6 we stress that the messages and the resulting signatures are not given to E^* . Also note that trapdoor t_1 is given to E^* .

We now show relations between the standard blindness and simulation blindness. Since simulation blindness captures blindness in a very strong way, it seems natural that the following lemma holds. Proofs for the following lemmas are in 11.

Lemma 1 (EqSimBLND \Rightarrow BL). If BS is equivocal simulation blind then it is blind.

Proof is done in a standard way. We construct D^* that successfully breaks equivocal simulation blindness by using B^* that breaks blindness.

Regarding the reverse direction, we do not know if blindness solely implies simulation blindness or not. We however can show that there exists a scheme that is blind and unforgeable but not simulation blind. Namely, for the schemes that provide both blindness and unforgeability the simulation blindness is a strictly stronger notion than blindness. This implication is limited but sufficiently meaningful since we are interested in schemes that provide both blindness and unforgeability. Proof can be done in the similar way as that of Theorem 11.

Lemma 2 (BL \wedge UF $\not\Rightarrow$ EqSimBLND). There exists BS that is blind and unforgeable but not equivocal simulation blind.

The following lemma states that it suffices to consider simulatability about sessions and signatures individually when trapdoors are separable for each purpose.

Lemma 3 (SesEq \wedge SigEq \Rightarrow EqSimBLND). If BS has a separable trapdoor generator and is signature equivocal and session equivocal with respect to the generator then BS is equivocal simulation blind.

Proof is done through three steps of game transformations starting from $\mathbf{EqSimBL}_{D^*}^{\text{BS}}(\lambda, 1)$ to $\mathbf{EqSimBL}_{D^*}^{\text{BS}}(\lambda, 0)$.

5.2 Protocol Wrapper $\mathbf{Wrap}()$

In Fig. 2, we show how to transform a blind signature scheme BS into a blind signature protocol by applying a simple wrapper algorithm, $\mathbf{Wrap}()$. The resulting protocol $\mathbf{Wrap}(\text{BS})$ is in the \mathcal{F}_{CRS} -hybrid model where \mathcal{F}_{CRS} is the CRS generation and distribution functionality whose output distribution is defined by BS .

Blind Signature Protocol $\mathbf{Wrap}(\text{BS})$ in \mathcal{F}_{CRS} -model

Key Generation: Upon receiving $(\text{KeyGen}, \text{sid})$ from the environment \mathcal{Z} , a party P_s verifies that $\text{sid} = (P_s, \text{sid}')$ for some sid' . If not, do nothing. Else, P_s derives CRS Σ from \mathcal{F}_{CRS} , computes $(vk, sk) \leftarrow \text{BS.Key}(\Sigma)$ and outputs $(\text{Generated}, \text{sid}, v)$ where $v(m, \sigma) = \text{BS.Vrf}(\Sigma, vk, \sigma, m)$.

Blind Signature Generation: Party P_u and P_s do the following.

P_u -side: On receiving $(\text{Request}, \text{sid}, \text{ssid}, v', m)$ from \mathcal{Z} , derive Σ from \mathcal{F}_{CRS} , send $(\text{Request}, \text{sid}, \text{ssid}, v')$ to P_s , invoke $\text{BS.User}(\Sigma, vk', m)$, and interact with P_s . Take vk' out from v' . If BS.User outputs σ such that $\text{BS.Vrf}(\Sigma, vk', \sigma, m) = 1$, then output $(\text{Received}, \text{sid}, \text{ssid}, \sigma)$.

P_s -side: On receiving $(\text{Request}, \text{sid}, \text{ssid}, v')$ from a user P_u , get Σ from \mathcal{F}_{CRS} , invoke $\text{BS.Signer}(\Sigma, sk)$ and interacts with P_u . If BS.Signer outputs completed, then output $(\text{Signed}, \text{sid}, \text{ssid})$.

Signature Verification: On receiving $(\text{Verify}, \text{sid}, \text{ssid}, v', m, \sigma)$ from \mathcal{Z} , a party P_v derives Σ from \mathcal{F}_{CRS} , takes vk' from v' , computes $f \leftarrow \text{BS.Vrf}(\Sigma, vk', \sigma, m)$, and outputs $(\text{Verified}, \text{sid}, \text{ssid}, f)$.

Common Reference Functionality \mathcal{F}_{CRS}

CRS Generation: On receiving $(\text{CrsGen}, \text{sid})$, \mathcal{F}_{CRS} computes $\Sigma \leftarrow \text{BS.Crs}(1^\lambda)$ for the first time and returns Σ . Simply return the same Σ for further requests.

Fig. 2. UC blind signature protocol transformed from stand-alone scheme BS

Note that the resulting protocol does not implement any mechanism to verify the given verification algorithm v' . It works as intended if $v' = v$ but no security is guaranteed for the user if $v' \neq v$. Also note that the signer ignores v' given from the user and uses the genuine secret key sk .

5.3 Equivalence

Theorem 2 ($\text{UF} \wedge \text{EqSimBLND} \Leftrightarrow \mathcal{F}_{\text{ncb}}$). *Protocol $\mathbf{Wrap}(\text{BS})$ securely realizes \mathcal{F}_{ncb} with respect to adaptive adversaries if and only if BS is unforgeable and equivocal simulation blind.*

“If” direction is proven by constructing a simulator, \mathcal{S} , that uses \mathcal{A} as a black-box. To run \mathcal{A} properly, \mathcal{S} simulates entities and their communication in

$\text{EXEC}_{\pi, \mathcal{A}, \mathcal{Z}}^{\mathcal{F}_{\text{ncb}}^{\text{crs}}}$. We then apply the game transformation technique starting from $\text{IDEAL}_{\mathcal{F}_{\text{ncb}}, \mathcal{S}, \mathcal{Z}}(\lambda, a)$ as Game 0. Game 1 removes the use of simulation algorithms SIM.Crs , SIM.User , SIM.Sig , and SIM.State from \mathcal{F}_{ncb} and \mathcal{S} . The difference is negligible due to the simulation blindness. Game 2 then modifies the verification process of \mathcal{F}_{ncb} so that it no longer care for the counters. This modification is justified by the unforgeability. Game 3 further modifies the verification process so that it completely follows the verification function. Justification is due to the completeness and consistency. Game 4 then modifies \mathcal{F}_{ncb} so that it does not record the signed messages any more. It is justified by the completeness and consistency again. Finally, Game 5 removes unused actions in \mathcal{F}_{ncb} and \mathcal{S} . This is just cosmetic to make sure that \mathcal{F}_{ncb} and \mathcal{S} do nothing but executing the real protocol. Thus Game 5 is equivalent to $\text{EXEC}_{\pi, \mathcal{A}, \mathcal{Z}}^{\mathcal{F}_{\text{ncb}}^{\text{crs}}}(\lambda, a)$.

“Only if” direction is more intricate. First, assuming that BS is not simulation blind, we show that, for any \mathcal{S} , there exists successful \mathcal{Z} . Second, assuming that BS is simulation blind but forgeable, we construct successful \mathcal{Z} that is not fooled by any \mathcal{S} . For the first part, we construct simulation algorithms SIM.Crs , SIM.User , SIM.Sig and SIM.State by using \mathcal{S} as a subroutine. For such simulation algorithms there exists adversary D^* that breaks simulation blindness since we assumed that BS is not simulation blind. Then we use such D^* to construct \mathcal{Z} . A tricky issue in constructing these simulation algorithms is that they do not share the internal state. Since individual copy of \mathcal{S} is run independently in these functions, it would output different CRS-es and public-keys. Our idea is to use the trapdoor as a container of the randomness given to \mathcal{S} so that every simulation algorithm can give the same randomness to \mathcal{S} . In this way, every copy of \mathcal{S} works on the same CRS and public-key so that all simulation algorithms work consistently. A formal proof is given in [11].

6 A Generic Construction

6.1 Overview

Our starting point is the “basic” blind signature scheme by Fischlin [13]. In his scheme, a user commits to message m by sending a commitment c and the signer returns a bare signature s on c . Then the user computes a final signature σ which actually is a non-interactive zero-knowledge proof of knowledge about the message m and the valid signature s . Unforgeability is based on the binding property of the commitment and the unforgeability of the bare signature scheme and the knowledge soundness of NIZK. Blindness is from the hiding property of the commitment scheme and the zero-knowledge property of NIZK. By $\text{BS}_{\mathcal{G}}$ we denote this generic scheme. When transformed by our wrapper, $\text{Wrap}(\text{BS}_{\mathcal{G}})$ securely realizes non-committing blind signature functionality \mathcal{F}_{ncb} with respect to *static* adversaries. (See [11] for details.) It is a surprise that such a conceptually simple scheme can provide universal composability even though the adversary is limited to be static.

An essential issue to handle adaptive security is the state reconstruction. Looking at the structure of $\text{BS}_{\mathcal{G}}$, the session equivocality can be easily achieved

by replacing the commitment scheme with a *trapdoor* commitment scheme. (In fact, with such a small modification to BS_G , the resulting $\text{Wrap}(\text{BS}_G)$ provides adaptive UC security *in the erasure model*.) On the other hand, the signature equivocability is not generally possible there. Recall that a signature is simulated by the zero-knowledge simulator. It therefore can be the case that there exists no randomness that is consistent to a real witness. To overcome this problem, we consider eliminating the use of zero-knowledge simulator by providing a correct witness to the proof system through the simulation of the bare signature in the signer-side. Namely, we make the signer's signing algorithm to be simulatable by using a signature scheme in the CRS model so that valid signatures can be created with the trapdoor of the CRS. In this way, we can always provide a witness to the proof system used in the user-side algorithm. Now, witness indistinguishability of the proof system assures that the same proof could have been created from any other witnesses. Accordingly, a consistent randomness always exists. This particular structure is suggested in [17] for the purpose of removing the CRS in the stand-alone model. We will take advantage of the structure for achieving adaptive security.

6.2 Building Blocks

–NIWI (*Non-interactive Witness Indistinguishable Proof System*). It is a non-interactive witness indistinguishable proof system of knowledge when the CRS is generated in the regular way. By NIWI.Crs , NIWI.Prf and NIWI.Vrf , we denote the CRS generation function, the proof generation function and the verification function, respectively. Additionally it must allow state reconstruction when the CRS is simulated. Namely, one can reconstruct a consistent randomness for a given witness and a valid transcript. The Groth-Sahai proof system [16], the GS proof system for short, meets these requirements under SXDH or DLIN assumption. It unfortunately does not work for any NP statement but works efficiently for relations represented by bilinear products. We thus need to choose other building blocks so that they fit to the GS proof system for instantiation.

–TC (*Trapdoor Commitment Scheme*). It is a standard trapdoor commitment scheme. By TC.Key , TC.Com and TC.Vrf , we denote the key generation function, the commitment function, and the verification function. There are two more functions such that one generates a random commitment and the other opens the commitment to an arbitrary value by using the trapdoor generated by TC.Key . See [1] an instantiation that works well with GS proof system under the SXDH assumption.

–SSIG (*Simulatable Signature Scheme*). It is a signature scheme in the CRS model with a special property such that valid signatures can be computed from the public-key and the trapdoor bind the CRS. By SSIG.Crs and SSIG.Key , we denote the CRS generation function and the key generation function. SSIG.Key takes the CRS and outputs a signing key and a verification key. Besides the signature generation function SSIG.Sign , there is a signature simulation function SSIG.Sim that generates valid signatures by using the public-key and the

trapdoor generated by SSIG.Crs . It is stressed that the simulated signatures must pass the verification by the verification function SSIG.Vrf but it is not demanded that they are indistinguishable from the real ones. Similarly, unforgeability is the standard unforgeability against chosen message attacks. In particular, the adversary is not given simulated signatures.

Any standard signature scheme can be turned into a simulatable one in an unconditional way as follows. Generate two key pairs by running the key generation algorithm twice independently. The first key pair is used as the CRS and the trapdoor while the second pair is used as the verification and signing key. Normal signing is done by using the second key. Simulation is done by the first key. A signature is accepted if it passes the original verification predicate with respect to either of the keys.

To fit to the other building blocks, SSIG must be able to sign group elements and the verification predicate must be represented as a product of pairings. For such a signature scheme a feasibility result based on DLIN assumption can be seen in [15].

6.3 The Scheme

The CRS generation function BS.Crs computes $(\Sigma_{\text{wi}}, t_{\text{wi}}) \leftarrow \text{NIWI.Crs}(1^\lambda)$, $(\Sigma_{\text{tc}}, t_{\text{tc}}) \leftarrow \text{TC.Key}(1^\lambda)$, and $(\Sigma_{\text{ssig}}, t_{\text{ssig}}) \leftarrow \text{SSIG.Crs}(1^\lambda)$, and outputs $\Sigma = (\Sigma_{\text{wi}}, \Sigma_{\text{bc}}, \Sigma_{\text{ssig}})$. Key generation function BS.Key is the same as SIG.Key , which outputs vk and sk . The signature generation protocol is illustrated in Fig. 3. The proof system NIWI proves the following relation between witness $w = (s, c, z)$ and instance $x = (vk, \Sigma_{\text{tc}}, \Sigma_{\text{ssig}}, m)$:

$$\text{TC.Vrf}(\Sigma_{\text{tc}}, c, m, z) = 1 \wedge \text{SSIG.Vrf}(\Sigma_{\text{ssig}}, vk, c, s) = 1$$

Verification function BS.Vrf takes $((\Sigma_{\text{wi}}, \Sigma_{\text{tc}}, \Sigma_{\text{ssig}}), vk, \sigma, m)$ as input and outputs $\varphi \in \{0, 1\}$ such that $\varphi \leftarrow \text{NIWI.Vrf}(\Sigma_{\text{wi}}, (vk, \Sigma_{\text{tc}}, \Sigma_{\text{ssig}}, m), \sigma)$.

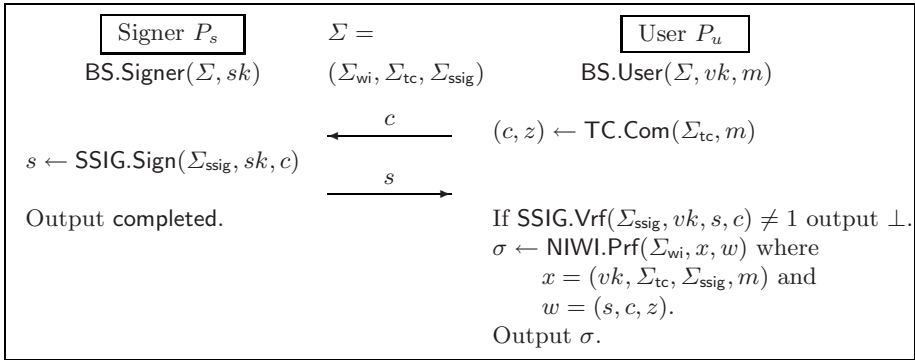


Fig. 3. Generic blind signature scheme BS_S . The signature generation protocol.

Theorem 3. *Protocol $\text{Wrap}(BS_S)$ securely realizes \mathcal{F}_{ncb} in the \mathcal{F}_{crs} -hybrid model with respect to adaptive adversaries without erasures.*

We claim that the scheme is session equivocal and signature equivocal. Observe that setting $t_1 = (t_{\text{wi}}, t_{\text{ssig}})$ and $t_2 = (t_{\text{tc}})$ forms separated trapdoors. Session equivocality is proven by constructing SIM.User and SIM.SesState by using the trapdoor property of TC. Signature equivocality can be shown by constructing SIM.Sig and SIM.SigState by using the simulation property of SSIG and state reconstructability of NIWI. Thus from Lemma 3, we can say that the scheme is equivocal simulation blind. We then argue that the scheme is unforgeable due to the binding property of TC, the unforgeability of SSIG and the proof of knowledge property of NIWI. Finally Theorem 3 is applied to complete the proof of Theorem 2.

References

1. Abe, M., Ohkubo, M.: A framework for universally composable non-committing blind signatures. IACR ePrint Archive 2009 (2009)
2. Buan, A.B., Kråkmo, K.G.L.: Universally composable blind signatures in the plain model. IACR ePrint Archive 2006/405 (2006)
3. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: 42nd FOCS, pp. 136–145 (2001)
4. Canetti, R.: On universally composable notions of security for signature, certification and authentication. IACR ePrint Archive 2003/239 (2003)
5. Canetti, R.: Universally composable signatures, certification and authentication. In: 17th Computer Security Foundations Workshop, CSFW (2004); Revised version available in IACR ePrint archive 2003/239
6. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. IACR ePrint Archive 2000/067. 2nd version updated on 13 Dec (2005)
7. Canetti, R., Fischlin, M.: Universally composable commitments. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 19–40. Springer, Heidelberg (2001)
8. Canetti, R., Rabin, T.: Universal composition with joint state. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 265–281. Springer, Heidelberg (2003)
9. Chaum, D.: Blind signatures for untraceable payments. In: CRYPTO 1982, pp. 199–204. Prentice Hall (1982)
10. Chaum, D.L.: Elections with unconditionally-secret ballots and disruptions equivalent to breaking RSA. In: Günther, C.G. (ed.) EUROCRYPT 1988. LNCS, vol. 330, pp. 177–182. Springer, Heidelberg (1988)
11. Fischlin, M.: Round-optimal composable blind signatures in the common reference model. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 60–77. Springer, Heidelberg (2006)
12. 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)
13. Fujioka, A., Okamoto, T., Ohta, K.: A practical secret voting scheme for large scale elections. In: Zheng, Y., Seberry, J. (eds.) AUSCRYPT 1992. LNCS, vol. 718, pp. 244–251. Springer, Heidelberg (1993)
14. Garay, J., Kiayias, A., Zhou, H.-S.: Sound and fine-grain specification of cryptographic tasks. IACR ePrint Archive 2008/132 (2008)

15. 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)
16. 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)
17. Hazay, C., Katz, J., Koo, C., Lindell, Y.: Concurrently-secure blind signatures without random oracles or setup assumptions. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 323–341. Springer, Heidelberg (2007)
18. Juels, A., Luby, M., Ostrovsky, R.: Security of blind digital signatures. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 150–164. Springer, Heidelberg (1997)
19. Kiayias, A., Zhou, H.: Equivocal blind signatures and adaptive UC-security. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 340–355. Springer, Heidelberg (2008)
20. Lindell, Y.: Bounded-concurrent secure two-party computation without setup assumptions. In: STOC, pp. 683–692. ACM, New York (2003)
21. Lindell, Y.: Lower bounds and impossibility results for concurrent self composition. *Journal of Cryptology* 21(2), 200–249 (2008)
22. Pointcheval, D., Stern, J.: Security arguments for digital signatures and blind signatures. *Journal of Cryptology* 13(3), 339–360 (2000)

Cryptanalysis of the Square Cryptosystems

Olivier Billet and Gilles Macario-Rat

Orange Labs, Issy-les-Moulineaux, France

billet@eurecom.fr, gilles.macariorat@orange-ftgroup.com

Abstract. Following the cryptanalyses of the encryption scheme HFE and of the signature scheme SFLASH, no serious alternative multivariate cryptosystems remained, except maybe the signature schemes UOV and HFE⁻. Recently, two proposals have been made to build highly efficient multivariate cryptosystems around a quadratic *internal* transformation: the first one is a signature scheme called square-vinegar and the second one is an encryption scheme called square introduced at CT-RSA 2009.

In this paper, we present a total break of both the square-vinegar signature scheme and the square encryption scheme. For the practical parameters proposed by the authors of these cryptosystems, the complexity of our attacks is about 2^{35} operations. All the steps of the attack have been implemented in the Magma computer algebra system and allowed to experimentally assess the results presented in this paper.

1 Introduction

There are mainly two motivations behind the construction of multivariate cryptosystems. The original one is to provide alternatives to the asymmetric schemes RSA and those based on Discrete Logarithm problems which are connected to number theoretic problems. Multivariate cryptosystems are instead connected to the hardness of solving randomly chosen systems of multivariate equations over a finite field, a problem which is NP-complete even in the case of quadratic polynomials defined over GF(2) when there are at least two such polynomials in the system. Moreover, this problem seems to be hard not only for very special instances but also on the average. Another incentive to develop multivariate cryptosystems is the expected efficiency that they might offer, a property that would be highly appreciated for constrained environments such as RFIDs and other embedded devices. Finally, some people argue about the fact that, contrary to the problem of factorisation and that of solving discrete logarithms [23], no quantum algorithm is known for the problem of solving sets of randomly chosen multivariate equations.

After the introduction of the C^* cryptosystem by Matsumoto and Imai in [13,16], there have been several other proposals. Among the most famous ones are certainly HFE (Hidden Field Equations) and SFLASH which can be thought of as two ways of generalising the C^* scheme. Some heuristic design principles have followed. A major one, which has been originally suggested by Shamir in [21], is to remove some equations from the public mapping in the case of signature schemes; this principle has proven to be successful in thwarting Patarin's

attack [17] against C^* (an attack that can be viewed as a preliminary to Gröbner basis attacks). Another one consists in adding a new set of variables to perturb the analysis as in the UOV (Unbalanced Oil and Vinegar) signature scheme [14].

Two of the most promising proposals, SFLASH and HFE have been cryptanalyzed during the last years. Some HFE instances have been shown to succumb to Gröbner basis attacks in [7] and the complexity of such attack has been argued to be quasi-polynomial in [12]. SFLASH has been entirely broken: the missing equations (due to the minus transformation) can be recovered in most cases as explained in [6] and the secret key of the resulting C^* scheme can be recovered following the cryptanalysis described in [10]. In this context, two new proposals were based on internal transformations that are not only quadratic on the base field, but also on the extension field: a signature scheme called square-vinegar was proposed in [2] and an encryption scheme called square appears in [4].

Our Results. In this paper, we expose a total break of both the square-vinegar signature and the square encryption proposals from a theoretical point of view as well as from a practical point of view. We indeed describe how to recover an equivalent secret key for both cryptosystems given the public key alone. For the parameters recommended by the authors, our attacks complete in a few minutes on a standard PC. These cryptanalyses also represent a theoretical break of the schemes as, under some reasonable assumptions, their complexity is shown to be polynomial with respect to the security parameter: the attacks have a time complexity of $O(\log^2(q)n^6)$ since they rely on standard linear algebra on n^2 unknowns over a finite field of size q and n is typically small because the time complexity of the public computation (signature or encryption) is $O(n^3)$. The attacks are sequences of steps including the discovery of new algebraic invariants leaking from the public key, a careful analysis of these invariants to sort out vinegar unknowns from the standard ones. We additionally implemented Magma [3] programs that were used to verify each of the steps of the cryptanalyses and to perform the attacks against the different sets of parameters recommended by the designers of the square encryption and square-vinegar signature schemes. Their source code is given in the appendix.

2 The Square Cryptosystems

The square cryptosystems are based on design ideas taken from both the HFE cryptosystem and the UOV cryptosystem. However, an important property of the square cryptosystems is that they are defined over fields of *odd* characteristic: as their internal transformations are quadratic, the systems would be linear over fields of characteristic 2. We begin by a brief reminder on HFE and UOV before proceeding to the description of the square cryptosystems themselves.

2.1 The HFE Cryptosystem

The HFE cryptosystem has been proposed by Patarin in [18] as a possible generalisation (and strengthening) of the C^* scheme proposed by Matsumoto and

Imai in [16]. Indeed C^* was broken by Patarin [17], whereas the best attack against HFE are Gröbner basis attacks which complexity was argued to be quasi-polynomial [7][12]. HFE is called hidden field equation because its internal transformation is kept secret. This internal transformation F is defined over an extension \mathbb{E} of degree n over some base field \mathbb{F}_q and is chosen to be \mathbb{F}_q -quadratic:

$$F : X \mapsto \sum_{\substack{0 \leq i < j < n \\ q^i + q^j \leq D}} \alpha_{i,j} X^{q^i + q^j} + \sum_{\substack{0 \leq k < n \\ q^k \leq D}} \beta_k X^{q^k} + \gamma, \tag{1}$$

where the coefficients $\alpha_{i,j}$, β_k , and γ lie in \mathbb{E} and D is an upper bound to the overall degree to make it practical to invert F through factorization. Since F is a \mathbb{F}_q -quadratic mapping, it can also be expressed over \mathbb{F}_q as an n -tuple (f_1, \dots, f_n) of quadratic polynomial mappings in n unknowns and so can the composition $T \circ F \circ S$ for any pair of one-to-one affine mappings $S : \mathbb{F}_q^n \rightarrow \mathbb{E}$ and $T : \mathbb{E} \rightarrow \mathbb{F}_q^n$. In the case of HFE, the mappings S and T are kept secret and together with F , constitute the secret key, whereas the public key is the mapping $G = T \circ F \circ S$. In order to decrypt, the legitimate user applies the inverse of T , finds roots of the univariate polynomials on the extension field \mathbb{E} and applies the inverse of S to each of these roots. The plaintext is one of the roots which can be singled out by using some redundancy. In this decryption process, the knowledge of the secrets S and T is crucial.

Additionally, Shamir’s proposal to remove some (say r) of the n polynomials that constitutes the public key can be applied in the case of a signature scheme: indeed, to sign a message (y_1, \dots, y_{n-r}) , the signer first completes the message with random values y_{n-r+1}, \dots, y_n and “decrypts” it normally. This operation is called the minus transformation and is used in the square-vinegar scheme.

With these notations, C^* is similar to HFE (with an unbounded total degree) where all coefficients of the internal transformation are set to zero but $\alpha_{0,\theta}$ for a well chosen θ . SFLASH in turn [11], is the original C^* scheme with the minus transformation applied.

2.2 The UOV Signature Scheme

Another ingredient in the design of the square-vinegar signature scheme is the use of additional unknowns meant to harden the analysis of the scheme by trying to break the structure used during the decryption process. Such an idea was first proposed in the oil and vinegar signature scheme. This scheme uses two sets of unknowns (x_1, \dots, x_n) and (z_1, \dots, z_v) respectively called the oil and the vinegar variables. The internal transformation then consists of an n -tuple of polynomials $F = (f_1, \dots, f_n)$ of the special form:

$$f_i(x, z) = \sum_{\substack{1 \leq i \leq n \\ 1 \leq j \leq v}} \alpha_{i,j} x_i z_j + \sum_{1 \leq i \leq n} \beta_i x_i + \sum_{1 \leq i \leq v} \gamma_i z_i + \sum_{1 \leq i \leq j \leq v} \delta_{i,j} z_i z_j + \epsilon, \tag{2}$$

where $\alpha_{i,j}$, β_i , γ_i , $\delta_{i,j}$, and ϵ are randomly chosen from the base field \mathbb{F}_q . The x_i are called oil variables because they do not mix, i.e. there is no cross-term $x_i x_j$.

Vinegar variables z_i in contrast, mix with other vinegar variables as well as with oil variables. The fact that the coefficients of the polynomials are chosen randomly is satisfactory since the resulting polynomials look closer to randomly chosen ones. However, the two types of variables makes it possible to create a signature scheme: in order to find some pre-image $y = (y_1, \dots, y_n)$ through F the signer first draws some random values for z_1, \dots, z_v and substitutes them in the description of F . The resulting set of polynomials becomes linear in the oil unknowns x_i and the associated $n \times n$ linear system (with y as right member) is easily solved: about $\frac{1}{e}$ of the time, the system has a solution (a_1, \dots, a_n) which makes (a, z) a pre-image of y through F and otherwise another choice for z is made until there is a solution. Obviously, this structure has to be hidden from the view of an attacker and the public key is the composition $G = F \circ S$ where $S : \mathbb{F}_q^{n+v} \rightarrow \mathbb{F}_q^{n+v}$ is a one-to-one affine application.

The message size over signature size for the UOV signature scheme is not optimal since the number of vinegar unknowns must be at least twice big as the number of oil unknowns for it to be secure [22][19][14].

2.3 The Square-Vinegar Signature Scheme

The square-vinegar signature scheme strives to provide an efficient alternative to UOV or HFE with the minus transformation applied. Let \mathbb{F}_q be a finite field and \mathbb{E} be an extension of degree n over \mathbb{F}_q . The internal transformation of the square-vinegar scheme is defined as:

$$F : \mathbb{E} \times \mathbb{F}_q^v \longrightarrow \mathbb{E} , \quad (X, X_v) \longmapsto \alpha X^2 + \beta(X_v)X + \gamma(X_v) , \quad (3)$$

where α is a constant randomly chosen from \mathbb{E} , $\beta : \mathbb{F}_q^v \rightarrow \mathbb{E}$ is a randomly chosen affine application, and $\gamma : \mathbb{F}_q^v \rightarrow \mathbb{E}$ is a randomly chosen \mathbb{F}_q -quadratic application. This internal transformation is hidden by two full rank affine applications $S : K^{n+v} \rightarrow \mathbb{E}$ and $T : \mathbb{E} \rightarrow \mathbb{F}_q^n$. Therefore S mixes the vinegar unknowns X_v with the “normal” unknowns X . In addition to T , a projection Π is applied where r of the n components have been removed as in SFLASH or HFE^{−−}. The affine transforms S and T together with the applications γ , β , and the constant α constitute the secret key. The public key P results from the composition of the three applications: $P = \Pi \circ T \circ F \circ S$.

The use of an odd characteristic base field is advertised by the authors as a means to thwart Gröbner bases attacks since introducing the corresponding field equations in the computation renders it unpractical. Mixing the vinegar unknowns with the normal ones breaks the algebraic relations between the input and the output that appeared in C^* (bilinear relations [17]) or HFE (algebraic relations of higher degree, as explained in [7][12]). Eventually, just as for HFE^{−−}, removing part of the output information further mitigates Gröbner bases attacks and prevents Kipnis and Shamir’s attack developed against UOV.

Signature. The signing process is highly efficient. It only requires the holder of the secret key to randomly pick r elements from \mathbb{F}_q to complete the message (m_1, \dots, m_{n-r}) to be signed into $\tilde{m} = (m_1, \dots, m_{n-r}, \tilde{m}_{n-r+1}, \dots, \tilde{m}_n)$

and to invert the public application in three steps: $S^{-1} \circ F^{-1} \circ T^{-1}(\tilde{m})$. Applying T^{-1} and S^{-1} is a matter of multiplying with precomputed matrices and inverting F requires to find the roots of a quadratic univariate polynomial over \mathbb{E} . In case there is no solution, the signer restarts the process by choosing another way of completing the message m into \tilde{m} .

2.4 The Square Encryption Scheme

A companion scheme to this square-veigar signature scheme has been proposed in [4]. The square encryption scheme strives to provide an efficient and secure alternative to HFE and, as the square-veigar scheme, has a square internal transformation: $F : \mathbb{E} \rightarrow \mathbb{E}, X \mapsto X^2$. The parameters are chosen so that the size of the base field verifies $q \equiv 3 \pmod 4$ and the degree n of \mathbb{E} over \mathbb{F}_q is odd. The transformation F is again hidden by two full rank affine mappings $S : \mathbb{F}_q^{n-r} \rightarrow \mathbb{E}$ and $T : \mathbb{E} \rightarrow \mathbb{F}_q^n$, which yields a public key $P = T \circ F \circ S$. (Following [5], the authors proposed to fix r of the input unknowns to a predefined value (say, zero) to prevent the attacker from controlling the differential of the public key as in Dubois, Fouque, Shamir, and Stern’s cryptanalysis [6].) This scheme is somewhat reminiscent of the C^* scheme, where $F(X) = X^{q^\theta+1}$ for a well chosen θ . But for the square encryption where $\theta = 0$, the bilinear relations $XY^{q^\theta} = X^{q^{2\theta}}Y$ between X and $Y = F(X)$ boils down to the tautology $XY = YX$. The embedding S aims to finish hiding the algebraic structure of the internal transformation.

Decryption. The secrets’ holder is able to decrypt very efficiently: in addition to finding pre-images through T and S which amounts to solve simple linear systems, the decryption process requires to compute a square root in the extension field \mathbb{E} . Computing the square root is done by the square and multiply algorithm $X = Y^{\frac{q^n+1}{4}}$ since $q^n \equiv 3 \pmod 4$. As there are two possible square roots, the right one is singled out as the one lying in the image of S .

3 Cryptanalysis of the Square-Veigar Signature Scheme

We now describe a generic and very efficient attack against the square-veigar signature scheme. Our attack proceeds in three steps: We first exhibit an invariant of the internal transformation and recover it through the analysis of the differential of the public key; Then, we use this information to recover an equivalent representation of the veigar space; In a third step, we transform the public key into a special shape that allows us to invert it efficiently. Put together, these three steps allow us to forge a signature for any given message.

3.1 Alternative Decompositions

Recall that the internal transformation of the square-veigar signature scheme has the following structure:

$$F : \mathbb{E} \times \mathbb{F}_q^v \longrightarrow \mathbb{E} , \quad (X, X_v) \longmapsto \alpha X^2 + \beta(X_v)X + \gamma(X_v) ,$$

where α is a constant, $\beta : \mathbb{F}_q^v \rightarrow \mathbb{E}$ is an affine \mathbb{F}_q -linear mapping, and $\gamma : \mathbb{F}_q^v \rightarrow \mathbb{E}$ is a \mathbb{F}_q -quadratic mapping, where \mathbb{E} is an extension of degree n over \mathbb{F}_q . The public key is the mapping $P = \Pi \circ T \circ F \circ S$, where Π is a projection that removes r polynomials, $S : \mathbb{F}_q^{n+v} \rightarrow \mathbb{E} \times \mathbb{F}_q^v$ and $T : \mathbb{E} \rightarrow \mathbb{F}_q^{n-r}$ are two affine linear mappings of full rank. The decomposition (T, F, S) of the public key is kept secret.

A major component of the internal transformation F is the mixing of vinegar unknowns with X . It makes it harder for an attacker to use the specific structure of a univariate quadratic polynomial of F viewed as a function of X . A crucial remark is that there exist linear mappings that, when composed with the internal transformation, not only conserve its special form, but also discard the part of F mixing the vinegar X_v with X . Indeed, consider the mappings $\sigma : (X, X_v) \mapsto (X - \frac{\alpha}{2}\beta(X_v), X_v)$ and $\tau : Y \mapsto \frac{1}{\alpha}Y$. (Remember that the scheme is defined over a field \mathbb{F}_q of odd characteristic.) It can be checked that these mappings provide an alternative decomposition $(T \circ \tau, \tilde{F}, \sigma \circ S)$ of the public key such that

$$\tilde{F} : (X, X_v) \mapsto X^2 + \tilde{\gamma}(X_v) , \tag{4}$$

where $\tilde{\gamma}$ is a \mathbb{F}_q -quadratic mapping. We stress here that an attacker does not need to know the mappings σ and τ but rather assumes without loss of generality that the public key follows the specific decomposition (4). (Also note that in a similar fashion, keeping secret the defining polynomial of the extension has no effect: as two fields of the same size are isomorphic and the isomorphism is a linear bijective application, any arbitrary choice made by the attacker is “absorbed” in S and T .) This last decomposition can be further tweaked as in [11] to remove the affine parts of the mappings S and T but at the expense of reintroducing a linear term in X , leading to an internal transformation of the following shape:

$$F' : (X, X_v) \mapsto X^2 + \beta'X + \gamma'(X_v) , \tag{5}$$

where β' is a *constant* from \mathbb{E} and γ' is some \mathbb{F}_q -quadratic mapping. In the following sections, the attacker can therefore just assume wlog that the public key is decomposed as (T', F', S') where S' and T' are linear mappings, and F' is as given in (5): then (T', F', S') contains enough information to forge valid signatures and thus constitutes an equivalent secret key. We call such a decomposition a “split decomposition” (the unknowns X and X_v are now separated in the internal transformation). A split decomposition is not unique: iterates of the Frobenius mapping $\varphi : z \mapsto z^q$ and multiplications $\Lambda_u : z \mapsto uz, u \in \mathbb{E}$, do not alter the prescribed shape of the internal transformation (though coefficients might change); In particular, if (T_0, F_0, S_0) is a split decomposition, so are $(T_0 \circ \Lambda_{u^{-2}}, \Lambda_{u^2} \circ F_0 \circ \Lambda_{u^{-1}}, \Lambda_u \circ S_0)$ and $(T_0 \circ \varphi^{-i}, \varphi^i \circ F_0 \circ \varphi^{-i}, \varphi^i \circ S_0)$.

3.2 Using the Multiplicative Property of the Differential

In the previous section we showed how to discard the cross-contribution of X_v and X . However, the contribution $\gamma(X_v)$ still disturbs the algebraic properties of the univariate quadratic in X . In order to circumvent this difficulty, we make

use of a tool first introduced by Fouque *et al.* in [9] that proved very useful in attacking multivariate cryptosystems: the differential of the public mapping. The differential of P in a is defined as: $DP_a(x) = P(x+a) - P(x) - P(a) + P(0)$.

In the case of an \mathbb{F}_q -quadratic mapping, DP_a is such that $(x, a) \mapsto DP_a(x)$ is a symmetric bilinear mapping. From now on, we denote by DP this bilinear mapping and call it differential of P . The differential map corresponding to the internal transformation $X \mapsto X^2 + \beta X + \gamma(X_v)$ of a square-vinegar instance is:

$$DF((X, X_v), (Y, Y_v)) = 2XY + D\gamma(X_v, Y_v) . \tag{6}$$

The success of the attack lies in the fact that normal (X) and vinegar (X_v) unknowns are separated in the expression of the differential DF . More precisely, the only linear mappings L such that for all (X, X_v) and all (Y, Y_v) :

$$DF((L(X), X_v), (Y, Y_v)) - DF((X, X_v), (L(Y), Y_v)) = 0 \Leftrightarrow L(X)Y = YL(X)$$

are $Z \mapsto \lambda Z$ for $\lambda \in \mathbb{E}$. Indeed, any solution $L : Z \mapsto \sum_{1 \leq i < n} l_i Z^{q^i}$ verifies $\sum_{1 \leq i < n} l_i XY^{q^i} = \sum_{1 \leq i < n} l_i X^{q^i} Y$ for all X and Y , and since $(X, Y) \mapsto X^{q^i} Y^{q^j}$ forms a basis of the space of bilinear forms we must have $l_i = 0$ for all $i > 0$.

In addition, we conjecture that with very high probability (with respect to the uniform choice of the coefficients of γ) the only linear mappings L verifying

$$\forall X_v \forall Y_v \quad D\gamma(L(X_v), Y_v) - D\gamma(X_v, L(Y_v)) = 0$$

are $Z_v \mapsto cZ_v$ for some $c \in \mathbb{F}_q$. This might be heuristically justified by the fact that the random choice of γ does not allow such an algebraic property to appear, and is verified experimentally. Assuming this conjecture is true, we have:

Proposition 1. *For a random instance of the square-vinegar scheme, it happens with very high probability that the only linear mappings L verifying:*

$$\forall (X, X_v) \forall (Y, Y_v) \quad DF(L(X, X_v), (Y, Y_v)) - DF((X, X_v), L(Y, Y_v)) = 0 \tag{7}$$

are $(Z, Z_v) \mapsto (\lambda Z, cZ_v)$, where $\lambda \in \mathbb{E}$ and $c \in \mathbb{F}_q$.

Proof. Write $L : (Z, Z_v) \mapsto (AZ + CZ_v, \tilde{C}Z + BZ_v)$ for some solution of (7). Since the equation holds for all inputs of DF , consider it specialised at $X_v = 0$ and $Y_v = 0$, with DF replaced by its expression (6):

$$\forall X \forall Y \quad [2A(X)Y + D\gamma(\tilde{C}(X), 0)] - [2XA(Y) - D\gamma(0, \tilde{C}(Y))] = 0 .$$

As $D\gamma(*, 0) = 0$ and $D\gamma(0, *) = 0$ for any $*$, this gives $A(X)Y = XA(Y)$ which, as we saw above, implies $A : Z \mapsto \lambda Z$ for $\lambda \in \mathbb{E}$. Similarly, at $X = 0$ and $Y = 0$, (7) becomes: $\forall X_v \forall Y_v \quad D\gamma(B(X_v), Y_v) - D\gamma(X_v, B(Y_v)) = 0$, implying $B : Z \mapsto cZ$ for $c \in \mathbb{F}_q$ by conjecture. Finally, at $X = 0$ and $Y_v = 0$, (7) becomes: $\forall X_v \forall Y \quad D\gamma(X_v, \tilde{C}(Y)) = 2C(X_v)Y$. Assume for a contradiction that C is not identically null. Then setting $X_v = x_1$ such that $C(x_1) \neq 0$, the right hand side spans a vector space of dimension n while the left hand side spans a vector space

of dimension at most v . Hence, when $v < n$ as in a square-vinegar instance, C must be identically null. Then, for all (X_v, Y) , we have $D\gamma(X_v, \tilde{C}(Y)) = 0$ or equivalently $\gamma(X_v + \tilde{C}(Y)) = \gamma(X_v) + \gamma(\tilde{C}(Y))$. In particular, this holds for $X_v = \tilde{C}(X)$ for any X and any Y so that $Z \mapsto \gamma(\tilde{C}(Z))$ is affine, that is, γ is affine over $\text{Im}(\tilde{C})$. For a random γ it is improbable that γ is affine over some (non-zero) sub-space. Hence, with high probability, \tilde{C} is identically null. \square

This property of F naturally transports to the public key, provided the removal of polynomials do not completely destroy its algebraic structure:

Claim 1. *If the number of coordinates removed by the projection Π is less than half and the coefficients of γ are randomly chosen, the set of linear mappings L satisfying*

$$\forall X \forall Y \quad DP(L(X), Y) - DP(X, L(Y)) = 0$$

is $\{S^{-1} \circ A_{u,c} \circ S\}_{u \in \mathbb{E}, c \in \mathbb{F}_q}$, i.e. the conjugates by the secret mapping S of all the multiplications $A_{u,c} : (X, X_v) \mapsto (uX, cX_v)$, where $u \in \mathbb{E}$ and $c \in \mathbb{F}_q$.

3.3 Extracting the Vinegar Vector Space

The solution set Σ of Claim \square can be easily determined as it amounts to solve a linear system of $(n - r)(n + v)^2$ equations in the $(n + v)^2$ unknowns of L over a finite field of size q . Let us call ‘‘vinegar vector space’’ the image through S of all the values v such that the n first coordinates of $S(v)$ are zero. Similarly, let us call ‘‘normal vector space’’ the image through S of all the values v such that the v last coordinates equal zero. Before explaining how to use the knowledge of Σ to recover these two vector spaces, let us state three useful lemmas.

Lemma 1. *Let u be in \mathbb{E} , π_u be the minimal polynomial of u over \mathbb{F}_q , and $\chi_{A_{u,c}}$ be the characteristic polynomial of $A_{u,c} : (X, X_v) \mapsto (uX, cX_v)$. Then:*

$$\chi_{A_{u,c}}(x) = (x - c)^v \cdot \pi_u(x)^{\frac{n}{\deg \pi_u}} .$$

Lemma 2. *Let u be in \mathbb{E} and π_u the minimal polynomial of u over \mathbb{F}_q . Then:*

$$\pi_u(x) = (x - u)(x - u^q) \cdots (x - u^{q^{\deg(\pi_u) - 1}}) .$$

Lemma 3 (Thm. 3.25 [15]). *The number of irreducible monic polynomials of degree n in $\mathbb{F}_q[X]$ is $\frac{1}{n} \sum_{d|n} \mu(d)q^{\frac{n}{d}}$ where μ is the Möbius function. \square It follows that the number of elements in \mathbb{E} with a minimal polynomial of degree n is at least $q^n - q^{\frac{n}{2}} - q^{\frac{n}{2} - 1} - \dots - q^2 - q$.*

Let M be any element picked at random from the solution set Σ of Claim \square . Since $M = S^{-1} \circ A_{u,c} \circ S$ for some $(u, c) \in \mathbb{E} \times \mathbb{F}_q$, M and $A_{u,c}$ are conjugate and thus have the same characteristic polynomial $\chi_M(x) = (x - c)^v \cdot \pi_u(x)^{\frac{n}{\deg \pi_u}}$ according to Lemma \square . In addition, Lemma \square shows that for u chosen uniformly at random

¹ $\mu(1) = 1$, $\mu(x) = (-1)^k$ for x a product of k distinct primes, and $\mu(x) = 0$ otherwise.

in \mathbb{E} , $\deg(\pi_u)$ has more than $1 - q/q^{\frac{n}{2}}$ chances to be n . We can therefore assume in the following that c and π_u are known from the factorization of χ .

The factorization of π_u over \mathbb{E} in turn discloses u^{q^i} for some unknown i . However, as stated at the end of Section 3.1, the split decomposition is not affected by iterates of the Frobenius mapping and thus it is enough to solve for \tilde{S} in the following linear system:

$$\tilde{S} \circ M = \Lambda_{u^{q^i}, c} \circ \tilde{S} .$$

Any particular solution S_0 of this system is sufficient, since the whole space of solutions is a coset of the commutant of $\Lambda_{u^{q^i}, c}$. The commutant of $X \mapsto u^{q^i} X$ is the space of multiplications, since u does not belong to any subfield of \mathbb{E} . On the contrary, the commutant of $X_v \mapsto cX_v$ is the whole space of \mathbb{F}_q -linear mappings, since precisely c lies in \mathbb{F}_q . At this point, the attacker is almost in the same position as the legitimate signer to produce a signature since he has access to the vinegar space through S_0 and can now work on

$$P \circ S_0^{-1}(X, X_v) = \Pi \circ T \circ (X^2 + \beta X + \gamma(X_v))$$

instead of the original public key P . Let us define $\tilde{P} = P \circ S_0^{-1}$.

The next step of the attack is to recover a mapping equivalent to T . To this end, we seek to cancel the part of \tilde{P} that is linear in X which can be achieved by using an adequate change of variables $X \mapsto (X - b)$, where b is to be determined. The expression of $\tilde{P}(X - b)$ with respect to X in turn contains a quadratic part, a linear part, and a constant part. Looking at the linear part alone, the attacker writes down that the set a coefficients of X are equal to zero; these coefficients are a set of $(n - r)$ affine functions with respect to b and solving for b allows the attacker to recover β . The final step is to recover an equivalent version of T . This is done by considering the part of \tilde{P} that is quadratic with respect to X : $Q(X) = \Pi \circ T(X^2)$. By composing with multiplications over \mathbb{E} , it is possible to complete the $(n - r)$ coordinates of Q into a full set $\tilde{Q}(X)$ of n coordinates by taking a basis of $\{Q(\lambda X)\}_{\lambda \in \mathbb{E}}$. Then, solving for \tilde{T} in $\tilde{Q}(X) = \tilde{T}(X^2)$ gives an equivalent representation T_0 of T .

At this point, the attacker gained the knowledge of S_0 , T_0 , and β_0 such that:

$$P \circ S_0^{-1}(X, X_v) = \Pi \circ T_0 \circ (X^2 + \beta_0 X) + P \circ S_0^{-1}(0, X_v) .$$

We claim that this is equivalent to the knowledge of the secret key since the attacker is then able to sign any message m as efficiently as the legitimate signer as follows. Draw some random value X_v from the vinegar space and randomly complete the $(n - r)$ coordinates of $m - P \circ S_0^{-1}(0, X_v)$ into an n coordinates value \tilde{m} . Compute $Y = T_0^{-1}(\tilde{m})$ and solve for X_0 in $(X + \frac{1}{2}\beta_0)^2 = Y + \frac{1}{4}\beta_0^2$. A signature of m is then given by $S_0^{-1}(X_0, X_v)$.

3.4 Complexity Analysis and Practical Parameters

Our attack requires $O(\log^2(q)(n + v)^6)$ operations to find the solution set Σ of Claim 1 and $O(\log^2(q)(n + v)^3)$ operations to factor the characteristic polynomial χ . The particular solution S_0 is found with $O(\log^2(q)(n + v)^6)$ operations.

The complexity of the other steps can be neglected and thus the attack has an overall complexity of $O(\log^2(q)(n + v)^6)$.

The authors of the square-vinegar signature scheme claimed a 80-bits security for the following parameter sets:

	parameter set 1	parameter set 2
field size q	31	13
normal unknowns n	31	36
vinegar unknowns v	4	4
removed polynomials r	3	3

The complexity of our attack is about 2^{35} and our Magma program in appendix completes within minutes for both parameter sets on a common desktop PC.

4 Cryptanalysis of the Square Encryption Scheme

The square encryption scheme poses new challenges to the attacker. Its design strategy of embedding the plaintext into a bigger space before applying the internal transformation makes it impossible to use the differential mapping as was done previously. This is due to the restricted view the attacker has on the input space which does not allow to manipulate the inner of the differential easily. In our attack against the square encryption scheme, we therefore use a different technique. Instead of peeling off the cryptosystem from the input, we peel it off from the output.

4.1 Equivalent Representation of the Secret Key

Due to the specific form of the internal transformation and without loss of generality, we may give the following alternative decomposition of the public key:

$$P(X) = T(S(X)^2) + T(s \cdot S(X)) + t \quad , \quad (8)$$

where S and T are the linear part of the original secret linear mappings and $s = \frac{1}{2}\sigma$ and $t = \tau + T(\sigma^2)$ with σ and τ the original secret constants from \mathbb{E} . Since the mappings S and T are linear, it can be easily seen that with respect to the input X , the first term of (8) is \mathbb{F}_q -quadratic, the second term is linear, and the third term is constant. Furthermore, these three homogeneous terms can be read directly on the public key itself, so that the attacker knows the following:

$$P_2(X) = T(S(X)^2) \quad , \quad P_1(X) = T(s \cdot S(X)) \quad , \quad P_0(X) = t \quad .$$

4.2 Looking for Invariant Subspaces

As with the signature scheme, the differential of the public key provides useful information to the attacker. In the case of the square encryption scheme, it can be expressed as:

$$DP(X, Y) = T(2 \cdot S(X) \cdot S(Y)) \quad .$$

Let consider the partial mappings $DP_y : X \mapsto DP(X, y)$. Since $S : \mathbb{F}_q^{n-r} \rightarrow \mathbb{E}$ has full rank, its image is of dimension $(n - r)$. Hence, choosing any linearly independent vectors y_1, \dots, y_{n-r} makes $DP_{y_1}, \dots, DP_{y_{n-r}}$ span the whole vector space of mappings $\{DP_z\}_{z \in \mathbb{E}}$. This shows that the attacker is able to derive a set of mappings $\Delta = \{P_1\} \cup \{DP_{y_i}\}_{i=1, \dots, n-r}$ each of which has the special form $T \circ A_\alpha \circ S$, where A_α stands for the multiplication by α in \mathbb{E} . This set of mappings can then be rewritten as $\Delta = \{T \circ A_{\lambda_i} \circ S\}_{i=1, \dots, n-r+1}$ where the $n - r + 1$ values $\lambda_1, \dots, \lambda_{n-r+1}$ are unknown, but linearly independent.

The attacker does not need to know the actual value of the λ_i since he can exploit this set of mappings in as follows. The general idea is to look for linear mappings L that can link the public equations, say two elements $D_1 = T \circ A_{\lambda_1} \circ S$ and $D_2 = T \circ A_{\lambda_2} \circ S$ from Δ . One natural idea is then to look for L such that:

$$L \circ D_1 = D_2 \quad , \tag{9}$$

since it can be easily checked that $L_0 = T \circ A_{\lambda_2 \lambda_1^{-1}} \circ T^{-1}$ is a particular solution of (9). However, the solution space of (9) is not restricted to multiplications. This is due to the ‘embedding’ mechanism, i.e. the fact that the mapping S is not a one-to-one mapping, which release some of the constraints and allows less structured linear mapping to be solutions.

A possible direction to solve this issue is to put more constraints on the mapping L while being careful to keep mappings of the form $T \circ A_* \circ T^{-1}$ in the solution space. This is why we not only look for a linear mapping that solves (9), but *several* equations similar to (9) simultaneously. This can be reformulated in terms of Δ as follows. We look for linear mappings L such that:

$$\forall i \in \{1, \dots, m\}, \quad L \circ (T \circ A_{\lambda_i} \circ S) \in \langle T \circ A_{\lambda_{m+1}} \circ S, \dots, T \circ A_{\lambda_{n-r+1}} \circ S \rangle \quad , \tag{10}$$

that is, the image through L of m elements of Δ must lie in the vector space spanned by the remaining elements of Δ . It is easy to see that if λ is such that:

$$\forall i \in \{1, \dots, m\}, \quad \lambda \cdot \lambda_i \in \langle \lambda_{m+1}, \dots, \lambda_{n-r+1} \rangle \quad , \tag{11}$$

then $T \circ A_\lambda \circ T^{-1}$ must be solution of (10).

The parameter m controls the number of solutions of (10) and (11). It can be used to simultaneously render system (11) under-determined and system (10) over-determined. This ensures that no other solutions except than the conjugates of multiplications. We can determine suitable values of m as follows. For $i \leq m$, the fact that $\lambda \cdot \lambda_i$ lies in $\langle \lambda_{m+1}, \dots, \lambda_{n-r+1} \rangle$ puts $n - ((n - r + 1) - m)$ constraints on the n coordinates of λ in \mathbb{F}_q . As $\lambda_1, \dots, \lambda_{n-r+1}$ are linearly independent, the above constraints are independent. Hence (11) admits solutions as soon as $n > m(n - (n - r + 1 - m))$. Similarly, the whole space of linear mappings L has dimension n^2 and each equation of (10) puts $n(n - r) - (n - r + 1 - m)$ constraints as mappings from Δ map \mathbb{F}_q^{n-r} to \mathbb{F}_q^n . Therefore, system (10) is over-determined as soon as $n^2 \leq m(n(n - r) - (n - r + 1 - m))$. These two conditions define a range of values of m such that the solution space of (10) becomes isomorphic to the solution space of (11). This behavior is entirely confirmed by our Magma implementation of the attack.

4.3 Recovery of the Secret Elements

Once a linear mapping $L = T \circ \Lambda_\lambda \circ T^{-1}$ has been recovered, every element of the secret key can be computed. By proceeding just as for the signature scheme, the underlying multiplication λ is revealed from the characteristic polynomial of L . An equivalent representation T_0 of T is then recovered by solving for \tilde{T} in $\tilde{T} \circ L = \Lambda_\lambda \circ \tilde{T}$. Let a be a randomly chosen element. The other component of the secret key can then be found via:

$$S(a) = \sqrt{T_0^{-1}(P_2(a))} \ , \quad s_0 = \frac{1}{S(a)} \cdot T_0^{-1}(P_1(a)) \ , \quad S_0 = \frac{1}{s_0} \cdot T_0^{-1} \circ P_1 \ .$$

(In the case where $T_0^{-1}(P_2(a))$ is not a square in \mathbb{E} , just replace T_0 by $-T_0$.)

4.4 Practical Parameters

The most time consuming step of our attack is to compute the solution space of (10) which requires $O(\log^2(q)n^6)$ operations. The authors of the square encryption scheme claimed a 80-bit security for the following parameter sets:

	parameter set 1	parameter set 2
field size q	31	31
unknowns $n - r$	34	51
polynomials n	37	54

but the complexity of our attack actually is about 2^{36} operations for the first parameter set and about 2^{39} for the second. Again, the key recovery written in Magma only requires a couple of seconds to complete on a standard workstation. During the attack, $m = 2$ was enough in practice to ensure that only conjugates of multiplications were solutions.

References

1. Akkar, M.-L., Courtois, N., Goubin, L., Duteuil, R.: A Fast and Secure Implementation of Sflash. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 267–278. Springer, Heidelberg (2002)
2. Baena, J., Clough, C., Ding, J.: Square-vinegar signature scheme. In: Buchmann, J., Ding, J. (eds.) PQCrypto 2008. LNCS, vol. 5299, pp. 17–30. Springer, Heidelberg (2008)
3. Bosma, W., Cannon, J., Playoust, C.: The Magma algebra system. I. The user language. J. Symbolic Comput. 24(3-4), 235–265 (1997)
4. Clough, C., Baena, J., Ding, J., Yang, B.-Y., Chen, M.-S.: Square, a New Multivariate Encryption Scheme. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 252–264. Springer, Heidelberg (2009)
5. Ding, J., Wolf, C., Yang, B.-Y.: ℓ -Invertible Cycles for Multivariate Quadratic Public Key Cryptography. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 266–281. Springer, Heidelberg (2007)

6. Dubois, V., Fouque, P.-A., Shamir, A., Stern, J.: Practical Cryptanalysis of SFLASH. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 1–12. Springer, Heidelberg (2007)
7. 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)
8. Faugère, J.-C., Perret, L.: Polynomial Equivalence Problems: Algorithmic and Theoretical Aspects. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 30–47. Springer, Heidelberg (2006)
9. Fouque, P.-A., Granboulan, L., Stern, J.: Differential cryptanalysis for multivariate schemes. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 341–353. Springer, Heidelberg (2005)
10. Fouque, P.-A., Macario-Rat, G., Stern, J.: Key Recovery on Hidden Monomial Multivariate Schemes. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 19–30. Springer, Heidelberg (2008)
11. Geiselmann, W., Steinwandt, R., Beth, T.: Attacking the Affine Parts of SFLASH. In: Honary, B. (ed.) Cryptography and Coding 2001. LNCS, vol. 2260, pp. 355–359. Springer, Heidelberg (2001)
12. 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)
13. Imai, H., Matsumoto, T.: Algebraic Methods for Constructing Asymmetric Cryptosystems. In: Calmet, J. (ed.) AAEC 1985. LNCS, vol. 229, pp. 108–119. Springer, Heidelberg (1986)
14. Kipnis, A., Patarin, J., Goubin, L.: Unbalanced Oil and Vinegar Signature Schemes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 206–222. Springer, Heidelberg (1999)
15. Lidl, R., Niederreiter, H.: Finite fields. Encyclopedia of mathematics and its applications, vol. 20. Cambridge university press, Cambridge (2003)
16. Matsumoto, T., Imai, H.: Public Quadratic Polynomial Tuples for Efficient Signature Verification and Message Encryption. In: Günther, C.G. (ed.) EUROCRYPT 1988. LNCS, vol. 330, pp. 419–453. Springer, Heidelberg (1988)
17. Patarin, J.: Cryptoanalysis of the Matsumoto and Imai Public Key Scheme of Eurocrypt 1988. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 248–261. Springer, Heidelberg (1995)
18. 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)
19. Patarin, J.: The Oil and Vinegar Algorithm for Signatures. Presented at the Dagstuhl Workshop on Cryptography (September 1997)
20. Patarin, J., Goubin, L., Courtois, N.T.: Improved Algorithms for Isomorphisms of Polynomials. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 184–200. Springer, Heidelberg (1998)
21. Shamir, A.: Efficient Signature Schemes Based on Birational Permutations. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 1–12. Springer, Heidelberg (1994)
22. Shamir, A., Kipnis, A.: Cryptanalysis of the Oil & Vinegar Signature Scheme. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 257–266. Springer, Heidelberg (1998)
23. Shor, P.W.: Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. SIAM J. Sci. Stat. Comp. 26, 1484 (1997)

A Simple Auxiliary Functions for Our Magma Scripts

Simple functions. The following function returns a root of $ax^2 + bx + c$.

```

1 SOLVE_2ND_DEGREE:=function(a,b,c)
2   is_,sqrt_delta:=ISSQUARE(b2-4*a*c);
3   return is_,(is_ select (-b+sqrt_delta)/(2*a) else 0);
4 end function;
```

Juggling between matrices and vectors:

```

5 MAT2VEC:=func< MAT | VECTOR(ELTSEQ(MAT)) >;
6 VEC2MAT:=func< vect, ncol | MATRIX(ncol, ELTSEQ(vect)) >;
```

SPACE returns the vector space spanned by a set of matrices MS viewed as vectors:

```

7 SPACE:=func< MS, KK, dim |
8   sub<VECTORSPACE(KK, dim)|[MAT2VEC(MS[i]) : i in [1..#MS]]> >;
```

The following returns the matrix of $x \mapsto \lambda x$:

```

9 MULBY:=func< λ, ETOV, VTOE, B |
10   MATRIX([ETOV(VTOE(B[i])*λ) : i in [1..#B]]) >;
```

Sequences of coefficients. It can be convenient to represent a quadratic polynomial as sequences of coefficients of its homogeneous degree 0, 1, and 2 components. C_{012} takes a function P viewed as a sequence of n_pol polynomials on n_var variables and outputs the corresponding sequences CS0, CS1, and CS2:

```

11 C012:=function(KK, V_INPUT, P, n_pol, n_var)
12   CS0:=[KK ! 0:ii in [1..n_pol]];
13   CS1:=[[KK ! 0:i in [1..n_var]:ii in [1..n_pol]];
14   CS2:=[[[KK ! 0:j in [1..i]:i in [1..n_var]:ii in [1..n_pol]];
15   x:=V_INPUT ! 0; y:=P(x);
16   for ii:=1 to n_pol do CS0[ii]:=y[ii]; end for; // constant
17   for i:=1 to n_var do
18     x:=V_INPUT ! 0; x[i]:=KK ! 1; y1:=P(x); x[i]:=KK ! -1; y2:=P(x);
19     for ii:=1 to n_pol do
20       CS1[ii][i]:=(y1[ii]-y2[ii])*(KK ! 2)-1; // coefficient of xi,
21       CS2[ii][i][i]:=(y1[ii]+y2[ii])*(KK ! 2)-1-CS0[ii]; // and xi2,
22     end for;
23   end for;
24   for i:=2 to n_var do for j:=1 to i-1 do
25     x:=V_INPUT ! 0; x[i]:=KK ! 1; x[j]:=KK ! 1; y:=P(x);
26     for ii:=1 to n_pol do
27       CS2[ii][i][j] := y[ii]-CS2[ii][i][i]-CS2[ii][j][j]-
28       CS1[ii][i]-CS1[ii][j]-CS0[ii]; // and xixj, i ≠ j
29     end for;
30   end for; end for;
31   return CS0, CS1, CS2;
32 end function;
```

Given three sequences of coefficients C_0 , C_1 , and C_2 defined with respect to a quadratic polynomial P as above, compute the value taken by P on input x :

```

33  EVAL:=func< C2, C1, C0, x, n_var |
34      &+[ &+[C2[i][j]*x[i]*x[j] : j in [1..i]] : i in [1..n_var]]
35      + &+[C1[i]*x[i]:i in [1..n_var]] + C0 >;

```

The next function computes the coefficients of the differential associated to the homogeneous form of degree 2 specified by the sequence of its coefficients:

```

36  DIFF:=function(CS2, KK, n_pol, n_var)
37    DP:=[ZEROMATRIX(KK, n_var, n_var): ii in [1..n_pol]];
38    for ii:=1 to n_pol do
39      for i:=1 to n_var do
40        DP[ii][i,i]:=2*CS2[ii][i][i];
41        for j:=1 to i-1 do
42          DP[ii][i,j]:=CS2[ii][i][j]; DP[ii][j,i]:=CS2[ii][i][j];
43        end for; end for; end for;
44    return DP; end function;

```

B Magma Script to Attack the Signature Scheme

An extension E of degree n over the base field K , also viewed as vector space V :

```

45  q:=31; n:=31; v:=4; r:=3; K:=GF(q); E:=ext<K|n>;
46  V, E2V:=VECTORSPACE(E, K); V2E:=E2V-1;
47  V_INPUT:=VECTORSPACE(K, n+v); V_VINEGAR:=VECTORSPACE(K, v);
48  V_MESSAGE:=VECTORSPACE(K, n-r); V_RANDOM:=VECTORSPACE(K, r);

```

We then randomly draw a secret key: the coefficient α , the linear mapping β , and the quadratic mapping γ to form the internal transformation

$$F : (X, X_v) \mapsto \alpha X^2 + \beta(X_v)X + \gamma(X_v) ,$$

```

49  alpha:=V2E(A[1]) where A is RANDOM(GL(n, K)); // ensures alpha != 0
50  beta_0:=RANDOM(E); beta_1:=[RANDOM(E):i in [1..v]];
51  beta:=func< Xv | &+[beta_1[i]*Xv[i]:i in [1..v]] + beta_0 >;
52  gamma_0:=RANDOM(E); gamma_1:=[RANDOM(E):i in [1..v]];
53  gamma_2:=[[RANDOM(E):j in [1..i]]:i in [1..v]];
54  gamma:=func< Xv |
55      &+[ &+[gamma_2[i][j]*Xv[i]*Xv[j] : j in [1..i]] : i in [1..v]] +
56      &+[gamma_1[i]*Xv[i]:i in [1..v]] + gamma_0 >;
57  F:=func< X, Xv | alpha*X2+beta(Xv)*X+gamma(Xv) >;

```

and randomly draw input and output linear layers S and T :

```

58  S1:=RANDOM(GL(n+v, K)); S0:=RANDOM(V_INPUT);
59  T1:=RANDOM(GL(n, K)); T0:=RANDOM(V);

```

The corresponding public key is obtained via $P = T \circ F \circ S$:

```

60 P:=function(input)
61   XX:=input*S1+S0;
62   X:=V2E(VECTOR([XX[i]:i in [1..n]])); // normal variables
63   XV:=VECTOR([XX[i]:i in [n+1..n+v]]); // vinegar variables
64   return E2V( F(X,Xv) )*T1+T0;
65 end function;
```

The coefficients of homogeneous parts for the set of forms corresponding to P is obtained via:

```

66 PUBC0, PUBC1, PUBC2:=C012(K, V_INPUT, P, n-r, n+v);
```

We are now able to verify if a signature is valid:

```

67 VERIFY:=function(msg, sig)
68   m:=[ EVAL(PUBC2[i], PUBC1[i], PUBC0[i], sig, n+v) : i in [1..n-r]];
69   return &and[ m[i] eq msg[i] : i in [1..n-r]];
70 end function;
```

We now compute an equivalent secret key. First, we look for the linear mappings M_X verifying: $M_X \times DP - DP \times M_X = 0$.

```

71 B:=BASIS(V); PR:=POLYNOMIALRING(K, (n+v)2);
72 MX:=MATRIX(n+v, [PR.i:i in [1..(n+v)2]]);
73 DP:=DIFF(PUBC2, K, n-r, n+v);
74 EQS:=[ELTSEQ(MX*DP[ii]-DP[ii]*TRANPOSE(MX)):ii in [1..n-r]];
75 GB:=[];
76 for ii:=1 to n-r do
77   GB:=GROEBNERBASIS(GB cat EQS[ii]);
78   if #GB + n + 1 eq (n+v)2 then break; end if;
79 end for;
```

We choose a particular solution M_- by removing the $n + 1$ degrees of freedom by fixing the remaining unknowns to random values, and extract the two roots $c \in K$ and $a \in E$ of the characteristic polynomial of M_- .

```

80 repeat W:=GROEBNERBASIS([PR.((n+v)2-i) + RANDOM(K):i in [0..n]]
```

at GB);

```

81 until not(W eq [PR ! 1]); // complete consistently
82 M_-:=MATRIX(n+v, [K ! EVALUATE(W[i], PR.i, 0):i in [1..(n+v)2]]);
83 CPOL:=FACTOREDCHARACTERISTICPOLYNOMIAL(M_-);
84 if not(#CPOL eq 2) then "Bad Char. Pol."; exit; end if;
85 c:=ROOTS(CPOL[1][1])[1][1]; // factor of degree 1
86 a:=ROOTS(POLYNOMIALRING(E) ! CPOL[2][1])[1][1]; // of degree n
```

M_- must be similar to the matrix of $(X, X_v) \mapsto (aX, cX_v)$, which will disclose a particular solution S_- as useful to sign as S :

```

87 A:=MULBY(a, E2V, V2E, B);
88 is_similar, S_-:=ISSIMILAR(M_-, DIAGONALJOIN(A, SCALARMATRIX(v, c) ));
89 if not(is_similar) then "Recovering S_- failed."; exit; end if;
```

Applying the change of base S_- , we get $(Z, Z_v) \mapsto T(Z^2 + \tilde{\beta} \cdot Z + \tilde{\gamma}(Z_v))$:

```

90 Z:=VECTOR([PR.i:i in [1..n+v]])*MATRIXALGEBRA(PR,n+v)!(S_-);
91 PUBZ:=[EVAL(PUBC2[i], PUBC1[i], PUBC0[i], Z, n+v):i in [1..n-r]];

```

To get rid of the term $\tilde{\beta} \cdot Z$, we look for Y such that the coefficient of Z in $T((X+Y)^2 + \tilde{\beta} \cdot (Z+Y) + \tilde{\gamma}(Z_v))$ becomes zero:

```

92 V0:=RANDOM(V_VINEGAR);
93 ZPY:=[PR!0:i in [1..(n+v)^2]]; // Z+Y
94 for i:=1 to n do ZPY[i]:=PR.i+PR.(i+n+v); end for;
95 for i:=1 to v do ZPY[i+n]:=V0[i]; end for;
96 PUBZV:=[EVALUATE(PUBZ[i],ZPY):i in [1..n-r]];
97 OY:=[PR!0:i in [1..(n+v)^2]]; // (Z,Y)=(0,Y)
98 for i:=1 to n do OY[i+n+v]:=PR.(i+n+v); end for;
99 EQLIN:=&cat[[EVALUATE(COEFFICIENT(PUBZV[i],PR,j,1),OY)
100           :j in [1..n]]:i in [1..n-r]]; // equations 2Y = \tilde{\beta}
101 Y0:=GROEBNERBASIS(EQLIN);
102 beta_:=VECTOR([K!EVALUATE(Y0[i],PR.(i+n+v),0):i in [1..n]]);

```

We are now able to get the polynomials corresponding to $T(Z^2 + \tilde{\gamma}(Z_v))$:

```

103 for i:=1 to n do ZPY[i]:=PR.i-beta_[i]; end for;
104 PUBZ0:=[EVALUATE(PUBZ[i],ZPY):i in [1..n-r]];

```

We recover $g_0 = \tilde{\gamma}(0)$ (remember vinegar part of ZPY was set to zero above):

```

105 g0:=[K!EVALUATE(PUBZ0[i],[PR!0:i in [1..(n+v)^2]]):i in [1..n-r]];

```

and thus $Z \mapsto T(Z^2)$ together with its differential $(X, Y) \mapsto 2XY$

```

106 PUBZ2:=[PUBZ0[i]-g0[i]:i in [1..n-r]];
107 DPUBZ2:=[SUBMATRIX(S_-*DP[i]*TRANSPPOSE(S_-),1,1,n,n):i in [1..n-r]];

```

but also $(X, Y) \mapsto 2a^2XY$:

```

108 DPUBZA:=[A*DPUBZ2[i]*TRANSPPOSE(A):i in [1..n-r]];

```

This allows us to complete T into a full rank mapping T_- via $T_-(X) = \frac{1}{2}DP(X, 1)$:

```

109 SPA:=SPACE(DPUBZ2 cat DPUBZA, K, n*n); SP2:=SPACE(DPUBZ2, K, n*n);
110 W:=BASIS(COMPLEMENT(SPA, SP2));
111 DPPLUS:=DPUBZ2 cat [VEC2MAT(W[i],n) : i in [1..#W]];
112 T_-:=(K!2)^-1*MATRIX([VECTOR([(B[i]*DPPLUS[j], B[1])
113           :j in [1..n]]) : i in [1..n]]);

```

and to forge a signature for any message:

```

114 msg:=RANDOM(V_MESSAGE);
115 repeat
116 Y:=VECTOR(ELTSEQ(msg-VECTOR(g0)) cat ELTSEQ(RANDOM(V_RANDOM)));
117 is_square, sqrX:=IS SQUARE( V2E(Y*T_-^-1) ); until is_square;
118 forged:=VECTOR(ELTSEQ( E2V(sqrX)-beta_ ) cat ELTSEQ(V0))*S_-;
119 if VERIFY(msg, forged) then "Forged signature."; end if;

```


C Magma Script to Attack the Encryption Scheme

```

120 q:=31; n:=37; r:=3; K:=GF(q); E:=ext<K|n>;
121 Vi:=VECTORSPACE(K, n-r); Vo, K2V:=VECTORSPACE(E, K); V2K:=K2V-1;

```

Build the secret key, the encryption function P , and coefficients:

```

122 L1:=SUBMATRIX(RANDOM(GL(n, K)), 1, 1, n-r, n); L2:=RANDOM(GL(n, K));
123 l1:=RANDOM(GL(n, K))[1]; l2:=RANDOM(Vo);
124 PENCRIPT:=func< plain | K2V(V2K(plain*L1+l1)2*L2+l2)>;
125 PUBC0, PUBC1, PUBC2 := C012(K, Vi, PENCRIPT, n, n-r);

```

The mappings $\Delta = \{DP_{y_i}\}_{i \in [1, n-r]}$ for linearly independant y_1, \dots, y_{n-r} :

```

126 DP:=DIFF(PUBC2, K, n, n-r); Y:=RANDOM(GL(n-r, K));
127 Δ:=[TRANPOSE(MATRIX([Y[k]*DP[i] : i in [1..n]])) : k in [1..n-r]];

```

The set Λ of linear mappings verifying (9) for some parameter m :

```

128 m:=2; δ:=[Δ[i] : i in [m+1..n-r]]; SP:=SPACE(δ, K, (n-r)*n);
129 DUAL:=TRANPOSE(NULLSPACEMATRIX(TRANPOSE(BASISMATRIX(SP))));
130 P1:=TRANPOSE(MATRIX(PUBC1)); B:=BASIS(VECTORSPACE(K, n2));
131 MMUL:=func< A | MATRIX([MAT2VEC(A*VEC2MAT(B[i], n)): i in [1..#B]])>;
132 Λ:=&meet[NULLSPACE(MMUL(Δ[i])*DUAL): i in [1..m]]
133      meet NULLSPACE(MMUL(P1)*DUAL);

```

Compute the characteristic polynomial CP of a random linear mapping in Λ :

```

134 M:=VEC2MAT(RANDOM(Λ), n); CP:=FACTOREDCHARACTERISTICPOLYNOMIAL(M);
135 a:=ROOTS(POLYNOMIALRING(E) ! CP[1][1])[1][1];
136 A:=MULBY(a, K2V, V2K, BASIS(Vo));

```

Recover the secret elements:

```

137 res, L2_:= ISSIMILAR(M, A); R:=RANDOM(Vi);
138 v:=V2K(VECTOR([(R*DP[j], R): j in [1..n]])*L2_-1)/2;
139 res, s:=ISSQUARE(v);
140 if not res then L2_:= -L2_; res, s:=ISSQUARE(-v); end if;
141 l1_:=K2V(V2K(R*P1*L2_-1)/(2*s));
142 L1_:=P1*L2_-1*MULBY(1/V2K(2*l1_), K2V, V2K, BASIS(Vo));
143 l2_:=PENCRIPT(Vi ! 0) - K2V(V2K(l1_2))*L2_;
144 lml1_:=sub<Vo|[L1_[i]:i in [1..n-r]]>;

145 DISCLOSE:=function(cipher) // unlegitimate decryption!
146   is_square, root:=ISSQUARE(V2K((cipher-l2_)*L2_-1));
147   if is_square then Z:=K2V(root);
148     if (Z-l1_) in lml1_ then return true, SOLUTION(L1_, Z-l1_);
149     else if (-Z-l1_) in lml1_ then return true, SOLUTION(L1_, -Z-l1_);
150     else return false, _; end if; end if; else return false, _; end if;
151 end function;

152 plain:=RANDOM(Vi); b, p:=DISCLOSE(PENCRIPT(plain));
153 if b and (p eq plain) then "Decryption successful."; end if;

```

Factoring pq^2 with Quadratic Forms: Nice Cryptanalyses

Guilhem Castagnos^{1,*}, Antoine Joux^{2,3}, Fabien Laguillaumie⁴,
and Phong Q. Nguyen⁵

¹ Institut de Mathématiques de Bordeaux – Université Bordeaux 1
guilhem.castagnos@math.u-bordeaux1.fr

² PRISM – Université de Versailles St-Quentin-en-Yvelines

³ DGA

antoine.joux@m4x.org

⁴ GREYC – Université de Caen Basse-Normandie

fabien.laguillaumie@info.unicaen.fr

⁵ INRIA and ENS, France

<http://www.di.ens.fr/~pnguyen/>

Abstract. We present a new algorithm based on binary quadratic forms to factor integers of the form $N = pq^2$. Its heuristic running time is exponential in the general case, but becomes polynomial when special (arithmetic) hints are available, which is exactly the case for the so-called NICE family of public-key cryptosystems based on quadratic fields introduced in the late 90s. Such cryptosystems come in two flavours, depending on whether the quadratic field is imaginary or real. Our factoring algorithm yields a general key-recovery polynomial-time attack on NICE, which works for both versions: Castagnos and Laguillaumie recently obtained a total break of *imaginary*-NICE, but their attack could not apply to *real*-NICE. Our algorithm is rather different from classical factoring algorithms: it combines Lagrange’s reduction of quadratic forms with a provable variant of Coppersmith’s lattice-based root finding algorithm for homogeneous polynomials. It is very efficient given either of the following arithmetic hints: the public key of *imaginary*-NICE, which provides an alternative to the CL attack; or the knowledge that the regulator of the quadratic field $\mathbb{Q}(\sqrt{p})$ is unusually small, just like in *real*-NICE.

Keywords: Public-key Cryptanalysis, Factorisation, Binary Quadratic Forms, Homogeneous Coppersmith’s Root Finding, Lattices.

1 Introduction

Many public-key cryptosystems require the hardness of factoring large integers of the special form $N = pq^2$, such as Okamoto’s Esign [Oka90], Okamoto and Uchiyama’s encryption [OU98], Takagi’s fast RSA variants [Tak98], and the large family (surveyed in [BTV04]) of cryptosystems based on quadratic fields, which

* This work was done while this author was with the PRISM – Université de Versailles.

was initiated by Buchmann and Williams’ key exchange [BW88], and which includes NICE¹ cryptosystems [HPT99, PT99, PT00, JSW08] (whose main feature is a quadratic decryption). These moduli are popular because they can lead to special functionalities (like homomorphic encryption) or improved efficiency (compared to RSA). And no significant weakness has been found compared to standard RSA moduli of the form $N = pq$: to the best of our knowledge, the only results on pq^2 factorisation are [PO96, Per01, BDH99]. More precisely, [PO96, Per01] obtained a linear speed-up of Lenstra’s ECM, and [BDH99, Sect. 6] can factor in time $\tilde{O}(N^{1/9})$ when p and q are balanced. Furthermore, computing the “squarefree part” of an integer (that is, given $N \in \mathbb{N}$ as input, compute $(r, s) \in \mathbb{N}^2$ such that $N = r^2s$ with s squarefree) is a classical problem in algorithmic number theory (cf. [AM94]), because it is polynomial-time equivalent to determining the ring of integers of a number field [Chi89].

However, some of these cryptosystems actually provide additional information (other than N) in the public key, which may render factorisation easy. For instance, Howgrave-Graham [How01] showed that the public key of [Oka86] disclosed the secret factorisation in polynomial time, using the gcd extension of Coppersmith’s root finding method [Cop97]. Very recently, Castagnos and Laguillaumie [CL09] showed that the public key in the imaginary version [HPT99, PT99, PT00] of NICE allowed to retrieve the secret factorisation in polynomial time. And this additional information in the public key was crucial to make the complexity of decryption quadratic in *imaginary*-NICE, which was the main claimed benefit of NICE. But surprisingly, the attack of [CL09] does not work against REAL-NICE [JSW08], which is the version of NICE with real (rather than imaginary) quadratic fields, and which also offers quadratic decryption. In particular, the public key of REAL-NICE only consists of $N = pq^2$, but the prime p has special arithmetic properties.

OUR RESULTS. We present a new algorithm to factor integers of the form $N = pq^2$, based on binary quadratic forms (or equivalently, ideals of orders of quadratic number fields). In the worst case, its heuristic running time is exponential, namely $\tilde{O}(p^{1/2})$. But in the presence of special hints, it becomes heuristically polynomial. These hints are different from the usual ones of lattice-based factoring methods [Cop97, BDH99, How01] where they are a fraction of the bits of the secret prime factors. Instead, our hints are arithmetic, and correspond exactly to the situation of NICE, including both the imaginary [HPT99, PT99, PT00] and real versions [JSW08]. This gives rise to the first general key-recovery polynomial-time attack on NICE, using only the public key.

More precisely, our arithmetic hints can be either of the following two:

- i. The hint is an ideal equivalent to a secret ideal of norm q^2 in an imaginary quadratic field of discriminant $-pq^2$: in NICE, such an ideal is disclosed by the public key. This gives an alternative attack of NICE, different from [CL09].
- ii. The hint is the knowledge that the *regulator* of the quadratic field $\mathbb{Q}(\sqrt{p})$ is unusually small, just like in REAL-NICE. Roughly speaking, the regulator is a real number which determines how “dense” the units of the ring of integers

¹ For *New Ideal Coset Encryption*.

of the number field $\mathbb{Q}(\sqrt{p})$ are. This number is known to lie in the large interval $\left[\log\left(\frac{1}{2}(\sqrt{p-4} + \sqrt{p})\right), \sqrt{\frac{1}{2}p}\left(\frac{1}{2}\log p + 1\right)\right]$. But for infinitely many p (including square-free numbers of the form $p = k^2 + r$, where $p > 5$, $r|4k$ and $-k < r \leq k$, see [Deg58]), the regulator is at most polynomial in $\log p$. For these unusually small regulators, our algorithm heuristically runs in time polynomial in the bit-length of $N = pq^2$, which gives the first total break of REAL-NICE [JSW08]. We stress that although such p 's are easy to construct, their density is believed to be arbitrary small.

Interestingly, our algorithm is rather different from classical factoring algorithms. It is a combination of Lagrange's reduction of quadratic forms with a provable variant of Coppersmith's lattice-based root finding algorithm [Cop97] for homogeneous polynomials. In a nutshell, our factoring method first looks for a reduced binary quadratic form $f(x, y) = ax^2 + bxy + cy^2$ representing properly q^2 with small coefficients, *i.e.* there exist small coprime integers x_0 and y_0 such that $q^2 = f(x_0, y_0)$. In case i., such a quadratic form is already given. In case ii., such a quadratic form is found by a walk along the principal cycle of the class group of discriminant pq^2 , using Lagrange's reduction of (indefinite) quadratic forms. Finally, the algorithm finds such small coprime integers x_0 and y_0 such that $q^2 = f(x_0, y_0)$, by using the fact that $\gcd(f(x_0, y_0), pq^2)$ is large. This discloses q^2 and therefore the factorisation of N . In both cases, the search for x_0 and y_0 is done with a new *rigorous* homogeneous bivariate variant of Coppersmith's method, which might be of independent interest: by the way, it was pointed out to us that Bernstein [Ber08] independently used a similar method in the different context of Goppa codes decoding.

Our algorithm requires "natural" bounds on the roots of reduced quadratic forms of a special shape. We are unable to prove rigorously all these bounds, which makes our algorithm heuristic (like many factoring algorithms). But we have performed many experiments supporting such bounds, and the algorithm works very well in practice.

FACTORISATION AND QUADRATIC FORMS. Our algorithm is based on quadratic forms, which share a long history with factoring (see [CP01]). Fermat's factoring method represents N in two intrinsically different ways by the quadratic form $x^2 + y^2$. It has been improved by Shanks with SQUFOF, whose complexity is $\tilde{O}(N^{1/4})$ (see [GW08] for a detailed analysis). Like ours, this method works with the infrastructure of a class group of positive discriminant, but is different in spirit since it searches for an *ambiguous* form (after having found a square form), and does not focus on discriminants of a special shape. Schoof's factoring algorithms [Sch82] are also essentially looking for ambiguous forms. One is based on computation in class groups of complex quadratic orders and the other is close to SQUFOF since it works with real quadratic orders by computing a good approximation of the regulator to find an ambiguous form. Like SQUFOF, this algorithm does not take advantage of working in a non-maximal order and is rather different from our algorithm. Both algorithms of [Sch82] runs in $\tilde{O}(N^{1/5})$ under the generalised Riemann hypothesis. McKee's method [McK99]

is a speedup of Fermat’s algorithm (and was presented as an alternative to SQUFOF) with a heuristic complexity of $\tilde{O}(N^{1/4})$ instead of $\tilde{O}(N^{1/2})$.

SQUFOF and other exponential methods are often used to factor small numbers (say 50 to 100 bits), for instance in the post-sieving phase of the Number Field Sieve algorithm. Some interesting experimental comparisons can be found in [Mil07]. Note that the currently fastest rigorous *deterministic* algorithm actually has exponential complexity: it is based on a polynomial evaluation method (for a polynomial of the form $x(x - 1) \cdots (x - B + 1)$ for some bound B) and its best variant is described in [BGS07]. Finally, all sieve factoring algorithms are somewhat related to quadratic forms, since their goal is to find random pairs (x, y) of integers such that $x^2 \equiv y^2 \pmod N$. However, these algorithms factor generic numbers and have a subexponential complexity.

ROAD MAP. The rest of the paper is organised as follows. The first section recalls facts on quadratic fields and quadratic forms, and present our heuristic supported by experiments. The next section describes the homogeneous Copper-smith method and the following exhibits our main result: the factoring algorithm. The last section consists of the two cryptanalyses of cryptosystems based on real quadratic fields (REAL-NICE) and on imaginary quadratic fields (NICE).

2 Background on Quadratic Fields and Quadratic Forms

2.1 Quadratic Fields

Let $D \neq 0, 1$ be a squarefree integer and consider the quadratic number field $K = \mathbb{Q}(\sqrt{D})$. If $D < 0$ (resp. $D > 0$), K is called an *imaginary* (resp. a *real*) quadratic field. The *fundamental discriminant* Δ_K of K is defined as $\Delta_K = D$ if $D \equiv 1 \pmod 4$ and $\Delta_K = 4D$ otherwise. An *order* \mathcal{O} in K is a subset of K such that \mathcal{O} is a subring of K containing 1 and \mathcal{O} is a free \mathbb{Z} -module of rank 2. The ring \mathcal{O}_{Δ_K} of algebraic integers in K is the *maximal* order of K . It can be written as $\mathbb{Z} + \omega_K \mathbb{Z}$, where $\omega_K = \frac{1}{2}(\Delta_K + \sqrt{\Delta_K})$. If we set $f = [\mathcal{O}_{\Delta_K} : \mathcal{O}]$ the *finite* index of any order \mathcal{O} in \mathcal{O}_{Δ_K} , then $\mathcal{O} = \mathbb{Z} + f\omega_K \mathbb{Z}$. The integer f is called the *conductor* of \mathcal{O} . The discriminant of \mathcal{O} is then $\Delta_f = f^2 \Delta_K$. Now, let \mathcal{O}_Δ be an order of discriminant Δ and \mathfrak{a} be a nonzero ideal of \mathcal{O}_Δ , its norm is $N(\mathfrak{a}) = |\mathcal{O}_\Delta/\mathfrak{a}|$. A *fractional* ideal is a subset $\mathfrak{a} \subset K$ such that $d\mathfrak{a}$ is an ideal of \mathcal{O}_Δ for $d \in \mathbb{N}$. A fractional ideal \mathfrak{a} is said to be *invertible* if there exists another fractional ideal \mathfrak{b} such that $\mathfrak{a}\mathfrak{b} = \mathcal{O}_\Delta$. The *ideal class group* of \mathcal{O}_Δ is $C(\mathcal{O}_\Delta) = I(\mathcal{O}_\Delta)/P(\mathcal{O}_\Delta)$, where $I(\mathcal{O}_\Delta)$ is the group of invertible fractional ideals of \mathcal{O}_Δ and $P(\mathcal{O}_\Delta)$ the subgroup consisting of principal ideals. Its cardinality is the *class number* of \mathcal{O}_Δ denoted by $h(\mathcal{O}_\Delta)$. A nonzero ideal \mathfrak{a} of \mathcal{O}_Δ , \mathfrak{a} is said to be *prime to f* if $\mathfrak{a} + f\mathcal{O}_\Delta = \mathcal{O}_\Delta$. We denote by $I(\mathcal{O}_\Delta, f)$ the subgroup of $I(\mathcal{O}_\Delta)$ of ideals prime to f . The group \mathcal{O}_Δ^* of units in \mathcal{O}_Δ is equal to $\{\pm 1\}$ for all $\Delta < 0$, except when Δ is equal to -3 and -4 (\mathcal{O}_{-3}^* and \mathcal{O}_{-4}^* are respectively the group of sixth and fourth roots of unity). When $\Delta > 0$, then $\mathcal{O}_\Delta^* = \langle -1, \varepsilon_\Delta \rangle$ where $\varepsilon_\Delta > 0$ is called the *fundamental unit*. The real number $R_\Delta = \log(\varepsilon_\Delta)$ is

the regulator of \mathcal{O}_Δ . The following important bounds on the regulator of a real quadratic field can be found in [JLW95]:

$$\log \left(\frac{1}{2}(\sqrt{\Delta-4} + \sqrt{\Delta}) \right) \leq R_\Delta < \sqrt{\frac{1}{2}\Delta} \left(\frac{1}{2} \log \Delta + 1 \right). \tag{1}$$

The lower bound is reached *infinitely often*, for instance with $\Delta = x^2 + 4$ with $2 \nmid x$. Finally, this last proposition is the heart of both NICE and REAL-NICE.

Proposition 1 ([Cox99, Proposition 7.20] [Wei04, Theorem 2.16]). *Let \mathcal{O}_{Δ_f} be an order of conductor f in a quadratic field K .*

- i. If \mathfrak{A} is an \mathcal{O}_{Δ_K} -ideal prime to f , then $\mathfrak{A} \cap \mathcal{O}_{\Delta_f}$ is an \mathcal{O}_{Δ_f} -ideal prime to f of the same norm.*
- ii. If \mathfrak{a} is an \mathcal{O}_{Δ_f} -ideal prime to f , then $\mathfrak{a}\mathcal{O}_{\Delta_K}$ is an \mathcal{O}_{Δ_K} -ideal prime to f of the same norm.*
- iii. The map $\varphi_f : I(\mathcal{O}_{\Delta_f}, f) \rightarrow I(\mathcal{O}_{\Delta_K}, f)$, $\mathfrak{a} \mapsto \mathfrak{a}\mathcal{O}_{\Delta_K}$ is an isomorphism.*

The map φ_f from Proposition 1 induces a surjection $\bar{\varphi}_f : C(\mathcal{O}_{\Delta_f}) \twoheadrightarrow C(\mathcal{O}_{\Delta_K})$ which can be efficiently computed (see [PT00]). In our settings, we will use a prime conductor $f = q$ and consider $\Delta_q = q^2\Delta_K$, for a fundamental discriminant Δ_K . In that case, the order of the kernel of $\bar{\varphi}_q$ is given by the classical *analytic class number formula* (see for instance [BV07])

$$\frac{h(\mathcal{O}_{\Delta_q})}{h(\mathcal{O}_{\Delta_K})} = \begin{cases} q - (\Delta_K/q) & \text{if } \Delta_K < -4, \\ (q - (\Delta_K/q))R_{\Delta_K}/R_{\Delta_q} & \text{if } \Delta_K > 0. \end{cases} \tag{2}$$

Note that in the case of real quadratic fields, $\epsilon_{\Delta_q} = \epsilon_{\Delta_K}^t$ for a positive integer t , hence $R_{\Delta_q}/R_{\Delta_K} = t$ and $t \mid (q - (\Delta_K/q))$.

2.2 Representation of the Classes

Working with ideals modulo the equivalence relation of the class group is essentially equivalent to work with binary quadratic forms modulo $\text{SL}_2(\mathbb{Z})$ (cf. Section 5.2 of [Coh00]). Moreover, quadratic forms are more suited to an algorithmic point of view. Every ideal \mathfrak{a} of \mathcal{O}_Δ can be written as $\mathfrak{a} = m \left(a\mathbb{Z} + \frac{-b+\sqrt{\Delta}}{2}\mathbb{Z} \right)$ with $m \in \mathbb{Z}$, $a \in \mathbb{N}$ and $b \in \mathbb{Z}$ such that $b^2 \equiv \Delta \pmod{4a}$. In the remainder, we will only consider *primitive* integral ideals, which are those with $m = 1$. This notation also represents the binary quadratic form $ax^2 + bxy + cy^2$ with $b^2 - 4ac = \Delta$. This representation of the ideal is unique if the form is normal (see below). We recall here some facts about binary quadratic forms.

Definition 1. *A binary quadratic form f is a degree 2 homogeneous polynomial $f(x, y) = ax^2 + bxy + cy^2$ where a, b and c are integers, and is denoted by $[a, b, c]$. The discriminant of the form is $\Delta = b^2 - 4ac$. If $a > 0$ and $\Delta < 0$, the form is called definite positive and indefinite if $\Delta > 0$.*

Let $M \in \text{SL}_2(\mathbb{Z})$ with $M = \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix}$, and $f = [a, b, c]$, a binary quadratic form, then $f.M$ is the equivalent binary quadratic form $f(\alpha x + \beta y, \gamma x + \delta y)$.

Definite Positive Forms. Let us first define the crucial notion of *reduction*.

Definition 2. *The form $f = [a, b, c]$ is called normal if $-a < b \leq a$. It is called reduced if it is normal, $a \leq c$, and if $b \geq 0$ for $a = c$.*

The procedure which transforms a form $f = [a, b, c]$ into a normal one consists in setting s such that $b + 2sa$ belongs to the right interval (see [BV07, (5.4)]) and producing the form $[a, b + 2sa, as^2 + bs + c]$. Once a form $f = [a, b, c]$ is normalised, a *reduction step* consists in normalising the form $[c, -b, a]$. We denote this form by $\rho(f)$ and by Rho a corresponding algorithm. The reduction then consists in normalising f , and then iteratively replacing f by $\rho(f)$ until f is reduced. The time complexity of this (Lagrange-Gauß) algorithm is quadratic (see [BV07]). It returns a reduced form g which is equivalent to f modulo $\text{SL}_2(\mathbb{Z})$. We will call *matrix of the reduction*, the matrix M such that $g = f.M$. The reduction procedure yields a *uniquely* determined reduced form in the class modulo $\text{SL}_2(\mathbb{Z})$.

Indefinite Forms. Our main result will deal with forms of positive discriminant. Here is the definition of a reduced indefinite form.

Definition 3. *The form $f = [a, b, c]$ of positive discriminant Δ is reduced if $|\sqrt{\Delta} - 2|a|| < b < \sqrt{\Delta}$ and normal if $-|a| < b \leq |a|$ for $|a| \geq \sqrt{\Delta}$, and $\sqrt{\Delta} - 2|a| < b < \sqrt{\Delta}$ for $|a| < \sqrt{\Delta}$.*

The reduction process is similar to the definite positive case. The time complexity of the algorithm is still quadratic (see [BV07, Theorem 6.6.4]). It returns a reduced form g which is equivalent to f modulo $\text{SL}_2(\mathbb{Z})$. The main difference with forms of negative discriminant is that there will in general not exist a unique reduced form per class, but several organised in a cycle structure *i. e.*, when f has been reduced then subsequent applications of ρ give other reduced forms.

Definition 4. *Let f be an indefinite binary quadratic form, the cycle of f is the sequence $(\rho^i(g))_{i \in \mathbb{Z}}$ where g is a reduced form which is equivalent to f .*

From Theorem 6.10.3 from [BV07], the cycle of f consists of all reduced forms in the equivalence class of f . Actually, the complete cycle is obtained by a finite number of application of ρ as the process is periodic. It has been shown in [BTW95] that the period length ℓ of the sequence of reduced forms in each class of a class group of discriminant Δ satisfies $\frac{R_\Delta}{\log \Delta} \leq \ell \leq \frac{2R_\Delta}{\log 2} + 1$.

Our factoring algorithm will actually take place in the principal equivalence class. The following definition exhibits the *principal form* of discriminant Δ .

Definition 5. *The reduced form $[1, \lfloor \sqrt{\Delta} \rfloor, (\lfloor \sqrt{\Delta} \rfloor^2 - \Delta)/4]$ of discriminant Δ is called the principal form of discriminant Δ , and will be denoted 1_Δ .*

2.3 Reduction of the Forms $[q^2, kq, (k^2 \pm p)/4]$ and Heuristics

In this subsection, p and q are two distinct primes of the same bit-size λ and $p \equiv 1 \pmod 4$ (resp. $p \equiv 3 \pmod 4$) when we deal with positive (resp. negative) discriminant. Our goal is to factor the numbers pq^2 with the special normalised quadratic forms $[q^2, kq, (k^2 + p)/4]$ or $[q^2, kq, (k^2 - p)/4]$, depending whether we work with a negative discriminant $\Delta_q = -pq^2$ or with a positive one $\Delta_q = pq^2$. If p and q have the same size, these forms are clearly not reduced neither in the imaginary setting nor in real one. But as we shall see, we can find the reduced forms which correspond to the output of the reduction algorithm applied on these forms.

Suppose that we know a form \hat{f}_k , either definite positive or indefinite, which is the reduction of a form $f_k = [q^2, kq, (k^2 \pm p)/4]$ where k is an integer. Then \hat{f}_k represents the number q^2 . More precisely, if $M_k = \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} \in \text{SL}_2(\mathbb{Z})$ is the matrix such that $\hat{f}_k = f_k.M_k$, then $\hat{f}_k.M_k^{-1} = f_k$ and $q^2 = f_k(1, 0) = \hat{f}_k(\delta, -\gamma)$. In Section 3, we will see that provided they are relatively small compared to Δ_q , the values δ and $-\gamma$ can be found in polynomial time with a new variant of Coppersmith method. Our factoring algorithm can be sketched as follows: find such a form \hat{f}_k and if the coefficients of M_k are sufficiently small, retrieve δ and $-\gamma$ and the non-trivial factor q^2 of Δ_q . In this paragraph, we give some heuristics on the size of such a matrix M_k and discuss their relevance. If M is a matrix we denote by $|M|$ the max norm, *i. e.*, the maximal coefficient of M in absolute value.

In the imaginary case, it is showed in the proof of [CL09, Theorem 2] that the forms f_k belong to different classes of the kernel of the map $\bar{\varphi}_q$, depending on k , so the reduced equivalent forms \hat{f}_k are the unique reduced elements of the classes of the kernel. To prove the correctness of our attack on NICE, we need the following heuristic (indeed, the root finding algorithm of Section 3 recovers roots up to $|\Delta_q|^{1/9}$):

Heuristic 1 (Imaginary case). *Given a reduced element \hat{f}_k of a nontrivial class of $\ker \bar{\varphi}_q$, the matrix of reduction M_k is such that $|M_k| < |\Delta_q|^{1/9}$ with probability asymptotically close to 1.*

In the full version, we prove a probabilistic version of Heuristic 1. From Lemma 5.6.1 of [BV07], $|M_k| < 2 \max\{q^2, (k^2 + p)/4\} / \sqrt{pq^2}$. As f_k is normalised, $|k| \leq q$ and $|M_k| < 2q/\sqrt{p} \approx |\Delta_q|^{1/6}$. Note that we cannot reach such a bound with our root finding algorithm. Experimentally, for random k , $|M_k|$ can be much smaller. For example, if the bit-size λ of p and q equals 100, the mean value of $|M_k|$ is around $|\Delta_q|^{1/11.7}$. Our heuristic can be explained as follows. A well-known heuristic in the reduction of positive definite quadratic forms (or equivalently, two-dimensional lattices) is that if $[a, b, c]$ is a reduced quadratic form of discriminant Δ , then a and c should be close to $\sqrt{\Delta}$. This cannot hold for all reduced forms, but it can be proved to hold for an overwhelming majority of reduced forms. Applied to $\hat{f}_k = [a, b, c]$, this means that we expect a and c to be

close to $|\Delta_q|^{1/2}$. Now, recall that $q^2 = \hat{f}_k(\delta, -\gamma) = a\delta^2 - b\delta\gamma + c\gamma^2$, which leads to δ and γ close to $\sqrt{q^2/a} = q/\sqrt{a} \approx q/|\Delta_q|^{1/4} \approx |\Delta_q|^{1/12}$. Thus, we expect that $|M_k| \leq |\Delta_q|^{1/12}$. And this explains why we obtained experimentally the bound $|\Delta_q|^{1/11.7}$. Figure 1(a) shows a curve obtained by experimentation, which gives the probability that $|M_k| < |\Delta_q|^{1/9}$ for random k , in function of λ . This curve also supports our heuristic.

In the real case, we prove in the following theorem that $R_{\Delta_q}/R_{\Delta_K}$ forms f_k are principal and we exhibit the generators of the corresponding primitive ideals.

Theorem 1. *Let Δ_K be a fundamental positive discriminant, $\Delta_q = \Delta_K q^2$ where q is an odd prime conductor. Let ε_{Δ_K} (resp. ε_{Δ_q}) be the fundamental unit of \mathcal{O}_{Δ_K} (resp. \mathcal{O}_{Δ_q}) and t such that $\varepsilon_{\Delta_K}^t = \varepsilon_{\Delta_q}$. Then the principal ideals of \mathcal{O}_{Δ_q} generated by $q\varepsilon_{\Delta_K}^i$ correspond to quadratic forms $f_{k(i)} = [q^2, k(i)q, (k(i)^2 - p)/4]$ with $i \in \{1, \dots, t - 1\}$ and $k(i)$ is an integer defined modulo $2q$ computable from $\varepsilon_{\Delta_K}^i \pmod q$.*

Proof. Let $\alpha_i = q\varepsilon_{\Delta_K}^i$ with $i \in \{1, \dots, t - 1\}$. Following the proof of [BTW95, Proposition 2.9], we detail here the computation of $\mathfrak{a}_i = \alpha_i \mathcal{O}_{\Delta_q}$. Let x_i and y_i be two integers such that $\varepsilon_{\Delta_K}^i = x_i + y_i \omega_K$. Then $\alpha_i = qx_i + y_i q \Delta_K (1 - q)/2 + y_i \frac{1}{2}(\Delta_q + \sqrt{\Delta_q})$, and α_i is an element of \mathcal{O}_{Δ_q} . Let m_i, a_i and b_i be three integers such that $\mathfrak{a}_i = m_i \left(a_i \mathbb{Z} + \frac{-b_i + \sqrt{\Delta_q}}{2} \right)$. As mentioned in the proof of [BTW95,

Proposition 2.9], m_i is the smallest positive coefficient of $\sqrt{\Delta_q}/2$ in \mathfrak{a}_i . As \mathcal{O}_{Δ_q} is equal to $\mathbb{Z} + (\Delta_q + \sqrt{\Delta_q})/2\mathbb{Z}$, $\alpha_i \mathcal{O}_{\Delta_q}$ is generated by α_i and $\alpha_i(\Delta_q + \sqrt{\Delta_q})/2$ as a \mathbb{Z} -module. So a simple calculation gives that $m_i = \gcd(y_i, q(x_i + y_i \Delta_K/2))$. As $\varepsilon_{\Delta_K}^i$ is not an element of \mathcal{O}_{Δ_q} , we have $\gcd(y_i, q) = 1$ so $m_i = \gcd(y_i, x_i + y_i \Delta_K/2)$. The same calculation to find m'_i for the ideal $\varepsilon_{\Delta_K}^i \mathcal{O}_{\Delta_K}$ reveals that $m_i = m'_i$. As $\varepsilon_{\Delta_K}^i \mathcal{O}_{\Delta_K} = \mathcal{O}_{\Delta_K}$ we must have $m'_i = 1$. Now, $N(\mathfrak{a}_i) = |N(\alpha_i)| = q^2$ and $N(\mathfrak{a}_i) = m'_i a_i = a_i$ and therefore $a_i = q^2$. Let us now find b_i . Note that b_i is defined modulo $2a_i$. Since $\alpha_i \in \alpha_i \mathcal{O}_{\Delta_q}$, there exist μ_i and ν_i such that $\alpha_i = a_i \mu_i + (-b_i + \sqrt{\Delta_q})/2\nu_i$. By identification in the basis $(1, \sqrt{\Delta_q})$, $\nu_k = 1$ and by a multiplication by 2, we obtain $2qx_i + qy_i \Delta_K \equiv -b_i y_i \pmod{2a_i}$. As $b_i \equiv \Delta_q \pmod{2}$, we only have to determine b_i modulo q^2 . As y_i is prime to q , we have $b_i \equiv k(i)q \pmod{q^2}$ with $k(i) \equiv -2x_i/y_i - \Delta_K \pmod{q}$. Finally, as we must have $-a_i < b \leq a_i$ if $a_i > \sqrt{\Delta_q}$ and else $\sqrt{\Delta_q} - 2a_i < b < \sqrt{\Delta_q}$, $k(i)$ is the unique integer with $k(i) \equiv \Delta_q \pmod{2}$ and $k(i) \equiv -2x_i/y_i - \Delta_K \pmod{q}$, such that $b = k(i)q$ satisfies that inequalities. Eventually, the principal ideal of \mathcal{O}_{Δ_q} generated by $q\varepsilon_{\Delta_K}^i$ corresponds to the form $[q^2, k(i)q, c_i]$ with $c_i = (b_i^2 - \Delta_q)/(4a_i) = (k(i)^2 - \Delta_K)/4$. \square

From this theorem, we see that if we go across the cycle of principal forms, then we will find reduced forms \hat{f}_k . To analyse the complexity of our factoring algorithm, we have to know the distribution of these forms on the cycle. An appropriate tool is the Shanks distance d (see [BV07, Definition 10.1.4]) which is close to the number of iterations of Rho between two forms. One has $d(\mathbf{1}_{\Delta_q}, f_{k(i)}) = iR_{\Delta_K}$. From Lemma 10.1.8 of [BV07], $|d(\hat{f}_{k(i)}, f_{k(i)})| < \log q$, for

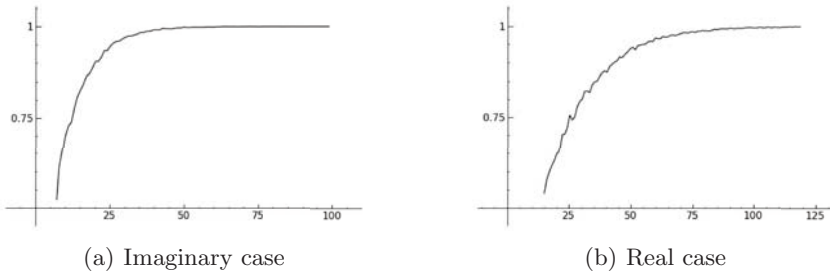


Fig. 1. Probability that $|M_k| < |\Delta_q|^{1/9}$ in function of the bit-size λ of p and q

all $i = 1, 2, \dots, t - 1$. Let j be the smallest integer such that $0 < jR_{\Delta_K} - 2 \log q$, then as $jR_{\Delta_K} = d(f_{k(i)}, f_{k(i+j)}) = d(f_{k(i)}, \hat{f}_{k(i)}) + d(\hat{f}_{k(i)}, \hat{f}_{k(i+j)}) + d(\hat{f}_{k(i+j)}, f_{k(i+j)})$, from the triangle inequality, one has $jR_{\Delta_K} < 2 \log(q) + |d(\hat{f}_{k(i)}, \hat{f}_{k(i+j)})|$. So, $|d(\hat{f}_{k(i)}, \hat{f}_{k(i+j)})| > jR_{\Delta_K} - 2 \log q > 0$. This inequality proves that $f_{k(i)}$ and $f_{k(i+j)}$ do not reduce to the same form. Experiments actually show that asymptotically, $|d(\hat{f}_{k(i)}, f_{k(i)})|$ is very small on average (smaller than 1). As a consequence, as pictured in figure 2, $d(\mathbf{1}_{\Delta_q}, \hat{f}_{k(i)}) \approx iR_{\Delta_K}$.

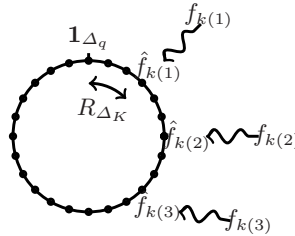


Fig. 2. Repartition of the forms $\hat{f}_{k(i)}$ along the principal cycle

Moreover, as in the imaginary case, experiments show that asymptotically the probability that the norm of the matrices of reduction, $|M_k|$ is smaller than $\Delta_q^{1/9}$ is close to 1 (see figure 1(b)). This leads to the following heuristic.

Heuristic 2 (Real case). *From the principal form $\mathbf{1}_{\Delta_q}$, a reduced form \hat{f}_k such that the matrix of the reduction, M_k , satisfy $|M_k| < \Delta_q^{1/9}$, can be found in $\mathcal{O}(R_{\Delta_K})$ successive applications of Rho .*

We did also some experiments to investigate the case where the bit-sizes of p and q are unbalanced. In particular when the size of q grows, the norm of the matrix of reduction becomes larger. For example, for a 100-bit p and a 200-bit q (resp. a 300-bit q), more than 95% (resp. 90%) of the \hat{f}_k have a matrix M_k with $|M_k| < \Delta_q^{1/6.25}$ (resp. $|M_k| < \Delta_q^{1/5.44}$).

3 A Rigorous Homogeneous Variant of Coppersmith’s Root Finding Method

Our factoring algorithm searches many times for small modular roots of degree two homogeneous polynomials and the most popular technique to find them is based on Coppersmith’s method (see [Cop97](#) or May’s survey [May07](#)). Our problem is the following: Given $f(x, y) = x^2 + bxy + cy^2$ a (monic) binary quadratic form and $N = pq^2$ an integer of unknown factorisation, find $(x_0, y_0) \in \mathbb{Z}^2$ such that $f(x_0, y_0) \equiv 0 \pmod{q^2}$, while $|x_0|, |y_0| \leq M$, where $M \in \mathbb{N}$. The usual technique for this kind of problems is only heuristic, since it is the gcd extension of bivariate congruences. Moreover, precise bounds cannot be found in the litterature. Fortunately, because our polynomial is homogeneous, we will actually be able to prove the method. This homogenous variant is quite similar to the one-variable standard Coppersmith method, but is indeed even simpler to describe and more efficient since there is no need to balance coefficients. We denote as $\|\cdot\|$ the usual Euclidean norm for polynomials. The main tool to solve this problem is given by the following variant of the widespread elementary Howgrave-Graham’s lemma [How97](#).

Lemma 1. *Let $B \in \mathbb{N}$ and $g(x, y) \in \mathbb{Z}[x, y]$ be a homogeneous polynomial of total degree δ . Let $M > 0$ be a real number and suppose that $\|g(x, y)\| < \frac{B}{\sqrt{\delta+1}M^\delta}$ then for all $x_0, y_0 \in \mathbb{Z}$ such that $g(x_0, y_0) \equiv 0 \pmod{B}$ and $|x_0|, |y_0| \leq M$, $g(x_0, y_0) = 0$.*

Proof. Let $g(x, y) = \sum_{i=0}^\delta g_i x^i y^{\delta-i}$ where some g_i s might be zero. We have

$$\begin{aligned} |g(x_0, y_0)| &\leq \sum_{i=0}^\delta |g_i| |x_0^i y_0^{\delta-i}| \leq M^\delta \sum_{i=0}^\delta |g_i| \\ &\leq M^\delta \sqrt{\delta+1} \|g(x, y)\| < B \end{aligned}$$

and therefore $g(x_0, y_0) = 0$. □

The trick is then to find only one small enough bivariate homogeneous polynomial satisfying the conditions of this lemma and to extract the rational root of the corresponding univariate polynomial with standard techniques. On the contrary, the original Howgrave-Graham’s lemma suggests to look for two polynomials of small norm having (x_0, y_0) as integral root, and to recover it *via* elimination theory. The usual way to obtain these polynomials is to form a lattice spanned by a special family of polynomials, and to use the LLL algorithm (cf. [LLL82](#)) to obtain the two “small” polynomials. Unfortunately, this reduction does not guarantee that these polynomials will be algebraically independent, and the elimination can then lead to a trivial relation. Consequently, this bivariate approach is heuristic. Fortunately, for homogeneous polynomials, we can take another approach by using Lemma [1](#) and then considering a univariate polynomial with a rational root. This makes the method rigorous and slightly simpler since we need a bound on $\|g(x, y)\|$ and not on $\|g(xX, yY)\|$ if X and Y are bounds on the roots and therefore the resulting lattice has smaller determinant than in the classical bivariate approach.

To evaluate the maximum of the bound we can obtain, we need the size of the first vector provided by LLL which is given by:

Lemma 2 (LLL). *Let L be a full-rank lattice in \mathbb{Z}^d spanned by an integer basis $\mathcal{B} = \{b_1, \dots, b_d\}$. The LLL algorithm, given \mathcal{B} as input, will output in time $O(d^6 \log^3(\max \|b_i\|))$ a non-zero vector $u \in L$ satisfying $\|u\| \leq 2^{(d-1)/4} \det(L)^{1/d}$.*

We will now prove the following general result regarding the modular roots of bivariate homogeneous polynomials which can be of independent interest.

Theorem 2. *Let $f(x, y) \in \mathbb{Z}[x, y]$ be a homogeneous polynomial of degree δ with $f(x, 0) = x^\delta$, N be a non-zero integer and α be a rational number in $[0, 1]$, then one can retrieve in polynomial time in $\log N$, δ and the bit-size of α , all the rationals x_0/y_0 , where x_0 and y_0 are integers such that $\gcd(f(x_0, y_0), N) \geq N^\alpha$ and $|x_0|, |y_0| \leq N^{\alpha^2/(2\delta)}$.*

Proof. Let b be a divisor of N for which there exists $(x_0, y_0) \in \mathbb{Z}^2$ such that $b = \gcd(f(x_0, y_0), N) \geq N^\alpha$. We define some integral parameters (to be specified later) m, t and t' with $t = m + t'$ and construct a family of $\delta t + 1$ homogeneous polynomials g and h of degree δt such that (x_0, y_0) is a common root modulo b^m . More precisely, we consider the following polynomials

$$\begin{cases} g_{i,j}(x, y) = x^j y^{\delta(t-i)-j} f^i N^{m-i} & \text{for } i = 0, \dots, m-1, j = 0, \dots, \delta-1 \\ h_i(x, y) = x^i y^{\delta t' - i} f^m & \text{for } i = 0, \dots, \delta t'. \end{cases}$$

We build the triangular matrix L of dimension $\delta t + 1$, containing the coefficients of the polynomials $g_{i,j}$ and h_i . We will apply LLL to the lattice spanned by the rows of L . The columns correspond to the coefficients of the monomials $y^{\delta t}, xy^{\delta t-1}, \dots, x^{\delta t-1}y, x^{\delta t}$. Let $\beta \in [0, 1]$ such that $M = N^\beta$. The product of the diagonal elements gives $\det(L) = N^{\delta m(m+1)/2}$. If we omit the quantities that do not depend on N , to satisfy the inequality of Lemma 1 with the root bound M , the LLL bound from Lemma 2 implies that we must have

$$\delta m(m+1)/2 \leq (\delta t + 1)(\alpha m - \delta t \beta) \tag{3}$$

and if we set λ such that $t = \lambda m$, this gives asymptotically $\beta \leq \frac{\alpha}{\delta \lambda} - \frac{1}{2\delta \lambda^2}$, which is maximal when $\lambda = \frac{1}{\alpha}$, and in this case, $\beta_{\max} = \alpha^2/(2\delta)$. The vector output by LLL gives a homogeneous polynomial $\tilde{f}(x, y)$ such that $\tilde{f}(x_0, y_0) = 0$ thanks to Lemma 1. Let $r = x/y$, any rational root of the form x_0/y_0 can be found by extracting the rational roots of $\tilde{f}'(r) = 1/y^{\delta t} \tilde{f}(x, y)$ with classical methods. \square

For the case we are most interested in, $\delta = 2$, $N = pq^2$ with p and q of the same size, *i. e.*, $\alpha = 2/3$ then $\lambda = 3/2$ and we can asymptotically get roots up to N^β with $\beta = \frac{1}{9}$. If we take $m = 4$ and $t = 6$, *i. e.*, we work with a lattice of dimension 13, we get from (3) that $\beta \approx \frac{1}{10.63}$ and with a 31-dimensional lattice ($m = 10$ and $t = 15$), $\beta \approx \frac{1}{9.62}$. If the size of q grows compared to p , *i. e.*, α increases towards 1, then β increases towards $1/4$. For example, if q is two times larger than p , *i. e.*, $\alpha = 4/5$ then $\beta = 1/6.25$. For $\alpha = 6/7$, we get $\beta \approx 1/5.44$.

We will call `HomogeneousCoppersmith` the algorithm which implements this method. It takes as input an integer $N = pq^2$ and a binary quadratic form $[a, b, c]$, from which we deduce the unitary polynomial $x^2 + b'xy + c'y^2$, by dividing both b and c by a modulo N , and the parameters m and t . In fact, this method will only disclose proper representations of q^2 , those for which x and y are coprime, but we note that f_k properly represents q^2 , and therefore so does our form $[a, b, c]$.

The case $\alpha = 1$ of Theorem 2 can already be found in Joux’s book [Jou09] and we mention that a similar technique has already been independently investigated by Bernstein in [Ber08].

4 A $\tilde{O}(p^{1/2})$ -Deterministic Factoring Algorithm for pq^2

We detail our new quadratic form-based factoring algorithm for numbers of the form pq^2 . In this section, p and q will be of same bit-size, and $p \equiv 1 \pmod{4}$.

4.1 The Algorithm

Roughly speaking, if $\Delta_q = N = pq^2$, our factoring algorithm, depicted in Fig. 3, exploits the fact that the non-reduced forms $f_k = [q^2, kq, -]$ reduce to forms \hat{f}_k for which there exists a small pair (x_0, y_0) such that $q^2 \mid \hat{f}_k(x_0, y_0)$ while $q^2 \nmid N$. From Theorem 1, we know that these reduced forms appear on the principal cycle of the class group of discriminant Δ_q . To detect them, we start a walk in the principal cycle from the principal form $\mathbf{1}_N$, and apply `Rho` until the Coppersmith-like method finds these small solutions.

Input: $N = pq^2, m, t$
Output: p, q
<ol style="list-style-type: none"> 1. $h \leftarrow \mathbf{1}_N$ 2. while (x_0, y_0) not found do <ol style="list-style-type: none"> 2.1. $h \leftarrow \text{Rho}(h)$ 2.2. $x_0/y_0 \leftarrow \text{HomogeneousCoppersmith}(h, N, m, t)$ 3. $q \leftarrow \text{Sqrt}(\text{Gcd}(h(x_0, y_0), N))$ 4. return $(N/q^2, q)$

Fig. 3. Factoring $N = pq^2$

4.2 Heuristic Correctness and Analysis of Our Algorithm

Assuming Heuristic 2, starting from $\mathbf{1}_N$, after $O(R_p)$ iterations, the algorithm will stop on a reduced form whose roots will be found with our Coppersmith-like method (for suitable values of m and t) since they will satisfy the expected $N^{1/9}$ bound. The computation of $\text{gcd}(h(x_0, y_0), N)$ will therefore expose q^2 and factor N . The time complexity of our algorithm is then heuristically $O(R_p \text{Poly}(\log N))$, whereas the space complexity is $O(\log N)$. The worst-case

complexity is $O(p^{1/2} \log p \text{Poly}(\log N))$. For small regulators, such as in REAL-NICE cryptosystem (see. Subsection 5.1), the time complexity is polynomial.

This algorithm can be generalised with a few modifications to primes p such that $p \equiv 3 \pmod{4}$, by considering $\Delta_q = 4pq^2$. Moreover if the bit-sizes of p and q are unbalanced, our experiments suggest that the size of the roots will be small enough (see end of Subsection 2.3 and Section 3), so the factoring algorithm will also work in this case, with the same complexity.

Comparison with other Deterministic Factorisation Methods. Boneh, Durfee and Howgrave-Graham presented in [BDH99] an algorithm for factoring integers $N = p^r q$. Their main result is the following:

Lemma 3 ([BDH99]). *Let $N = p^r q$ be given, and assume $q < p^c$ for some c . Furthermore, assume that P is an integer satisfying $|P - p| < p^{1 - \frac{r}{r+c} - 2\frac{1}{2}}$. Then the factor p may be computed from N , r , c and P by an algorithm whose running time is dominated by the time it takes to run LLL on a lattice of dimension d .*

For $r = 2$ and $c = 1$, this leads to a deterministic factoring algorithm which consists in exhaustively search for an approximation P of p and to solve the polynomial equation $(P + X)^2 \equiv 0 \pmod{p^2}$ with a method à la Coppersmith. The approximation will be found after $O(p^{1/3}) = O(N^{1/9})$ iterations.

The fastest deterministic generic integer factorisation algorithm is actually a version of Strassen's algorithm [Str76] from Bostan, Gaudry and Schost [BGS07], who ameliorates a work of Chudnovsky and Chudnovsky [CC87] and proves a complexity of $O(M_{\text{int}}(\sqrt[4]{N} \log N))$ where M_{int} is a function such that integers of bit-size d can be multiplied in $M_{\text{int}}(d)$ bit operations. More precisely, for numbers of our interest, Lemma 13 from [BGS07] gives the precise complexity:

Lemma 4 ([BGS07]). *Let b, N be two integers with $2 \leq b < N$. One can compute a prime divisor of N bounded by b , or prove that no such divisor exists in $O\left(M_{\text{int}}(\sqrt{b} \log N) + \log b M_{\text{int}}(\log N) \log \log N\right)$ bit operations and space $O(\sqrt{b} \log N)$ bits.*

In particular, for $b = N^{1/3}$, the complexity is $\tilde{O}(N^{1/6})$, with a very large space complexity compared to our algorithm. Moreover, none of these two last of algorithms can actually factor an integer of cryptographic size. The fact that a prime divisor has a small regulator does not help in these algorithms, whereas it makes the factorisation polynomial in our method.

5 Cryptanalysis of the NICE Cryptosystems

Hartmann, Paulus and Takagi proposed the elegant NICE encryption scheme (see [HPT99, PT99, PT00]), based on imaginary quadratic fields and whose main feature was a quadratic decryption time. Later on, several other schemes, including (special) signature schemes relying on this framework have been proposed. The public key of these NICE cryptosystems contains a discriminant $\Delta_q = -pq^2$

together with a reduced ideal \mathfrak{h} whose class belongs to the kernel of $\bar{\varphi}_q$. The idea underlying the NICE cryptosystem is to hide the message behind a random element $[\mathfrak{h}]^r$ of the kernel. Applying $\bar{\varphi}_q$ will make this random element disappear, and the message will then be recovered.

In [JSW08], Jacobson, Scheidler and Weimer embedded the original NICE cryptosystem in *real* quadratic fields. Whereas the idea remains essentially the same as the original, the implementation is very different. The discriminant is now $\Delta_q = pq^2$, but because of the differences between imaginary and real setting, these discriminant will have to be chosen carefully. Among these differences, the class numbers are expected to be small with very high probability (see the Cohen-Lenstra heuristics [CL84]). Moreover, an equivalence class does not contain a *unique* reduced element anymore, but a multitude of them, whose number is governed by the size of the fundamental unit. The rough ideas to understand these systems and our new attacks are given in the following. The full description of the systems is omitted for lack of space but can be found in [HPT99, JSW08].

5.1 Polynomial-Time Key Recovery in the Real Setting

The core of the design of the REAL-NICE encryption scheme is the very particular choice of the secret prime numbers p and q such that $\Delta_K = p$ and $\Delta_q = pq^2$. They are chosen such that the ratio $R_{\Delta_q}/R_{\Delta_K}$ is of order of magnitude of q and that R_{Δ_K} is bounded by a polynomial in $\log(\Delta_K)$. To ensure the first property, it is sufficient to choose q such that $q - \left(\frac{\Delta_K}{q}\right)$ is a small multiple of a large prime. If the second property is very unlikely to naturally happen since the regulator of p is generally of the order of magnitude of \sqrt{p} , it is indeed quite easy to construct fundamental primes with small regulator. The authors of [JSW08] suggest to produce a prime p as a so-called *Schinzel sleeper*, which is a positive squarefree integer of the form $p = a^2x^2 + 2bx + c$ with a, b, c, x in \mathbb{Z} , $a \neq 0$ and $b^2 - 4ac$ dividing $4\gcd(a^2, b)^2$. Schinzel sleepers are known to have a regulator of the order $\log(p)$ (see [CW05]). Some care must be taken when setting the (secret) a, b, c, x values, otherwise the resulting $\Delta_q = pq^2$ is subject to factorisation attacks described in [Wei04]. We do not provide here more details on these choices since the crucial property for our attack is the fact that the regulator is actually of the order $\log(p)$. The public key consists of the sole discriminant Δ_q . The message is carefully embedded (and padded) into a primitive \mathcal{O}_{Δ_q} -ideal so that it will be recognised during decryption. Instead of moving the message ideal \mathfrak{m} to a different equivalence class (like in the imaginary case), the encryption actually hides the message in the cycle of reduced ideal of its own equivalent class by multiplication of a random principal \mathcal{O}_{Δ_q} -ideal \mathfrak{h} (computed during encryption). The decryption process consists then in applying the (secret) map $\bar{\varphi}_q$ and perform an exhaustive search for the padded message in the *small* cycle of $\bar{\varphi}_q([\mathfrak{m}\mathfrak{h}])$. This exhaustive search is actually possible thanks to the choice of p which has a very small regulator. Like in the imaginary case, the decryption procedure has a quadratic complexity and significantly outperforms an RSA decryption for any given security level (see Table 3 from [JSW08]).

Unfortunately, due to the particular but necessary choice of the secret prime p , the following result states the total insecurity of the REAL-NICE system.

Result 1. *Algorithm [3] recovers the secret key of REAL-NICE in polynomial time in the security parameter under Heuristic [2] since the secret fundamental discriminant p is chosen to have a regulator bounded by a polynomial in $\log p$.*

We apply the cryptanalysis on the following example. The Schinzel polynomial $S(X) = 2725^2 X^2 + 2 \cdot 3815X + 2$ produces a suitable 256-bit prime p for the value $X_0 = 103042745825387139695432123167592199$. This prime has a regulator $R_{\Delta_K} \simeq 90.83$. The second 256-bit prime q is chosen following the recommendations from [Wei04]. This leads to a the discriminant

$$\Delta_q = 28736938823310044873380716142282073396186843906757463274792638734144060602830510 \\ 80738669163489273592599054529442271053869832485363682341892124500678400322719842 \\ 63278692833860326257638544601057379571931906787755152745236263303465093$$

Our algorithm recovers the prime

$$q = 60372105471499634417192859173853663456123015267207769653235558092781188395563$$

from Δ_q after 45 iterations in 42.42 seconds on a standard laptop. The rational root is $\frac{x_0}{y_0}$ equal to $-\frac{2155511611710996445623}{3544874277134778658948}$, where x_0 and y_0 satisfy $\frac{\log(\Delta_q)}{\log(|x_0|)} \simeq 10.8$ and $\frac{\log(\Delta_q)}{\log(|y_0|)} \simeq 10.7$.

5.2 Polynomial-Time Key Recovery of the Original NICE

As mentioned above, the public key of the original NICE cryptosystem contains the representation of a reduced ideal \mathfrak{h} whose class belongs to the kernel of the surjection $\bar{\varphi}_q$. The total-break of the NICE cryptosystem is equivalent to solving the following *kernel problem*.

Definition 6 (Kernel Problem [BPT04]). *Let λ be an integer, p and q be two λ -bit primes with $p \equiv 3 \pmod{4}$. Fix a non-fundamental discriminant $\Delta_q = -pq^2$. Given an element $[\mathfrak{h}]$ of $\ker \bar{\varphi}_q$, factor the discriminant Δ_q .*

Castagnos and Laguillaumie proposed in [CL09] a polynomial-time algorithm to solve this problem. We propose here a completely different solution within the spirit of our factorisation method and whose complexity is also polynomial-time. As discuss in Subsection 2.3, the idea is to benefit from the fact that the public ideal \mathfrak{h} corresponds to a reduced quadratic form, \hat{f}_k , which represents q^2 . We thus find these x_0 and y_0 such that $\gcd(\hat{f}_k(x_0, y_0), \Delta_q) = q^2$ with the Coppersmith method of Section [3].

Result 2. *The Homogeneous Coppersmith method from Section [3] solves the Kernel Problem in polynomial time in the security parameter under Heuristic [7].*

We apply our key recovery on the example of NICE proposed in [JJ00, CL09]:

$$\Delta_q = -1001133619402846750073919037082619174565372425946674915149340539464219927955168 \\ 18216760083640752198709726199732701843864411853249644535365728802022498185665592 \\ 9837085464532821079127759142567629134901322152002224671621236001656120923$$

$$a = 5702268770894258318168588438117558871300783180769995195092715895755173700399 \\ 141486895731384747$$

$$b = 3361236040582754784958586298017949110648731745605930164666819569606755029773 \\ 074415823039847007$$

The public key consists in Δ_q and $\mathfrak{h} = (a, b)$. Our Coppersmith method finds in less than half a second the root $u_0 = \frac{-103023911}{349555951} = \frac{x_0}{y_0}$ and

$$h(x_0, y_0) = 5363123171977038839829609999282338450991746328236957351089 \\ 4245774887056120365979002534633233830227721465513935614971 \\ 593907712680952249981870640736401120729 = q^2.$$

All our experiments have been run on a standard laptop under Linux with software Sage. The lattice reduction has been performed with Stehlé's `fpLLL` [Ste].

Acknowledgements. We warmly thank Denis Simon and Brigitte Vallée for helpful discussions and the reviewers for their useful comments. Part of this work was supported by the Commission of the European Communities through the ICT program under contract ICT-2007-216676 ECRYPT II.

References

- [AM94] Adleman, L.M., McCurley, K.S.: Open problems in number theoretic complexity, II. In: Huang, M.-D.A., Adleman, L.M. (eds.) ANTS 1994. LNCS, vol. 877, pp. 291–322. Springer, Heidelberg (1994)
- [BDH99] Boneh, D., Durfee, G., Howgrave-Graham, N.: Factoring $N = p^r q$ for large r . In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 326–337. Springer, Heidelberg (1999)
- [Ber08] Bernstein, D.J.: List decoding for binary Goppa codes, (2008) Preprint <http://cr.yp.to/papers.html#goppalist>
- [BGS07] Bostan, A., Gaudry, P., Schost, É.: Linear Recurrences with Polynomial Coefficients and Application to Integer Factorization and Cartier-Manin Operator. *SIAM J. Comput.* 36(6), 1777–1806 (2007)
- [BPT04] Biehl, I., Paulus, S., Takagi, T.: Efficient Undeniable Signature Schemes based on Ideal Arithmetic in Quadratic Orders. *Des. Codes Cryptography* 31(2), 99–123 (2004)
- [BTV04] Buchmann, J., Takagi, T., Vollmer, U.: Number Field Cryptography. In: van der Poorten, Stein (eds.) *High Primes & Misdemeanours: Lectures in Honour of the 60th Birthday of Hugh Cowie Williams*. Fields Institute Communications, vol. 41, pp. 111–125. AMS (2004)
- [BTW95] Buchmann, J., Thiel, C., Williams, H.C.: Short Representation of Quadratic Integers. In: Proc. of CANT 1992, *Math. Appl.*, vol. 325, pp. 159–185. Kluwer Academic Press, Dordrecht (1995)
- [BV07] Buchmann, J., Vollmer, U.: *Binary Quadratic Forms. An Algorithmic Approach*. Springer, Heidelberg (2007)

- [BW88] Buchmann, J., Williams, H.C.: A Key-Exchange System based on Imaginary Quadratic Fields. *J. Cryptology* 1, 107–118 (1988)
- [CC87] Chudnovsky, D.V., Chudnovsky, G.V.: Approximations and Complex Multiplication According to Ramanujan. In: *Ramanujan Revisited: Proceedings*, pp. 375–472. Academic Press, Boston (1987)
- [Chi89] Chistov, A.L.: The complexity of constructing the ring of integers of a global field. *Dokl. Akad. Nauk. SSSR*, 306, 1063–1067 (1989); English translation: *Soviet. Math. Dokl.* 39, 597–600 (1989)
- [CL84] Cohen, H., Lenstra Jr., H.W.: Heuristics on class groups. *Springer LNM*, vol. 1052, pp. 26–36 (1984)
- [CL09] Castagnos, G., Laguillaumie, F.: On the Security of Cryptosystems with Quadratic Decryption: The Nicest Cryptanalysis. In: Joux, A. (ed.) *EUROCRYPT 2009*. LNCS, vol. 5479, pp. 260–277. Springer, Heidelberg (2009)
- [Coh00] Cohen, H.: *A Course in Computational Algebraic Number Theory*. Springer, Heidelberg (2000)
- [Cop97] Coppersmith, D.: Small Solutions to Polynomial Equations, and Low Exponent RSA Vulnerabilities. *J. Cryptology* 10(4), 233–260 (1997)
- [Cox99] Cox, D.A.: *Primes of the form $x^2 + ny^2$* . John Wiley & Sons, Chichester (1999)
- [CP01] Crandall, R., Pomerance, C.: *Prime Numbers: A Computational Perspective*. Springer, Heidelberg (2001)
- [CW05] Cheng, K.H.F., Williams, H.C.: Some Results Concerning Certain Periodic Continued Fractions. *Acta Arith.* 117, 247–264 (2005)
- [Deg58] Degert, G.: Über die Bestimmung der Grundeinheit gewisser reell- quadratischer Zahlkörper. *Abh. Math. Sem. Univ. Hamburg* 22, 92–97 (1958)
- [GW08] Gower, J.E., Wagstaff Jr., S.S.: Square form factorization. *Math. Comput.* 77(261), 551–588 (2008)
- [How97] Howgrave-Graham, N.: Finding small roots of univariate modular equations revisited. In: Darnell, M.J. (ed.) *Cryptography and Coding 1997*. LNCS, vol. 1355, pp. 131–142. Springer, Heidelberg (1997)
- [How01] Howgrave-Graham, N.: Approximate Integer Common Divisors. In: Silverman, J.H. (ed.) *CaLC 2001*. LNCS, vol. 2146, pp. 51–66. Springer, Heidelberg (2001)
- [HPT99] Hartmann, M., Paulus, S., Takagi, T.: NICE - New Ideal Coset Encryption. In: Koç, Ç.K., Paar, C. (eds.) *CHES 1999*. LNCS, vol. 1717, pp. 328–339. Springer, Heidelberg (1999)
- [JJ00] Jaulmes, É., Joux, A.: A NICE Cryptanalysis. In: Preneel, B. (ed.) *EUROCRYPT 2000*. LNCS, vol. 1807, pp. 382–391. Springer, Heidelberg (2000)
- [JLW95] Jacobson Jr., M.J., Lukes, R.F., Williams, H.C.: An investigation of bounds for the regulator of quadratic fields. *Experimental Mathematics* 4(3), 211–225 (1995)
- [Jou09] Joux, A.: *Algorithmic Cryptanalysis*. CRC Press, Boca Raton (2009)
- [JSW08] Jacobson Jr., M.J., Scheidler, R., Weimer, D.: An Adaptation of the NICE Cryptosystem to Real Quadratic Orders. In: Vaudenay, S. (ed.) *AFRICACRYPT 2008*. LNCS, vol. 5023, pp. 191–208. Springer, Heidelberg (2008)
- [LLL82] Lenstra, A.K., Lenstra Jr., H.W., Lovász, L.: Factoring Polynomials with Rational Coefficients. *Math. Ann.* 261, 515–534 (1982)
- [May07] May, A.: Using LLL-Reduction for Solving RSA and Factorization Problems: A Survey. In: *LLL+25 Conference in honour of the 25th birthday of the LLL algorithm* (2007)

- [McK99] McKee, J.: Speeding Fermat's factoring method. *Math. Comput.* 68(228), 1729–1737 (1999)
- [Mil07] Milan, J.: Factoring Small Integers: An Experimental Comparison. INRIA report (2007), <http://hal.inria.fr/inria-00188645/en/>
- [Oka86] Okamoto, T.: Fast public-key cryptosystem using congruent polynomial equations. *Electronic Letters* 22(11), 581–582 (1986)
- [Oka90] Okamoto, T.: A fast signature scheme based on congruential polynomial operations. *IEEE Transactions on Information Theory* 36(1), 47–53 (1990)
- [OU98] Okamoto, T., Uchiyama, S.: A New Public-Key Cryptosystem as Secure as Factoring. In: Nyberg, K. (ed.) *EUROCRYPT 1998*. LNCS, vol. 1403, pp. 308–318. Springer, Heidelberg (1998)
- [Per01] Peralta, R.: Elliptic curve factorization using a partially oblivious function. In: *Cryptography and computational number theory*, *Progr. Comput. Sci. Appl. Logic.*, vol. 20, pp. 123–128 (2001)
- [PO96] Peralta, R., Okamoto, E.: Faster Factoring of Integers of a Special Form. *IEICE Trans. Fundamentals* E79-A, 4, 489–493 (1996)
- [PT99] Paulus, S., Takagi, T.: A generalization of the Diffie-Hellman problem and related cryptosystems allowing fast decryption. In: *Proc. of ICISC 1998*, pp. 211–220 (1999)
- [PT00] Paulus, S., Takagi, T.: A New Public-Key Cryptosystem over a Quadratic Order with Quadratic Decryption Time. *J. Cryptology* 13(2), 263–272 (2000)
- [Sch82] Schoof, R.: Quadratic fields and factorization. *Computational Methods in Number Theory*, MC-Tracts 154/155, 235–286 (1982)
- [Ste] Stehlé, D.: fp11l-3.0, <http://perso.ens-lyon.fr/damien.stehle/#software>
- [Str76] Strassen, V.: Einige Resultate über Berechnungskomplexität. *Jber. Deutsch. Math.-Verein.*, 78, 1–8 (1976/1977)
- [Tak98] Takagi, T.: Fast RSA-type cryptosystem modulo p^kq . In: Krawczyk, H. (ed.) *CRYPTO 1998*. LNCS, vol. 1462, pp. 318–326. Springer, Heidelberg (1998)
- [Wei04] Weimer, D.: An Adaptation of the NICE Cryptosystem to Real Quadratic Orders. Master's thesis, Technische Universität Darmstadt (2004)

Attacking Power Generators Using Unravelled Linearization: When Do We Output Too Much?*

Mathias Herrmann and Alexander May

Horst Görtz Institute for IT-Security
Faculty of Mathematics
Ruhr University Bochum, Germany
mathias.herrmann@rub.de, alex.may@rub.de

Abstract. We look at iterated power generators $s_i = s_{i-1}^e \bmod N$ for a random seed $s_0 \in \mathbb{Z}_N$ that in each iteration output a certain amount of bits. We show that heuristically an output of $(1 - \frac{1}{e}) \log N$ most significant bits per iteration allows for efficient recovery of the whole sequence. This means in particular that the Blum-Blum-Shub generator should be used with an output of less than half of the bits per iteration and the RSA generator with $e = 3$ with less than a $\frac{1}{3}$ -fraction of the bits.

Our method is lattice-based and introduces a new technique, which combines the benefits of two techniques, namely the method of linearization and the method of Coppersmith for finding small roots of polynomial equations. We call this new technique *unravelled linearization*.

Keywords: power generator, lattices, small roots, systems of equations.

1 Introduction

Pseudorandom number generators (PRGs) play a crucial role in cryptography. An especially simple construction is provided by iterating the RSA function $s_i = s_{i-1}^e \bmod N$ for an RSA modulus $N = pq$ of bit-size n and a seed $s_0 \in \mathbb{Z}_N$. This so-called power generator outputs in each iteration a certain amount of bits of s_i , usually the least significant bits. In order to minimize the amount of computation per iteration, one typically uses small e such as $e = 3$. With slight modifications one can choose $e = 2$ as well when replacing the iteration function by the so-called absolute Rabin function [3,4], where $s^2 \bmod N$ is defined to be $\min\{s^2 \bmod N, N - s^2 \bmod N\}$, N is a Blum integer and s_0 is chosen from $\{0, \dots, \frac{N-1}{2}\}$ with Jacobi symbol $+1$.

It is well-known that under the RSA assumption one can safely output up to $\Theta(\log n) = \Theta(\log \log N)$ bits per iteration [18]. At Asiacrypt 2006, Steinfeld, Pieprzyk and Wang [14] showed that under a stronger assumption regarding the optimality of some well-studied lattice attacks, one can securely output

* This research was supported by the German Research Foundation (DFG) as part of the project MA 2536/3-1 and by the European Commission through the ICT programme under contract ICT-2007-216676 ECRYPT II.

$(\frac{1}{2} - \frac{1}{e} - \epsilon - o(1))n$ bits. The assumption is based on a specific RSA one-wayness problem, where one is given an RSA ciphertext $c = m^e \bmod N$ together with a certain fraction of the plaintext bits of m , and one has to recover the whole plaintext m . We call this generator the SPW generator. The SPW generator has the desirable property that one can output a constant fraction $\Omega(\log N)$ of all bits per iteration. Using an even stronger assumption, Steinfeld, Pieprzyk and Wang could improve the output size to $(\frac{1}{2} - \frac{1}{2e} - \epsilon - o(1))n$ bits.

A natural question is whether the amount of output bits of the SPW generator is maximal. Steinfeld et al.’s security proof uses in a black-box fashion the security proof of Fischlin and Schnorr for RSA bits [8]. This proof unfortunately introduces a factor of $\frac{1}{2}$ for the output rate of the generator. So, Steinfeld et al. conjecture that one might improve the rate to $(1 - \frac{1}{e} - \epsilon)n$ using a different proof technique. Here, ϵ is a security parameter and has to be chosen such that performing $2^{\epsilon n}$ operations is infeasible. We show that this bound is essentially the best that one can hope for by giving an attack up to the bound $(1 - \frac{1}{e})n$.

In previous cryptanalytic approaches, upper bounds for the number of output bits have been studied by Blackburn, Gomez-Perez, Gutierrez and Shparlinski [2]. For $e = 2$ and a class of PRGs similar to power generators (but with prime moduli), they showed that provably $\frac{2}{3}n$ bits are sufficient to recover the secret seed s_0 . As mentioned in Steinfeld et al., this bound can be generalized to $(1 - \frac{1}{e+1})n$ using the heuristic extension of Coppersmith’s method [7] to multivariate equations.

Our contribution: We improve the cryptanalytic bound to $(1 - \frac{1}{e})n$ bits using a new heuristic lattice-based technique. Notice that the two most interesting cases are $e = 2, 3$, the Blum-Blum-Shub generator and the RSA generator. For these cases, we improve on the best known attack bounds from $\frac{2}{3}n$ to $\frac{1}{2}n$ and from $\frac{3}{4}n$ to $\frac{2}{3}n$, respectively. Unfortunately — similar to the result of Blackburn et al. [2] — our results are restricted to power generators that output most significant bits in each iteration. It remains an open problem to show that the bounds hold for least significant bits as well.

Our improvement comes from a new technique called *unravelled linearization*, which is a hybrid of lattice-based linearization (see [13] for an overview) and the lattice-based technique due to Coppersmith [7]. Let us illustrate this new technique with a simple example. Assume we want to solve a polynomial equation $x^2 + ax + b = y \bmod N$ for some given $a, b \in \mathbb{Z}_N$ and some unknowns x, y . This problem can be considered as finding the modular roots of a univariate polynomial $f(x) = x^2 + ax + b$ with some error y .

It is a well-known heuristic that a linear modular equation can be easily solved by computing a shortest lattice vector, provided that the absolute value of the product of the unknowns is smaller than the modulus [13]. In order to linearize our equation, we substitute $u := x^2$ and end up with a linear equation in u, x, y . This can be solved whenever $|uxy| < N$. If we assume for simplicity that the unknowns x, y are of the same size, this yields the condition $|x| < N^{\frac{1}{4}}$.

However, in the above case it is easy to see that this linearization is not optimal. A better linearization would define $u := x^2 - y$, leaving us with a linear

equation in u, x only. This yields the superior condition $|x| < N^{\frac{1}{3}}$. So one benefits from the fact that one can easily glue variables together, in our case x^2 and y , whenever this does not change the size of the larger variable. In our example this would also work when y had a known coefficient c of size $|c| \approx |y|$.

The main benefit from the attack of Blackburn et al. [2] comes from a clever linearization of the variables that occur in the case of power generators. While on the one hand such a linearization of a polynomial equation offers some advantages, on the other hand we lose the algebraic structure. Performing e.g. the substitution $u := x^2$, one obtains a linear equation in u, x, y but the property that u and x are algebraically dependent — one being the square of the other — is completely lost. Naturally, this drawback becomes more dramatic when looking at higher degree polynomials.

As a consequence, Coppersmith [6,5,7] designed in 1996 a lattice-based method that is well-suited for exploiting polynomial structures. The underlying idea is to additionally use algebraic relations before linearization. Let us illustrate this idea with our example polynomial $f(x, y) = x^2 + ax + b - y$. We know that whenever f has a small root modulo N , then also $xf = x^3 + ax^2 + bx - xy$ shares this root. Using xf as well, we obtain *two* modular equations in five unknowns x^3, x^2, x, y, xy . Notice that the unknowns x^2 and x are re-used in the second equation which reflects the algebraic structure. So even after linearizing both equations, Coppersmith’s method preserves some polynomial structure. In addition to multiplication of f by powers of x and y — which is often called shifting in the literature — one also allows for powers f^i with the additional benefit of obtaining equations modulo larger moduli N^i .

When we compute the enabling condition with Coppersmith’s method for our example $f(x, y)$ using an optimal shifting and powering, we obtain a bound of $|x| < N^{\frac{1}{3}}$. So the method yields a better bound than naive linearization, but cannot beat the bound of the more clever linearization with $u := x^2 - y$. Even worse, Coppersmith’s method results in the use of lattices of much larger dimension.

To summarize, linearization makes use of the similarity of coefficients in a polynomial equation, whereas Coppersmith’s method basically makes use of the structure of the polynomial’s monomial set.

Motivation for unravalled linearization: Our new technique of *unravalled linearization* aims to bring together the best of both worlds. Namely, we allow for clever linearization but still exploit the polynomial structure. *Unravalled linearization* proceeds in three steps: linearization, basis construction, and unravelling. Let us illustrate these steps with our example $f(x, y)$, where we use the linearization $u := x^2 - y$ in the first step. In this case, we end up with a linear polynomial $g(u, x)$. Similar to Coppersmith’s approach, in the second step we use shifts and powers of this polynomial. E.g., g^2 defines an equation in the unknowns u^2, ux, x^2, u, x modulo N^2 . But since we start with a *linear* polynomial g , this alone will not bring us any benefits, because the algebraic structure got lost in the linearization process from f to g .

Therefore, in the third step we partially unravel the linearization for g^2 using the relation $x^2 = y + u$. The unravelled form of g^2 defines a modular equation in the unknowns u^2, ux, y, u, x , where we basically substitute the unknown x^2 by the unknown y . Notice here, that we can reuse the variable u which occurs in g^2 anyway. This substitution leads to a significant gain, since y is much smaller in size than x^2 .

In the present paper, we elaborate on this simple observation that *unravelling of linearization* brings benefits to lattice reduction algorithms. We use the equations that result from the power generator as a case study for demonstrating the power of *unravelled linearization*, but we are confident that our new technique will also find new applications in various other contexts.

The paper is organized as follows. In Section 2 we will fix some very basic notions for lattices. In Section 3 we define our polynomials from the power generator with $e = 2$ and give a toy example with only two PRG iterations that illustrates how *unravelled linearization* works. This already leads to an improved bound of $\frac{7}{11}n$. In Section 4 we generalize to arbitrary lattice dimension (bound $\frac{3}{5}n$) and in Section 5 we generalize to an arbitrary number of PRG iterations (bound $\frac{1}{2}n$). In Section 6 we finally generalize to an arbitrary exponent e . Since our attacks rely on Coppsmith-type heuristics, we verify the heuristics experimentally in Section 7.

2 Basics on Lattices

Let $\mathbf{b}_1, \dots, \mathbf{b}_d \in \mathbb{Q}^d$ be linearly independent. Then the set

$$L := \left\{ \mathbf{x} \in \mathbb{Q}^d \mid \mathbf{x} = \sum_{i=1}^d a_i \mathbf{b}_i, a_i \in \mathbb{Z} \right\}$$

is called a lattice L with basis matrix $B \in \mathbb{Q}^{d \times d}$, having the vectors $\mathbf{b}_1, \dots, \mathbf{b}_d$ as row vectors. The parameter d is called the lattice dimension, denoted by $\dim(L)$. The determinant of the lattice is defined as $\det(L) := |\det(B)|$.

The famous LLL algorithm [10] computes a basis consisting of short and pairwise almost orthogonal vectors. Let $\mathbf{v}_1, \dots, \mathbf{v}_d$ be an LLL-reduced lattice basis with Gram-Schmidt orthogonalized vectors $\mathbf{v}_1^*, \dots, \mathbf{v}_d^*$. Intuitively, the property of pairwise almost orthogonal vectors $\mathbf{v}_1, \dots, \mathbf{v}_d$ implies that the norm of the Gram-Schmidt vectors $\mathbf{v}_1^*, \dots, \mathbf{v}_d^*$ cannot be too small. This is quantified in the following theorem of Jutla [9] that follows from the LLL paper [10].

Theorem 1 (LLL). *Let L be a lattice spanned by $B \in \mathbb{Q}^{d \times d}$. On input B , the L^3 -algorithm outputs an LLL-reduced lattice basis $\{\mathbf{v}_1, \dots, \mathbf{v}_d\}$ with*

$$\|\mathbf{v}_i^*\| \geq 2^{\frac{1-i}{4}} \left(\frac{\det(L)}{b_{max}^{d-i}} \right)^{\frac{1}{4}} \quad \text{for } i = 1, \dots, d$$

in time polynomial in d and in the bit-size of the largest entry b_{max} of the basis matrix B .

3 Power Generators with $e = 2$ and Two Iterations

Let us consider power generators defined by the recurrence sequence

$$s_i = s_{i-1}^e \pmod N,$$

where N is an RSA modulus and $s_0 \in \mathbb{Z}_N$ is the secret seed.

Suppose that the power generator outputs in each iteration the most significant bits k_i of s_i , i.e. $s_i = k_i + x_i$, where the k_i are known for $i \geq 1$ and the x_i are unknown.

Our goal is to recover all x_i for a number of output bits k_i that is as small as possible. In other word, if we define $x_i < N^\delta$ then we have to find an attack that maximizes δ .

Let us start with the most simple case of two iterations and $e = 2$. The best known bound is $\delta = \frac{1}{3}$ due to Blackburn et al. [2]. We will later generalize to an arbitrary number of iterations and also to an arbitrary e .

For the case of two iterations, we obtain

$$s_1 = k_1 + x_1 \quad \text{and} \quad s_2 = k_2 + x_2,$$

for some unknown s_i, x_i . The recurrence relation of the generator $s_2 = s_1^2 \pmod N$ yields $k_2 + x_2 = (k_1 + x_1)^2 \pmod N$, which results in the polynomial equation

$$x_1^2 - x_2 + \underbrace{2k_1}_{a} x_1 + \underbrace{k_1^2 - k_2}_{b} = 0 \pmod N.$$

Thus, we search for small modular roots of $f(x_1, x_2) = x_1^2 - x_2 + ax_1 + b$ modulo N .

Let us first illustrate our new technique called *unravelled linearization* with a small-dimensional lattice attack before we apply it in full generality in Section 4.

Step 1: Linearize $f(x_1, x_2)$ into g .

We make the substitution $u := x_1^2 - x_2$. This leaves us with a linear polynomial $g(u, x_1) = u + ax_1 + b$.

Step 2: Basis construction.

Defining standard shifts and powers for g is especially simple, since g is a linear polynomial. If we fix a total degree bound of $m = 2$, then we choose g, xg and g^2 .

Let $X := N^\delta$ be an upper bound for x_1, x_2 . Then $U := N^{2\delta}$ is an upper bound for u . The choice of the shift polynomials results in a lattice L spanned by the rows of the lattice basis B depicted in Figure 1.

Let (u_0, x_0) be a root of g . Then the vector $\mathbf{v} = (1, x_0, x_0^2, u_0, u_0x_0, u_0^2, k_1, k_2, k_3)B$ has its right-hand three last coordinates equal to 0 for suitably chosen $k_i \in \mathbb{Z}$. Hence we can write \mathbf{v} as $\mathbf{v} = (1, \frac{x_0}{X}, \dots, \frac{u_0^2}{U^2}, 0, 0, 0)$. Since $|u_0| \leq U$ and $|x_0| \leq X$, we obtain $\|\mathbf{v}\| \leq \sqrt{6}$.

To summarize, we are looking for a short vector \mathbf{v} in the 6-dimensional sublattice $L' = L \cap (\mathbb{Q}^6 \times 0^3)$ with $\|\mathbf{v}\| \leq \sqrt{\dim(L')}$. Let $\mathbf{b}_1, \dots, \mathbf{b}_6$ be an LLL-reduced basis of L' with orthogonalized basis $\mathbf{b}_1^*, \dots, \mathbf{b}_6^*$. Coppersmith [7] showed that

$$\begin{pmatrix} & & & & & & g & x_1g & g^2 \\ & & & & & & b & b & b^2 \\ & & & & & & a & b & ab \\ & & & & & & a & a & a^2 \\ & & & & & & 1 & 1 & b \\ & & & & & & & 1 & a \\ & & & & & & & & 1 \\ & & & & & & N & & \\ & & & & & & & N & \\ & & & & & & & & N^2 \end{pmatrix}$$

Fig. 1. After linearization and standard shifts and powers for $m = 2$

any vector $\mathbf{v} \in L'$ that is smaller than \mathbf{b}_6^* must lie in the sub-space spanned by $\mathbf{b}_1, \dots, \mathbf{b}_5$, i.e. \mathbf{v} is orthogonal to \mathbf{b}_6^* . This immediately yields a coefficient vector of a polynomial $h(u, x_1)$, which has the same roots as $g(u, x_1)$, but over the integers instead of modulo N . Assume that we can find two such polynomials h_1, h_2 , then we can compute all small roots by resultant computation provided that h_1, h_2 do not share a common divisor. The only heuristic of our method is that the polynomials h_1, h_2 are indeed coprime.

By the LLL-Theorem (Theorem [1](#)), an orthogonalized LLL-basis contains a vector \mathbf{b}_d^* in L' with $\|\mathbf{b}_d^*\| \geq c(d) \det(L')^{\frac{1}{d}}$, where $c(d) = 2^{\frac{1-d}{4}}$. Thus, if the condition

$$c(d) \det(L')^{\frac{1}{d}} \geq \sqrt{d}$$

holds, then $\bar{\mathbf{v}} = (1, \frac{x_0}{X}, \dots, \frac{u_0^2}{U^2})$ will be orthogonal to the vector \mathbf{b}_d^* .

Since $\det(L')$ is a function of N , we can neglect $d = \dim(L')$ for large enough N . This in turn simplifies our condition to

$$\det(L') \geq 1.$$

Moreover, one can show by a unimodular transformation of B that $\det(L') = \det(L)$.

For our example, the enabling condition $\det(L) \geq 1$ translates to $U^4 X^4 \leq N^4$. Plugging in the values of $X := N^\delta$ and $U := N^{2\delta}$, this leads to the condition $\delta \leq \frac{1}{3}$. Notice that this is exactly the condition from Blackburn et al. [\[2\]](#). Namely, if the PRG outputs $\frac{2}{3}n$ bits per iteration, then the remaining $\frac{1}{3}n$ bits can be found in polynomial time.

We will now improve on this result by unravelling the linearization of g .

Step 3: Unravel g 's linearization.

We unravel the linearization by back-substitution of $x_1^2 = u + x_2$. This slightly changes our lattice basis (see Fig. [2](#)).

The main difference is that the determinant of the new lattice L_u increases by a factor of X . Thus our enabling condition $\det(L_u) \geq 1$ yields $U^4 X^3 \leq N^4$ or equivalently $\delta \leq \frac{4}{11}$. This means that if the PRG outputs $\frac{7}{11}n$ of the bits in

$$\begin{pmatrix} & & & & & & g & x_1g & g^2 \\ & & & & & & b & & b^2 \\ & & & & & & a & b & ab \\ & & & & & & & a & a^2 \\ & & & & & & 1 & a & a^2 + b \\ & & & & & & & 1 & a \\ & & & & & & & & 1 \\ & & & & & & N & & \\ & & & & & & & N & \\ & & & & & & & & N^2 \end{pmatrix}$$

Fig. 2. After unravelling the linearization

each of two iterations, then we can reconstruct the remaining $\frac{4}{11}n$ bits of both iterations in polynomial time. This beats the previous bound of $\frac{1}{3}n$.

We would like to stress again that our approach is heuristic. We construct two polynomials h_1, h_2 . The polynomials h_1, h_2 contain a priori three variables x_1, x_2, u , but substituting u by $x_1^2 - x_2$ results in two bivariate polynomials h'_1, h'_2 . Then, we hope that h'_1 and h'_2 are coprime and thus allow for efficient root finding. We verified this heuristic with experiments in Section 7.

4 Generalization to Lattices of Arbitrary Dimension

The linearization step from $f(x_1, x_2)$ to $g(u, x_1)$ is done as in the previous section using $u := x_1^2 - x_2$. For the basis construction step, we fix an integer m and define the following collection of polynomials

$$g_{i,j}(u, x_1) := x_1^j g^i(u, x_1) \quad \text{for } i = 1, \dots, m \text{ and } j = 0, \dots, m - i. \quad (1)$$

In the unravelling step, we substitute each occurrence of x_1^2 by $u + x_2$ and change the lattice basis accordingly. It remains to compute the determinant of the resulting lattice. This appears to be a non-trivial task due to the various back-substitutions. Therefore, we did not compute the lattice determinant as a function of m by hand. Instead, we developed an automated process that might be useful in other contexts as well.

We observe that the determinant can be calculated by knowing first the product of all monomials that appear in the collection of the $g_{i,j}$ after unravelling, and second the product of all N . Let us start with the product of the N , since it is easy to compute from Equation (1):

$$\prod_{i=1}^m \prod_{j=0}^{m-i} N^i = \prod_{i=1}^m N^{(m+1)i-i^2} = N^{\frac{1}{6}m^3 + o(m^3)}.$$

¹ The polynomial h_2 can be constructed from \mathbf{b}_{d-1}^* with a slightly more restrictive condition on $\det(L)$ coming from Theorem 4. However, in practical experiments the simpler condition $\det(L) \geq 1$ seems to suffice for h_2 as well. In the subsequent chapters, this minor detail is captured by the asymptotic analysis.

Now let us bound the product of all monomials. Each variable x_1, x_2, u appears in the unravelled form of $g_{i,j}$ with power at most $2m$. Therefore, the product of all monomials that appear in all $\frac{1}{2}m^2 + o(m^2)$ polynomials has in each variable degree at most m^3 . Thus, we can express the exponent of each variable as a polynomial function in m of degree 3 with rational coefficients — similar to the exponent of N .

But since we know that the exponents are polynomials in m of degree at most 3, we can uniquely determine them by a polynomial interpolation at 4 points. Namely, we explicitly compute the unravelled basis for $m = 1, \dots, 4$ and count the number of variables that occur in the unravelled forms of the $g_{i,j}$. From these values, we interpolate the polynomial function for arbitrary m .

This technique is much less error-prone than computing the determinant functions by hand and it allows for analyzing very complicated lattice basis structures. Applying this interpolation process to our unravelled lattice basis, we obtain $\det(L) = X^{-p_1(m)}U^{-p_2(m)}N^{p_3(m)}$ with

$$p_1(m) = \frac{1}{12}m^3 + o(m^3), \quad p_2(m) = \frac{1}{6}m^3 + o(m^3), \quad p_3(m) = \frac{1}{6}m^3 + o(m^3).$$

Our condition $\det(L) \geq 1$ thus translates into $\frac{5}{12}\delta \leq \frac{1}{6}$ resp. $\delta \leq \frac{2}{5}$. Interestingly, this is exactly the bound that Blackburn et al. [2] conjectured to be the best possible bound one can obtain by looking at two iterations of the PRG.

In the next section, we will also generalize our result to an arbitrary fixed number of iterations of the PRG. This should intuitively help to further improve the bounds and this intuition turns out to be true. To the best of our knowledge, our attack is the first one that is capable of exploiting more than two equations in the contexts of PRGs.

5 Using an Arbitrary Fixed Number of PRG Iterations

We illustrate the basic idea of generalizing to more iterations by using three iterations of the generator before analyzing the general case.

Let $s_i = k_i + x_i$ for $i = 1, 2, 3$, where the k_i are the output bits and the x_i are unknown. For these values, we are able to use two iterations of the recurrence relation, namely

$$s_2 = s_1^2 \bmod N \quad s_3 = s_2^2 \bmod N$$

from which we derive two polynomials

$$f_1 : \underbrace{x_1^2 - x_2}_{u_1} + \underbrace{2k_1}_{a_1} x_1 + \underbrace{k_1^2 - k_2}_{b_1} = 0 \bmod N$$

$$f_2 : \underbrace{x_2^2 - x_3}_{u_2} + \underbrace{2k_1}_{a_2} x_2 + \underbrace{k_2^2 - k_3}_{b_2} = 0 \bmod N.$$

We perform the linearization step $f_1 \rightarrow g_1$ and $f_2 \rightarrow g_2$ by using the substitutions $u_1 := x_1^2 - x_2$ and $u_2 := x_2^2 - x_3$.

The intuition behind the definition of this collection of polynomials follows the same reasoning as in the example for $m = 2$. We wish to keep small the number of new monomials introduced by the shifts with g_2 . Notice that the monomials x_2^i for $i = 0, \dots, \lfloor \frac{m}{2} \rfloor$ already appeared in the g_1 shifts — since we back-substituted $x_1^2 \rightarrow u_1 + x_2$. Therefore, it is advantageous to use the g_2 shifts only up to $\lfloor \frac{m}{2} \rfloor$.

With the interpolation technique introduced in Section 4, we derive a bound of $\delta \leq \frac{6}{13}$ for the case of 2 polynomials, i.e. three output values of the generator.

5.1 Arbitrary Number of PRG Iterations

Given $n + 1$ iterations of the PRG, we select a collection of shift polynomials following the intuition given in the previous section:

$$g_{i_1, \dots, i_n, k} := x_1^k g_1^{i_1} \dots g_n^{i_n}$$

$$\text{for } \begin{cases} i_1 = 0, \dots, m \\ i_2 = 0, \dots, \lfloor \frac{m-i_1}{2} \rfloor \\ \vdots \\ i_n = 0, \dots, \lfloor \frac{m - \sum_{j=1}^{n-1} 2^{j-1} i_j}{2^{n-1}} \rfloor \\ k = 0, \dots, m - \sum_{j=1}^n 2^{j-1} i_j \end{cases} \quad \text{with } i_1 + \dots + i_n \geq 1.$$

To perform the asymptotic analysis we need to determine the value of the determinant of the corresponding lattice basis. This means, we have to count the exponents of all occurring monomials in the set of shift polynomials. We would like to point out that because of the range of the index k , the shifts with x_1^k do not introduce additional monomials over the set defined by the product of the g_i alone. For this product the monomials can be enumerated as follows (see Appendix A for a proof):

$$x_1^{a_1} \dots x_n^{a_n} u_1^{i_1 - a_1} \dots u_{n-1}^{i_{n-1} - a_{n-1}} u_n^{i_n - 2b_n - a_n} x_{n+1}^{b_n}$$

$$\text{with } \begin{cases} i_1 = 0, \dots, m & a_1 = 0, 1 \\ i_2 = 0, \dots, \lfloor \frac{m-i_1}{2} \rfloor & a_2 = 0, 1 \\ \vdots & \vdots \\ i_n = 0, \dots, \lfloor \frac{m - \sum_{j=1}^{n-1} 2^{j-1} i_j}{2^{n-1}} \rfloor & a_n = 0, 1 \\ b_n = 0, \dots, \lfloor \frac{i_n - a_n}{2} \rfloor. \end{cases}$$

We are only interested in the asymptotic behavior, i.e. we just consider the highest power of m . We omit the floor function as it only influences a lower order term. Analogously, we simplify the exponents of u_j by omitting the value a_j , since it is a constant polynomial in m . Furthermore, for the same reason the contribution to the determinant of all x_i with $i \leq n$ can be neglected.

To derive the final condition, we have to compute the polynomials $p_j(m)$ of the following expression for the determinant (resp. the coefficients of the highest power of m):

$$\det(L) = X_{n+1}^{-p_x(m)} U_1^{-p_1(m)} \dots U_n^{-p_n(m)} N^{p_N(m)}.$$

It seems to be a complicated task to compute these polynomials explicitly. Therefore, we follow a different approach and compute the sizes of their leading coefficients in relation to each other. This turns out to be enough to derive a bound on the sizes of the unknowns. In Appendix B we explain how to derive the following expressions for the polynomials:

$$p_j(m) = \frac{1}{2^{j-1}} p_1(m) \text{ for } j \leq n, \quad p_x(m) = \frac{1}{2^n} p_1(m), \quad p_N(m) = \frac{2^n - 1}{2^{n-1}} p_1(m),$$

where we again omit low order terms. We use these expressions in the enabling condition $\det(L) \geq 1$ and plug in upper bounds $X_{n+1} \leq N^\delta$ and $U_i \leq N^{2^\delta}$. It is sufficient to consider the condition for the exponents:

$$\delta \frac{1}{2^n} p_1(m) + 2\delta \sum_{j=1}^n \frac{1}{2^{j-1}} p_1(m) \leq \frac{2^n - 1}{2^{n-1}} p_1(m).$$

Simplifying this condition and solving for δ , we obtain

$$\delta \leq \frac{2^{n+1} - 2}{2^{n+2} - 3},$$

which converges for $n \rightarrow \infty$ to $\delta \leq \frac{1}{2}$.

6 Extending to Higher Powers

In the previous sections, we have considered PRGs with exponent $e = 2$ only, i.e. a squaring operation in the recurrence relation. A generalization to arbitrary exponents is straight forward.

Suppose the PRG has the recurrence relation $s_2 = s_1^e \text{ mod } N$. Let, as in Section B, the output of the generator be k_1, k_2 , i.e. we have $s_1 = k_1 + x_1$ and $s_2 = k_2 + x_2$, for some unknown s_i, x_i .

Using the recurrence relation, this yields the polynomial equation

$$\underbrace{x_1^e - x_2 + e k_1 x_1^{e-1} + \dots + e k_1^{e-1} x_1 + k_1^e - k_2}_u = 0 \text{ mod } N.$$

The linearization step is analog to the case where $e = 2$, however, the unravelling of the linearization only applies for higher powers of x_1 , in this case x_1^e .

The collection of shift polynomials using n PRG iterations is

$$g_{i_1, \dots, i_n, k} := x_1^k g_1^{i_1} \dots g_n^{i_n}$$

$$\text{for } \begin{cases} i_1 = 0, \dots, m \\ i_2 = 0, \dots, \lfloor \frac{m-i_1}{e} \rfloor \\ \vdots \\ i_n = 0, \dots, \lfloor \frac{m - \sum_{j=1}^{n-1} e^{j-1} i_j}{e^{n-1}} \rfloor \\ k = 0, \dots, m - \sum_{j=1}^n e^{j-1} i_j \end{cases} \quad \text{with } i_1 + \dots + i_n \geq 1.$$

Taking a closer look at the analysis in Appendix [A](#) and [B](#) shows that the generalization for arbitrary e is straightforward. Working through the analysis we obtain for arbitrary e an asymptotic bound for an arbitrary number of polynomials of $\delta \leq \frac{1}{e}$.

7 Experiments

Since our technique uses a heuristic concerning the algebraic independence of the obtained polynomials, we have to experimentally verify our results. Therefore, we implemented the unravelled linearization using SAGE 3.4.1. including the L^2 reduction algorithm from Nguyen and Stehlé [\[12\]](#). In [Table 1](#) some experimental results are given for a PRG with $e = 2$ and 256 bit modulus N .

Table 1. Experimental Results for $e = 2$

polys	m	δ	exp. δ	$\dim(L)$	time(s)
1	4	0.377	0.364	15	1
1	6	0.383	0.377	28	5
1	8	0.387	0.379	45	45
2	4	0.405	0.390	22	10
2	6	0.418	0.408	50	1250
3	4	0.407	0.400	23	5

In the first column we denote the number of polynomials. The second column shows the chosen parameter m , which has a direct influence on how close we approach the asymptotic bound. On the other hand, the parameter m increases the lattice dimension and therefore the time required to compute a solution. The theoretically expected δ is given in the third column, whereas the actually verified δ is given in the fourth column. The last column denotes the time required to find the solution on a Core2 Duo 2.2 GHz running Linux 2.6.24.

It is worth mentioning that most of the time to find the solution is not spend on doing the lattice reduction, but for extracting the common root from the set of polynomials using resultant computations. The resultant computations yielded the desired solutions of the power generators.

Acknowledgement. We would like to thank Dan Bernstein for bringing this research topic to our attention during an Ecrypt meeting.

References

1. Ben-Or, M., Chor, B., Shamir, A.: On the cryptographic security of single rsa bits. In: STOC, pp. 421–430. ACM, New York (1983)
2. Blackburn, S.R., Gomez-Perez, D., Gutierrez, J., Shparlinski, I.: Reconstructing noisy polynomial evaluation in residue rings. *J. Algorithms* 61(2), 47–59 (2006)
3. Blum, L., Blum, M., Shub, M.: A simple unpredictable pseudo-random number generator. *SIAM J. Comput.* 15(2), 364–383 (1986)
4. Blum, M., Micali, S.: How to generate cryptographically strong sequences of pseudo-random bits. *SIAM J. Comput.* 13(4), 850–864 (1984)
5. Coppersmith, D.: Finding a small root of a bivariate integer equation; factoring with high bits known. In: Maurer [11], pp. 178–189
6. Coppersmith, D.: Finding a small root of a univariate modular equation. In: Maurer [11], pp. 155–165
7. Coppersmith, D.: Small solutions to polynomial equations, and low exponent rsa vulnerabilities. *J. Cryptology* 10(4), 233–260 (1997)
8. Fischlin, R., Schnorr, C.-P.: Stronger security proofs for rsa and rabin bits. *J. Cryptology* 13(2), 221–244 (2000)
9. Jutla, C.S.: On finding small solutions of modular multivariate polynomial equations. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 158–170. Springer, Heidelberg (1998)
10. Lenstra, A.K., Lenstra, H.W., Lovász, L.: Factoring Polynomials with Rational Coefficients. *Mathematische Annalen* 261(4), 515–534 (1982)
11. Maurer, U.M. (ed.): EUROCRYPT 1996. LNCS, vol. 1070. Springer, Heidelberg (1996)
12. Nguyen, P.Q., Stehlé, D.: Floating-point lll revisited. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 215–233. Springer, Heidelberg (2005)
13. Nguyễn, P.Q., Stern, J.: The Two Faces of Lattices in Cryptology. In: Silverman, J.H. (ed.) CaLC 2001. LNCS, vol. 2146, pp. 146–180. Springer, Heidelberg (2001)
14. Steinfeld, R., Pieprzyk, J., Wang, H.: On the provable security of an efficient rsa-based pseudorandom generator. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 194–209. Springer, Heidelberg (2006)

A Describing the Set of Monomials

Theorem 1 *Suppose we have n polynomials of the form*

$$f_i(x_i, x_{i+1}) = x_i^2 + a_i x_i + b_i - x_{i+1}$$

and define the collection of polynomials

$$f_1^{i_1} \dots f_n^{i_n} \quad \text{for} \quad \begin{cases} i_1 &= 0, \dots, m \\ i_2 &= 0, \dots, \lfloor \frac{m-i_1}{2} \rfloor \\ \vdots & \\ i_n &= 0, \dots, \lfloor \frac{m - \sum_{j=1}^{n-1} 2^{j-1} i_j}{2^{n-1}} \rfloor. \end{cases}$$

After performing the substitutions $x_i^2 \mapsto u_i + x_{i+1}$, the set of all occurring monomials can be described as

$$x_1^{a_1} \dots x_n^{a_n} u_1^{i_1 - a_1} \dots u_{n-1}^{i_{n-1} - a_{n-1}} u_n^{i_n - 2b_n - a_n} x_{n+1}^{b_n}$$

$$\text{with } \begin{cases} i_1 = 0, \dots, m & a_1 = 0, 1 \\ i_2 = 0, \dots, \lfloor \frac{m-i_1}{2} \rfloor & a_2 = 0, 1 \\ \vdots & \vdots \\ i_n = 0, \dots, \lfloor \frac{m - \sum_{j=1}^{n-1} 2^{j-1} i_j}{2^{n-1}} \rfloor & a_n = 0, 1 \\ b_n = 0, \dots, \lfloor \frac{i_n - a_n}{2} \rfloor. \end{cases}$$

Proof. By induction: **Basic step:** $n = 1$

For one polynomial $f_1(x_1, x_2) = x_1^2 + a_1x_1 + b_1 - x_2$ we perform the substitution $x_1^2 \mapsto u_1 + x_2$ to obtain $g_1(u_1, x_1) = u_1 + a_1x_1 + b_1$. The set of all monomials that are introduced by the powers of $g_1(u_1, x_1)$ can be described as

$$x_1^{j_1} u_1^{i_1 - j_1} \quad \text{for } \begin{cases} i_1 = 0, \dots, m \\ j_1 = 0, \dots, i_1. \end{cases}$$

It remains to perform the substitution on this set. Therefore, we express the counter j_1 by two counters a_1 and b_1 and let $j_1 = 2b_1 + a_1$, i.e. we write the set as

$$(x_1^2)^{b_1} x_1^{a_1} u_1^{i_1 - 2b_1 - a_1} \quad \text{for } \begin{cases} i_1 = 0, \dots, m \\ a_1 = 0, 1 \\ b_1 = 0, \dots, \lfloor \frac{i_1 - a_1}{2} \rfloor. \end{cases}$$

Imagine that we enumerate the monomials for fixed i_1, a_1 and increasing b_1 , and simultaneously perform the substitution $x_1^2 \mapsto u_1 + x_2$. The key point to notice is that all monomials that occur after the substitution, i.e. all of $(u_1 + x_2)^{b_1} x_1^{a_1} u_1^{i_1 - 2b_1 - a_1}$, have been enumerated by a previous value of b_1 , except for the single monomial $x_2^{b_1} x_1^{a_1} u_1^{i_1 - 2b_1 - a_1}$.

Thus, the set of monomials after the substitution can be expressed as

$$x_2^{b_1} x_1^{a_1} u_1^{i_1 - 2b_1 - a_1} \quad \text{for } \begin{cases} i_1 = 0, \dots, m \\ a_1 = 0, 1 \\ b_1 = 0, \dots, \lfloor \frac{i_1 - a_1}{2} \rfloor. \end{cases}$$

This concludes the basic step.

Inductive Step: $n - 1 \rightarrow n$

Suppose the assumption is correct for $n - 1$ polynomials. By the construction of the shift polynomials and the induction hypothesis, we have the set of monomials

$$\underbrace{x_1^{a_1} \dots x_{n-1}^{a_{n-1}} u_1^{i_1 - a_1} \dots u_{n-2}^{i_{n-2} - a_{n-2}} u_{n-1}^{i_{n-1} - 2b_{n-1} - a_{n-1}} x_n^{b_{n-1}}}_{\text{Hypothesis}} \underbrace{x_n^{j_n} u_n^{i_n - j_n}}_{f_n}$$

$$\text{for } \begin{cases} i_1 = 0, \dots, m & a_1 = 0, 1 \\ i_2 = 0, \dots, \lfloor \frac{m-i_1}{2} \rfloor & a_2 = 0, 1 \\ \vdots & \vdots \\ i_{n-1} = 0, \dots, \lfloor \frac{m-\sum_{j=1}^{n-2} 2^{j-1} i_j}{2^{n-2}} \rfloor & a_{n-1} = 0, 1 \\ b_{n-1} = 0, \dots, \lfloor \frac{i_{n-1}-a_{n-1}}{2} \rfloor \\ i_n = 0, \dots, \lfloor \frac{m-\sum_{j=1}^{n-1} 2^{j-1} i_j}{2^{n-1}} \rfloor & j_n = 0, \dots, i_n. \end{cases}$$

By adding the n -th polynomial, we also get the new relation $x_n^2 = u_n + x_{n+1}$. Before performing the substitutions, however, we have to take a closer look at the powers of x_n . The problem seems to be that we have a contribution from the n -th polynomial as well as from some previous substitutions. It turns out that this can be handled quite elegantly. Namely, we will show that all occurring monomials are enumerated by just taking $b_{n-1} = 0$.

Consider the set of monomials for $b_{n-1} = c$ for some constant $c \geq 1$:

$$x_1^{a_1} \dots u_{n-1}^{i_{n-1}-2c-a_{n-1}} x_n^{j_n+c} \quad \text{for } j_n \in \{0, \dots, i_n\}.$$

Exactly the same set of monomials is obtained by considering the index $i'_{n-1} = i_{n-1} - 2$ and $b_{n-1} = c - 1$. Notice that in this case the counter i'_n , which serves as an upper bound of j'_n , runs from 0 through

$$\left\lfloor \frac{m - \sum_{j=1}^{n-2} 2^{j-1} i_j - 2^{n-2} i'_{n-1}}{2^{n-1}} \right\rfloor = \left\lfloor \frac{m - \sum_{j=1}^{n-2} 2^{j-1} i_j - 2^{n-2} i_{n-1} + 2^{n-1}}{2^{n-1}} \right\rfloor = i_n + 1.$$

Thus, we have the same set of monomials as with $b_{n-1} = c - 1$:

$$x_1^{a_1} \dots u_{n-1}^{i'_{n-1}-2(c-1)-a_{n-1}} x_n^{j'_n+(c-1)} \quad \text{for } j'_n \in \{0, \dots, i'_n\}.$$

Iterating this argument, we conclude that all monomials are enumerated by $b_{n-1} = 0$.

Having combined the occurring powers of x_n , we continue by performing an analog step as in the basic step: introduce a_n and b_n representing j_n . This leads to

$$x_1^{a_1} \dots u_{n-1}^{i_{n-1}-a_{n-1}} (x_n^2)^{b_n} x_n^{a_n} u_n^{i_n-2b_n-a_n}$$

$$\text{for } \begin{cases} i_1 = 0, \dots, m & a_1 = 0, 1 \\ i_2 = 0, \dots, \lfloor \frac{m-i_1}{2} \rfloor & a_2 = 0, 1 \\ \vdots & \vdots \\ i_{n-1} = 0, \dots, \lfloor \frac{m-\sum_{j=1}^{n-2} 2^{j-1} i_j}{2^{n-2}} \rfloor & a_{n-1} = 0, 1 \\ i_n = 0, \dots, \lfloor \frac{m-\sum_{j=1}^{n-1} 2^{j-1} i_j}{2^{n-1}} \rfloor & a_n = 0, 1 \\ b_n = 0, \dots, \lfloor \frac{i_n-a_n}{2} \rfloor. \end{cases}$$

Finally we substitute $x_n^2 = u_n + x_{n+1}$. Using the same argument as in the basic set, we note that new monomials only appear for powers of x_{n+1} .

B Relations among Exponent Polynomials

For the determinant computation we need to sum up the exponents of the occurring monomials. Take for example u_ℓ with $\ell < n$: using the description of the set from Appendix A, we need to compute

$$\sum_{i_1=0}^m \sum_{a_1=0}^1 \sum_{i_2=0}^{\lfloor \frac{m-i_1}{2} \rfloor} \sum_{a_2=0}^1 \dots \sum_{i_n=0}^{\lfloor \frac{m-\sum_{j=1}^{n-1} 2^{j-1} i_j}{2^{n-1}} \rfloor} \sum_{a_n=0}^1 \sum_{b_n=0}^{\lfloor \frac{i_n-a_n}{2} \rfloor} (i_\ell - a_\ell).$$

We will step by step simplify this expression using the fact that in the asymptotic consideration only the highest power of the parameter m is important.

In the first step we notice that we may remove the $-a_\ell$ from the summation, because a_ℓ does not depend on m , while i_ℓ does. Therefore, the a_ℓ just affects lower order terms. With the same argument we can omit the a_n in the upper bound of the sum over b_n . Further, the floor function in the limit of the sums does only affect lower order terms and therefore may be omitted. Next, we can move all the sums of the a_i to the front, since they are no longer referenced anywhere, and replace each of these sums by a factor of 2, making altogether a global factor of 2^n .

For further simplification of the expression, we wish to eliminate the fractions that appear in the bounds of the sums. To give an idea how to achieve this, consider the expression

$$\sum_{i_1=0}^m \sum_{i_2=0}^{\frac{m-i_1}{2}} i_2.$$

Our intuition is to imagine an index i'_2 of the second sum that performs steps with a width of 2 and is upper bounded by $m - i_1$. To keep it equivalent, we have to compute the sum of over all integers of the form $\lfloor \frac{i'_2}{2} \rfloor$. However, when changing the index to i'_2 , the sum surely does not perform steps with width 2. I.e. we count every value exactly twice. Thus, to obtain a correct reformulation, we have to divide the result by 2. Note that asymptotically we may omit the floor function and simply sum over $\frac{i'_2}{2}$.

In the same way we are able to reformulate all sums from i_1 to i_n . For better readability we replaced i'_j with i_j again.

$$2^n \cdot \frac{1}{2} \cdot \frac{1}{4} \cdot \dots \cdot \frac{1}{2^{n-1}} \sum_{i_1=0}^m \sum_{i_2=0}^{m-i_1} \dots \sum_{i_n=0}^{m-\sum_{j=1}^{n-1} i_j} \sum_{b_n=0}^{\frac{i_n}{2^n}} \frac{1}{2^{\ell-1}} i_\ell. \tag{2}$$

It seems to be a complicated task to explicitly evaluate a sum of this form. Therefore, we follow a different approach, namely we relate the sums over different i_ℓ to each other. We start with the discussion of a slightly simpler observation:

Sums of the form $\sum_{i_1=0}^m \sum_{i_2=0}^{m-i_1} \dots \sum_{i_n=0}^{m-\sum_{j=1}^{n-1} i_j} i_\ell$ are equal for all $\ell \leq n$.

An explanation can be given as follows. Imagine the geometric object that is represented by taking the i_j as coordinates in an n -dimensional space. This set describes an n -dimensional simplex, e.g. a triangle for $n = 2$, a tetrahedron for $n = 3$, etc. Considering its regular structure, i.e. the symmetry in the different coordinates, it should be clear that the summation over each of the i_ℓ results in the same value.

In the sum of Equation (2) there is an additional inner summation with index b_n and limit $i_n/2^n$. For the indices $\ell < n$ this innermost sum is constant for all values of ℓ and thus with the previous argumentation the whole sums are equal for all $\ell < n$. We only have to take care of the leading factors, i.e. the powers of 2 that came from replacing the summation variables.

This gives us already a large amount of the exponent polynomials in the determinant expression. Namely, we are able to formulate the polynomials p_ℓ (which is the sum over the i_ℓ) in terms of p_1 for all $\ell < n$. The difference is exactly the factor $\frac{1}{2^{\ell-1}}$ that has been introduced when changing the index from i_ℓ to i'_ℓ .

For the exponent polynomial of the variable u_n , however, we have to be careful because we do not compute the summation of $i_n - a_n$, but of $i_n/2^{n-1} - 2b_n - a_n$ instead ($i_n/2^{n-1}$ since we changed the summation index i_n). The value $-a_n$ can be omitted with the same argument as before. To derive a relation of p_n to p_1 , we start by evaluating the inner sums:

$$\begin{aligned}
 p_1 : \dots & \sum_{i_n=0}^{m-\sum_{j=1}^{n-1} i_j} \sum_{b_n=0}^{\frac{i_n}{2^n}} i_1 = \dots \sum_{i_n=0}^{m-\sum_{j=1}^{n-1} i_j} \frac{i_n}{2^n} i_1 \\
 p_n : \dots & \sum_{i_n=0}^{m-\sum_{j=1}^{n-1} i_j} \sum_{b_n=0}^{\frac{i_n}{2^n}} \left(\frac{i_n}{2^{n-1}} - 2b_n \right) = \dots \sum_{i_n=0}^{m-\sum_{j=1}^{n-1} i_j} \left(\frac{i_n^2}{2^{2n-1}} - 2 \frac{1}{2} \frac{i_n^2}{2^{2n}} \right) \\
 & = \dots \sum_{i_n=0}^{m-\sum_{j=1}^{n-1} i_j} \frac{i_n^2}{2^{2n-1}}.
 \end{aligned}$$

Notice that once again, for the asymptotic analysis we have only considered the highest powers.

Because of the previously mentioned symmetry between i_1 and i_n , we finally derive $p_n = \frac{1}{2^{n-1}} p_1$. The same argument can be used to derive the bound on the variable x_{n+1} for which we have to compute the sum

$$p_x : \dots \sum_{i_n=0}^{m-\sum_{j=1}^{n-1} i_j} \sum_{b_n=0}^{\frac{i_n}{2^n}} b_n = \dots \sum_{i_n=0}^{m-\sum_{j=1}^{n-1} i_j} \frac{i_n^2}{2^{2n}}.$$

The multiplicative relation between p_1 and p_x is therefore $p_x = \frac{1}{2^n} p_1$.

Finally, to compute the exponent of N in the determinant, we have to sum up all exponents that occur in the enumeration of the shift polynomials given in

Section 5.1. The simplifications are equivalent to the ones used before and we obtain:

$$p_N = \sum_{\ell=1}^n \left(\frac{1}{2} \cdot \frac{1}{4} \cdots \frac{1}{2^{n-1}} \sum_{i_1=0}^m \cdots \sum_{i_n=0}^{m-\sum_{j=0}^{n-1} i_j} \sum_{k=0}^{m-\sum_{j=0}^n i_j} \frac{1}{2^{\ell-1}} i_\ell \right).$$

We first note that for $\ell < n$ we may write

$$\cdots \frac{1}{2^{\ell-1}} i_\ell \sum_{i_n=0}^c \sum_{k=0}^{c-i_n} 1 \quad \text{with } c = m - \sum_{j=0}^{n-1} i_j.$$

This is asymptotically equivalent to

$$\cdots \frac{1}{2^{\ell-1}} i_\ell \sum_{i_n=0}^c \sum_{k=0}^{i_n} 1 = 2^n \cdots \frac{1}{2^{\ell-1}} i_\ell \sum_{i_n=0}^c \sum_{k=0}^{\frac{i_n}{2^n}} 1 = \frac{1}{2^{\ell-1}} p_1.$$

For $\ell = n$ we argue again that the summations for different i_ℓ behave the same way. Thus it follows $\frac{1}{2} \cdot \frac{1}{4} \cdots \frac{1}{2^{n-1}} \sum_{i_1=0}^m \cdots \sum_{i_n=0}^{m-\sum_{j=0}^{n-1} i_j} \sum_{k=0}^{m-\sum_{j=0}^n i_j} \frac{i_n}{2^{n-1}} = \frac{1}{2^{n-1}} p_1$. Summing up, we obtain

$$p_N = \left(1 + \frac{1}{2} + \frac{1}{4} + \cdots + \frac{1}{2^{n-1}} \right) p_1 = \frac{2^n - 1}{2^{n-1}} p_1.$$

Security Notions and Generic Constructions for Client Puzzles

Liquan Chen¹, Paul Morrissey², Nigel P. Smart², and Bogdan Warinschi²

¹ Hewlett-Packard Laboratories, Bristol, UK
liquan.chen@hp.com

² Computer Science Department, University of Bristol, UK
{paulm,nigel,bogdan}@cs.bris.ac.uk

Abstract. By a computational puzzle we mean a mildly difficult computational problem that requires resources (processor cycles, memory, or both) to solve. Puzzles have found a variety of uses in security. In this paper we are concerned with *client puzzles*: a type of puzzle used as a defense against Denial of Service (DoS) attacks. The main contribution of this paper is a formal model for the security of client puzzles. We clarify the interface that client puzzles should offer and give two security notions for puzzles. Both functionality and security are inspired by, and tailored to, the use of puzzles as a defense against DoS attacks. Our definitions fill an important gap: breaking either of the two properties immediately leads to successful DoS attacks. We illustrate this point with an attack against a previously proposed puzzle construction. We also provide a generic construction of a client puzzle which meets our security definitions.

1 Introduction

A Denial of Service (DoS) attack on a server aims to render it unable to provide some service by depleting its internal resources. For example, the famous TCP-SYN flooding attack [9] prevents further connections to a server by starting a large number of TCP sessions which are then left uncompleted. The effort of the attacker is rather small, whereas the server quickly runs out of resources (which are allocated to the unfinished sessions).

One countermeasure against connection depletion DoS attacks uses client puzzles [14]. When contacted by some unauthenticated, potentially malicious, client to execute some protocol and before allocating any resources for the execution, the server issues a client puzzle – a moderately hard computational problem. The server only engages in the execution of the protocol (and thus allocates resources) when the client returns a valid solution to the puzzle. The idea is that the server spends its resources only after the client has spent a significant amount of resources itself. To avoid the burden of running the above mechanism when no attackers are present, the defense only kicks in whenever the server resources drop below a certain threshold.

Client puzzles have received a lot of attention in the cryptographic community [2,5,10,11,14,24,27,28] but most of the prior work consists of proposing puzzle constructions and arguing that those constructions do indeed work. Although sometimes technical, such security arguments are with respect to intuitive security notions for puzzles since rigorous formal models for the security of such puzzles are missing. The absence of such models has (at least) two undesirable consequences. On the one hand the investigation of puzzle constructions usually concentrates on some security aspects and omits others which are of equal importance when puzzles are used as part of other protocols. More importantly, the absence of formal models prevents a rigorous, reduction-based analysis of the effectiveness of puzzles against DoS in the style of modern cryptography (where the existence of a successful DoS attacker implies the existence of an attacker against client puzzles).

In this paper we aim to solve the first problem outlined above as a first key step towards solving the second one. The main contribution of this paper is a formal framework for the design and analysis of client puzzles. In addition to fixing their formal syntax, we design security notions inspired by, and therefore tailored for, the use of client puzzles as a defense against DoS attacks. Specifically, we require that an adversary cannot produce valid puzzles on his own (*puzzle-unforgeability*) and that puzzles are non-trivial – the client needs indeed to spend at least a specified amount of resources to solve them – (*puzzle difficulty*). The use of client puzzles that do not fulfill at least one of our notions immediately leads to a successful DoS attack. Our definitions use well-established intuition and techniques for defining one-wayness and authentication properties. Apart from some design decisions regarding the measure for resources and the precise oracles an adversary should have access to, there are no deep surprises here. However, we highlight that the lack of rigorous definitions such as those we put forward in this paper is dangerous. Constructions that are secure at an intuitive level, may be in fact insecure when used. Indeed, we explicitly demonstrate that a popular construction, that does not meet our notion of unforgeability, does not protect and in fact facilitates DOS attacks in systems that use it.

Furthermore, we give a generic construction of a client puzzle that is secure in the sense we define. Many existing client puzzle constructions can be obtained as an instantiation of our generic construction, with only minor modifications if any. Our construction uses a pseudorandom function family to provide puzzle-unforgeability and puzzle-difficulty is obtained from a one-way function given a large part of the preimage. We prove our construction secure via an asymptotic reduction for unforgeability and a concrete reduction for difficulty. Next, we discuss our results in more details.

Our Contribution

FORMAL SYNTAX OF A CLIENT PUZZLE. Our first contribution is a formal syntax for client puzzles. We define a client puzzle as a tuple of algorithms for system setup, puzzle generation, solution finding, puzzle authenticity checking, and solution checking. The definition is designed to capture the main functionality

required from client puzzles from the perspective of their use against DoS attacks.

SECURITY NOTIONS FOR CLIENT PUZZLES. The use of puzzles against DoS attacks also inspired the two (orthogonal) security notions for client puzzles that we design.

To avoid storing puzzles handed out to clients (a resource consuming task), the server gives puzzles away and expects the client to hand back both the puzzle and its solution. Obviously, the server needs to be sure the client cannot produce puzzles on its own, as this would lead to trivial attacks. We remark that this aspect is often overlooked in existing constructions since it is only apparent when puzzles are considered in the precise context for which they are intended. We capture this requirement via the notion of puzzle-unforgeability. Formally, we define a security game where the adversary is given certain querying capabilities (he can for example request to see puzzles and their solutions, can verify the authenticity of puzzles, etc) and aims to output a new puzzle which the server deems as valid.

The second notion, puzzle-difficulty, reflects the idea that the client needs to spend a certain amount of resources to solve a puzzle. In our definition we took adversary resources to mean “clock cycles”, as this design decision allows us to abstract away undesirable details like the distributed nature of many DoS adversaries. We define a security game where the adversary is given various querying capabilities sufficient for mimicking a DoS attack-like environment: he can see puzzles and their solutions, obtain solutions for puzzles he chooses, etc. The challenge for the adversary is to solve a given challenge puzzle spending less than a certain number of clock cycles, with probability better than a certain threshold.

AN ATTACK ON THE JUELS AND BRAINARD PUZZLES. Most of the previous work on puzzles concentrates exclusively on the difficulty aspect and overlooks, or only partially considers, the unforgeability property. One such work is the puzzle construction proposed by Juels and Brainard [14]. We demonstrate the usefulness of our definitions by showing the Juels and Brainard construction is forgeable. We then explain how a system using this kind of puzzle can be attacked by exploiting the weakness we have identified.

GENERIC CONSTRUCTIONS. We provide a generic construction of a client puzzle inspired by the Juels and Brainard sub-puzzle construction [14]. First, we evaluate a pseudorandom function (PRF), keyed by some secret value, on inputs including a random nonce, hardness parameter and a system specific string. This stage ensures uniqueness of the puzzle and the desired unforgeability; only the server that possesses the hidden key is able to perform this operation and hence generate valid puzzles. The remaining information to complete the puzzle is then computed by evaluating a one way function (OWF), for which finding preimages has a given difficulty, on the output of the PRF; the goal in solving the puzzle is to find such a preimage given the inputs to the PRF and the target. The idea is that the client would need to do an exhaustive search on the possible preimage

space to find such a preimage. We certify the intuition by rigorous proofs that the generic construction meets the security definitions that we put forth, for appropriately chosen parameters. Importantly, many *secure variants* of previously proposed constructions can be obtained as instances of our generic construction. For example, the puzzle constructions proposed by Juels and Brainard [14] puzzle and the two-party variant of the Waters et al. puzzles [28] can be seen as variants of our generic construction. Finally, we provide concrete security bounds for the first of these puzzles. We do so in the random oracle model which we use to obtain secure and efficient instantiations of the two primitives used by our generic construction.

Related Work

MERKLE PUZZLES. The use of puzzles in cryptography was pioneered by Merkle [18] who used puzzles to establish a secret key between parties over an insecure channel. Since then the optimality of Merkle puzzles has been analyzed by Impagliazzo and Rudich [12] and Barak and Mahmoody–Ghidary [3]. The possibility of basing weak public key cryptography on one-way functions, or some variant of them was recently explored by Biham, Goren and Ishai [4]. Specifically, a variant of Merkle’s protocol is suggested whose security is based on the one-wayness of the underlying primitive.

CLIENT PUZZLES. Client puzzles were first introduced as a defense mechanism against DoS attacks by Juels and Brainard [14]. The construction they proposed uses hash function inversion as the source of puzzle-difficulty. They also attempt to obtain puzzle-unforgeability but partially fail in two respects. By neglecting the details of how puzzles are to be used against a DoS attack, the construction suffers from a flaw (which we explain how to exploit later in this paper) that can be used to mount a DoS attack. Secondly, despite intuitive claims that security is based on the one-wayness of the hash function used in the construction, security requires much stronger assumptions, namely one-wayness with partial information about the preimage. The authors also present a method to combine a key agreement or authentication protocol with a client puzzle, and present a set of informal desirable properties of puzzles. Building on this work, Aura et al. [2] use the same client puzzle protocol construction but present a new client puzzle mechanism, also based on hash function inversion, and extend the set of desirable properties.

An alternative method for constructing client puzzles and client puzzle protocols was proposed by Waters et al. [28]. This technique assumes the client puzzle protocol is a three party protocol and constructs a client puzzle based on the discrete logarithm problem for which authenticity and correctness can be verified using a Diffie–Hellman based technique. One of the main advantages of this approach is that puzzle generation can be outsourced from the server to another external bastion, yet verification of solutions can be performed by the server itself.

More recently Tritilanunt et al. [27] proposed a client puzzle based on the subset sum problem. Schaller et al. [24] have also used what they refer to as cryptographic puzzles for broadcast authentication in networks.

An interesting line of work analyzes ways to construct stronger puzzles out of weaker ones. The concept of chaining together client puzzles to produce a new and more difficult client puzzle was introduced by Groza and Petrica [11]. Their construction enforces a sequential solving strategy, and thus yields a harder puzzle. A related work is that of Canetti, Halevi, and Steiner [5] who are concerned with relating the difficulty of solving one single puzzle to that of solving several independent puzzles. They consider the case of “weakly” verifiable puzzles (puzzles for which the solution can only be checked by the party that produced the puzzles). That paper does not consider the use of puzzles in the context of DoS attacks, and thus is not concerned with authenticity.

CLIENT PUZZLE PROTOCOLS. In an interesting paper that analyzes resistance of client puzzle protocols to man-in-the-middle attacks [21], Price concludes that in any secure protocol the server needs to resort to digital signatures. We note that such concerns are related but orthogonal to the goals that we pursue in this paper. Indeed, in prior literature there is no clear distinction between client puzzles (the problems that the server hands out for clients to solve) and client puzzle protocols (the ensemble that includes, in addition to the particular puzzles that are constructed, the way state is maintained by the server, the mechanism for deeming a puzzle as expired etc.) We emphasize that in this paper we are mainly concerned with the former so the results of [21] do not apply.

DoS ATTACKS. A classification of remote DoS attacks, countermeasures and a brief consideration of Distributed Denial of Service (DDoS) attacks were given by Karig and Lee [15]. Following this Specht and Lee [26] give a classification of DDoS attacks, tools and countermeasures. In [26] the adversarial model of [15] is extended to include Internet Relay Chat (IRC) based models. The authors of [26] also classify the types of software used for such attacks and the most common known countermeasures. Other classifications of DDoS attacks and countermeasures were later given by [7,19].

A number of protocols have been designed to resist DDoS attacks. The most important examples are the JFK protocol [1] and the HIP protocol [20]. The JFK protocol of Aiello et al. [1] trades the forward secrecy property, known as adaptive forward secrecy, for denial of service resistance. The original protocol does not use client puzzles. In [25] the cost based technique of Meadows [16,17] is used to analyze the JFK protocol. Two denial of service attacks are found and both can be prevented by introducing a client puzzle into the JFK protocol.

SPAM AND TIME-LOCK CRYPTO. Other proposals for the use of puzzles include the work of Dwork and Naor who propose to use a pricing function (a particular type of puzzles) to combat junk email [8]. The basic principle is the pricing function costs a given amount of computation to compute and this computation can be verified cheaply without any additional information. A service provider could then issue a “stamp duty” on bulk mailings. Finally, Rivest et al. introduced the notion of timed-release crypto in [22] and instantiate this notion with a time-lock

puzzle. The overall goal of timed-release crypto is to encrypt a message such that nobody, even the sender, can decrypt it before a given length of time has passed.

Paper Overview. We start with a sample client puzzle from Section 2. Our formal definition of a client puzzle and a client puzzle protocol is in Section 3. In Section 4 we give security notions for client puzzles in terms of unforgeability and difficulty. We demonstrate that the Juels and Brainard client puzzles is insecure in Section 5. Finally, our generic construction of a client puzzle is given in Section 6. We also include a sample instantiation based on hash functions which we analyze in the random oracle model.

2 Juels and Brainard Puzzles

To illustrate some of the basic ideas behind the construction of puzzles, we first give a brief description of the puzzle generation process for the Juels and Brainard construction [14]. In our description we refer to the (authorized) puzzle generation entity (or user) as the *generator* and the (authorized) puzzle solving entity (or user) as the *solver*. We use the term “puzzle” from here onwards for individual puzzle instances. We write $\{0, 1\}^t$ for the set of binary strings of length t and $\{0, 1\}^*$ for the set of binary strings of arbitrary finite length. If $x = x_0, x_1, \dots, x_i, \dots, x_j, \dots, x_n$ is a bit string then we let $x\langle i, j \rangle$ denote the sub string x_i, \dots, x_j .

For this construction the generator (generally some server) holds a long term secret value s chosen uniformly from a space large enough to prevent exhaustive key search attacks. The server also chooses a hardness parameter: a pair $Q = (\alpha, \beta) \in \mathbb{N}^2$ which ensure puzzles have a certain amount of difficulty to solve. We let $H : \{0, 1\}^* \mapsto \{0, 1\}^m$ be some hash function. To generate a new puzzle the generator performs the following steps to compute the required sub-puzzle instances P_j for $j \in \{1, 2, \dots, \beta\}$:

- A bit string σ_j is computed as $\sigma_j = H(s, \text{str}, j)$. The value str has the structure $\text{str} = t\|M$ for t some server time value¹ and M some unique value². We denote $x_j = \sigma_j\langle 1, \alpha \rangle$ and $z_j = \sigma_j\langle \alpha + 1, m \rangle$.
- A value y_j is computed as $y_j = H(\sigma_j)$ and the sub-puzzle instance is $P_j = (z_j, y_j)$.

The full puzzle instance is then the required parameters plus the tuple of sub-puzzle instances $\text{puz} = (Q, \text{str}, P = (P_1, P_2, \dots, P_\beta))$. The sub-puzzle instance generation process is summarized in Figure 1.

A solution to a given sub-puzzle P_j is any string x'_j such that $H(x'_j \parallel z_j) = y_j$. The solution to the full puzzle instance is a tuple of solutions to the sub-puzzles. To verify a potential solution $\text{soln} = (Q, \text{str}, \text{soln}_1, \dots, \text{soln}_\beta)$ the generator verifies

¹ The details of the type of value this is are not described in [14] but here we will assume this is as a bit string.

² In [14] this is specified as the first message flow of a protocol or some other unique data. Again, we will assume this is encoded as a bit string since this is not specified.

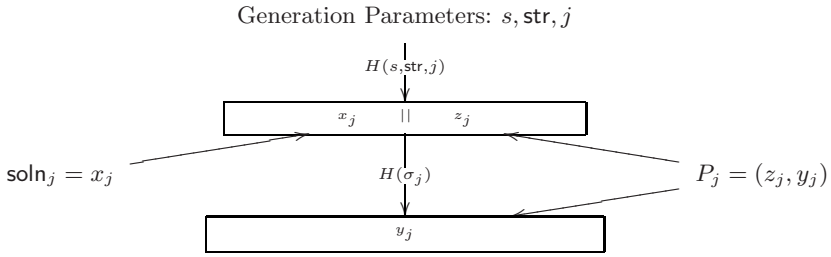


Fig. 1. The Juels and Brainard Sub-Puzzle Instance Generation

each P_j and soln_j by checking that $H(\text{soln}_j || z_j) = y_j$ for each j . The authenticity of a given puzzle is checked by regenerating each P_j using s and comparing this to the puzzle submitted.

To incorporate this client puzzle into a client puzzle protocol the server, on receiving a valid solution, allocates buffer slots, by using a hash table on the values of M , for each puzzle and correct solution submitted. This ensures that only one puzzle instance and solution are accepted for a given value of M .

3 Client Puzzles

The role of a client puzzle in a protocol is to give one party some assurance that the other party has spent at least a given amount of effort computing a solution to a given puzzle instance. In this section we give a formal definition of a client puzzle in the most general sense.

FORMAL SYNTAX OF A CLIENT PUZZLE. A client puzzle is a tuple of algorithms: a setup algorithm for generating long term public and private parameters, an algorithm for generating puzzle instances of a given difficulty, a solution finding algorithm, an algorithm for verifying authenticity of a puzzle instance and an algorithm for verifying correctness of puzzle and solution pairs. We formally define a client puzzle as follows.

Definition 1 (Client Puzzle). A client puzzle $\text{CPuz} = (\text{Setup}, \text{GenPuz}, \text{FindSoln}, \text{VerAuth}, \text{VerSoln})$ is given by the following algorithms:

- **Setup** is a p.p.t. setup algorithm. On input of 1^k , for security parameter k , it performs the following operations:
 - Selects the long term secret key space sSpace , hardness space QSpace , string space strSpace , puzzle instance space puzSpace and solution space solnSpace .
 - Selects the long term puzzle generation key $s \xleftarrow{\$} \text{sSpace}$.
 - Sets Π additional public information, such as some description of algorithms required for the client puzzle.
 - Sets $\text{params} \leftarrow (\text{sSpace}, \text{puzSpace}, \text{solnSpace}, \text{QSpace}, \Pi)$ and outputs (params, s) .

The tuple params is the public system parameters and as such is not explicitly given as an input to other algorithms. The value s is kept private by the puzzle generator.

- GenPuz is a p.p.t. puzzle generation algorithm. On input of $s \in \text{sSpace}$, $Q \in \text{QSpace}$ and $\text{str} \in \text{strSpace}$ it outputs a puzzle instance $\text{puz} \in \text{puzSpace}$.
- FindSoln is a probabilistic solution finding algorithm. On input of $\text{puz} \in \text{puzSpace}$ and a run time $\tau \in \mathbb{N}$ it outputs a potential solution $\text{soln} \in \text{solnSpace}$ after at most τ clock cycles of execution.
- VerAuth is a d.p.t. puzzle authenticity verification algorithm. On input of $s \in \text{sSpace}$ and $\text{puz} \in \text{puzSpace}$ this outputs **true** or **false**.
- VerSoln is a deterministic solution verification algorithm. On input of $\text{puz} \in \text{puzSpace}$ and a potential solution $\text{soln} \in \text{solnSpace}$ this outputs **true** or **false**.

For correctness we require that if $(\text{params}, s) \leftarrow \text{Setup}(1^k)$ and $\text{puz} \leftarrow \text{GenPuz}(s, Q, \text{str})$, for $Q \in \text{QSpace}$ and $\text{str} \in \text{strSpace}$, then

- $\text{VerAuth}(s, \text{puz}) = \mathbf{true}$ and
- $\exists \tau \in \mathbb{N}$ such that $\text{soln} \leftarrow \text{FindSoln}(\text{puz}, \tau)$ and $\text{VerSoln}(\text{puz}, \text{soln}) = \mathbf{true}$.

REMARK 1. Typically client puzzles use a set of system parameters, most notably system time, as input to the puzzle generation algorithm. This is so the server has a mechanism for expiring puzzles handed out to clients. In our model we use str to capture this input and do not enforce any particular structure on it.

REMARK 2. To prevent DoS attacks that exhaust the server memory it is desirable that the server stores as little state as possible for uncompleted protocol runs (i.e. before a puzzle has been solved). We refer to this concern of client puzzles as “state storage costs” [2]. We build this into our definition of a client puzzle by insisting that *only* a single value, s , is stored by a server; all the data necessary to solve a given puzzle and to re-generate, and hence verify authenticity of a puzzle and solution pair, is included in the puzzle description puz .

REMARK 3. Generally, for a puzzle to be “secure” when used within a client puzzle protocol, we want puzzles generated to be unique and for puzzle and solution pairs to only be validly used once by a client. In actual usage, a server can filter out resubmitted correctly solved puzzle and solution pairs by, for example, using a hash table mechanism. Uniqueness of puzzles can be ensured by having GenPuz select a random nonce n_s and use this in the puzzle generation.

REMARK 4. Our definition assumes private verifiability for VerAuth . Generally the only party concerned with checking who generated a given puzzle is the puzzle generator (client puzzles are used *before* any other transactions take place and to protect the generator and no other party). Although in some cases it may be useful to have publicly verifiable puzzles it would complicate the definition and we choose to keep our definition practical yet as simple as possible.

4 Security Notions for Client Puzzles

We define two notions for client puzzles. The first measures the ability of an adversary to produce a correctly authenticating puzzle with an unknown private key. We refer to this as the ability of an adversary to forge a client puzzle. The second notion gives a measure of the likelihood of an adversary finding a solution to a given puzzle within a given number of clock cycles of execution. We refer to this as the difficulty of a client puzzle. Intuitively, these are both what one would expect to require from a client puzzle given its role in defenses against DoS attacks; being able to either forge puzzles or solve them faster than expected allows an adversary to mount a DoS attack.

We first review the definition of a function family since we will use function families to express security of a given client puzzle in terms of difficulty. A *function family* is a map $F : I \times D \mapsto R$. The set I is the set of all possible indices, D the domain and R the range. Unless otherwise specified we assume $I = \mathbb{N}$. The set R is finite and all sets are nonempty. We write $F_i : D \mapsto R$ for $F_i(d) = F(i, d)$ where $i \in I$ and refer to F_i as an instance of F .

UNFORGEABILITY OF PUZZLES. We first define our notion of unforgeability of client puzzles. Intuitively, we require an adversary that sees puzzles generated by the server (possibly together with their associated solutions), and that can verify the authenticity of any puzzle it chooses, cannot produce a valid looking puzzle on his own.

To formalize unforgeability of a client puzzle we use the following game $\text{Exec}_{\mathcal{A}, \text{CPuz}}^{\text{UF}}(k)$ between a challenger \mathcal{C} and an adversary \mathcal{A} .

- (1) The challenger \mathcal{C} first runs **Setup** on input 1^k to obtain (params, s) . The tuple params is given to \mathcal{A} and s is kept secret by \mathcal{C} .
- (2) The adversary \mathcal{A} gets to make as many **CreatePuz**(Q, str) and **CheckPuz**(puz) queries as it likes which \mathcal{C} answers as follows.
 - **CreatePuz**(Q, str) queries. A new puzzle is generated $\text{puz} \leftarrow \text{GenPuz}(s, Q, \text{str})$ and output to \mathcal{A} .
 - **CheckPuz**(puz) queries. If $\text{VerAuth}(s, \text{puz}) = \text{true}$ and puz was not output by \mathcal{C} in response to a **CreatePuz** query then \mathcal{C} terminates the game setting the output to 1. Otherwise **false** is returned to \mathcal{A} .
- (3) If \mathcal{C} does not terminate the game in response to a **Check** query then eventually \mathcal{A} terminates and the output of the game is set to 0.

We say the adversary \mathcal{A} wins if $\text{Exec}_{\mathcal{A}, \text{CPuz}}^{\text{UF}}(k) = 1$ and loses otherwise. We define the advantage of such an adversary as

$$\text{Adv}_{\mathcal{A}, \text{CPuz}}^{\text{UF}}(k) = \Pr \left[\text{Exec}_{\mathcal{A}, \text{CPuz}}^{\text{UF}}(k) = 1 \right].$$

Puzzle-unforgeability then means that no efficient adversary can win the above game with non-negligible probability.

Definition 2 (Puzzle-unforgeability). A client puzzle CPuz is UF secure if for any p.p.t. adversary \mathcal{A} its advantage $\text{Adv}_{\mathcal{A}, \text{CPuz}}^{\text{UF}}(k)$ is a negligible function of k .

REMARK 1. In the game $\text{Exec}_{\mathcal{A}, \text{CPuz}}^{\text{UF}}(k)$ we allow \mathcal{A} access to all algorithms defined in CPuz. In particular, we allow unlimited access to the GenPuz algorithm for any given chosen inputs. This allows \mathcal{A} to generate as many puzzles as it wishes (since \mathcal{A} is p.p.t. it will anyway generate at most polynomially many) with any given chosen key and difficulty values. Notice that the adversary can find solutions to any puzzle by running the FindSoln algorithm which is public. These abilities are sufficient to mimic the environment in which a DoS attacker would sit.

DIFFICULTY OF SOLVING PUZZLES. We formalize the idea that a puzzle CPuz cannot be solved trivially via the game $\text{Exec}_{\mathcal{A}, \text{CPuz}}^{Q, \text{DIFF}}(k)$ between a challenger \mathcal{C} and an adversary \mathcal{A} . The game is defined for each hardness parameter $Q \in \mathbb{N}$ as follows:

- (1) The challenger \mathcal{C} runs Setup on input 1^k to obtain (params, s) and passes params to \mathcal{A} .
- (2) The adversary \mathcal{A} is allowed to make any number of CreatePuzSoln(str) queries throughout the game. In response to each such query \mathcal{C} generates a new puzzle as $\text{puz} \leftarrow \text{GenPuz}(s, Q, \text{str})$ and finds a solution soln such that $\text{VerSoln}(\text{puz}, \text{soln}) = \text{true}$. The pair $(\text{puz}, \text{soln})$ is then output to \mathcal{A} .
- (3) At any point during the execution \mathcal{A} is allowed to make a single Test(str[†]) query. The challenger then generates a challenge puzzle as $\text{puz}^\dagger \leftarrow \text{GenPuz}(s, Q, \text{str}^\dagger)$ which it returns to \mathcal{A} .

Adversary \mathcal{A} terminates its execution by outputting a potential solution soln^\dagger . We define the running time τ of \mathcal{A} as being the running time of all of the experiment $\text{Exec}_{\mathcal{A}, \text{CPuz}}^{Q, \text{DIFF}}(k)$.

We say the adversary wins $\text{Exec}_{\mathcal{A}, \text{CPuz}}^{Q, \text{DIFF}}(k)$ if $\text{VerSoln}(\text{puz}^\dagger, \text{soln}^\dagger) = \text{true}$. In this case we set the output of $\text{Exec}_{\mathcal{A}, \text{CPuz}}^{Q, \text{DIFF}}(k)$ to be 1 and otherwise to 0. We then define the success of an adversary \mathcal{A} against CPuz as

$$\text{Succ}_{\mathcal{A}, \text{CPuz}}^{Q, \text{DIFF}}(k) = \Pr \left[\text{Exec}_{\mathcal{A}, \text{CPuz}}^{Q, \text{DIFF}}(k) = 1 \right].$$

We define the difficulty of puzzle solving by requiring that for any puzzle hardness the success of any adversary that runs in a bounded number of steps falls below a certain threshold (that is related to the hardness of the puzzle).

Definition 3 (Puzzle-difficulty). Let $\varepsilon : \mathbb{N}^2 \mapsto (\mathbb{N} \mapsto [0, 1])$ be a family of monotonically increasing functions. We use the notation $\varepsilon_{k, Q}(\cdot)$ for the function within this family corresponding to security parameter k and hardness parameter Q . We say a client puzzle CPuz is $\varepsilon(\cdot)$ -DIFF if for all $\tau \in \mathbb{N}$, for all adversaries \mathcal{A} in $\text{Exec}_{\mathcal{A}, \text{CPuz}}^{\text{DIFF}, Q}(k)$, for all security parameters $k \in \mathbb{N}$, and for all $Q \in \mathbb{N}$ it holds that

$$\text{Succ}_{\mathcal{A}_\tau, \text{CPuz}}^{Q, \text{DIFF}}(k) \leq \varepsilon_{k, Q}(\tau)$$

where \mathcal{A}_τ is the adversary \mathcal{A} restricted to at most τ clock cycles of execution.

REMARK 1. The security game above allows \mathcal{A} to obtain many puzzle and solution pairs by making queries to model actual usage in DoS settings; when a client puzzle is used as part of a client puzzle protocol an adversary may see many such puzzles and solutions exchanged between a given generator and solver on a network. The adversary could then learn something from these.

REMARK 2. In the definition of the Test query, we do allow the string str^\dagger to be one previously submitted as a CreatePuzSoln query and allow CreatePuzSoln queries on any string including str^\dagger after the Test query. It then immediately follows that a difficult puzzle needs to be such that each puzzle generated is unique. Otherwise, a previously obtained solution through the CreatePuzSoln query may serve as a solution to the challenge query. Furthermore, it also follows that solutions to some puzzles should not be related to the solutions of other puzzles, as otherwise a generalization of the above attack would work.

REMARK 3. The queries CreatePuz (used in the game for puzzle-unforgeability) and CreatePuzSoln used in the above game are related, but different. The query CreatePuzSoln outputs a puzzle together with its solution. The second is more subtle: in a CreatePuz query we allow \mathcal{A} to specify the value of Q used but in CreatePuzSoln we do not (the value of Q is fixed throughout the difficulty game).

REMARK 4. Clearly any puzzle that is $\varepsilon(\cdot)$ -DIFF is also $(\varepsilon(\cdot) + \mu)$ -DIFF where $\mu \in \mathbb{R}_{>0}$ is such that $\varepsilon(\tau) + \mu \leq 1$ (since $\text{Succ}_{\mathcal{A}_\tau, \text{CPuz}}^{Q, \text{DIFF}}(k) \leq \varepsilon_{k, Q}(\tau) \leq \varepsilon_{k, Q}(\tau) + \mu$). The most accurate measure of difficulty for a given puzzle CPuz is then the function $\varepsilon(\tau) = \inf_{\mathcal{A}_\tau} \text{Succ}_{\mathcal{A}_\tau, \text{CPuz}}^{Q, \text{DIFF}}(k)$.

REMARK 5. Since we measure the running time of the adversary in clock cycles, the model abstracts away the possibility that the adversary may be distributed and thus facilitates further analysis (for example of the effectiveness of client puzzle defense against DoS attacks).

5 An Attack on the Juels and Brainard Puzzles

In this section we describe an attack on the Juels and Brainard [14] client puzzle mechanism as described in Section 2. The attack works because puzzles are forgeable, which is due to a crucial weakness in puzzle generation; each set of generation parameters defines a family of puzzles each with a different hardness value. Finally we construct a DDoS attack on servers using certain client puzzle protocols based on this construction. This attack clearly demonstrates the applicability of our definitions and how they can be used to find problems with a given client puzzle construction.

PROVING FORGEABILITY. The reason the construction is forgeable is the authentication is not unique to a given instance but covers a number of instances of varying difficulty. This occurs because the puzzle instance difficulty is not included in the first preimage of the sub-puzzle construction. We exploit this weakness and

construct an adversary \mathcal{A} with $\mathbf{Adv}_{\mathcal{A}, \text{CPuz}}^{\text{UF}}(k) = 1$. We have the following lemma regarding the forgeability of the Juels and Brainard construction.

Lemma 1. *The client puzzle construction of Juels and Brainard [14] is not UF secure.*

Proof. To prove this we construct an adversary \mathcal{A} against the UF security of the construction that can win the security game $\text{Exec}_{\mathcal{A}, \text{CPuz}}^{\text{UF}}(k)$ with probability 1. We now describe the details of \mathcal{A} .

At the start of the security game \mathcal{A} is given a set of public parameters. The adversary then makes a query $\text{CreatePuz}(Q, \text{str})$ for some random choices of Q and str where $Q = (\alpha, \beta)$ and receives a puzzle instance $\text{puz} = (Q, \text{str}, P = (P_1, P_2, \dots, P_\beta))$ in response. Next \mathcal{A} removes the first bit of each P_i to obtain P_i^\dagger and constructs $Q^\dagger = (\alpha + 1, \beta)$ and $\text{puz}^\dagger = (Q^\dagger, \text{str}, P^\dagger = (P_1^\dagger, P_2^\dagger, \dots, P_\beta^\dagger))$. The adversary then makes a query $\text{CheckPuz}(\text{puz}^\dagger)$.

Clearly \mathcal{A} wins with probability 1 since puz and puz^\dagger are both generated from the same s and str hence puz^\dagger will correctly verify yet was not output from a CreatePuz query. \square

REMARK 1. One could also prove Lemma 1 by having the adversary construct the forgery as $Q^\dagger = (\alpha, \beta - 1)$ and then $\text{puz}^\dagger = (Q^\dagger, \text{str}, (P_1^\dagger, \dots, P_{\beta-1}^\dagger))$. One could also vary the number of bits moved between α and each P_i or change the number of sub-puzzles deleted. The reason we choose to give the proof in the manner given is because this specific method allows for the construction of a DDoS attack with the given assumptions we make about the protocol using this particular client puzzle. We describe this attack next.

CONSTRUCTING A DDoS ATTACK. We now use the forgeability of the construction to mount a DDoS attack on client puzzle protocols based on this client puzzles. The attack works when the difficulty parameter is increased in a certain way and when the hash table, mentioned in [14] and used to prevent multiple puzzle instance and solution submissions, is based on some unique data for each instance that is not in the preimage of any sub-puzzle. A hash table mechanism that depends on some unique data contained in each sub-puzzle preimage, as is mentioned in [14], would thwart the following DDoS attack on client puzzle protocols based on this client puzzle.

We first assume the client puzzle is used in the client puzzle protocol of [14] and the generator increases Q by increasing α many times for each increase in β . We also assume any hash tables used are computed using either the puzzle instance alone or the correct solutions alone.

To mount the DDoS attack the adversary commands each of its zombies (platforms the adversary controls) to start a run of the protocol with the server under attack. The server will begin to issue puzzle instances and then, when enough requests are received, will increase Q by incrementing α . Each zombie computes a solution to the first puzzle it receives and to submits this to the server. Then, while this puzzle has not expired, each time α is incremented, a new puzzle and solution pair is trivially computed by removing the first bit from each x_i and

concatenating this to the end of each soln_i previously computed. The new puzzle and solution pair are submitted to the server and will correctly verify and will then be allocated buffer space (due to our assumptions on the hash table mechanism). When a zombies' puzzle expires it obtains a new one. As the value Q is increased then so will the puzzle expiry period and hence more forged puzzles can be used per valid puzzle obtained eventually exhausting the memory resources of the server.

Also, even if we assume that the buffer allocation based on the hash table mechanism is as in [14] the attack will still consume a huge amount of server computational resources. This is because the adversary can trivially spoof new puzzle instances and solutions from previous ones. These will not be allocated buffer space due to the hash table mechanism, but will consume computation via server verification computations. In the next section we give an example instantiation of a generic construction that is a repaired version of the sub-puzzle mechanism; an unforgeable version of the sub-puzzle construction.

6 A Generic Client Puzzle Construction

In this section we provide a generic construction for a client puzzle which also repairs the flaw identified in the previous section with respect to the Juels and Brainard puzzle. Our construction is based on a pseudorandom function (PRF) and a one way function (OWF). We prove our generic construction is secure according to the definitions we put forth in this paper, and show one possible instantiation. Intuitively, the unforgeability of puzzles is ensured by the use of the PRF and the difficulty of solving puzzles is ensured by the hardness of inverting the one-way function. We first review some notational conventions and definitions regarding function families, pseudorandom functions, and concrete notions for pseudorandom function families and one way function families.

If F is a function family then we use the notation $f \stackrel{\$}{\leftarrow} F$ for $i \stackrel{\$}{\leftarrow} I$; $f \leftarrow F_i$. We denote the set of all possible functions mapping elements of D to R by $\text{Func}(D, R)$. A *random function* from D to R is then a function selected uniformly at random from $\text{Func}(D, R)$.

PSEUDORANDOMNESS. We define the PRF game $\text{Exec}_{\mathcal{B}, F}^{\text{PRF}, b}(k)$ for an adversary \mathcal{B} against the function family $F : \mathcal{K} \times D \mapsto R$, where $|\mathcal{K}| = 2^k$, as follows.

- (1) For $b = 1$ the adversary \mathcal{B} has black box access to a truly random function \mathcal{R} from the set $\text{Func}(D, R)$ and for $b = 0$ the adversary \mathcal{B} has black box access to a function F_s chosen at random from F .
- (2) The adversary \mathcal{B} is allowed to ask as many queries as it wants to whichever function it has black box access to. Eventually \mathcal{B} terminates outputting a bit b^* .

We set the output of $\text{Exec}_{\mathcal{B}, F}^{\text{PRF}, b}(k)$ to 1 if $b^* = b$ and set the output to 0 otherwise. We then define the advantage of an adversary against F in terms of PRF as

$$\mathbf{Adv}_{\mathcal{B}, F}^{\text{PRF}}(k) = \left| \Pr[\text{Exec}_{\mathcal{B}, F}^{\text{PRF}, 0}(k) = 1] - \Pr[\text{Exec}_{\mathcal{B}, F}^{\text{PRF}, 1}(k) = 1] \right|.$$

CONCRETE PSEUDORANDOM AND ONE WAY FUNCTION FAMILIES. Here we briefly review concrete notions of security for pseudorandom function and one way function families. We depart from the typical “ (ε, t) -hardness” style of definitions, as they are not sufficient for our purposes. Instead we view ε , the probability of a break, as a function of the running time τ of the adversary. So, a primitive is $\varepsilon(\cdot)$ -secure if for all adversaries running in time τ the probability of breaking the primitive is at most $\varepsilon(\tau)$.

Definition 4 ($\nu_k(\cdot)$ -PRFF). *Let $F : \mathcal{K} \times D \mapsto R$ be a function family and $\nu : \mathbb{N} \mapsto [\mathbb{N} \mapsto [0, 1]]$ be a family of monotonically increasing functions. We say F is a $\nu_k(\cdot)$ -PRFF if for all $k \in \mathcal{K}$ and for all adversaries \mathcal{A} it holds that $\text{Adv}_{\mathcal{A}, F}^{\text{PRF}}(k) \leq \nu_k(\tau)$.*

Note that, in the definition of an $\nu_k(\cdot)$ -PRFF, the security parameter k specifies the size of the keyspace for the game $\text{Exec}_{\mathcal{A}, F}^{\text{PRF}, b}(k)$ and the actual key, and hence function from the family used, is chosen at random from this keyspace.

Definition 5 ($\varepsilon_i(\cdot)$ -OWF). *For an adversary \mathcal{A} we define its advantage against a function $\psi : \mathcal{X} \mapsto \mathcal{Y}$, where \mathcal{X} is fixed and finite, in terms of OWF as*

$$\text{Adv}_{\mathcal{A}, \psi}^{\text{OWF}} = \Pr[x \xleftarrow{\$} \mathcal{X}; y \leftarrow \psi(x); (\tilde{x} \leftarrow \mathcal{A}(y) \wedge \psi(\tilde{x}) = y)].$$

Let $\varepsilon_i : \mathbb{N} \mapsto [0, 1]$ be a monotonically increasing function. Then, the function ψ is an $\varepsilon_i(\cdot)$ -OWF if for all adversaries \mathcal{A} it holds that $\text{Adv}_{\mathcal{A}, \psi}^{\text{OWF}} \leq \varepsilon_i(\tau)$.

We then extend this definition to a family of functions as follows:

Definition 6 ($\varepsilon(\cdot)$ -OWFF). *Let $\varphi : \mathbb{N} \mapsto (\mathcal{X} \mapsto \mathcal{Y})$ and $\varepsilon : \mathbb{N} \mapsto (\mathbb{N} \mapsto [0, 1])$ be function families. We say φ is an $\varepsilon(\cdot)$ -OWFF if for all $i \in \mathbb{N}$ the function $\varphi_i : \mathcal{X} \mapsto \mathcal{Y}$ is an $\varepsilon_i(\cdot)$ -OWF.*

THE GENERIC CONSTRUCTION. Our generic construction is based on the method of Juels and Brainard [14]. Most client puzzle constructions based on one way functions, such as the discrete log based scheme of [28], and the RSA based scheme of [10], can be described in this manner with some minor modifications. So, our generic construction pins down sufficient assumptions on the building blocks that imply security of the resulting puzzle. We let $k \in \mathbb{N}$ then let $F : \mathcal{K} \times D \mapsto \mathcal{X}$ where $|\mathcal{X}| \geq |\mathcal{K}| = 2^k$ be a function family indexed by elements of \mathcal{K} . The domain D of F_s is 3-tuples of the form $\mathbb{N} \times \{0, 1\}^* \times \{0, 1\}^k \in \{0, 1\}^*$. We write $F_s((\cdot, \cdot, \cdot))$ when we want to specify the exact encoding of an element of D explicitly as an input to F_s . We further let $\varphi : \mathbb{N} \mapsto (\mathcal{X} \mapsto \mathcal{Y})$ be a family of functions indexed by Q . We assume there is a polynomial time algorithm to compute φ_Q for each value of Q and input. The various algorithms in the scheme are then as follows:

Setup(1^k). The various spaces are chosen; $\text{sSpace} \leftarrow \mathcal{K}$, $\text{QSpace} \leftarrow \mathbb{N}$, $\text{strSpace} \leftarrow \{0, 1\}^*$, $\text{solnSpace} \leftarrow \mathcal{X}$ and $\text{puzSpace} \leftarrow \text{QSpace} \times \text{strSpace} \times \{0, 1\}^k \times \mathcal{Y}$. The parameter Π is assigned to be the polynomial time algorithm to compute φ_Q for all $Q \in \text{QSpace}$ and $x \in \mathcal{X}$. Finally, the value s is chosen as $s \xleftarrow{\$} \text{sSpace}$ and the tuple params constructed then output.

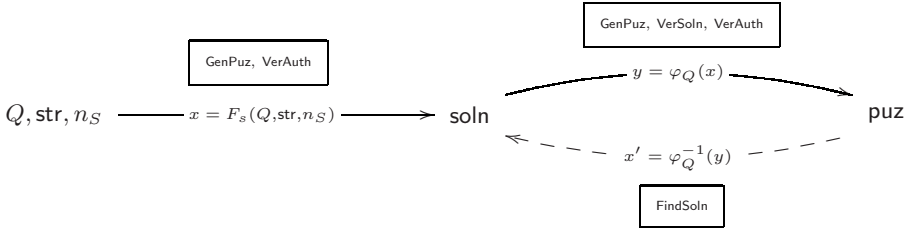


Fig. 2. Solid arrows are actions performed by a generator and dashed ones by a solver. The lists of algorithms above/below arrows imply the actions are performed as part of these algorithms. The details of how each action is used in the given algorithm are given in the full description.

GenPuz(s, Q, str). A nonce is selected $n_S \xleftarrow{\$} \{0, 1\}^k$. Next x is computed as $x \leftarrow F_s(Q, \text{str}, n_S)$. The value $y \in \mathcal{Y}$ is computed as $y \leftarrow \varphi_Q(x)$ and the puzzle assigned to be $\text{puz} = (Q, \text{str}, n_S, y)$ and output.

FindSoln(puz, τ). While this algorithm is within the allowed number of clock cycles of execution it randomly samples elements from the set of possible solutions without replacement and for each potential preimage $x' \in \mathcal{X}$ computes $y' \leftarrow \varphi_Q(x')$. If $y' = y$ this outputs x' then halts and otherwise continues with random sampling. If this algorithm reaches the last clock cycle of execution then it outputs a random element of the remaining unsampled preimage space. The set of possible solutions is generally a subset of \mathcal{X} that is defined by the value y of size dependent upon Q in some manner; the details of how the size varies depends upon the function family φ .

VerAuth(s, puz'). For a puzzle $\text{puz}' = (Q', \text{str}', n'_S, y')$ this computes x' as $x' \leftarrow F_s(Q', \text{str}', n'_S)$ then $y \leftarrow \varphi_Q(x')$. If $y' = y$ this outputs **true** and otherwise outputs **false**.

VerSoln($\text{puz}', \text{soln}'$). Given a potential solution $\text{soln}' = x'$ this checks if $\varphi_Q(x') = y$ and if so outputs **true** and otherwise outputs **false**.

We use the notation $\text{CPuz} = \text{PROWF}(F; \varphi)$ for the generic construction in this manner. The construction is summarized in Figure 2.

REMARK 1. In the definition of an $\varepsilon(\cdot)$ -OWF we specify the domain \mathcal{X} is fixed and finite but do not specify the exact size or shape of this; in our generic construction this is set to be the output space of some PRF.

REMARK 2. The exact specification of the **FindSoln** algorithm is not important for our theorems and proofs, nor is it unique. Indeed, other techniques such as exhaustive search may even be faster than the algorithm given. The important point is such an algorithm exists and can be described.

REMARK 3. The domain D of F is given as 3 tuples of the form $\mathbb{N} \times \{0, 1\}^* \times \{0, 1\}^k$ which is the same as $\{0, 1\}^*$. However, we will always construct elements

of D from a given tuple rather than taking a bit string and encoding it as an element of D . Hence we do not refer to this as a uniquely recoverable encoding on D .

REMARK 4. In reality the variable n_S need not be sampled at random; it just has to be a nonce and could be instantiated with, for example, a counter. We specify uniform sampling from the domain of n_S since it makes our proofs simpler and easier to follow.

REMARK 5. Our generic construction is similar to the Juels and Brainard scheme [14] but avoids the forgeability problems by including the hardness parameter Q in the input to F .

REMARK 6. Finally, we remark that the generic construction where the PRF function is replaced by a MAC is not necessarily secure. Indeed, one-wayness of the generic construction is guaranteed as long as the one-way function is applied to randomly chosen bit-string. While this property is ensured through the use of a pseudo-random function, it does not always hold for a MAC. For example, the combination of an artificial MAC function for which the first half of the output bits are constant with the OWF that discards the first half of its input is clearly an insecure puzzle construction.

The following theorems capture the unforgeability and the level of hardness enjoyed by our generic construction. Their proofs can be found in the full version of the paper.

Theorem 1. *Let F be a PRF family and φ a family of functions as described above such that for each value of Q and for all $y \in \mathcal{Y}$ we have $|\varphi_Q^{-1}(y)|/|\mathcal{X}| \leq 1/2^k$, where k is the security parameter. Then the client puzzle defined by $\text{CPuz} = \text{PROWF}(F; \varphi)$ is UF secure.*

To understand the rôle of the condition that $|\varphi_Q^{-1}(y)|/|\mathcal{X}| \leq 1/2^k$ consider the (extreme) case when F has a small constant number of images, that each corresponds to roughly the same number of possible inputs to F . Notice that this condition does not contradict the pseudorandomness of F , but such a function is not sufficient to ensure unforgeability. Indeed, an attacker can select a random $y \in \mathcal{Y}$, obtain some x such that $\varphi(x) = y$ and select some random triple (Q, str, n_S) as the solution to the puzzle. With probability about half, the image of (Q, str, n_S) is x . The adversary can therefore produce solved puzzles that are valid without interacting with the server.

Theorem 2. *Let F be a $\nu(\cdot)$ -PRFF family for the function family $\nu(\cdot)$, φ an $\varepsilon(\cdot)$ -OWFF for the function family $\varepsilon(\cdot)$ and $\text{CPuz} = \text{PROWF}(F; \varphi)$. Then the client puzzle $\text{PROWF}(F; \varphi)$ is $\gamma(\cdot)$ -DIFF where*

$$\gamma_{k,Q}(\tau) = 2 \cdot \nu_k(\tau + \tau^0) + (1 + \tau/(2^k - \tau)) \cdot \varepsilon_Q(\tau + \tau^1)$$

and $\tau^0, \tau^1 \in \mathbb{N}$ are some constants.

An adversary may try to solve puzzles by either computing the value $F_s(Q, \text{str}, n_S)$ for an unknown value of s or by computing a preimage of φ_Q for the value

y provided. The function ν_k in Theorem 2 captures that computing F_s for an unknown value of s should not be easy; the function F needs to be a good PRF. Intuitively, k should be chosen to be large enough that it is easier to compute a preimage of y under φ_Q than computing the corresponding value $F_s(Q, \text{str}, n_S)$.

Impact on Practical Implementations of Puzzles

Some of the most popular proposals of puzzles are based on hash functions. In this section we instantiate our generic construction from the previous section using hash functions to construct the needed PRF and one-way function families. We obtain essentially a modified Juels and Brainard scheme that incorporates the defence against the attack that we present in Section 5. The security analysis is in the random oracle model.

Given a hash function $H : \{0, 1\}^* \mapsto \{0, 1\}^m$ a standard construction for a PRF family F is as follows. Key generation selects a random string $s \xleftarrow{\$} \{0, 1\}^k$ where k is the security parameter. Function application is defined by $F_s(x) = H(s || x)$ for any $x \in \{0, 1\}^*$. Furthermore, given a hash function $G : \{0, 1\}^* \rightarrow \{0, 1\}^n$ we define the function family φ of functions $\varphi_Q : \{0, 1\}^m \rightarrow \{0, 1\}^{m-Q} \times \{0, 1\}^n$ by $\varphi_Q(x) = (x \langle Q + 1, m \rangle, G(x))$. In the full version of the paper we prove that in the random oracle model, F is a $\nu_k(\cdot)$ -PRFF function, for some function family ν with $\nu_k(\tau) \leq \frac{m}{2^k}$ and that φ is $\varepsilon(\cdot)$ -OWFF for some function family ε with $\varepsilon(\tau) \leq \tau/2^m + \tau/(2^{m-Q})$. Concrete bounds for the security of our construction follow by instantiating the bounds in Theorems 1 and 2.

Acknowledgements. The authors would like to thank EPSRC and the EU FP7 project eCrypt-2 for partially supporting the work in this paper. The third author was supported by a Royal Society Wolfson Research Merit Award. The authors would also like to thank Colin Boyd for helpful discussion in this work.

References

1. Aiello, W., Bellare, S.M., Blaze, M., Canetti, R., Ioannidis, J., Kermoytis, A.D., Reingold, O.: Just Fast Keying: Key Agreement In A Hostile Internet. *ACM Trans. on Info. and Syst. Sec.* 4, 1–30 (2004)
2. Aura, T., Nikander, P., Leiwo, J.: DoS-Resistant Authentication with Client Puzzles. In: Christianson, B., Crispo, B., Malcolm, J.A., Roe, M. (eds.) *Security Protocols 2000*. LNCS, vol. 2133, pp. 170–181. Springer, Heidelberg (2001)
3. Barak, B., Mahmoody-Ghidary, M.: Merkle Puzzles are Optimal. *Cryptology ePrint archive*, report 2008/032 (2008)
4. Biham, E., Goren, Y.J., Ishai, Y.: Basing Weak Public-Key Cryptography on Strong One-Way Functions. In: Canetti, R. (ed.) *TCC 2008*. LNCS, vol. 4948, pp. 55–72. Springer, Heidelberg (2008)
5. 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)

6. Chen, L., Mao, W.: An Auditable Metering Scheme for Web Advertisement Applications. In: Davida, G.I., Frankel, Y. (eds.) *ISC 2001*. LNCS, vol. 2200, pp. 475–485. Springer, Heidelberg (2001)
7. Douligeris, C., Mitrokotsa, A.: DDoS Attacks and Defence mechanisms: Classification and State-of-the-Art. *Computer Networks* 44, 643–666 (2004)
8. Dwork, C., Naor, M.: Pricing via Processing or Combatting Junk Email. In: Brickell, E.F. (ed.) *CRYPTO 1992*. LNCS, vol. 740, pp. 139–147. Springer, Heidelberg (1993)
9. Eddy, W.: TCP SYN Flooding Attacks and Common Mitigations. RFC 4987
10. Gao, Y.: Efficient Trapdoor-Based Client Puzzle System Against DoS Attacks. M.Sc Thesis, University of Wollongong, Computer Science Department (2005)
11. Groza, B., Petrica, D.: On Chained Cryptographic Puzzles. In: 3rd Romanian-Hungarian Joint Symp. on Applied Comput. Intel. – SACI, pp. 25–26 (2006)
12. Impagliazzo, R., Rudich, S.: Limits on the Provable Consequences of One-Way Permutations. In: *ACM Symp. on the Theory of Comp. – STOC 1989*, pp. 44–61 (1989)
13. Jakobsson, M., Juels, A.: Proofs of Work and Bread Pudding Protocols. In: *Joint Working Conference on Secure Information Networks: Communications and Multimedia Security*. IFIP Conference Proceedings, vol. 152, pp. 258–272 (1999)
14. Juels, A., Brainard, J.: Client Puzzles: A Cryptographic Countermeasure Against Connection Depletion Attacks. In: *ISOC Network and Distributed System Security Symposium*, pp. 151–165 (1999)
15. Karig, D., Lee, R.: Remote Denial of Service Attacks and Countermeasures. Princeton University Department of Electrical Engineering Technical Report CE-L2001–002 (2001)
16. Meadows, C.: A Formal Framework and Evaluation Method for Network Denial of Service. In: *12th Computer Security Foundations Workshop*, pp. 4–13. IEEE Computer Society Press, Los Alamitos (1999)
17. Meadows, C.: A Cost-Based Framework for Analysis of Denial of Service in Networks. *Journal of Computer Security* 9, 143–164 (2001)
18. Merkle, R.: Secure Communications Over Insecure Channels. *Communications of the ACM* 21, 294–299 (1978)
19. Mirkovic, J., Martin, J., Reiher, P.: A Taxonomy of DDoS Attack and DDoS Defense Mechanisms. *ACM SIGCOMM Computer Communication Review* 34, 39–53 (2004)
20. Moskowitz, R., Nikander, P., Jokela, P., Henderson, T.: Host Identity Protocol. Internet Draft (October 2007)
21. Price, G.: A General Attack Model of Hash-Based Client Puzzles. In: Paterson, K.G. (ed.) *Cryptography and Coding 2003*. LNCS, vol. 2898, pp. 319–331. Springer, Heidelberg (2003)
22. Rivest, R.L., Shamir, A., Wagner, D.: Time-lock Puzzles and Timed-release Crypto. Massachusetts Institute of Technology Technical Report TR-684 (1996)
23. Rogaway, P.: Formalizing Human Ignorance. In: Nguy en, P.Q. (ed.) *VIETCRYPT 2006*. LNCS, vol. 4341, pp. 211–228. Springer, Heidelberg (2006)
24. Schaller, P., Capkun, S., Basin, D.: BAP: Broadcast Authentication Using Cryptographic Puzzles. In: Katz, J., Yung, M. (eds.) *ACNS 2007*. LNCS, vol. 4521, pp. 401–419. Springer, Heidelberg (2007)

25. Smith, J., González-Nieto, J.M., Boyd, C.: Modelling Denial of Service Attacks on JFK with Meadows's Cost-Based Framework. In: Proceedings of the 2006 Australasian workshop on Grid computing and e-research, vol. 54, pp. 125–134 (2006)
26. Specht, S., Lee, R.: Distributed Denial of Service: Taxonomies of Attacks, Tools, and Countermeasures. In: 17th International Conference on Parallel and Distributed Computing Systems, pp. 543–550 (2004)
27. Tritilanunt, S., Boyd, C., Foo, E., González Nieto, J.M.: Toward Non-parallelizable Client Puzzles. In: Bao, F., Ling, S., Okamoto, T., Wang, H., Xing, C. (eds.) CANS 2007. LNCS, vol. 4856, pp. 247–264. Springer, Heidelberg (2007)
28. Waters, B., Juels, A., Halderman, J.A., Felten, E.W.: New Client Puzzle Outsourcing Techniques for DoS Resistance. In: 11th ACM Conference on Computer and Communication Security – CCS, pp. 246–256. ACM Press, New York (2004)

Foundations of Non-malleable Hash and One-Way Functions

Alexandra Boldyreva¹, David Cash¹, Marc Fischlin², and Bogdan Warinschi³

¹ Georgia Institute of Technology, USA
{aboldyre,cdc}@cc.gatech.edu

² Darmstadt University of Technology, Germany
marc.fischlin@gmail.com

³ University of Bristol, UK
bogdan@cs.bris.ac.uk

Abstract. Non-malleability is an interesting and useful property which ensures that a cryptographic protocol preserves the independence of the underlying values: given for example an encryption $\mathcal{E}(m)$ of some unknown message m , it should be hard to transform this ciphertext into some encryption $\mathcal{E}(m^*)$ of a related message m^* . This notion has been studied extensively for primitives like encryption, commitments and zero-knowledge. Non-malleability of one-way functions and hash functions has surfaced as a crucial property in several recent results, but it has not undergone a comprehensive treatment so far. In this paper we initiate the study of such non-malleable functions. We start with the design of an appropriate security definition. We then show that non-malleability for hash and one-way functions can be achieved, via a theoretical construction that uses perfectly one-way hash functions and simulation-sound non-interactive zero-knowledge proofs of knowledge (NIZKPoK). We also discuss the complexity of non-malleable hash and one-way functions. Specifically, we show that such functions imply perfect one-wayness and we give a black-box based separation of non-malleable functions from one-way permutations (which our construction bypasses due to the “non-black-box” NIZKPoK based on trapdoor permutations). We exemplify the usefulness of our definition in cryptographic applications by showing that (some variant of) non-malleability is necessary and sufficient to securely replace one of the two random oracles in the IND-CCA encryption scheme by Bellare and Rogaway, and to improve the security of client-server puzzles.

1 Introduction

MOTIVATION. Informally, non-malleability of some function f is a cryptographic property that asks that learning $f(x)$ for some x does not facilitate the task of generating some $f(x^*)$ so that x^* is related to x in some non-trivial way. This notion is especially useful when f is used to build higher-level multi-user protocols where non-malleability of the protocol itself is crucial (e.g., for voting or auctioning). Non-malleability has been rather extensively studied for some cryptographic primitives. For example, both definitions as well as constructions from

standard cryptographic assumptions are known for encryption, commitments and zero-knowledge [17,5,29,16,20,14,11,15,27,28,2]. Non-malleability in the case of other primitives, notably for one-way functions and for hash functions,¹ has only recently surfaced as a crucial property in several works [7,8,11,19], which we discuss below.

For instance, plenty of cryptographic schemes are only proved secure in the random oracle (RO) model [4], where one assumes that a hash function behaves as a truly random function to which every party has access to. It is well-known that such proofs do not strictly guarantee security for instantiations with hash functions whose only design principles are based on one-wayness and/or collision-resistance, because random functions possess multiple properties the proofs may rely on. Hiding all partial information about pre-images, i.e. perfect one-wayness, is one of these properties, and has been studied in [9,12]. Non-malleability is another example of such a property.

An illustrative example is the encryption scheme of Bellare and Rogaway [4], where a ciphertext of message M has the form $(f(r), G(r) \oplus M, H(r, M))$ for a trapdoor permutation f , hash functions G, H and random r . The scheme is known to be IND-CCA secure in the random oracle model. However, an instantiation of H with a malleable function for which given $H(r, M)$ it is possible to compute $H(r, M \oplus M')$, for some fixed M' known to the attacker, renders the scheme insecure: the attacker can recover M by submitting to the decryption oracle the valid ciphertext $(f(r), G(r) \oplus M \oplus M', H(r, M \oplus M'))$.

It was shown in [7] that a similar attack can be carried out against the popular OAEP encryption scheme whenever the instantiation of the underlying hash function is malleable. A subsequent work [8] showed that some form of non-malleability permits positive results about security of an alleviated version of the OAEP scheme in the standard model. However, it remains unclear if the approach to non-malleability in [8] expands beyond the OAEP example, and the work left open the construction of non-malleable primitives.

Another motivating example is the abstraction used to model hash functions in symbolic (Dolev-Yao) security analysis. In this setting it is *axiomatized* that an adversary can compute some hash only when it knows the underlying value. Clearly, malleable hash functions do not satisfy this axiom. Therefore, non-malleability for hash functions is necessary in order to ensure that symbolic analysis is (in general) sound with respect to the standard cryptographic model. Otherwise, real attacks that use malleability can not be captured/discovered in the more abstract symbolic model.

In a different vein, and from a more conceptual perspective, higher-level protocols could potentially benefit from non-malleable hash functions as a building block. A recent concrete example is the recommended use of such non-malleable hash functions in a human-computer interaction protocol for protecting local storage [11]. There, access should be linked to the ability to answer human-solvable

¹ In the sequel we aggregate both one-way functions and hash functions under the term hash functions for simplicity.

puzzles (similar to CAPTCHAs), but it should be infeasible for a machine to maul puzzles and redirect them under a different domain to other human beings.

We will also discuss a construction of a cryptographic puzzle from [25] designed to prevent DoS attacks, and show that malleability of the underlying hash function leads to insecure constructions.

Hence, non-malleability is a useful design principle that designers of new hash functions should keep in mind. At this point, however, it is not even clear what the exact requirements from a theoretical viewpoint are. Therefore, a first necessary step is to find a suitable definition which is (a) achievable, and (b) applicable. The next step would be to design practical hash functions and compression functions which are non-malleable, or which at least satisfy some weaker variant of non-malleability.

CONTRIBUTIONS. In this paper we initiate the study of non-malleable hash functions. We start with the design of an appropriate security definition. Our definition uses the standard simulation paradigm, also employed in defining non-malleability for encryption and commitment schemes. It turns out however that a careless adjustment of definitions for other primitives yield definitions for non-malleable hash functions that cannot be realized. We therefore motivate and provide a meaningful variation of the definition which ensure that the notion is achievable and may be useful in applications.

Testifying to the difference to other cryptographic primitives, we note that for non-malleable encryption the original simulation-based definition of [17] was later shown to be equivalent to an indistinguishability-based definition [5]. For our case here, finding an equivalent indistinguishability-based definition for non-malleable hash functions appears to be far from trivial, and we leave the question as an interesting open problem.

We then show that our definition can be met. Our construction of a non-malleable hash function employs a perfectly one-way hash function (POWHF) [9,12], i.e., a probabilistic hash function which hides all information about its pre-image. Notice that this form of secrecy in itself does not ensure non-malleability, so we make the function non-malleable by appending a simulation-sound non-interactive zero-knowledge proof of knowledge (NIZKPoK) [29,14] of the hashed value.² Both primitives exist, for example, if trapdoor permutations exist.³

The construction we provide is probabilistic and does not achieve the desired level of efficiency for practical applications. We emphasize that our construction should be regarded as a feasibility result that shows that, in principle, non-malleable hash functions can be built from standard assumptions. We leave open

² Analogously to Canetti's terminology of perfectly one-way *hash* functions [9] we refer to our construction as a hash function since we require collision resistance, although it does not compress.

³ We remark that the intuitively appealing approach of using non-malleable encryption or commitment schemes to directly construct non-malleable hashes does not work. One of the reasons is that the former primitives rely on secret randomness, whereas hash values need to be publicly verifiable given the pre-image.

the problem of finding a practical, deterministic solution. We note that our definition is general enough to allow such constructions.

Next, we investigate necessary cryptographic assumptions for building non-malleable functions. We provide two results. First we show that a non-malleable hash function needs to hide any information about the pre-image. This result justifies the use of POWHFs in our construction. Then we show (in the style of Impagliazzo-Rudich [24]) that black-box constructions of non-malleable *one-way* functions from one-way permutations are in fact impossible even if the collision-resistance requirement is dropped. To be more precise, we follow the approach of Hsiao and Reyzin [23] and show that no black-box security reduction is possible. Notice that our construction circumvents the impossibility result due to the use of a “non-black-box” NIZKPoK.

Finally, we study the applicability of our definition. We show that non-malleability is in fact sufficient for secure partial instantiation of the aforementioned encryption scheme of Bellare and Rogaway [4], i.e., that the scheme remains IND-CCA secure when H is replaced with a non-malleable hash function. Although G is still a random oracle, this partial instantiation helps to better understand the necessary properties of the primitives and also provides a better security heuristic.

We also sketch an application to the framework of cryptographic puzzles [25] as a defense against DoS attacks, where non-malleability surfaces as an important property. The usefulness of the definition has also been shown in [19], using a special case of a preliminary version of our definition to prove that HMAC [3] is a secure message authentication code, assuming that the compression function of the hash function is non-malleable. We expect further applications of non-malleable hash functions in other areas, and some of the techniques used in our proof here may be helpful for these scenarios.

RELATED WORK. Independently of our work, Canetti and Dakdouk [10] and Pandey et al. [26] recently also suggested one-way functions with special properties related to, yet different from non-malleability, and Canetti and Varia [13] investigated non-malleable obfuscation. The work of Canetti and Dakdouk [10] introduces the notion of extractable perfect one-way functions where generating an image also guarantees that one knows a preimage. This should even hold if an adversary sees related images, a setting which somewhat resembles the one that we give for non-malleability. Yet, extractability in [10] is defined by requiring the existence of a knowledge extractor which generates a preimage from the adversary’s view, including the other images. In contrast, the common approach to non-malleability (which we also adopt) is to deny the simulator access to the other images, in order to capture the idea that these images should not help. Hence the security definition from [10] is incomparable to ours. Moreover using the notion of [10] to show insecurity of candidate practical hashes seems difficult: arguing about the success of an attacker under their definition involves, in particular, showing that it is impossible to extract a pre-image when someone produces an image. In contrast, security as defined by our notion is easier to refute. For example, the hash functions from [7] for which flipping a bit in the

pre-image results in flipping a bit in the image are clearly insecure under our definition.

The work by Pandey et al. [26] defines adaptive one-way function families where inversion for an image under some key is still infeasible, even if one is allowed to obtain preimages under different keys. This notion is also related to non-malleability and turns out to be useful to design non-malleable protocols like commitments and zero-knowledge proofs. Unfortunately, this strong notion is not known to be realizable.

It is noteworthy that, analogously to our work here, both papers choose the Bellare-Rogaway encryption function as an important test case, and succeed in instantiating the second random oracle of the scheme. Together with the notion that we develop in this paper, these give three different alternatives for the requirements needed for this instantiation. Those works also show that the first random oracle could be instantiated in the standard model with a function which in addition to the notions they define is also pseudorandom. Unfortunately, no construction from standard assumptions that meets either one of the two resulting notions is known. In contrast, our single-oracle instantiation through a non-malleable hash function is possible under standard assumptions.

The work by Canetti and Varia [13] independently considers the notion of verifiable non-malleable obfuscation where an adversary, given an obfuscated circuit, tries to produce an (obfuscated) circuit which is functionally related. The adversary's success is measured against the success of a simulator given only an oracle implementing the original circuit functionality. Their notion of verifiable non-malleable obfuscators comes closest to our notion of non-malleable hash functions, and their construction for achieving a weaker notion of verifiable non-malleable obfuscation resembles our feasibility construction closely.

The two notions are, nonetheless, different in spirit. For obfuscators the adversary's task is to find something *functionally* related, whereas for non-malleable hash functions the adversary's task is to find a hash of a related pre-image, thus capturing relations about *specific* values like relations among the bits. There are further technical differences like the fact that the (achievable) notion of weakly verifiable non-malleable obfuscators does not support auxiliary information—as required for our encryption case, for example—making the two notions incomparable. More details are given in Section 3.

2 Preliminaries

Definition 1 (Hash Functions). A hash function $\mathcal{H} = (\text{HK}, \text{H}, \text{HVf})$ consists of PPTAs for key generation, evaluation and verification, where

- PPTA HK for security parameter 1^k outputs a key K (which contains 1^k and implicitly defines a domain D_K),
- PPTA H for inputs K and $x \in D_K$ returns a value $y \in \{0, 1\}^*$,
- PTA HVf on inputs K, x, y returns a decision bit.

It is required that for any $K \stackrel{\$}{\leftarrow} \text{HK}(1^k)$, any $x \in D_K$, any $y \stackrel{\$}{\leftarrow} \text{H}(K, x)$, algorithm $\text{HVf}(K, x, y)$ outputs 1.

Note that we consider a very general syntax, comprising the “classical” notions of one-way functions (with a public key) and of collision-resistant hash functions which compress the input to a shorter digest (see [22] for definitions). In our case the evaluation algorithm H may be probabilistic, as long the correctness of hash values is verifiable given the pre-image only (via HVf). Also, we do not demand the length of the output of the hash function to be smaller than that of the input. However, while we capture a large class of primitives, the generalized syntax may not preserve all properties of the special cases, e.g., if the evaluation algorithm is probabilistic, two independent parties hashing the same input will not necessarily get the same value.

We now recall the definitions of one-wayness and collision resistance. For one-wayness the definition that we give is more general than the standard one in that it considers specific input distributions \mathcal{X} for the function, and also accounts for the possibility that the adversary may have some partial information about the pre-image (modeled through a probabilistic function hint):

Definition 2 (One-wayness and Collision-resistance). *A hash function $\mathcal{H} = (HK, H, HVf)$ is called*

- one-way (wrt \mathcal{X} and hint) if for any PPTA \mathcal{A} the probability that for $K \xleftarrow{\$} HK(1^k)$, $x \xleftarrow{\$} \mathcal{X}(1^k)$, $h_x \xleftarrow{\$} \text{hint}(K, x)$, $y \xleftarrow{\$} H(K, x)$ and $x^* \xleftarrow{\$} \mathcal{A}(K, y, h_x)$ we have $HVf(K, x^*, y) = 1$, is negligible.
- collision-resistant if for any PPTA \mathcal{A} the probability for $K \xleftarrow{\$} HK(1^k)$, $(x, x', y) \xleftarrow{\$} \mathcal{A}(K)$ that $x \neq x'$ but $HVf(K, x, y) = 1$ and $HVf(K, x', y) = 1$, is negligible.

3 Non-malleability of Hash and One-Way Functions

Our definition for hash functions follows the classical (simulation-based) approach for defining non-malleability [17]. Informally, our definition requires that for any adversary which, on input a hash value y , finds another value y^* such that the pre-images are related, there exists a simulator which does just as well without ever seeing y .

In the adversary’s attack we consider a three-stage process. The adversary first selects a distribution \mathcal{X} from which a secret input x is then sampled (and passes on some state information). In the second stage the algorithm sees a hash value y of this input x , and the adversary’s goal is to create another hash value y^* (usually different from y). In the third stage the adversary is given x and now has to output a pre-image x^* to y^* which is “related” to x (we make the definition stronger by giving the challenge pre-image to the adversary). The simulator may also pick a distribution \mathcal{X} according to which x is sampled, but then it needs to specify x^* directly from the key of the hash function only.

In the second stage the adversary (and consequently the simulator) also gets as input a “hint” h_x about the original pre-image x , to represent some a-priori information potentially gathered from other executions of other protocols in which x is used. In fact, such side information is often crucial for the deployment

in applications, e.g., for the encryption example in Section 6. As in the case of non-malleable commitments and encryption, related pre-images are defined via a relation $R(x, x^*)$. This relation may also depend on the distribution \mathcal{X} to catch significantly diverging choices of the adversary and the simulator and to possibly restrict the choices for \mathcal{X} , say, to require a certain min-entropy. However, unlike for other primitives, we do not measure the success of the adversary and the simulator for arbitrary relations R between x and x^* , but instead restrict the relations to a class \mathcal{R} of admissible relations. We discuss this and other subtleties after the definition:

Definition 3 (NM-Hash). *A hash function $\mathcal{H} = (\text{HK}, \text{H}, \text{HVf})$ is called non-malleable (with respect to probabilistic function hint and relation class \mathcal{R}) if for any PPTA $\mathcal{A} = (\mathcal{A}_d, \mathcal{A}_y, \mathcal{A}_x)$ there exists a PPTA $\mathcal{S} = (\mathcal{S}_d, \mathcal{S}_x)$ such that for every relation $R \in \mathcal{R}$ the difference*

$$\Pr \left[\mathbf{Exp}_{\mathcal{H}, \mathcal{A}}^{\text{nmh-1}}(k) = 1 \right] - \Pr \left[\mathbf{Exp}_{\mathcal{H}, \mathcal{S}}^{\text{nmh-0}}(k) = 1 \right] \text{ is negligible, where:}$$

<p>Experiment $\mathbf{Exp}_{\mathcal{H}, \mathcal{A}}^{\text{nmh-1}}(k)$</p> <p>$K \stackrel{\\$}{\leftarrow} \text{HK}(1^k)$</p> <p>$(\mathcal{X}, st_d) \stackrel{\\$}{\leftarrow} \mathcal{A}_d(K) \quad // \text{ for state } st_d$</p> <p>$x \stackrel{\\$}{\leftarrow} \mathcal{X}(1^k), h_x \stackrel{\\$}{\leftarrow} \text{hint}(K, x)$</p> <p>$y \stackrel{\\$}{\leftarrow} \text{H}(K, x)$</p> <p>$(y^*, st_y) \stackrel{\\$}{\leftarrow} \mathcal{A}_y(y, h_x, st_d)$</p> <p>$x^* \stackrel{\\$}{\leftarrow} \mathcal{A}_x(x, st_y)$</p> <p>Return 1 iff</p> <p style="padding-left: 20px;">$R(\mathcal{X}, x, x^*)$</p> <p style="padding-left: 20px;">$\wedge (x, y) \neq (x^*, y^*)$</p> <p style="padding-left: 20px;">$\wedge \text{HVf}(K, x^*, y^*) = 1$</p>	<p>Experiment $\mathbf{Exp}_{\mathcal{H}, \mathcal{S}}^{\text{nmh-0}}(k)$</p> <p>$K \stackrel{\\$}{\leftarrow} \text{HK}(1^k)$</p> <p>$(\mathcal{X}, st_d) \stackrel{\\$}{\leftarrow} \mathcal{S}_d(K)$</p> <p>$x \stackrel{\\$}{\leftarrow} \mathcal{X}(1^k), h_x \stackrel{\\$}{\leftarrow} \text{hint}(K, x)$</p> <p>$x^* \stackrel{\\$}{\leftarrow} \mathcal{S}_x(h_x, st_d)$</p> <p>Return 1 iff</p> <p style="padding-left: 20px;">$R(\mathcal{X}, x, x^*)$</p>
---	---

REMARK 1. Our definition is parameterized by a class of relations \mathcal{R} . This is because for some relations the definition is simply not achievable, as in the case when the relation involves the hash of x instead of x itself. More specifically, consider the relation $R(x, x^*)$ which parses x^* as K, y and outputs $\text{HVf}(K, x, y)$. Then, an adversary on input y, h_x, st_d may output $y^* \stackrel{\$}{\leftarrow} \text{H}(K, (K, y))$ and then, given x , returns $x^* = (K, y)$. This adversary succeeds in experiment $\mathbf{Exp}_{\mathcal{H}, \mathcal{A}}^{\text{nmh-1}}(k)$ with probability 1. In contrast, any simulator is likely to fail, as long as the hash function does not have “weak” keys, i.e., keys for which the distribution of generated images is trivial (such that the simulator can guess y with sufficiently high probability).

We resolve this problem by requiring the definition to hold for a subset \mathcal{R} of all relations. It is of course desirable to seek secure constructions with respect

⁴ Throughout the paper all hint functions and relations are assumed to be efficient. We furthermore assume that the security parameter is given in unary to all algorithms as additional input (if not mentioned explicitly).

to very broad classes of relations (cf. our construction in Section 4) which are more handy for general deployment. At the same time, certain scenarios may only require non-malleability with respect to a small set of relations (cf. the application example discussed in Section 6). Our definition is general and permits easy tuning for the needs of a particular application or a class of applications.

REMARK 2. For virtually all “interesting” functions \mathcal{H} and relation classes \mathcal{R} the definition is achievable only for adversaries and simulators that output descriptions of well-spread distributions \mathcal{X} (i.e., with super-logarithmic min-entropy). For the construction in next section we also require hint to be a so-called uninvertible function 9 (for which finding the exact pre-image is infeasible). Note that uninvertibility is a weaker requirement than one-wayness, as it holds for example for constant functions. We prefer to keep the definition as general as possible, so we do not explicitly impose such restrictions on the adversary, simulator, and hint .

REMARK 3. In our definition we demand that the simulator outputs x^* given K and h_x only. A weaker condition would be to have a simulator $\mathcal{S}_y(h_x, \text{st}_d)$ first output y^* , like the adversary \mathcal{A}_y , and then $x^* \leftarrow \mathcal{S}_x(x, \text{st}_y)$, before checking that $R(\mathcal{X}, x, x^*)$ and that $\text{Hvf}(K, x^*, y^*) = 1$. Since in this case the simulator in the second stage is also given x we call this a *weak simulator* and hash functions achieving this notion *weakly non-malleable*. This distinction resembles the notions of non-malleable commitments with respect to commitment and with respect to opening [16,20]. Depending on the application scenario of non-malleable hash functions the stronger or weaker version might be required. As an example, the result about the Bellare-Rogaway encryption scheme uses the stronger definition above, and our construction in the next section achieves this stronger notion, which obviously implies the weaker one.

REMARK 4. Similarly to the previous variation one can let the adversary only output a hash value y^* , and omit the step where it later also has to give x^* . The simulator’s task, too, is then to only output a hash value. Then one defines meaningful relations through existential quantifications (“... if there exists a pre-image x^* such that $R(x, x^*)$ holds”). This is essentially the approach taken by Canetti and Varia [13] for (weakly) verifiable non-malleable obfuscators.

On the one hand the “hash-only” approach above facilitates the adversary’s task if it does not need to know a specific pre-image. On the other hand, it also simplifies the simulator’s task. As an example the adversary in our definition may decide upon a *specific* x^* satisfying the relation, after seeing x . Security against such an attack cannot be captured by the above notion of relaxed simulators, whereas the simulator in our definition also needs to find an appropriate x^* . This particular example demonstrates that our approach and the definition for (weakly) verifiable non-malleable obfuscators in [13] are incomparable. Further differences between the notions are the lack of auxiliary information and the dependency of the simulator on the relation in the definition of Canetti and

Varia [13]. In addition, the feasibility results presented later in our paper and the solutions in [13] are for incomparable classes of relations.

REMARK 5. Note that we only demand that $(x, y) \neq (x^*, y^*)$ for the adversary's choice (instead of demanding $x \neq x^*$ or $y \neq y^*$ instead), yielding a stronger definition, especially when the randomized hash function has multiple images for some input. Again, the particular need depends on the application and our solution meets this stronger requirement.

REMARK 6. In the case of non-malleable encryption the original simulation-based definition of [17] was later shown to be equivalent to an indistinguishability-based definition [5]. The superficial similarity between our definition of non-malleable hash functions and the one of non-malleable encryption suggests that this may be possible here as well. Surprisingly, straightforward attempts to define non-malleability of hash functions through indistinguishability do not seem to yield an equivalent definition. We discuss this issue in the full version [6] in more detail (because of lack of space), and leave it as an interesting open problem to find a suitable indistinguishability-based definition for non-malleable hash functions.

REMARK 7. The usual security notions for hash functions include one-wayness and collision-resistance. However, neither property is known to follow from Definition 3. Consider a *constant* function H which is clearly not one-way nor collision-resistant. But the function is weakly non-malleable as a simulator can simulate \mathcal{A} in a black-box way by handing the adversary the constant value. We keep these rather orthogonal security properties separate, as some applications may require one but not the others.

REMARK 8. Some applications (like the HMAC example in [19]) require a multi-valued version of the definition in which the adversary can adaptively generate several distributions and receive the images (with side information) before deciding upon y^* . One can easily extend our definition accordingly, letting \mathcal{A}_d loop several times, in each round i generating a distribution \mathcal{X}_i and receiving y_i and h_{x_i} at the beginning of the next round and before outputting an image y^* . In general, it is possible to extend our construction to this case using stronger, adaptive versions of POWHFs and NIZKPoKs. See Remark 1 after Theorem 1.

4 Constructing Non-malleable Hash Functions

In this section we give feasibility results via constructions for non-malleable hash functions. The main ingredient of our constructions is a perfectly one-way hash function (POWHF) [9,12], which hides all information about the pre-image but which may still be malleable [7]. To ensure non-malleability we tag the hash value with a simulation-sound non-interactive zero-knowledge proof of knowledge of the pre-image. We first recall the definitions of these two primitives.

For POWHFs we slightly adapt the definition from [9,12] to our setting. Originally, POWHFs have been defined to have a specific input distribution \mathcal{X} (like the uniform distribution in [12,18]). Here we let the adversary choose the input

distribution adaptively, and merely demand that this distribution \mathcal{X} satisfies a certain efficient predicate $P_{\text{pow}}(\mathcal{X})$; this is analogous to the non-malleability experiment in which the adversary chooses \mathcal{X} and the relation R takes \mathcal{X} as additional input. We call the side information here aux (as opposed to hint for non-malleability) in order to distinguish between the two primitives. In fact, in our construction aux uses hint as a subroutine but generates additional output.

Definition 4 (POWHF). A hash function $\mathcal{P} = (\text{POWK}, \text{POW}, \text{POWvf})$ is called a perfectly one-way hash function (with respect to predicate P_{pow} and probabilistic function aux) if it is collision resistant, and if for any PPTA $\mathcal{B} = (\mathcal{B}_d, \mathcal{B}_b)$, where \mathcal{B}_b has binary output, the following random variables are computationally indistinguishable:

$$\begin{array}{l|l}
 K \stackrel{\$}{\leftarrow} \text{POWK}(1^k); x \stackrel{\$}{\leftarrow} \mathcal{X}(1^k) & K \stackrel{\$}{\leftarrow} \text{POWK}(1^k) \\
 a_x \stackrel{\$}{\leftarrow} \text{aux}(K, x); y \stackrel{\$}{\leftarrow} \text{POW}(K, x) & (\mathcal{X}, \text{std}) \stackrel{\$}{\leftarrow} \mathcal{B}_d(K) \\
 b \stackrel{\$}{\leftarrow} \mathcal{B}_b(y, a_x, \text{std}) & x \stackrel{\$}{\leftarrow} \mathcal{X}(1^k), x' \stackrel{\$}{\leftarrow} \mathcal{X}(1^k) \\
 \text{return } (K, x, b) \text{ if } P_{\text{pow}}(\mathcal{X}) = 1 & a_x \stackrel{\$}{\leftarrow} \text{aux}(K, x); y' \stackrel{\$}{\leftarrow} \text{POW}(K, x') \\
 \text{else } \perp & b \stackrel{\$}{\leftarrow} \mathcal{B}_b(y', a_x, \text{std}) \\
 & \text{return } (K, x, b) \text{ if } P_{\text{pow}}(\mathcal{X}) = 1 \\
 & \text{else } \perp
 \end{array}$$

REMARK 1. As pointed out in [9][12] the definition only makes sense if aux is an uninvertible function of the input (such that finding the pre-image x from a_x is infeasible) and \mathcal{B}_x only outputs descriptions of well-spread distributions (with super-logarithmic min-entropy). Otherwise the notion is impossible to achieve. For generality, we do not restrict \mathcal{X} and aux explicitly here.

REMARK 2. Perfectly one-way hash functions (in the sense above) can be constructed from any one-way permutation [12][18] (for the uniform input distribution), any regular collision-resistant hash function [12] (for any distribution with fixed, super-logarithmic min-entropy), or under the decisional Diffie-Hellman assumption [9] (for the uniform distribution). Usually these general constructions are not known to be secure assuming arbitrary functions aux , yet for the particular function aux required by the application they can often be adapted accordingly. A concrete example is given in Section [6] in our discussion of the Bellare-Rogaway encryption scheme.

ON THE CHOICE OF THE RELATION CLASS. Recall that the definition of non-malleability is parametrized by a class of relations. As explained earlier in the paper, no non-malleable hash function for an arbitrary class exists (see Remark 1 after Definition [3]). In the sequel, we exhibit a class of relations for which we show how to construct non-malleable hash functions, and then present our provably secure construction.

Specifically, we consider the class of relations $\mathcal{R}_{\text{pred}}^{\text{rinfo}}$, parameterized by an optional function rinfo and which consists of all relations of the form $R(x, x^*) = P(x, P^*(\text{rinfo}(x), x^*))$, for all efficient predicates P, P^* . [5] The function $\text{rinfo}(x)$

⁵ Where we neglect the distribution \mathcal{X} as part of the relation’s input for the moment.

may be empty or consist of a small fraction of bits of x (e.g., up to logarithmically many), and should be interpreted as the information about x that may be used in evaluating the relation R . It is important that rinfo is an uninvertible function, as otherwise, if one could recover x from $\text{rinfo}(x)$, then $\mathcal{R}_{\text{pred}}^{\text{rinfo}}$ would comprise all efficient relations, $R(x, x^*) = P^*(x, x^*)$, and non-malleability with respect to this class, again, would not be achievable.

As an example consider the empty function rinfo such that $\mathcal{R}_{\text{pred}}$ consists of all relations $R(x, x^*) = P(x, P^*(x^*))$. This class of relations allows to check for instance that individual bits of x and x^* are complement of each other, i.e., if π_j denotes the projection onto the j -th bit then one sets $P^*(x^*) = \pi_j(x^*)$ and lets $P(x, P^*(x^*))$ output 1 if $\pi_j(x) \neq \pi_j(x^*)$. This example has also been used by Boldyreva and Fischlin [7] to show the necessity of non-malleability for OAEP, and to give an example of a perfectly one-way hash function that is malleable in the sense that flipping the first bit of an image produces a hash of the pre-image whose first bit is also flipped.

In the examples above rinfo has been the empty function. Of course, using non-trivial functions rinfo allows for additional relations and enriches the class $\mathcal{R}_{\text{pred}}^{\text{rinfo}}$. Consider for example a hash function H that is malleable in the sense that an adversary, given $H(K, r||m)$ for random $r \in \{0, 1\}^k$, can compute $H(K, r||m')$ for some $m' \neq m$. One way to capture that the two pre-images coincide on the first k bits is to set $\text{rinfo}(r||m) = r$ and to set $P^*(r, x^*) = 1$ if and only if r is the prefix of x^* . Since rinfo should be uninvertible, the function should rather return only a fraction of r , though. Similarly, one can see that the class $\mathcal{R}_{\text{pred}}^{\text{rinfo}}$ “captures” relations like $R(x, x^*) = 1$ iff $x \oplus x^* = \delta$ for some constant δ , and many other useful relations.

Finally, we note that each relation from the class also checks that the chosen input distribution \mathcal{X} “complies” with the eligible distributions from the underlying POWHF. That is, each relation also checks that the predicate $P_{\text{pow}}(\mathcal{X})$ of the POWHF is satisfied. The full relation $R(\mathcal{X}, x, x^*)$ then evaluates to 1 iff $P(x, P^*(\text{rinfo}(x), x^*)) = 1$ and $P_{\text{pow}}(\mathcal{X}) = 1$. More formally, for any predicate P_{pow} and uninvertible function rinfo we define the class of relations:

$$\mathcal{R}_{\text{pred}}^{\text{rinfo}, P_{\text{pow}}} = \left\{ R : \begin{array}{l} \text{there exist efficient (probabilistic) predicates } P, P^* \\ \text{such that } R(\mathcal{X}, x, x^*) = P(x, P^*(\text{rinfo}(x), x^*)) \wedge P_{\text{pow}}(\mathcal{X}) \end{array} \right\}.$$

Our construction also uses a simulation-sound zero-knowledge proof of knowledge $\Pi = (\text{CRS}, \text{P}, \text{V})$ for the NP-relation R_{pow} defined by:

$$R_{\text{pow}} = \{(K_{\text{pow}}||y_{\text{pow}}, x||r) : \text{POW}(K_{\text{pow}}, x; r) = y_{\text{pow}}\}.$$

which essentially says that one “knows” a pre-image of a hash value. Simulation-sound NIZK proofs of knowledge for such relations can be derived from trapdoor permutations [29,14]. We recall the definition of the former in the full version.

THE CONSTRUCTION AND ITS SECURITY. The following theorem captures the security of our construction.

Theorem 1. Let $\mathcal{P} = (\text{POWK}, \text{POW}, \text{POWf})$ be a perfectly one-way hash function with respect to P_{pow} and aux , where $\text{aux} = (\text{hint}, \text{rinfo})$ for probabilistic functions hint and rinfo . Let $\Pi = (\text{CRS}, \text{P}, \text{V})$ be a simulation-sound non-interactive zero-knowledge proof of knowledge for relation R_{pow} . Then the following hash function $\mathcal{H} = (\text{HK}, \text{H}, \text{HVf})$ is non-malleable with respect to hint and $\mathcal{R}_{\text{pred}}^{\text{rinfo}, P_{\text{pow}}}$:

- PPTA HK on input 1^k samples $K_{\text{pow}} \xleftarrow{\$} \text{POWK}(1^k)$ and $\text{crs} \xleftarrow{\$} \text{CRS}(1^k)$ and outputs $K = (K_{\text{pow}}, \text{crs})$. The associated domain D_K is given by $D_{K_{\text{pow}}}$.
- PPTA H on input K and $x \in D_K$ computes $y_{\text{pow}} \leftarrow \text{POW}(K_{\text{pow}}, x; r)$ for random $r \xleftarrow{\$} \text{RND}_{K_{\text{pow}}}$ as well as $\pi \xleftarrow{\$} \text{P}(\text{crs}, K_{\text{pow}} || y_{\text{pow}}, x || r)$. It outputs $y = (y_{\text{pow}}, \pi)$.
- PTA HVf for inputs $K = (K_{\text{pow}}, \text{crs})$, x and $y = (y_{\text{pow}}, \pi)$ outputs 1 if and only if $\text{POWf}(K_{\text{pow}}, x, y_{\text{pow}}) = 1$ and $\text{V}(\text{crs}, K_{\text{pow}} || y_{\text{pow}}, \pi) = 1$.

In addition, \mathcal{H} is collision-resistant.

Due to space limitations we provide the detailed proof in the full version of the paper [6].

REMARK 1. The malleability adversary has access to essentially two different sources of partial information about x : $\text{hint}(x)$ which it receives explicitly as input, and $\text{rinfo}(x)$ which it can use indirectly through the relation R . This motivates the requirement that \mathcal{P} be perfectly one-way with respect to partial information $\text{aux} = (\text{hint}, \text{rinfo})$.

REMARK 2. As mentioned after the definition of non-malleable hash functions, some applications (like the one about HMAC [19]) may require a stronger notion in which the adversary can adaptively generate distributions and receives the images, before deciding upon y^* . Our construction above can be extended to this case, assuming that the POWHF obeys a corresponding “adaptiveness” property and that the zero-knowledge proof of knowledge is multiple simulation-sound and multiple zero-knowledge. Such adaptively-secure POWHFs (for uniform distributions) can be built from one-way permutations [18] and suitable zero-knowledge proofs exist, assuming trapdoor permutations [29,14].

5 On the Complexity of Non-malleable Functions

In this section we discuss the existential complexity of non-malleable functions. We first indicate, via an oracle separation result, that deriving non-malleable hash and one-way functions via one-way permutations is infeasible. In the full version [6] we also discuss the relation between non-malleability and one-wayness.

5.1 On the Impossibility of Black-Box Reductions

We first show that, under reasonable conditions, there is no black-box reduction from non-malleable hash functions (which might not even be collision-resistant

but rather one-way only) to one-way permutations. For space reasons most of the proofs have been moved to the full version of the paper [6].

BLACK-BOX REDUCTIONS. In their seminal paper Impagliazzo and Rudich [24] have shown that some cryptographic primitives cannot be derived from other primitives, at least if the starting primitive is treated as a black box. Instead of separating primitives as in [24] here we follow the more accessible approach of Hsiao and Reyzin [23], giving a relaxed separation result with respect to black-box security reductions. We give a formalization of the oracle-based black-box separation approach that we use in the full version.

For our result we assume that the algorithms of the hash function \mathcal{H} are granted oracle access to a random permutation oracle \mathcal{P} (which is one-way, of course). A black-box reduction to \mathcal{P} is now an algorithm which, with oracle access to \mathcal{P} and a putative successful attacker \mathcal{A} on the non-malleability property, inverts \mathcal{P} with noticeable probability. Such an attacker \mathcal{A} may take advantage of another oracle \mathcal{O} (related to \mathcal{P}) which allows it to break the non-malleability but does not help to invert the one-way permutation \mathcal{P} . Since neither the construction nor the reduction are given access to \mathcal{O} , the reduction must be genuinely black-box.

DEFINING ORACLES \mathcal{P} AND \mathcal{O} . For now we let \mathcal{P} be a random permutation oracle which in particular is a one-way function. Below we show through de-randomization techniques that some fixed \mathcal{P} must also work. For our separation we let the side information of the non-malleable hash function include an image of the uniformly distributed input x under \mathcal{P} . More precisely, consider the function $\text{hint}_{\text{sep}}^{\mathcal{P}}$ which on input $(1^k, K, x)$ for random x computes $h_x = \mathcal{P}(0^k || x || \langle \text{HVf} \rangle || K)$ for the description $\langle \text{HVf} \rangle$ of the verification algorithm and finally outputs h_x .⁶

We next construct the oracle \mathcal{O} that helps to break non-malleability. The idea is that using \mathcal{O} it is possible to extract from the image y and “hint” h_x (described above) the pre-image x of y . Since the adversary gets y as input, but the simulator does not, the oracle is only helpful to the adversary. Note that breaking non-malleability means that no simulator of comparable complexity is able to approximate the success probability of $\mathcal{A}^{\mathcal{P}, \mathcal{O}}$ closely. To ensure that the simulator has the equal power as $\mathcal{A}^{\mathcal{P}, \mathcal{O}}$ we grant the simulator $\mathcal{S}^{\mathcal{P}, \mathcal{O}}$ therefore access to both oracles \mathcal{P}, \mathcal{O} .

Construction 1. *Let oracle \mathcal{O} take as input a parameter 1^k , an image y and a “hint” h_x . The oracle first finds the pre-image $z || x || \langle \text{HVf} \rangle || K$ of h_x under \mathcal{P} and verifies that $z = 0^k$; if not it immediately returns \perp . Else it checks that $\text{HVf}^{\mathcal{P}}(K, x, y) = 1$ and returns x if so (and outputs \perp otherwise).*

⁶ We note that the side information h_x does not reveal any essential information about x in the sense that one can show that, for any non-malleable hash function for the uniform input distribution and no side information at all, the hash function remains non-malleable with respect to h_x relative to the random permutation \mathcal{P} (but not relative to \mathcal{O} , of course). Also observe that the common strategy of using black-box simulators usually works for any side information, and in particular for the one here.

We show that \mathcal{O} does not help to invert \mathcal{P} , thus showing that relative to the oracles there still exists one-way permutations:

Proposition 1. *For any efficient algorithm $\mathcal{B}^{?,?}$, the probability that $\mathcal{B}^{\mathcal{P},\mathcal{O}}$ breaks the one-wayness of \mathcal{P} is negligible.*

In light of this lemma we conclude that there exists a particular \mathcal{P} that is hard to invert for all PPT adversaries with oracles \mathcal{P}, \mathcal{O} . The argument is the same as in [23]. For a fixed PPT adversary \mathcal{B} , we define the sequence of events (indexed by k) where \mathcal{B} inverts strings of length k with some good probability; for a suitable choice of parameters, the sum of the probabilities (over \mathcal{P}) of these events converges and by the first Borel-Cantelli lemma only finitely many of these events may occur, almost surely. Then taking the countable intersection over all PPT \mathcal{B} , we get that there is at least one \mathcal{P} with the desired property.

SEPARATION. We require some mild, technical conditions for our non-malleable hash function and the relation. Namely, we assume that

- the hash function is *non-trivial* meaning that it is infeasible to predict an image for uniformly distributed input over $\{0, 1\}^k$ (thus ruling out trivial examples like constant hash functions), and
- the relation class \mathcal{R} contains the relation R_{sep} which on input (\mathcal{X}, x, x^*) checks that \mathcal{X} is the uniform distribution on $\{0, 1\}^k$, and that $\text{parity}(x) = \bigoplus x_i = \text{parity}(x^*) = \bigoplus x_i^*$. Note that $R_{\text{sep}} \in \mathcal{R}_{\text{pred}}$ for our predicate-based relations, even for the empty function rinfo , and can thus be achieved in principle.

Theorem 2. *Let $\mathcal{H}^{\mathcal{P}} = (\text{HK}^{\mathcal{P}}, \text{H}^{\mathcal{P}}, \text{HVf}^{\mathcal{P}})$ be a non-trivial non-malleable hash function with respect to $\text{hint}_{\text{sep}}^{\mathcal{P}}$ and $\mathcal{R} \ni R_{\text{sep}}$. Then there exists an adversary $\mathcal{A}^{\mathcal{P},\mathcal{O}}$ that breaks non-malleability of $\mathcal{H}^{\mathcal{P}}$ (for any simulator $\mathcal{S}^{\mathcal{P},\mathcal{O}}$).*

Corollary 1. *There exists no black-box reduction from non-trivial non-malleable functions (with respect to $\text{hint}_{\text{sep}}^{\mathcal{P}}$ and $\mathcal{R} \ni R_{\text{sep}}$) to one-way permutations.*

At first glance it seems as if our result would transfer (after some minor modifications) to other non-malleable primitives like commitments. This is not the case. The oracle \mathcal{O} in our construction relies on the ability to check whether a pre-image x matches an image y (public verifiability of hash functions), while other primitives such as encryption $\mathcal{E}(m; r)$ and commitments $\text{Com}(m; r)$ use hidden randomness (which is not part of the input of function hint).

RELATING NON-MALLEABILITY AND PERFECT ONE-WAYNESS. In the full version we show that non-malleability implies a variant of perfect-one-wayness.

6 Applications

In this section we study the usefulness of our notion for cryptographic applications. As an example we show that when one of the two random oracles in the

mentioned encryption scheme proposed by Bellare and Rogaway in [4] is instantiated with a non-malleable hash function, the scheme remains IND-CCA secure. In addition, we argue that non-malleability is useful in preventing off-line computation attacks against a certain class of cryptographic puzzles.

INSTANTIATING RANDOM ORACLES. We start with recalling the scheme. Let \mathcal{F} be a family of trapdoor permutations and G, H be random oracles. The message space of the scheme $\text{BR}^{G,H}[\mathcal{F}] = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is the range of G . The key generation algorithm \mathcal{K} outputs a random \mathcal{F} -instance f and its inverse f^{-1} as the public and secret key, respectively. The encryption algorithm \mathcal{E} on inputs f and m picks random r in the domain of f (we assume that $r \in \{0, 1\}^k$) and outputs $(f(r), G(r) \oplus m, H(r||m))$. The decryption algorithm on inputs f^{-1} and (y, g, h) first computes $r \leftarrow f^{-1}(y)$, then $m \leftarrow g \oplus G(r)$, and outputs m iff $H(r||m) = h$. The scheme $\text{BR}^{G,H}[\mathcal{F}]$ is proven to be IND-CCA secure in the random oracle model assuming that \mathcal{F} is one-way [4].

Here we study the possibility of realizing the random oracle \mathcal{H} with an actual hash function family $\mathcal{H} = (\text{HK}, \text{H}, \text{HVf})$, a so-called *partial H -instantiation* of the scheme. More precisely, we modify the scheme so that the public key and secret key also contain a key $K \xleftarrow{\$} \text{HK}(1^k)$ specifying a function. Then \mathcal{E} computes $\text{H}(K, r||m)$ instead of $H(r||m)$, and \mathcal{D} computes $\text{HVf}(K, r||m, h)$ instead of checking that $H(r||m) = h$. We refer to this scheme as $\text{BR}^{G,\mathcal{H}}[\mathcal{F}]$. The following shows that functions that meet our notion of non-malleability are sufficient for a secure partial H -instantiation.

Before stating the sufficient conditions for security to hold, we fix some notation. Below we let the function $\text{rinfo}_{\text{BR}}(x) = \text{msb}_{k/2}(x)$ output the $k/2$ most significant bits of its input. The class of relations we require here for non-malleability is only a subset of the achievable class discussed in Section 4. Namely, we only require a relation of the form $R_{\text{BR}}(\mathcal{X}, x, x^*) = P^*(\text{rinfo}_{\text{BR}}(x), x^*) \wedge P_{\text{pow}}(\mathcal{X})$, where P_{pow} is the predicate that checks that \mathcal{X} is the canonical representation of the uniform distribution on the first k bits, and P^* is the predicate that simply verifies that $\text{msb}_{k/2}(x^*) = \text{rinfo}_{\text{BR}}(x)$. We choose this specific predicate R_{BR} so that it can check if $x = x^*$, while erring with only negligible probability, but still admit the construction of non-malleable hash functions.

Below we will require that the trapdoor permutation family is *msb $_{k/2}$ -partial one-way*, meaning that it is hard to compute the $k/2$ most significant bits of the random input r given a random instance f and $f(r)$ (cf. [21] for the formal definition). This is a rather mild assumption to impose on \mathcal{F} . For example, RSA was shown to be partial one-way under the RSA assumption in [21]. A general approach to construct such a partial one-way family \mathcal{F} is to define $f(r) = g(\text{msb}_{k/2}(r))||g(\text{lsb}_{k/2}(r))$ for a trapdoor permutation g [7].

⁷ In fact, this construction also has the useful property that $f(r)$ is still hard to invert, even if given $\text{msb}_{k/2}(r)$. Thus this trapdoor permutation is suitable for constructing POWHFs secure with respect to side information $(\text{msb}_{k/2}(r), f(r))$ and therefore, via our construction, non-malleable hash functions for side information $\text{hint}_{\text{BR}}(r) = f(r)$ and the relation R_{BR} . In other words, non-malleable hash functions for hint_{BR} and R_{BR} exist under common cryptographic assumptions.

We need one more technical detail before stating the theorem. We start with some hash function family $\mathcal{H} = (\text{HK}, \text{H}, \text{HVf})$ and trapdoor permutation family \mathcal{F} . We write $\mathcal{H} = (\text{HK}_{\mathcal{F}}, \text{H}, \text{HVf})$ for the modified hash function for which key generation outputs a random instance of \mathcal{F} along with the original hash key. Below we write hint_{BR} for the function that takes as input a key (K, f) and string x , and outputs $f(r)$, where r are the first k bits of the input x . We note the IND-CPA version of the scheme by Bellare and Rogaway was shown secure in the standard model by Canetti [9], assuming the hash function is a POWHF with respect to a similar hint function.

Theorem 3. *Let \mathcal{F} be an $\text{msb}_{k/2}$ -partial one-way trapdoor permutation family and let $\mathcal{H} = (\text{HK}_{\mathcal{F}}, \text{H}, \text{HVf})$ be a collision-resistant hash function which is non-malleable with respect to the function hint_{BR} and to the relation R_{BR} . Assume further that \mathcal{H} is a perfectly one-way hash function with respect to P_{pow} and hint_{BR} . Then $\text{BR}^{G, \mathcal{H}}[\mathcal{F}]$ is IND-CCA secure (in the RO model).*

REMARK. Although the non-malleability property of the hash implies that no partial information about pre-images is leaked (cf. the full version for a formal statement of this implication), the theorem above requires the hash to be perfectly one-way in the sense of Definition 4, which is a stronger requirement in general. The proof of the theorem is in the full version [6].

APPLICATION TO CRYPTOGRAPHIC PUZZLES. Cryptographic puzzles are a defense mechanism against denial of service attacks (DoS). The idea is that, before spending any resources for the execution of a session between a client and a server, the server requires the client to solve a puzzle. Since solving puzzles requires spending cycles, the use of puzzles prevents a malicious client to engage in a large number of sessions without spending itself a significant amount of resources. One desirable condition is that the server does not store any client-related state.

A simple construction for such puzzles proposed by Juels and Brainard [25] is based on any arbitrary one-way function $h : \{0, 1\}^l \rightarrow \{0, 1\}^l$. First, select at random $x \xleftarrow{\$} \{0, 1\}^l$ and compute $y = h(x)$. Then, a puzzle is given by the tuple $(x[1..l - k], y)$ consisting of the first $l - k$ bits of x together with y . To prove it solved the puzzle, the client has to return (x, y) . It can be easily seen that the construction above is not entirely satisfactory. In particular, it either fails against replay attacks —where the clients present the same puzzle-solution pair to the server— or the server needs to store all of the x 's used to compute the puzzles.

The solution proposed to mitigate the above problem is to compute x as $H(S, t)$, where S is some large bitstring known only to the server, and t is some bitstring that somehow “expires” after a certain amount of time (this can be for example the current system time). The puzzle is then given by $(t, x[1..l - k], y)$, where $y = h(x)$. A solution (or solved puzzle) is (t, x, y) which needs to satisfy the obvious equations, and moreover, t is not an expired bitstring.

In the setting above, non-malleability of H surfaces as an important property. If out of the first two elements $(t, H(S, t))$ of a puzzle solution the adversary can

efficiently construct $(t', H(S, t'))$ for $t' \neq t$, a string which has not yet expired, then the defense sketched above is rendered useless: the adversary can easily construct new puzzles (together with their solutions). Requiring that the function H is non-malleable with respect to the relation $R(s_1, s_2) = 1$ iff $s_1 = (S, t)$ and $s_2 = (S, t')$ for $t \neq t'$ is sufficient to prevent the above attack.

Acknowledgments

We thank all the reviewers for their comments. We also thank Vipul Goyal for stimulating discussions. Alexandra Boldyreva is supported in part by NSF CAREER award 0545659 and NSF Cyber Trust award 0831184. David Cash is supported in part by the aforementioned grants. Marc Fischlin is supported by the Emmy Noether Program Fi 940/2-1 of the German Research Foundation DFG. The work is partially funded by the European Commission through the ICT programme under Contract ICT-2007-216646 ECRYPT II.

References

1. Barak, B.: Constant-round coin-tossing with a man in the middle or realizing the shared random string model. In: FOCS 2002, pp. 345–355. IEEE, Los Alamitos (2002)
2. Barak, B., Prabhakaran, M., Sahai, A.: Concurrent non-malleable zero knowledge. In: FOCS 2005, pp. 563–572. IEEE, Los Alamitos (2005)
3. Bellare, M., Canetti, R., Krawczyk, H.: Keying hash functions for message authentication. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 1–15. Springer, Heidelberg (1996)
4. Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols. In: CCS 1993, pp. 62–73. ACM, New York (1993)
5. Bellare, M., Sahai, A.: Non-malleable encryption: Equivalence between two notions, and an indistinguishability-based characterization. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 519–536. Springer, Heidelberg (1999)
6. Boldyreva, A., Cash, D., Fischlin, M., Warinschi, B.: Foundations of non-malleable hash and one-way functions. Full version of this paper. Cryptology ePrint Archive, Report 2009/065 (2009)
7. Boldyreva, A., Fischlin, M.: Analysis of random-oracle instantiation scenarios for OAEP and other practical schemes. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 412–429. Springer, Heidelberg (2005)
8. Boldyreva, A., Fischlin, M.: On the security of OAEP. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 210–225. Springer, Heidelberg (2006)
9. Canetti, R.: Towards realizing random oracles: Hash functions that hide all partial information. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 455–469. Springer, Heidelberg (1997)
10. Canetti, R., Dakdouk, R.R.: Extractable perfectly one-way functions. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfssdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 449–460. Springer, Heidelberg (2008)

11. Canetti, R., Halevi, S., Steiner, M.: Mitigating dictionary attacks on password-protected local storage. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 160–179. Springer, Heidelberg (2006)
12. Canetti, R., Micciancio, D., Reingold, O.: Perfectly one-way probabilistic hash functions. In: STOC 1998, pp. 131–140. ACM, New York (1998)
13. Canetti, R., Varia, M.: Non-malleable obfuscation. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 73–90. Springer, Heidelberg (2009)
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. Damgård, I., Groth, J.: Non-interactive and reusable non-malleable commitment schemes. In: STOC 2003, pp. 426–437. ACM, New York (2003)
16. Di Crescenzo, G., Ishai, Y., Ostrovsky, R.: Non-interactive and non-malleable commitment. In: STOC 1998, pp. 141–150. ACM, New York (1998)
17. Dolev, D., Dwork, C., Naor, M.: Non-malleable cryptography. *SIAM Journal on Computing* 30(2), 391–437 (2000)
18. Fischlin, M.: Pseudorandom function tribe ensembles based on one-way permutations: Improvements and applications. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 429–444. Springer, Heidelberg (1999)
19. Fischlin, M.: Security of NMAC and HMAC based on non-malleability. In: Malkin, T.G. (ed.) RSA-CT 2008. LNCS, vol. 4964, pp. 138–154. Springer, Heidelberg (2008)
20. Fischlin, M., Fischlin, R.: Efficient non-malleable commitment schemes. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 414–432. Springer, Heidelberg (2000)
21. 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, pp. 260–274. Springer, Heidelberg (2001)
22. Goldreich, O.: *The foundations of cryptography, vol. 1*. Cambridge University Press, Cambridge (2004)
23. 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)
24. Impagliazzo, R., Rudich, S.: Limits on the provable consequences of one-way permutations. In: STOC 1989, pp. 44–61. ACM, New York (1989)
25. Juels, A., Brainard, J.: Client puzzles: A cryptographic countermeasure against connection depletion attacks. In: NDSS 1999, pp. 151–165 (1999)
26. Pandey, O., Pass, R., Vaikuntanathan, V.: Adaptive one-way functions and applications. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 57–74. Springer, Heidelberg (2008)
27. Pass, R., Rosen, A.: Concurrent non-malleable commitments. In: FOCS 2005, pp. 563–572. IEEE, Los Alamitos (2005)
28. Pass, R., Rosen, A.: New and improved constructions of non-malleable cryptographic protocols. In: STOC 2005, pp. 533–542. ACM Press, New York (2005)
29. Sahai, A.: Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In: FOCS 1999, p. 543. IEEE, Los Alamitos (1999)

Improved Cryptanalysis of Skein

Jean-Philippe Aumasson^{1,*}, Çağdaş Çalkı², Willi Meier^{1,**}, Onur Özen³,
Raphael C.-W. Phan⁴, and Kerem Varıcı^{5,***}

¹ FHNW, Klosterzelgstrasse 2, 5210 Windisch, Switzerland

² Middle East Technical University, Institute of Applied Mathematics,
06531 Ankara, Turkey

³ EPFL IC LACAL Station 14, 1015 Lausanne, Switzerland

⁴ Loughborough Uni, Electronic and Electrical Engineering, LE11 3TU, UK

⁵ K.U.Leuven, Dept. of Electrical Engineering, ESAT/SCD/COSIC and IBBT
Kasteelpark Arenberg 10, 3001 Heverlee, Belgium

Abstract. The hash function Skein is the submission of Ferguson et al. to the NIST Hash Competition, and is arguably a serious candidate for selection as SHA-3. This paper presents the first third-party analysis of Skein, with an extensive study of its main component: the block cipher Threefish. We notably investigate near collisions, distinguishers, impossible differentials, key recovery using related-key differential and boomerang attacks. In particular, we present near collisions on up to 17 rounds, an impossible differential on 21 rounds, a related-key boomerang distinguisher on 34 rounds, a known-related-key boomerang distinguisher on 35 rounds, and key recovery attacks on up to 32 rounds, out of 72 in total for Threefish-512. None of our attacks directly extends to the full Skein hash. However, the pseudorandomness of Threefish is required to validate the security proofs on Skein, and our results conclude that at least 36 rounds of Threefish seem required for optimal security guarantees.

1 Introduction

The hash function research scene has seen a surge of works since devastating attacks [1,2,3,4] on the two most deployed hash functions, MD5 and SHA-1. This led to a lack of confidence in the current U.S. (and de facto worldwide) hash standard, SHA-2 [5], because of its similarity with MD5 and SHA-1.

As a response to the potential risks of using SHA-2, the U.S. National Institute of Standards and Technology (NIST) launched a public competition—the NIST

* Supported by the Swiss National Science Foundation under project no. 113329.

** Supported by GEBERT RÜF STIFTUNG, project no. GRS-069/07.

*** This work was sponsored by the Research Fund K.U.Leuven, by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy) and by the European Commission through the ICT Programme under Contract ICT-2007-216676 (ECRYPT II). The information in this paper is provided as is, and no warranty is given or implied that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

Hash Competition—to select a new hash standard [6]. The new hash function, SHA-3, is expected to have at least the security of SHA-2, and to achieve this with significantly improved efficiency. By the deadline of October 2008, NIST received 64 submissions, 51 were accepted as first round candidates, and in July 2009 14 were selected as second round candidates, including Skein. Due to the critical role of hash functions in security protocols, this competition catches the attention not only from academia, but also from industry—with candidates from IBM, Hitachi, Intel, Sony—and from governmental organizations.

Skein [7] is the submission of Ferguson et al. to the NIST Hash Competition. According to its designers, it combines “speed, security, simplicity and a great deal of flexibility in a modular package that is easy to analyze” [7, p.i]. Skein supports three different internal state sizes (256-, 512-, and 1024-bit), and is one of the fastest contestants on 64-bit machines.

Skein is based on the “UBI (The Unique Block Iteration) chaining mode” that itself uses a compression function made out of the Threefish-512 block cipher. Below we give a brief top-down description of these components:

- **Skein** makes three invocations to the UBI mode with different *tags*: the first hashes the configuration block with a tag “Cfg”, the second hashes the message with a tag “Msg”, and the third hashes a null value with a tag “Out”.
- **UBI** mode hashes an arbitrary-length string by iterating invocations to a compression function, which takes as input a chaining value, a message block, and a tweak. The tweak encodes the number of bytes processed so far, and special flags for the first and the last block.
- **The compression function** inside the UBI mode is the Threefish-512 block cipher in MMO (Matyas-Meyer-Oseas) mode, i.e., from a chaining value h , a message block m , and a tweak t it returns $E_h(t, m) \oplus m$ as new chaining value.
- **Threefish** is a family of tweakable block ciphers based on a simple permutation of two 64-bit words: $\text{MIX}(x, y) = (x + y, (x + y) \oplus (y \lll R))$. Threefish-512 is the version of Threefish with 512-bit key and 512-bit blocks, and is used in the default version of Skein.

So far, no third-party cryptanalysis of Skein has been published, and the only cryptanalytic results are in its documentation [7, §9]. It describes a near collision on eight rounds for the compression function, a distinguisher for 17 rounds of Threefish, and it conjectures the existence of key recovery attacks on 24 to 27 rounds (depending on the internal state size). Furthermore, [7, §9] discusses the possibility of a trivial related-key boomerang attack on a modified Threefish, and concludes that it cannot work on the original version. A separate document [8] presents proofs of security for Skein when assuming that some of its components behave ideally (e.g., that Threefish is an ideal cipher).

This paper presents the first external analysis of Skein, with a focus on the main component of its default version: the block cipher Threefish-512. Table [9] summarizes our results.

Table 1. Summary of the known results on Threefish-512 (near collisions are for Threefish-512 in MMO mode, related-key boomerang attacks make use of four related-keys , “√” designates the present paper)

Rounds	Time	Memory	Type	Authors
8	1	–	511-bit near-collision	[7]
16	2 ⁶	–	459-bit near-collision	√
17	2 ²⁴	–	434-bit near-collision	√
17	2 ^{8.6}	–	related-key distinguisher*	[7]
21	2 ^{3.4}	–	related-key distinguisher	√
21	–	–	related-key impossible differential	√
25	?	–	related-key key recovery (conjectured)	[7]
25	2 ^{416.6}	–	related-key key recovery	√
26	2 ^{507.8}	–	related-key key recovery	√
32	2 ³¹²	2 ⁷¹	related-key boomerang key recovery	√
34	2 ³⁹⁸	–	related-key boomerang distinguisher	√
35	2 ⁴⁷⁸	–	known-related-key boomerang distinguisher	√

*: complexity deduced from the biases in [7, Tab.22].

The rest of the paper is organized as follows: §2 describes Threefish-512; §3 studies near-collisions for Skein’s compression function with a reduced Threefish-512; §4 describes impossible differentials; §5 discusses and improves the key-recovery attacks sketched in [7, §§9.3]. Finally, §6 uses the boomerang technique to describe our best distinguishers and key-recovery attacks on Threefish. §7 concludes.

2 Brief Description of Threefish-512

Threefish-512 works on 64-bit words, and we write their hexadecimal value in sans-serif font (e.g., 0123456789ABCDEF). The letter Δ stands for a difference in the most significant bit (MSB), i.e., Δ = 8000000000000000. Notations are the same as in the specification of Threefish [7, §§2.2]: a 512-bit plaintext block is parsed as eight words v_{0,0}, . . . , v_{0,7}, and is encrypted through N_r = 72 rounds, where round number d ∈ {0, . . . , N_r − 1} operates as follows:

1. If d ≡ 0 mod 4, add a subkey by setting e_{d,i} ← v_{d,i} + k_{d,i}, i = 0, . . . , 7, otherwise, just copy the state e_{d,i} ← v_{d,i}, i = 0, . . . , 7.
2. Set (f_{d,2i}, f_{d,2i+1}) ← MIX_{d,i}(e_{d,2i}, e_{d,2i+1}), i = 0, . . . , 3, where

$$\text{MIX}_{d,i}(x, y) = (x + y, (x + y) \oplus (y \lll R_{d,i})) ,$$

with R_{d,i} a rotation constant dependent on d and i.

3. Permute the state words:

$$\begin{aligned} v_{d+1,0} &\leftarrow f_{d,2} & v_{d+1,1} &\leftarrow f_{d,1} & v_{d+1,2} &\leftarrow f_{d,4} & v_{d+1,3} &\leftarrow f_{d,7} \\ v_{d+1,4} &\leftarrow f_{d,6} & v_{d+1,5} &\leftarrow f_{d,5} & v_{d+1,6} &\leftarrow f_{d,0} & v_{d+1,7} &\leftarrow f_{d,3} . \end{aligned}$$

After $N_r \equiv 0 \pmod 4$ rounds, the ciphertext is set to

$$(v_{N_r,0} + k_{N_r,0}), \dots, (v_{N_r,7} + k_{N_r,7}) .$$

The s -th keying (counting from zero, thus which occurs at round $d = 4s$) uses subkeys $k_{s,0}, \dots, k_{s,7}$. These are derived from the key k_0, \dots, k_7 and from the tweak t_0, t_1 as

$$\begin{array}{l|l} k_{s,0} \leftarrow k_{(s+0) \bmod 5} & k_{s,4} \leftarrow k_{(s+4) \bmod 5} \\ k_{s,1} \leftarrow k_{(s+1) \bmod 5} & k_{s,5} \leftarrow k_{(s+5) \bmod 5} + t_s \bmod 3 \\ k_{s,2} \leftarrow k_{(s+2) \bmod 5} & k_{s,6} \leftarrow k_{(s+6) \bmod 5} + t_{(s+1) \bmod 3} \\ k_{s,3} \leftarrow k_{(s+3) \bmod 5} & k_{s,7} \leftarrow k_{(s+7) \bmod 5} + s \end{array}$$

where $k_8 = 5555555555555555 \oplus \bigoplus_{i=0}^7 k_i$ and $t_2 = t_0 \oplus t_1$.

3 Near Collisions for the UBI Compression Function

We extend the analysis presented in [7, §9] to find near-collisions for the compression function of Skein’s UBI mode; [7, §9] exploits *local collisions*, i.e., collisions in the intermediate values of the state, which occur when particular differences are set in the key, the plaintext, and the tweak.

The compression function outputs $E_k(t, m) \oplus m$, where E is Threefish-512. Our strategy is simple: like in [7, §9], we prepend a four-round differential trail to the first local collision at round four so as to avoid differences until the 13-th round. Then, we follow the trail induced by the introduced difference.

The next two sections work out the details as follows:

- §3.1 shows how to adapt the differential trail found in [7, §9] when a 4-round trail is prepended.
- §3.2 describes the differential trails used and evaluates the probability that a random input conforms.
- §3.3 explains how to reduce the complexity of the attack by precomputing a single conforming pair for the first 4-round trail, and using some conditions to speed up the search.

3.1 Adapting Differences in the Key and the Tweak

In [7, §§9.3.4], Skein’s designers suggest to prepend a 4-round trail that leads to the difference $(0, 0, \dots, 0, \Delta)$, previously used for the 8-round collision. However, the technique as it is presented does not work. This is because the order of keyings is then shifted, and so the original difference in the key and in the tweak does not cancel the $(0, 0, \dots, 0, \Delta)$ difference at the second keying.

Therefore, for differences to vanish at the third keying, one needs a difference Δ in k_7 and t_0 , which gives a difference $(0, \dots, 0, \Delta)$ at the second keying, and $(0, 0, 0, 0, \Delta, 0, 0)$ after the fourth. The difference in the state after $(4 + 8)$ rounds is thus the same as originally after eight rounds. Note that, as observed in [7, §§9.4], at least seven keyings separate two vanishing keyings. See Table 2 for details.

Table 2. Details of the subkeys and of their differences, given a difference Δ in k_7 and t_0 (leading to Δ differences in k_8 and t_2)

s	d	$k_{s,0}$	$k_{s,1}$	$k_{s,2}$	$k_{s,3}$	$k_{s,4}$	$k_{s,5}$	$k_{s,6}$	$k_{s,7}$
		Differences							
0	0	k_0 0	k_1 0	k_2 0	k_3 0	k_4 0	$k_5 + t_0$ Δ	$k_6 + t_1$ 0	k_7 Δ
1	4	k_1 0	k_2 0	k_3 0	k_4 0	k_5 0	$k_6 + t_1$ 0	$k_7 + t_2$ 0	$k_8 + 1$ Δ
2	8	k_2 0	k_3 0	k_4 0	k_5 0	k_6 0	$k_7 + t_2$ 0	$k_8 + t_0$ 0	$k_0 + 2$ 0
3	12	k_3 0	k_4 0	k_5 0	k_6 0	k_7 Δ	$k_8 + t_0$ 0	$k_0 + t_1$ 0	$k_1 + 3$ 0
4	16	k_4 0	k_5 0	k_6 0	k_7 Δ	k_8 Δ	$k_0 + t_1$ 0	$k_1 + t_2$ Δ	$k_2 + 4$ 0
5	20	k_5 0	k_6 0	k_7 Δ	k_8 Δ	k_0 0	$k_1 + t_2$ Δ	$k_2 + t_0$ Δ	$k_3 + 5$ 0
6	24	k_6 0	k_7 Δ	k_8 Δ	k_0 0	k_1 0	$k_2 + t_0$ Δ	$k_3 + t_1$ 0	$k_4 + 6$ 0

3.2 Differential Trails

We now trace the difference when prepending four rounds, i.e., when the difference is in k_7 and in t_0 only (and in the plaintext).

4-Round Trail. To prepend four rounds and reach the difference $(0, \dots, 0, \Delta)$, one uses the trail provided in the full version [9] of this paper. The plaintext difference is modified by the first keying (the MSB differences in the sixth and eighth word vanish). The probability that a random input successfully crosses the 4-round differential trail is 2^{-33} (either forward or backward).

12-Round Trail. The second keying adds Δ to the last state word, making its difference vanish. The state remains free of any difference up to the fourth keying, after the twelfth round, which sets a difference Δ in the fifth word state. Table 3 presents the corresponding trail for up to the 17-th round. After 17 rounds, the weight becomes too large to obtain near collisions. On 16 rounds, adding the final keying and the feedforward, one obtains a collision on $512 - 53 = 459$ bits. Likewise, for 17 rounds, a collision can be found on $512 - 78 = 434$ bits.

3.3 Optimizing the Search

A direct application of the differential trails in the previous section gives a cost 2^{33} to cross the first four rounds; then, after the twelfth round,

Table 3. Differential trail (linearization) used for near collisions, of probability 2^{-24}

Rd	Difference				Pr
13	0000000000000000	0000000000000000	8000000000000000	0000000000000000	1
14	8000000000000000	0000000000000000	8000000000000000	0000000000000000	1
15	8000000000000000	8000000000000000	8000100000000000	8000000000000100	2^{-1}
16	0000010000000100	0000001000000000	0008010000000400	0000000400000000	2^{-5}
17	8008010400000400	0000010100000140	800A014004008400	A805018020000100	2^{-18}

- With 16 rounds: complexity is $2^{1+5} = 2^6$, so 2^{39} in total, for finding a collision over 459 bits.
- With 17 rounds: complexity is $2^{1+5+18} = 2^{24}$, so 2^{56} in total, for finding a collision over 434 bits.

A simple trick allows us to avoid the cost of crossing the first 4-round trail: note that the first keying adds $(k_5 + t_0)$ to the sixth state word, and $(k_6 + t_1)$ to the seventh; hence, given one conforming pair, one can modify k_5, k_6, t_0, t_1 while preserving the values of $(k_5 + t_0)$ and $(k_6 + t_1)$, and the new input will also follow the differential trail. It is thus sufficient to precompute a single conforming pair to avoid the cost due to the prepended rounds.

To carry out this precomputation efficiently, a considerable speedup of the 2^{33} complexity can be obtained by finding sufficient conditions to cross the first round with probability one (instead of 2^{-21}):

- A first set of conditions is on the words (v_{2i}, v_{2i+1}) : whenever there is a nonzero difference at a same offset, the bit should have a different value in the first and in the second word (otherwise carries induce additional differences).
- A second set of conditions concerns the differences that do not “collide”: one should ensure that no carry propagates from the leftmost bits.

In total, there are $13 + 8 = 21$ such conditions, which lets enough degrees of freedom to satisfy the subsequent differential tails. Using techniques like neutral bits [10], the probability may be reduced further, but the complexity 2^{12} is low enough for efficiently finding a conforming pair. By choosing inputs according to the above conditions, while being careful to avoid contradictions, we can find a pair that conforms within a few thousand trials (see Appendix A for an example).

We can now use this pair to search for near collisions. It suffices to pick random values for k_5 and k_6 , then set $t_0 = -k_5$ and $t_1 = -k_6$ to get a set of 2^{128} distinct inputs. Experiments were consistent with our analysis, and examples of near collisions are given in Appendix B.

3.4 Improved Distinguisher

Based on our trick to cross the first twelve rounds “for free”, we can improve the distinguisher suggested in [7]. This distinguisher exploited the observation of a bias $0.01 < \varepsilon \leq 0.05$ after 17 rounds (thus leading to a distinguisher requiring at least $1/0.05^2 \approx 400$ samples). [7] suggested to combine it with the prepending of four rounds, though no further details were given. Our observations show that with the adapted difference in the key and the tweak, a bias about 0.3 exists at the 385-th bit, after 21 rounds. We detected this bias using a frequency test similar to that in [11, §§2.1]. This directly gives a distinguisher on 21 rounds, and requiring only about $1/0.3^2 \approx 11$ samples.

4 Impossible Differentials

The *miss-in-the-middle* technique (a term coined by Biham et al. in [12]), was first applied by Knudsen [13] to construct a 5-round impossible differential for the block cipher DEAL. The idea was generalized by Biham et al. [12] to find impossible differentials for ciphers of any structure. The idea is as follows: Consider a cascade cipher $E = E^\beta \circ E^\alpha$ such that for E^α there exists a differential $(\Delta_{in}^\alpha \rightarrow \Delta_{out}^\alpha)$ and for $(E^\beta)^{-1}$ there exists a differential $(\Delta_{in}^\beta \rightarrow \Delta_{out}^\beta)$, both with *probability one*, where the equality is impossible $(\Delta_{out}^\alpha \neq \Delta_{out}^\beta)$. It follows that the differential $(\Delta_{in}^\alpha \rightarrow \Delta_{in}^\beta)$ cannot occur, for it requires $\Delta_{out}^\alpha = \Delta_{out}^\beta$. This technique can be extended to the related-key setting. For example, related-key impossible differentials were found for 8-round AES-192 [14, 15].

Below we first present probability-1 truncated differentials on the first 13 rounds (forward) and on the last seven rounds (backward) of 20-round Threefish-512. A “miss-in-the-middle” observation then allows us to deduce the existence of impossible differentials on 20 and 21 rounds.

4.1 Forward Differential

The first keying ($s = 0$) adds to the state $v_{0,i}, \dots, v_{0,7}$ the values $k_0, k_1, \dots, k_4, k_5 + t_0, k_6 + t_1, k_7$. Then, the second keying ($s = 1$) adds $k_1, \dots, k_5, k_6 + t_1, k_7 + t_2, k_8 + 1$. By setting a difference Δ in k_6, k_7, t_1 and in the plaintext $v_{0,7}$, we ensure that differences vanish in the first two keyings, and thus nonzero differences only appear after the eighth round, for third keying.

The third keying ($s = 2$) adds $k_2, \dots, k_6, k_7 + t_2, k_8 + t_0, k_0 + t_2$. Hence the difference Δ is introduced in $e_{8,4}$ only. It gives a difference Δ in $f_{8,4}, f_{8,5}$, thus in $v_{9,2}, v_{9,5}$. After the tenth round, the state $v_{10,\cdot}$ has the following difference with probability one.

```
8000000000000000  0000000000000000  8000000000000000  0000000000000000
0000000000000000  8000040000000000  0000000000000000  8000000000000000 .
```

After the twelfth round (before the fourth keying), the state $v_{12,\cdot}$ has again some differences that occur with probability one (the X differences are uncertain, that is, have probability strictly below one):

```
XXXXXXXXXX4000000  0000000002000000  XXXXXXXXXXXXX4000  0000000000000040
00000000000000000  XXXXXXXXXXXXXXX100  0000000000000000  XXXXXXXXXXX4000800 .
```

Given this class of differences, after the 13-th round (which starts by making the fourth keying) we have the class of differences

```
XXXXXXXXXXXXXXXX40  XXXXXXXXXXX2000000  XXXXXXXXXXXXXXX100  XXXXXXXXXXXXXXX10
XXXXXXXXXXXXXXXX800  XXXXXXXXXXXXXXXXXX  XXXXXXXXXXX2000000  XXXXXXXXXXXXXXX40 .
```

There are in total 92 bits with probability-1 differences between the 13-th and the 14-th round. These differences were empirically verified.

4.2 Backward Differential

The sixth keying ($s = 5$), which occurs after the 20-th round, returns the ciphertext

$$\begin{array}{l|l} c_0 = v_{20,0} + k_5 & c_4 = v_{20,4} + k_0 \\ c_1 = v_{20,1} + k_6 & c_5 = v_{20,5} + k_1 + t_2 \\ c_2 = v_{20,2} + k_7 & c_6 = v_{20,6} + k_2 + t_0 \\ c_3 = v_{20,3} + k_8 & c_7 = v_{20,7} + k_3 + 5 \end{array}$$

By setting a difference Δ in k_6, k_7, t_1 (like for the forward differential), and in the ciphertext words c_1, c_2, c_5 , we ensure that differences vanish in the sixth keying, and thus nonzero differences only appear after the 17-th round, when making the fifth keying (by computing backwards from the 20-th round).

The fifth keying ($s = 4$), after the 16-th round, subtracts from the state the values $k_4, \dots, k_8, k_0 + t_1, k_1 + t_2, k_2 + 4$. Hence, the difference Δ is introduced (backwards) in $v_{16,2}, v_{16,3}, v_{16,5}, v_{16,6}$. After inverting the 16-th round, we obtain with probability one the difference

```
XXXXXXXXX40000000  0000000040000000  0000000000000000  0000000000000000
8000000000000000  0000000000000000  XXXXXXXX10000000  0000000010000000 .
```

Finally, after inverting the 14-th round, we have the following difference with probability one:

```
XXXXXXXXXXXX8000  XXXXXXXXXXXXXXX8000  XXXXXXXXXXXXXXXXXX  XXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXX  XXXXXXXXXXX400000  XXXXXXXXXXX800000  XX50000000800000 .
```

In total there are 134 bits of difference with probability one between the 14-th and the 13-th round.

4.3 Miss-in-the-Middle

We showed that if there's a difference Δ in the key in k_6 and k_7 , and in the tweak in t_1 , then a difference Δ in the plaintext word $v_{0,7}$ propagates to give probability-1 differences after up to 13 rounds. Then we showed that for the same difference in the key and in the tweak, a difference Δ in the ciphertext

words c_1, c_2, c_5 guarantees (probability one) that between the 13-th and the 14-th rounds we also have probability-1 differences.

Looking for example at the first word of the state: the forward differential leads to a difference in the seventh bit, whereas the backward differential requires this bit to be unchanged. Therefore, it is impossible that a difference Δ in the plaintext $v_{0,7}$ leads to a difference Δ in c_1, c_2, c_5 with 20-round Threefish-512.

We can extend this impossible differential one more round: after the 20-th round and the sixth keying the state has only differences Δ in $e_{20,1}, e_{20,2}, e_{20,3}$. These differences always give the same difference after the 21-st round, because they are only in MSB's. This directly gives an impossible differential on 21 rounds of Threefish-512 (e.g., 21 out of 72). However contrary to the 20-round impossible differential, it is irrelevant to Threefish-512 with exactly $N_r = 21$ rounds, because of the final keying that occurs after the 21-st round (which makes some differences uncertain, because before the keying we have differences in non-MSB's).

5 Improved Key-Recovery Attacks

The documentation of Skein sketches key-recovery attacks on all Threefish versions, though the complexity is not studied. We analyzed these observations, and could find better attacks than conjectured by the Skein designers.

To optimize the attack strategy in [7, §§9.3], the attacker has to determine which key bits should be guessed. This is to minimize the noise over the bias after a partial inversion of the last rounds, and thus to minimize the complexity of the attack. The less key bits guessed, the better for the attacker (up to the bound of half the key bits). One can easily determine which key bits do not affect the bias when inverting one or two rounds. For example, two rounds after round 21 (where the bias occurs), the 385-th bit does not affect the second, third, fourth, and sixth state words. Hence, it is not affected by a wrong guess of the key words k_0, k_2, k_6 . The bias is slightly affected by erroneous guesses of k_3 (which modifies the last state word in the keying), but it is still large (about $0.12 \approx 2^{-3}$). It is thus sufficient to guess half the key (k_1, k_4, k_5, k_7) to be able to observe the bias.

Note that the cost of the prepended rounds depends on which key words are guessed: indeed, when guessing a word, one can adapt the corresponding plaintext word in order to satisfy the conditions of the differential. Here the non-guessed words imply a cost $2^{12+18} = 2^{30}$ to cross the first differential. The total cost of recovering the 512-bit key on 23 rounds is thus about $2^{30} \times 2^6 \times 2^{256} = 2^{292}$.

To attack more rounds, a more advanced search for the optimal set of bits to be guessed is likely to reduce the complexity of our attacks. For this, we used the same strategy as in the analysis of the Salsa20 and ChaCha stream ciphers [16]. Namely, we computed the *neutrality* of each key bit (i.e., the probability that flipping the bit preserves the difference), and we chose to guess the bits that affect the bias the most, using some threshold on their neutrality. More precisely, we sort key bits according to their neutrality, then filter them with respect to some

threshold value. According to [16]’s terminology, this corresponds to partitioning the key bits into “significant” and “non-significant” ones.

Recall that in §3.4 we observed a bias at the 385-th bit after $4 + 17$ rounds of Threefish-512. A key recovery attack on $21 + n$ rounds consists in guessing some key bits, inverting n rounds based on this guess, letting the other key bits be random, and observing a bias in that bit. Complexity is determined by the number of guessed bits and the value of the observed bias.

Inverting four rounds with all key bits whose neutrality is greater than 0.29 (we found 125 of those), we observe a bias 0.0365. Since some key bits are not guessed, and thus assumed random, some of the conditions to conform to the first round’s differential cannot be controlled. There are eight such additional conditions, which means that the 4-round initial differential will be followed with probability 2^{-12-8} . Since our bias approximately equals to $2^{-4.8}$, and since we need to guess $512 - 125$ key bits, the overall complexity of the attack on 25-round Threefish-512 is about $2^{12+8} \times 2^{2 \times 4.8} \times 2^{387} = 2^{416.6}$. Below we give the mask corresponding to the 125 *non-guessed* bits, for each key word:

```
0000070060FFF836  0040030021FFFC0E  803C02F03FFFF83F  001001001603C006
00780E30007F000E  0000000000000000  0000000000000000  007001800E03F801 .
```

We can apply the same method on 26 rounds: with a neutrality threshold 0.17 we obtain 30 “significant” key bits, and we observe a bias about 0.017 when all of them are random. The non-guessed bits give two additional conditions for the first 4-round differential. In total, the complexity of the attack is thus about $2^{12+2} \times 2^{2 \times 5.9} \times 2^{482} = 2^{507.8}$. Memory requirements are negligible.

6 Boomerang Attacks

Boomerang attacks were introduced by Wagner and first applied to block ciphers [17]. Roughly speaking, in boomerang attacks one uses two short differential trails rather than a long one to exploit the efficiency of the former trails. Let E denote the encryption function of Threefish. View E as a cascade of four subciphers

$$E = E^\omega \circ E^\gamma \circ E^\beta \circ E^\alpha, \quad (1)$$

so that E is composed of a core $E' = E^\gamma \circ E^\beta$ sandwiched by rounds E^α and E^ω . The boomerang distinguisher is generally described for E' only, but for key recovery attacks on Threefish we need to generalize the attack to the construction in Eq. (1).

Recall that in related-key attacks, one assumes that the attacker can query the cipher with other keys that have some specified relation with the original key. This relation is often an XOR-difference. A related-key differential is thus a triplet $(\Delta_{\text{in}}, \Delta_{\text{out}}, \Delta_k)$, associated with the probability

$$\Pr_{k,m} [E_k(m) \oplus E_{k \oplus \Delta_k}(m \oplus \Delta_{\text{in}}) = \Delta_{\text{out}}] = p.$$

Here, Δ_{in} and Δ_{out} are the input and output differences, Δ_k is the key difference, and p the probability of the differential.

For (related-key) boomerang attacks based on four related-keys, one exploits two short related-key differentials: $(\Delta_{\text{in}}^\beta, \Delta_{\text{out}}^\beta, \Delta_k^\beta)$ for E^β , of probability p and $(\Delta_{\text{in}}^\gamma, \Delta_{\text{out}}^\gamma, \Delta_k^\gamma)$ for E^γ , of probability q .

A distinguisher then works as follows:

1. Pick a random plaintext m_1 and form $m_2 = m_1 \oplus \Delta_{\text{in}}^\beta$.
2. Obtain $c_1 = E'_k(m_1)$ and $c_2 = E'_{k \oplus \Delta_k^\beta}(m_2)$.
3. Set $c_3 = c_1 \oplus \Delta_{\text{out}}^\gamma$ and $c_4 = c_2 \oplus \Delta_{\text{out}}^\gamma$.
4. Obtain $m_3 = E'_{k \oplus \Delta_k^\gamma}(c_3)$ and $m_4 = E'_{k \oplus \Delta_k^\beta \oplus \Delta_k^\gamma}(c_4)$.
5. Check $m_3 \oplus m_4 = \Delta_{\text{in}}^\beta$.

For an ideal cipher, the final equality is expected to hold with probability 2^{-n} where n is the block length. The probability of the related-key boomerang distinguisher, on the other hand, is approximately p^2q^2 (see [17,18,19,20] for details).

Note that the boomerang attack can be generalized to exploit multiple differentials. The success probability then becomes $\hat{p}^2\hat{q}^2$, where \hat{p} and \hat{q} are the square roots of the sums of the squares of the differentials exploited [1].

6.1 Exploiting Nonlinear Differentials

Differentials are often found via linearization, i.e., assuming that integer additions behave as XOR's. One then evaluates the probability of the differential with respect to the probability that each active addition behaves as XOR. This probability equals 2^{-w} , where w is the Hamming weight of the logical OR of the two difference masks, excluding the MSB.

Yet one is not limited to such “linear” differentials, and the best differential—in terms of probability—is not necessarily a linearization, as illustrated by the work of Lipmaa and Moriai [21]: for integer addition, they presented efficient algorithms for computing the probability of any differential, and for finding the optimal differential. The problem was later studied using formal rational series with linear representation [22].

We used the algorithms in [21] to find the differentials of our boomerang attacks. Note that it is not guaranteed that our trails are optimal, for the combination of local optimal differential trails (with respect to their probability) may contribute to a faster increase of the weight than (non-necessarily optimal) linear differentials. Yet our best differentials are not completely linear.

6.2 Related-Key Distinguishers

Like in our previous attacks, we exploit differences in the key and in the plaintext that vanish until the twelfth round (both for the forward and backward differentials). Then, we follow a nonlinear differential trail until the middle of

¹ Throughout the paper, our differentials do not make use of this multiple differential approach. One can further improve upon the differentials provided in this work by using this technique.

the cipher, i.e., between the 16-th and 17-th rounds. Our differential trail for E^β has probability $p = 2^{-86}$, and the one for E^γ has probability 2^{-113} , leading to a boomerang distinguisher on 34 rounds requiring about $(pq)^{-2} = 2^{398}$ trials (see full version [9]). Note that for the second part, MSB differences are set in the key words k_2 and k_3 , and in the tweak words t_0 and t_1 (thus giving no difference in the seventh subkey).

6.3 Known-Related-Key Distinguishers

Although the standard notion of distinguisher requires a secret (key), the notion of *known-key distinguisher* [23] is also relevant to set apart a block cipher from a randomly chosen permutation. Moreover, when a block cipher is used within a compression function, as Threefish is, known-key distinguishers may lead to distinguishers for the hash function because all inputs are known to the adversary. If differences in the keys are used, we shall thus talk of *known-related-key distinguisher*. An example of such distinguisher is the exhibition of input/output pairs that have some specific relation, as presented in [23] for seven rounds of AES-128. Here, we shall consider tuples $(m_1, m_2, m_3, m_4, c_1, c_2, c_3, c_4)$ that satisfy the boomerang property.

To build a known-related-key boomerang distinguisher on Threefish, we consider the decryption function, i.e., we start from the end of the cipher: when the key is known, the attacker can easily find a ciphertext that conforms to the first differential (e.g., to the weight-83 differential at round 35), which we could verify experimentally. In other words, the final differential (including the differences caused by the final key) is “free” when launching the boomerang. When it returns, however, the 2^{83} factor cannot be avoided if we want to exactly follow the differential (which is not strictly necessary to run a distinguisher). We thus obtain a distinguisher on 35-round Threefish-512 with complexity 2^{83} times that of the related-key distinguisher on 34 rounds, that is, approximately 2^{478} encryptions.

Several tricks may be used to obtain a similar distinguisher at a reduced cost. For example, observing that the first and fourth (resp. second and third) MIX functions of round 34 depend only on the first and second (resp. third and fourth) MIX’s of round 35, one can speed-up the search for inputs conforming to the first two rounds of the boomerang.

6.4 Extension to Key-Recovery

We now show how to build a key-recovery attack on top of a boomerang distinguisher for 32-round Threefish-512. We present some preliminary observations before describing and analyzing our attack.

Using notations of Eq. (11): E^β starts from the beginning and ends after the key addition in round 16, and E^γ starts from round 17 and ends just before the key addition after round 32. Our goal is to recover the *last subkey*. Restricted to 32 rounds, the boomerang distinguisher has probabilities $p = 2^{-86}$ for E^β and $q = 2^{-37}$ for E^γ , yielding an overall boomerang probability of $p^2q^2 = 2^{-246}$. We now introduce some notions required to facilitate the analysis of our attack.

Definition 1 (CS-sequence). Let δ be a 64-bit word of Hamming weight $0 \leq w \leq 64$. The CS-sequence of δ is

$$S_\delta = (|s_0|, |s_1|, \dots, |s_{w-1}|),$$

where $|s_i|$ is the bit length of the i -th block of consecutive zeros in δ finishing with a one.

For example, for $\delta = 1000010402000000$ we have

$$\delta = \underbrace{0001}_{s_0} \underbrace{0000 \ 0000 \ 0000 \ 0000 \ 0001}_{s_1} \underbrace{0000 \ 0100}_{s_2} \underbrace{0000 \ 0010}_{s_3} 0000 \ \dots \ 0000,$$

and so the CS-sequence of δ is $S_\delta = (|s_0|, |s_1|, |s_2|, |s_3|) = (4, 20, 6, 9)$.

The following result is extensively used in the key recovery attack using boomerang distinguisher, whose proof is provided in the full version of this paper [9].

Theorem 1. The number of possible differences N_δ after addition of difference δ with zero or $\Delta = 8000000000000000$ difference modulo 2^{64} can be directly computed from the CS-sequence of δ as

$$N_\delta = |s_0| \sum_{(k_1, k_2, \dots, k_{w-1}) \in \{0,1\}^{w-1}} \prod_{i=1}^{w-1} |s_i|^{k_i}.$$

For instance, if $\delta = 1000010402000000$ then

$$\begin{aligned} N_\delta &= 4 \sum_{(k_1, k_2, k_3) \in \{0,1\}^3} (20^{k_1} \times 6^{k_2} \times 9^{k_3}) \\ &= 4 \times (1 + 9 + 6 + (6 \times 9) + 20 + (20 \times 6) + (20 \times 9) + (20 \times 9 \times 6)) \\ &= 4 \times 1470 = 5880. \end{aligned}$$

Applying Theorem [1], we have the number of possible output differences caused by Δ_{out}^γ just after the key addition followed by the related-key boomerang distinguisher for Threefish-512 is approximately 2^{62} . We obtain this number by multiplying the number of possibilities for each word of the state (see Table [4]).

Table 4. Number of possible output differences after the key addition in Threefish-512, for each word. Multiplying these numbers, we obtain in total approximately 2^{62} possible differences.

$v_{32,i}$	$S_{\Delta_{out}^\gamma}$	$N_{\Delta_{out}^\gamma}$
$v_{32,0}$	(24, 15)	384
$v_{32,1}$	(32)	32
$v_{32,2}$	(0)	1
$v_{32,3}$	(4, 20, 6, 9)	5880
$v_{32,4}$	(1)	1
$v_{32,5}$	(13, 2, 9, 2, 12, 11, 5)	957840
$v_{32,6}$	(13, 11, 30)	4836
$v_{32,7}$	(14)	14

The Attack. Our attack works in three steps: in the first step, we obtain quartets satisfying the related-key boomerang relation; in the second, we recover the partial key by using the possible right quartets obtained from the first step; the last step is the brute force search of the rest of the key. The attack works as follows.

1. Find right quartets

for $i = 1, \dots, 2^{248}$

- Generate a random unique pair of chosen plaintexts (m_1^i, m_2^i) with an Δ_{in}^β difference and encrypt each plaintext with key k^1 and k^2 (having Δ_k^β difference) respectively to obtain the corresponding ciphertexts (c_1^i, c_2^i) .
- for $j = 1, \dots, 2^{62}$
 - Set $c_3^{i,j} = c_1^i \oplus \Delta_{\text{out}}^{\prime,j}$ where $\Delta_{\text{out}}^{\prime,j}$ is set to the j -th possible difference caused by $\Delta_{\text{out}}^\gamma$.
 - Decrypt $c_3^{i,j}$ with k^3 and obtain the plaintext $m_3^{i,j}$.
 - Store the values $c_3^{i,j}$ and $m_3^{i,j}$.
- for $k = 1, \dots, 2^{62}$
 - Set $c_4^{i,k} = c_2^i \oplus \Delta_{\text{out}}^{\prime,k}$ where $\Delta_{\text{out}}^{\prime,k}$ is set to the k -th possible difference caused by $\Delta_{\text{out}}^\gamma$.
 - Decrypt $c_4^{i,k}$ with k^4 and obtain the plaintext $m_4^{i,k}$.
 - Calculate $M = m_4^{i,k} \oplus \Delta_{\text{in}}^\beta$ and check whether M exists among the stored values of $m_3^{i,j}$. If this is the case, store the possible right quartet.
- Free the memory allocated for the stored values of (possibly wrong) $c_3^{i,j}$ and $m_3^{i,j}$. Increment i .

2. Recover the partial key

For each ciphertext word having a nonzero difference of a (possibly) right quartet (c_1, c_2, c_3, c_4) guess the corresponding output whitening key word $k_{\omega,l}$ for $l = 0, 3, 5, 6$, and check

$$(c_{1,l} - k_{\omega,l}) \oplus (c_{3,l} - k_{\omega,l}^2) = (c_{2,l} - k_{\omega,l}^3) \oplus (c_{4,l} - k_{\omega,l}^4) = \Delta_{\text{out},l}^\gamma,$$

where $k_{\omega,l}^2 = k_{\omega,l} \oplus \Delta_{k,l}^\gamma$ and $k_{\omega,l}^3 = k_{\omega,l}^4 \oplus \Delta_{k,l}^\gamma$. If this is the case, store this $k_{\omega,l}$.

3. Recover the full key

Run an exhaustive search of the remaining bits of the subkey.

Complexity Analysis. The goal of step 1 is to find enough quartets satisfying the related-key boomerang trail. For each distinct 2^{248} plaintext-ciphertext pairs

(m_1, m_2) and (c_1, c_2) , we correspondingly generate 2^{62} new plaintext-ciphertext pairs (m_3, c_3) and (m_4, c_4) by using the possible number of output differences given in Table 4. We know that a right quartet has to satisfy one of the possible number of output differences Δ'_{out} ; hence it is guaranteed to find the right quartet once it exists as we consider all possible combinations. Note that, increasing the number of quartets in that manner does not increase the number of right quartets, the reason simply being the newly generated plaintext-ciphertext pairs (m_3, c_3) and (m_4, c_4) can only have one *root* right plaintext-ciphertext pair (m_1, m_2) and (c_1, c_2) . Therefore, the expected number of right quartets is $2^{248} \cdot 2^{-246} = 2^2$. On the other hand, we expect $2^{372} \cdot 2^{-512} = 2^{-140}$ additional false quartets.

The first loop at step 1 requires 2^{62} reduced round Threefish decryptions and approximately $2^{70.5}$ bytes of memory. The second loop can be implemented independently and requires 2^{62} reduced round Threefish decryptions and 2^{62} memory accesses. On the other hand, we need additional memory complexity of $2^{69.5}$ bytes for storing Δ'_{out} values. Therefore, the overall complexity of the first step is bounded by 2^{312} reduced round Threefish decryptions and about 2^{71} bytes of memory. Note that the memory requirement for the surviving quartets is negligible.

Step 2 tries to recover the last subkey by using the quartets that passed the previous step. For each surviving quartet, we guess 64 bits of the final key at each word, decrypt one round and check the output difference $\Delta'_{\text{out},l}$. As the computation at each word can be processed independently, the overall complexity of this step is dominated by the previous step.

The probability that a false combination of quartets and key bits is counted in step 2 is upper bounded by 2^{-2w_l} where w_l is the minimum hamming weight of the corresponding output difference $\Delta'_{\text{out},l}$. Therefore, the right key is suggested $4 + 2^{-140} \cdot 2^{-2w_l} \approx 4$ times by the right and additional false quartets. On the other hand, a wrong key is expected to be hit $4 \cdot 2^{-2w_l} + 2^{-140} \cdot 2^{-2w_l} \approx 2^{-2}$ times. Note that this only holds for the words having an XOR difference of hamming weight two, for the rest the number of hits is strictly less than 2^{-2} . We can use Poisson distribution to calculate the success rate of our attack. For an expected number of 2^{-2} , the probability that a wrong key is suggested at most once is 0.97. However, the probability that the right key is suggested more than once is more than 0.90. Therefore, we can find the right key or at least eliminate most of the keys with high probability. The complexity of the rest of the attack is dominated by the first step.

7 Conclusion

We applied a wide range of attack strategies to the core algorithm of Skein (the block cipher Threefish-512), culminating with a distinguisher on 35-round Threefish-512, and a key-recovery attack on 32 rounds. Other versions of Threefish are vulnerable to similar attack strategies (for example, our related-key boomerang distinguisher works on up to 33 rounds of Threefish-256). To the

best of our knowledge, this is the first application of a key-recovery boomerang attack to an “ARX” algorithm, and also the first application of the boomerang technique to known-key distinguishers.

Despite its relative simplicity, the full Threefish seems to resist state-of-the-art cryptanalytic techniques. Its balanced “ARX” structure combined with large words provides a good balance between diffusion and non-linearity, and avoids any particular structure exploitable by attackers. Using attacks on Threefish to attack the hash function Skein (or its compression function) seems difficult, because of the rather complex mode of operation of Skein. Although none of our attacks directly extends to the hash mode, the pseudorandomness of Threefish is required to validate the security proofs on Skein. Hence, 36 or more rounds of Threefish seem to be required to provide optimal security.

Future works might apply the recent rebound attack [24] to Threefish, although it looks difficult to combine it with the trick discussed in §3.1; this forces the attacker to use specific differences. Another research direction relates to optimization of boomerang known- or chosen-key distinguishers.

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. Cannière, C.D., Rechberger, C.: Finding SHA-1 Characteristics: General Results and Applications. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 1–20. Springer, Heidelberg (2006)
4. 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)
5. NIST: FIPS 180-2 Secure Hash Standard (2002)
6. NIST: Cryptographic Hash Competition, <http://www.nist.gov/hash-competition>
7. Ferguson, N., Lucks, S., Schneier, B., Whiting, D., Bellare, M., Kohno, T., Callas, J., Walker, J.: The Skein Hash Function Family. Submission to NIST (2008)
8. Bellare, M., Kohno, T., Lucks, S., Ferguson, N., Schneier, B., Whiting, D., Callas, J., Walker, J.: Provable Security Support for the Skein Hash Family, <http://www.skein-hash.info/sites/default/files/skein-proofs.pdf> (Draft) (February 18, 2009)
9. Aumasson, J.P., Çalk, Ç., Meier, W., Özen, O., Phan, R.C.W., Varici, K.: Improved Cryptanalysis of Skein. In: Cryptology ePrint Archive (2009)
10. Biham, E., Chen, R.: Near-Collisions of SHA-0. In: Franklin, M.K. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 290–305. Springer, Heidelberg (2004)
11. NIST: SP 800-22, A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications (2001)
12. Biham, E., Biryukov, A., Shamir, A.: Miss in the Middle Attacks on IDEA and Khufu. In: Knudsen, L.R. (ed.) FSE 1999. LNCS, vol. 1636, pp. 124–138. Springer, Heidelberg (1999)

13. Knudsen, L.R.: DEAL - a 128-bit Block Cipher. Technical Report 151, University of Bergen (1998); submitted as an AES candidate
14. Jakimoski, G., Desmedt, Y.: Related-Key Differential Cryptanalysis of 192-bit Key AES Variants. In: Matsui, M., Zuccherato, R.J. (eds.) SAC 2003. LNCS, vol. 3006, pp. 208–221. Springer, Heidelberg (2004)
15. Biham, E., Dunkelman, O., Keller, N.: Related-Key Impossible Differential Attacks on 8-Round AES-192. In: Pointcheval, D. (ed.) CT-RSA 2006. LNCS, vol. 3860, pp. 21–33. Springer, Heidelberg (2006)
16. Aumasson, J.P., Fischer, S., Khazaei, S., Meier, W., Rechberger, C.: New Features of Latin Dances: Analysis of Salsa, ChaCha, and Rumba. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 470–488. Springer, Heidelberg (2008)
17. Wagner, D.: The Boomerang Attack. In: Knudsen, L.R. (ed.) FSE 1999. LNCS, vol. 1636, pp. 156–170. Springer, Heidelberg (1999)
18. Biham, E., Dunkelman, O., Keller, N.: Related-Key Boomerang and Rectangle Attacks. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 507–525. Springer, Heidelberg (2005)
19. Biham, E., Dunkelman, O., Keller, N.: New Combined Attacks on Block Ciphers. In: Gilbert, H., Handschuh, H. (eds.) FSE 2005. LNCS, vol. 3557, pp. 126–144. Springer, Heidelberg (2005)
20. Dunkelman, O.: Techniques for Cryptanalysis of Block Ciphers. PhD thesis, Technion, Israel (February 2006)
21. Lipmaa, H., Moriai, S.: Efficient Algorithms for Computing Differential Properties of Addition. In: Matsui, M. (ed.) FSE 2001. LNCS, vol. 2355, pp. 336–350. Springer, Heidelberg (2002)
22. Lipmaa, H., Wallén, J., Dumas, P.: On the Additive Differential Probability of Exclusive-Or. In: Roy, B., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 317–331. Springer, Heidelberg (2004)
23. Knudsen, L.R., Rijmen, V.: Known-Key Distinguishers for Some Block Ciphers. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 315–324. Springer, Heidelberg (2007)
24. Mendel, F., Rechberger, C., Schläffer, M., Thomsen, S.S.: The Rebound Attack: Cryptanalysis of Reduced Whirlpool and Grøstl. In: Dunkelman, O. (ed.) FSE 2009. LNCS, vol. 5665, pp. 260–276. Springer, Heidelberg (2009)

A Conforming Pair for the 4-Round Differential

When the key and the tweak are zero, the following two message blocks conform to the differential described in §3.2

E979D16280002004	32B29AE900000000	D921590E00000000	5771CC9000000400
A62FF22800000000	484B245000040080	D3BEA4E800008010	7A72784300000000
A971917200100020	72B2DAE980002004	DD61588E01000400	5331CC1000000000
A62FF22800040090	C84B245000000000	D1BEA4E800000000	FA72784300008010

B Examples of Near Collisions

We provide an example of near collision on 459 bits for the reduced compression function of Skein's UBI mode. Both inputs always have $k_0 = \dots = k_4 = k_7 = 0$, and

$$k_5 = \text{CODECODECODECODE}.$$

On the 16-round compression function, the first input has message block

```
E979D16280002004  32B29AE900000000  D921590E00000000  5771CC9000000400
A62FF22800000000  484B245000040080  D3BEA4E800008010  7A72784300000000
```

and

$$k_6 = 6B9B2C1000000000 \quad t_0 = 3F213F213F213F22 \quad t_1 = 9464D3F000000000$$

The second input has message block

```
A971917200100020  72B2DAE980002004  DD61588E01000400  5331CC1000000000
A62FF22800040090  C84B245000000000  D1BEA4E800000000  FA72784300008010
```

and

$$k_6 = 6B9B2C1000000000 \quad t_0 = \text{BF213F213F213F22} \quad t_1 = 9464D3F000000000$$

The corresponding digests are respectively

```
2A6DE91E3E8CDE3B  BADAF451F59D3145  7C298A43FB73463F  D8309C9E9E2594D5
35431D226A2022E3  0EA42EB45F9EEEB9  DF038EECD6504300  588A798B1266D67A
```

and

```
6A65A80EBE9CFF1F  FADAB450759D1141  78618AC3FA73463F  5C709C1A9E2590D5
B5431D226A242273  8EAE2FF45B9A6A39  5D038EECD650C310  D08E788B1266576A
```

Linearization Framework for Collision Attacks: Application to CubeHash and MD6^{*}

Eric Brier¹, Shahram Khazaei², Willi Meier³, and Thomas Peyrin¹

¹ Ingenico, France

² EPFL, Switzerland

³ FHNW, Switzerland

Abstract. In this paper, an improved differential cryptanalysis framework for finding collisions in hash functions is provided. Its principle is based on linearization of compression functions in order to find low weight differential characteristics as initiated by Chabaud and Joux. This is formalized and refined however in several ways: for the problem of finding a conforming message pair whose differential trail follows a linear trail, a condition function is introduced so that finding a collision is equivalent to finding a preimage of the zero vector under the condition function. Then, the dependency table concept shows how much influence every input bit of the condition function has on each output bit. Careful analysis of the dependency table reveals degrees of freedom that can be exploited in accelerated preimage reconstruction under the condition function. These concepts are applied to an in-depth collision analysis of reduced-round versions of the two SHA-3 candidates CubeHash and MD6, and are demonstrated to give by far the best currently known collision attacks on these SHA-3 candidates.

Keywords: Hash functions, collisions, differential attack, SHA-3, CubeHash and MD6.

1 Introduction

Hash functions are important cryptographic primitives that find applications in many areas including digital signatures and commitment schemes. A hash function is a transformation which maps a variable-length input to a fixed-size output, called message *digest*. One expects a hash function to possess several security properties, one of which is *collision resistance*. Being collision resistant, informally means that it is *hard* to find two distinct inputs which map to the same output value. In practice, the hash functions are mostly built from a fixed input size compression function, *e.g.* the renowned Merkle-Damgård construction. To any hash function, no matter how it has been designed, we can always attribute fixed input size compression functions, such that a collision for a derived compression function results in a direct collision for the hash function itself. This way, firstly we are working with fixed input size compression functions rather than varying input size ones, secondly we can attribute compression functions to those hash functions which are not explicitly based on a fixed input size compression function, and

^{*} An extended version is available at <http://eprint.iacr.org/2009/382>

thirdly we can derive different compression functions from a hash function. For example multi-block collision attack [27] benefits from the third point. Our task is to find two messages for an attributed compression function such that their digests are preferably equal (a collision) or differ in only a few bits (a near-collision).

The goal of this work is to *revisit* collision-finding methods using linearization of the compression function in order to find differential characteristics for the compression function. This method was initiated by Chabaud and Joux on SHA-0 [11] and was later extended and applied to SHA-1 by Rijmen and Oswald [26]. The recent attack on EnRUPT by Indesteege and Preneel [15] is another application of the method. In particular, in [26] it was observed that the codewords of a linear code, which are defined through a linearized version of the compression function, can be used to identify differential paths leading to a collision for the compression function itself. This method was later extended by Pramstaller et al. [25] with the general conclusion that finding high probability differential paths is related to low weight codewords of the attributed linear code. In this paper we further investigate this issue.

The first contribution of our work is to present a more concrete and tangible relation between the linearization and differential paths. In the case that modular addition is the only involved nonlinear operation, our results can be stated as follows. Given the parity check matrix \mathcal{H} of a linear code, and two matrices \mathcal{A} and \mathcal{B} , find a codeword Δ such that $\mathcal{A}\Delta \vee \mathcal{B}\Delta$ is of low weight. This is clearly different from the problem of finding a low weight codeword Δ . We then consider the problem of finding a conforming message pair for a given differential trail for a certain linear approximation of the compression function. We show that the problem of finding conforming pairs can be reformulated as finding preimages of zero under a function which we call the *condition* function. We then define the concept of *dependency table* which shows how much influence every input bit of the condition function has on each output bit. By carefully analyzing the dependency table, we are able to profit not only from neutral bits [7] but also from probabilistic neutral bits [2] in a backtracking search algorithm, similar to [6, 24, 14]. This contributes to a better understanding of freedom degrees uses.

We consider compression functions working with n -bit words. In particular, we focus on those using modular addition of n -bit words as the only nonlinear operation. The incorporated linear operations are XOR, shift and rotation of n -bit words in practice. We present our framework in detail for these constructions by approximating modular addition with XOR. We demonstrate its validity by applying it on reduced-round variants of CubeHash [4] (one of the NIST SHA-3 [22] competitors) which uses addition, XOR and rotation. CubeHash instances are parametrized by two parameters r and b and are denoted by `CubeHash- r/b` which process b message bytes per iteration; each iteration is composed of r rounds. Although we can not break the original submission `CubeHash-8/1`, we provide real collisions for the much weaker variants `CubeHash-3/64` and `CubeHash-4/48`. Interestingly, we show that neither the more secure variants `CubeHash-6/16` and `CubeHash-7/64` do provide the desired collision security for 512-bit digests by providing theoretical attacks with complexities $2^{222.6}$ and $2^{203.0}$ respectively; nor that `CubeHash-6/4` with 512-bit digests is second-preimage resistant, as with probability 2^{-478} a second preimage can be produced by only one hash evaluation. Our theory can be easily generalized to arbitrary nonlinear operations. We discuss

this issue and as an application we provide collision attacks on 16 rounds of MD6 [23]. MD6 is another SHA-3 candidate whose original number of rounds varies from 80 to 168 when the digest size ranges from 160 to 512 bits.

2 Linear Differential Cryptanalysis

Let's consider a compression function $H = \text{Compress}(M, V)$ which works with n -bit words and maps an m -bit message M and a v -bit initial value V into an h -bit output H . Our aim is to find a collision for such compression functions with a randomly given initial value V . In this section we consider *modular-addition-based* Compress functions, that is, they use only modular additions in addition to linear transformations. This includes the family of AXR (Addition-XOR-Rotation) hash functions which are based on these three operations. In Section 5 we generalize our framework to other family of compression functions. For these Compress functions, we are looking for two messages with a difference Δ that result in a collision. In particular we are interested in a Δ for which two randomly chosen messages with this difference lead to a collision with a high probability for a randomly chosen initial value. For modular-addition-based Compress functions, we consider a linearized version for which all additions are replaced by XOR. This is a common linear approximation of addition. Other possible linear approximations of modular addition, which are less addressed in literature, can be considered according to our generalization of Section 5. As addition was the only nonlinear operation, we now have a linear function which we call $\text{Compress}_{\text{lin}}$. Since $\text{Compress}_{\text{lin}}(M, V) \oplus \text{Compress}_{\text{lin}}(M \oplus \Delta, V) = \text{Compress}_{\text{lin}}(\Delta, 0)$ is independent of the value of V , we adopt the notation $\text{Compress}_{\text{lin}}(M) = \text{Compress}_{\text{lin}}(M, 0)$ instead. Let Δ be an element of the kernel of the linearized compression function, *i.e.* $\text{Compress}_{\text{lin}}(\Delta) = 0$. We are interested in the probability $\Pr\{\text{Compress}(M, V) \oplus \text{Compress}(M \oplus \Delta, V) = 0\}$ for a random M and V . In the following we present an algorithm which computes this probability, called the *raw (or bulk) probability*.

2.1 Computing the Raw Probability

We consider a general n -bit vector $x = (x_0, \dots, x_{n-1})$ as an n -bit integer denoted by the same variable, *i.e.* $x = \sum_{i=0}^{n-1} x_i 2^i$. The Hamming weight of a binary vector or an integer x , $\text{wt}(x)$, is the number of its nonzero elements, *i.e.* $\text{wt}(x) = \sum_{i=0}^{n-1} x_i$. We use $+$ for modular addition of words and \oplus, \vee and \wedge for bit-wise XOR, OR and AND logical operations between words as well as vectors. We use the following lemma which is a special case of the problem of computing $\Pr\{((A \oplus \alpha) + (B \oplus \beta)) \oplus (A + B) = \gamma\}$ where α, β and γ are constants and A and B are independent and uniform random variables, all of them being n -bit words. Lipmaa and Moriai have presented an efficient algorithm for computing this probability [19]. We are interested in the case $\gamma = \alpha \oplus \beta$ for which the desired probability has a simple closed form.

Lemma 1. $\Pr\{((A \oplus \alpha) + (B \oplus \beta)) \oplus (A + B) = \alpha \oplus \beta\} = 2^{-\text{wt}((\alpha \vee \beta) \wedge (2^{n-1} - 1))}$.

Lemma 1 gives us the probability that modular addition behaves like the XOR operation. As $\text{Compress}_{\text{lin}}$ approximates Compress by replacing modular addition with

XOR, we can then devise a simple algorithm to compute (estimate) the raw probability $\Pr\{\text{Compress}(M, V) \oplus \text{Compress}(M \oplus \Delta, V) = \text{Compress}_{\text{lin}}(\Delta)\}$. Let's first introduce some notation.

Notation. Let n_{add} denote the number of additions which `Compress` uses in total. In the course of evaluation of $\text{Compress}(M, V)$, let the two addends of the i -th addition ($1 \leq i \leq n_{\text{add}}$) be denoted by $A^i(M, V)$ and $B^i(M, V)$, for which the ordering is not important. The value $C^i(M, V) = (A^i(M, V) + B^i(M, V)) \oplus A^i(M, V) \oplus B^i(M, V)$ is then called the carry word of the i -th addition. Similarly, in the course of evaluation of $\text{Compress}_{\text{lin}}(\Delta)$, denote the two inputs of the i -th linearized addition by $\alpha^i(\Delta)$ and $\beta^i(\Delta)$ in which the ordering is the same as that for A^i and B^i . We define five more functions $\mathbf{A}(M, V)$, $\mathbf{B}(M, V)$, $\mathbf{C}(M, V)$, $\alpha(\Delta)$ and $\beta(\Delta)$ with $(n - 1)n_{\text{add}}$ -bit outputs. These functions are defined as the concatenation of all the n_{add} relevant words excluding their MSBs. For example $\mathbf{A}(M, V)$ and $\alpha(\Delta)$ are respectively the concatenation of the n_{add} words $(A^1(M, V), \dots, A^{n_{\text{add}}}(M, V))$ and $(\alpha^1(\Delta), \dots, \alpha^{n_{\text{add}}}(\Delta))$ excluding the MSBs.

Using this notation, the raw probability can be simply estimated as follows.

Lemma 2. *Let `Compress` be a modular-addition-based compression function. Then for any message difference Δ and for random values M and V , $p_{\Delta} = 2^{-\text{wt}(\alpha(\Delta) \vee \beta(\Delta))}$ is a lower bound for $\Pr\{\text{Compress}(M, V) \oplus \text{Compress}(M \oplus \Delta, V) = \text{Compress}_{\text{lin}}(\Delta)\}$.*

Proof. We start with the following definition.

Definition 1. *We say that a message M (for a given V) conforms to (or follows) the trail of Δ iff¹*

$$((A^i \oplus \alpha^i) + (B^i \oplus \beta^i)) \oplus (A^i + B^i) = \alpha^i \oplus \beta^i, \text{ for } 1 \leq i \leq n_{\text{add}}, \quad (1)$$

where A^i , B^i , α^i and β^i are shortened forms for $A^i(M, V)$, $B^i(M, V)$, $\alpha^i(\Delta)$ and $\beta^i(\Delta)$, respectively.

It is not difficult to prove that under some reasonable independence assumptions p_{Δ} , which we call conforming probability, is the probability that a random message M follows the trail of Δ . This is a direct corollary of Lemma 1 and Definition 1. The exact proof can be done by induction on n_{add} , the number of additions in the compression function. Due to other possible non-conforming pairs that start from message difference Δ and lead to output difference $\text{Compress}_{\text{lin}}(\Delta)$, p_{Δ} is a lower bound for the desired probability in the lemma. □

If $\text{Compress}_{\text{lin}}(\Delta)$ is of low Hamming weight, we get a near collision in the output. The interesting Δ 's for collision search are those which belong to the kernel of $\text{Compress}_{\text{lin}}$, i.e. those that satisfy $\text{Compress}_{\text{lin}}(\Delta) = 0$. From now on, we assume that $\Delta \neq 0$ is in the kernel of $\text{Compress}_{\text{lin}}$, hence looking for collisions. According to Lemma 2, one needs to try around $1/p_{\Delta}$ random message pairs in order to find a collision which conforms to the trail of Δ . However in a random search it is better not to restrict oneself

¹ If and only if.

to the conforming messages as a collision at the end is all we want. Since p_Δ is a lower bound for the probability of getting a collision for a message pair with difference Δ , we might get a collision sooner. In Section 3 we explain a method which might find a *conforming* message by avoiding random search.

2.2 Link with Coding Theory

We would like to conclude this section with a note on the relation between the following two problems: (I) finding low-weight codewords of a linear code, (II) finding a high probability linear differential path. Since the functions $\text{Compress}_{\text{in}}(\Delta)$, $\alpha(\Delta)$ and $\beta(\Delta)$ are linear, we consider Δ as a column vector and attribute three matrices \mathcal{H} , \mathcal{A} and \mathcal{B} to these three transformations, respectively. In other words we have $\text{Compress}_{\text{in}}(\Delta) = \mathcal{H}\Delta$, $\alpha(\Delta) = \mathcal{A}\Delta$ and $\beta(\Delta) = \mathcal{B}\Delta$. We then call \mathcal{H} the *parity check matrix* of the compression function.

Based on an initial work by Chabaud and Joux [11], the link between these two problems has been discussed by Rijmen and Oswald in [26] and by Pramstaller et al. in [25] with the general conclusion that finding highly probable differential paths is related to low weight codewords of the attributed linear code. In fact the relation between these two problems is more delicate. For problem (I), we are provided with the parity check matrix \mathcal{H} of a linear code for which a codeword Δ satisfies the relation $\mathcal{H}\Delta = 0$. Then, we are supposed to find a low-weight nonzero codeword Δ . This problem is believed to be hard and there are some heuristic approaches for it, see [10] for example. For problem (II), however, we are given three matrices \mathcal{H} , \mathcal{A} and \mathcal{B} and need to find a nonzero Δ such that $\mathcal{H}\Delta = 0$ and $\mathcal{A}\Delta \vee \mathcal{B}\Delta$ is of low-weight, see Lemma 2. Nevertheless, low-weight codewords Δ 's matrix \mathcal{H} might be good candidates for providing low-weight $\mathcal{A}\Delta \vee \mathcal{B}\Delta$, *i.e.* differential paths with high probability p_Δ . In particular, this approach is promising if these three matrices are sparse.

3 Finding a Conforming Message Pair Efficiently

The methods that are used to accelerate the finding of a message which satisfies some requirements are referred to as *freedom degrees use* in the literature. This includes message modifications [27], neutral bits [7], boomerang attacks [16, 20], tunnels [18] and submarine modifications [21]. In this section we show that the problem of finding conforming message pairs can be reformulated as finding preimages of zero under a function which we call the *condition* function. One can carefully analyze the condition function to see how freedom degrees might be used in efficient preimage reconstruction. Our method is based on measuring the amount of influence which every input bit has on each output bit of the condition function. We introduce the dependency tables to distinguish the influential bits, from those which have no influence or are less influential. In other words, in case the condition function does not mix its input bits well, we profit not only from neutral bits [7] but also from probabilistic neutral bits [2]. This is achieved by devising a backtracking search algorithm, similar to [6, 24, 14], based on the dependency table.

3.1 Condition Function

Let's assume that we have a differential path for the message difference Δ which holds with probability $p_\Delta = 2^{-y}$. According to Lemma 2 we have $y = \text{wt}(\alpha(\Delta) \vee \beta(\Delta))$. In this section we show that, given an initial value V , the problem of finding a conforming message pair such that $\text{Compress}(M, V) \oplus \text{Compress}(M \oplus \Delta, V) = 0$ can be translated into finding a message M such that $\text{Condition}_\Delta(M, V) = 0$. Here $Y = \text{Condition}_\Delta(M, V)$ is a function which maps m -bit message M and v -bit initial value V into y -bit output Y . In other words, the problem is reduced to finding a preimage of zero under the Condition_Δ function. As we will see it is quite probable that not every output bit of the Condition function depends on all the message input bits. By taking a good strategy, this property enables us to find the preimages under this function more efficiently than random search. But of course, we are only interested in preimages of zero. In order to explain how we derive the function Condition from Compress we first present a quite easy-to-prove lemma. We recall that the *carry word* of two words A and B is defined as $C = (A + B) \oplus A \oplus B$.

Lemma 3. *Let A and B be two n -bit words and C represent their carry word. Let $\delta = 2^i$ for $0 \leq i \leq n - 2$. Then,*

$$((A \oplus \delta) + (B \oplus \delta)) = (A + B) \Leftrightarrow A_i \oplus B_i \oplus 1 = 0, \quad (2)$$

$$(A + (B \oplus \delta)) = (A + B) \oplus \delta \Leftrightarrow A_i \oplus C_i = 0, \quad (3)$$

and similarly

$$((A \oplus \delta) + B) = (A + B) \oplus \delta \Leftrightarrow B_i \oplus C_i = 0. \quad (4)$$

For a given difference Δ , a message M and an initial value V , let $\mathbf{A}_k, \mathbf{B}_k, \mathbf{C}_k, \alpha_k$ and $\beta_k, 0 \leq k < (n - 1)n_{\text{add}}$, respectively denote the k -th bit of the output vectors of the functions $\mathbf{A}(M, V), \mathbf{B}(M, V), \mathbf{C}(M, V), \alpha(\Delta)$ and $\beta(\Delta)$, as defined in Section 2.1. Let $\{i_0, \dots, i_{y-1}\}, 0 \leq i_0 < i_1 < \dots < i_{y-1} < (n - 1)n_{\text{add}}$ be the positions of 1's in the vector $\alpha \vee \beta$. We define the function $Y = \text{Condition}_\Delta(M, V)$ as:

$$Y_j = \begin{cases} \mathbf{A}_{i_j} \oplus \mathbf{B}_{i_j} \oplus 1 & \text{if } (\alpha_{i_j}, \beta_{i_j}) = (1, 1), \\ \mathbf{A}_{i_j} \oplus \mathbf{C}_{i_j} & \text{if } (\alpha_{i_j}, \beta_{i_j}) = (0, 1), \\ \mathbf{B}_{i_j} \oplus \mathbf{C}_{i_j} & \text{if } (\alpha_{i_j}, \beta_{i_j}) = (1, 0), \end{cases} \quad (5)$$

for $j = 0, 1, \dots, y - 1$. This equation can be equivalently written as equation 7.

Proposition 1. *For a given V and Δ , a message M conforms to the trail of Δ iff $\text{Condition}_\Delta(M, V) = 0$.*

3.2 Dependency Table for Freedom Degrees Use

For simplicity and generality, let's adopt the notation $F(M, V) = \text{Condition}_\Delta(M, V)$ in this section. Assume that we are given a general function $Y = F(M, V)$ which maps m message bits and v initial value bits into y output bits. Our goal is to reconstruct preimages of a particular output, for example the zero vector, efficiently. More precisely,

we want to find V and M such that $F(M, V) = 0$. If F mixes its input bits very well, one needs to try about 2^y random inputs in order to find one mapping to zero. However, in some special cases, not every input bit of F affects every output bit. Consider an ideal situation where message bits and output bits can be divided into ℓ and $\ell + 1$ disjoint subsets respectively as $\bigcup_{i=1}^{\ell} \mathcal{M}_i$ and $\bigcup_{i=0}^{\ell} \mathcal{Y}_i$ such that the output bits \mathcal{Y}_j ($0 \leq j \leq \ell$) only depend on the input bits $\bigcup_{i=1}^j \mathcal{M}_i$ and the initial value V . In other words, once we know the initial value V , we can determine the output part \mathcal{Y}_0 . If we know the initial value V and the input portion \mathcal{M}_1 , the output part \mathcal{Y}_1 is then known and so on. Refer to Section 6 to see the partitioning of a condition function related to MD6. This property of F suggests Algorithm 1 for finding a preimage of zero. Algorithm 1 is a backtracking search algorithm in essence, similar to [6, 24, 14], and in practice is implemented recursively with a tree-based search to avoid memory requirements. The values $q_0, q_1, \dots, q_{\ell}$ are the parameters of the algorithm to be determined later. To discuss the complexity of the algorithm, let $|\mathcal{M}_i|$ and $|\mathcal{Y}_i|$ denote the cardinality of \mathcal{M}_i and \mathcal{Y}_i respectively, where $|\mathcal{Y}_0| \geq 0$ and $|\mathcal{Y}_i| \geq 1$ for $1 \leq i \leq \ell$. We consider an *ideal behavior* of F for which each output part depends in a complex way on all the variables that it depends on. Thus, the output segment changes independently and uniformly at random if we change any part of the relevant input bits.

Algorithm 1. Preimage finding

Require: $q_0, q_1, \dots, q_{\ell}$

Ensure: some preimage of zero under F

- 0: Choose 2^{q_0} initial values at random and keep those $2^{q'_1}$ candidates which make \mathcal{Y}_0 part null.
 - 1: For each candidate, choose $2^{q_1 - q'_1}$ values for \mathcal{M}_1 and keep those $2^{q'_2}$ ones making \mathcal{Y}_1 null.
 - 2: For each candidate, choose $2^{q_2 - q'_2}$ values for \mathcal{M}_2 and keep those $2^{q'_3}$ ones making \mathcal{Y}_2 null.
 - ⋮
 - i : For each candidate, choose $2^{q_i - q'_i}$ values for \mathcal{M}_i and keep those $2^{q'_{i+1}}$ ones making \mathcal{Y}_i null.
 - ⋮
 - ℓ : For each candidate, choose $2^{q_{\ell} - q'_{\ell}}$ values for \mathcal{M}_{ℓ} and keep those $2^{q'_{\ell+1}}$ final candidates making \mathcal{Y}_{ℓ} null.
-

To analyze the algorithm, we need to compute the optimal values for q_0, \dots, q_{ℓ} . The time complexity of the algorithm is $\sum_{i=0}^{\ell} 2^{q_i}$ as at each step 2^{q_i} values are examined. The algorithm is successful if we have at least one candidate left at the end, *i.e.* $q'_{\ell+1} \geq 0$. We have $q'_{i+1} \approx q_i - |\mathcal{Y}_i|$, coming from the fact that at the i -th step 2^{q_i} values are examined each of which makes the portion \mathcal{Y}_i of the output null with probability $2^{-|\mathcal{Y}_i|}$. Note that we have the restrictions $q_i - q'_i \leq |\mathcal{M}_i|$ and $0 \leq q'_i$ since we have $|\mathcal{M}_i|$ bits of freedom degree at the i -th step and we require at least one surviving candidate after each step. Hence, the optimal values for q_i 's can be recursively computed as $q_{i-1} = |\mathcal{Y}_{i-1}| + \max(0, q_i - |\mathcal{M}_i|)$ for $i = \ell, \ell - 1, \dots, 1$ with $q_{\ell} = |\mathcal{Y}_{\ell}|$.

How can we determine the partitions \mathcal{M}_i and \mathcal{Y}_i for a given function F ? We propose the following heuristic method for determining the message and output partitions in practice. We first construct a $y \times m$ binary valued table T called *dependency table*.

The entry $T_{i,j}$, $0 \leq i \leq m - 1$ and $0 \leq j \leq y - 1$, is set to one iff the j -th output bit is highly affected by the i -th message bit. To this end we empirically measure the probability that changing the i -th message bit changes the j -th output bit. The probability is computed over random initial values and messages. We then set $T_{i,j}$ to one iff this probability is greater than a threshold $0 \leq th < 0.5$, for example $th = 0.3$. We then call Algorithm 2.

Algorithm 2. Message and output partitioning

Require: Dependency table T

Ensure: ℓ , message partitions $\mathcal{M}_1, \dots, \mathcal{M}_\ell$ and output partitions $\mathcal{Y}_0, \dots, \mathcal{Y}_\ell$.

- 1: Put all the output bits j in \mathcal{Y}_0 for which the row j of T is all-zero.
 - 2: Delete all the all-zero rows from T .
 - 3: $\ell := 0$;
 - 4: **while** T is not empty **do**
 - 5: $\ell := \ell + 1$;
 - 6: **repeat**
 - 7: Determine the column i in T which has the highest number of 1's and delete it from T .
 - 8: Put the message bit which corresponds to the deleted column i into the set \mathcal{M}_ℓ .
 - 9: **until** There is at least one all-zero row in T OR T becomes empty
 - 10: If T is empty set \mathcal{Y}_ℓ to those output bits which are not in $\bigcup_{i=0}^{\ell-1} \mathcal{Y}_i$ and stop.
 - 11: Put all the output bits j in \mathcal{Y}_ℓ for which the corresponding row of T is all-zero.
 - 12: Delete all the all-zero rows from T .
 - 13: **end while**
-

In practice, once we make a partitioning for a given function using the above method, there are two issues which may cause the ideal behavior assumption to be violated:

1. The message segments $\mathcal{M}_1, \dots, \mathcal{M}_i$ do not have full influence on \mathcal{Y}_i ,
2. The message segments $\mathcal{M}_{i+1}, \dots, \mathcal{M}_\ell$ have influence on $\mathcal{Y}_0, \dots, \mathcal{Y}_i$.

With regard to the first issue, we ideally would like that all the message segments $\mathcal{M}_1, \mathcal{M}_2, \dots, \mathcal{M}_i$ as well as the initial value V have full influence on the output part \mathcal{Y}_i . In practice the effect of the last few message segments $\mathcal{M}_{i-d_i}, \dots, \mathcal{M}_i$ (for some small integer d_i) is more important, though. Theoretical analysis of deviation from this requirement may not be easy. However, with some tweaks on the tree-based (backtracking) search algorithm, we may overcome this effect in practice. For example if the message segment \mathcal{M}_{i-1} does not have a great influence on the output segment \mathcal{Y}_i , we may decide to backtrack two steps at depth i , instead of one (the default value). The reason is as follows. Imagine that you are at depth i of the tree and you are trying to adjust the i -th message segment \mathcal{M}_i , to make the output segment \mathcal{Y}_i null. If after trying about $2^{\min(|\mathcal{M}_i|, |\mathcal{Y}_i|)}$ choices for the i -th message block, you do not find an appropriate one, you will go one step backward and choose another choice for the $(i-1)$ -st message segment \mathcal{M}_{i-1} ; you will then go one step forward once you have successfully adjusted the $(i-1)$ -st message segment. If \mathcal{M}_{i-1} has no effect on \mathcal{Y}_i , this would be useless and increase our search cost at this node. Hence it would be appropriate if we backtrack two steps at this depth. In general, we may tweak our tree-based search by setting the number of steps which we want to backtrack at each depth.

In contrast, the theoretical analysis of the second issue is easy. Ideally, we would like that the message segments $\mathcal{M}_i, \dots, \mathcal{M}_\ell$ have no influence on the output segments $\mathcal{Y}_0, \dots, \mathcal{Y}_{i-1}$. The smaller the threshold value th is chosen, the less the influence would be. Let 2^{-p_i} , $1 \leq i \leq \ell$, denote the probability that changing the message segment \mathcal{M}_i does not change any bit from the output segments $\mathcal{Y}_0, \dots, \mathcal{Y}_{i-1}$. The probability is computed over random initial values and messages, and a random non-zero difference in the message segment \mathcal{M}_i . Algorithm [1](#) must be reanalyzed in order to recompute the optimal values for q_0, \dots, q_ℓ . Algorithm [1](#) also needs to be slightly changed by reassuring that at step i , all the output segments $\mathcal{Y}_0, \dots, \mathcal{Y}_{i-1}$ remain null. The time complexity of the algorithm is still $\sum_{i=0}^{\ell} 2^{q_i}$ and it is successful if at least one surviving candidate is left at the end, *i.e.* $q_{\ell+1} \geq 0$. However, here we set $q'_{i+1} \approx q_i - |\mathcal{Y}_i| - p_i$. This comes from the fact that at the i -th step 2^{q_i} values are examined each of which makes the portion \mathcal{Y}_i of the output null with probability $2^{-|\mathcal{Y}_i|}$ and keeping the previously set output segments $\mathcal{Y}_0, \dots, \mathcal{Y}_{i-1}$ null with probability 2^{-p_i} (we assume these two events are independent). Here, our restrictions are again $0 \leq q'_i$ and $q_i - q'_i \leq |\mathcal{M}_i|$. Hence, the optimal values for q_i 's can be recursively computed as $q_{i-1} = p_{i-1} + |\mathcal{Y}_{i-1}| + \max(0, q_i - |\mathcal{M}_i|)$ for $i = \ell, \ell - 1, \dots, 1$ with $q_\ell = |\mathcal{Y}_\ell|$.

Remark 1. When working with functions with a huge number of input bits, it might be appropriate to consider the m -bit message M as a string of u -bit units instead of bits. For example one can take $u = 8$ and work with bytes. We then use the notation $M = (M[0], \dots, M[m/u-1])$ (assuming u divides m) where $M[i] = (M_{iu}, \dots, M_{iu+u-1})$. In this case the dependency table must be constructed according to the probability that changing every message unit changes each output bit.

4 Application to CubeHash

CubeHash [\[4\]](#) is Bernstein's proposal for the NIST SHA-3 competition [\[22\]](#). CubeHash variants, denoted by CubeHash- r/b , are parametrized by r and b which at each iteration process b bytes in r rounds. Although CubeHash-8/1 was the original official submission, later the designer proposed the tweak CubeHash-16/32 which is almost 16 times faster than the initial proposal [\[5\]](#). Nevertheless, the author has encouraged cryptanalysis of CubeHash- r/b variants for smaller r 's and bigger b 's.

4.1 CubeHash Description

CubeHash works with 32-bit words ($n = 32$) and uses three simple operations: XOR, rotation and modular addition. It has an internal state $S = (S_0, S_1, \dots, S_{31})$ of 32 words and its variants, denoted by CubeHash- r/b , are identified by two parameters $r \in \{1, 2, \dots\}$ and $b \in \{1, 2, \dots, 128\}$. The internal state S is set to a specified value which depends on the digest length (limited to 512 bits) and parameters r and b . The message to be hashed is appropriately padded and divided into b -byte message blocks. At each iteration one message block is processed as follows. The 32-word internal state S is considered as a 128-byte value and the message block is XORed into the first b bytes of the internal state. Then, the following fixed permutation is applied r times to the internal state to prepare it for the next iteration.

1. Add S_i into $S_{i\oplus 16}$, for $0 \leq i \leq 15$.
2. Rotate S_i to the left by seven bits, for $0 \leq i \leq 15$.
3. Swap S_i and $S_{i\oplus 8}$, for $0 \leq i \leq 7$.
4. XOR $S_{i\oplus 16}$ into S_i , for $0 \leq i \leq 15$.
5. Swap S_i and $S_{i\oplus 2}$, for $i \in \{16, 17, 20, 21, 24, 25, 28, 29\}$.
6. Add S_i into $S_{i\oplus 16}$, for $0 \leq i \leq 15$.
7. Rotate S_i to the left by eleven bits, for $0 \leq i \leq 15$.
8. Swap S_i and $S_{i\oplus 4}$, for $i \in \{0, 1, 2, 3, 8, 9, 10, 11\}$.
9. XOR $S_{i\oplus 16}$ into S_i , for $0 \leq i \leq 15$.
10. Swap S_i and $S_{i\oplus 1}$, for $i \in \{16, 18, 20, 22, 24, 26, 28, 30\}$.

Having processed all message blocks, a fixed transformation is applied to the final internal state to extract the hash value as follows. First, the last state word S_{31} is ORed with integer 1 and then the above permutation is applied $10 \times r$ times to the resulting internal state. Finally, the internal state is truncated to produce the message digest of desired hash length. Refer to [4] for the full specification.

4.2 Definition of the Compression Function Compress

To be in the line of our general method, we need to deal with fixed-size input compression functions. To this end, we consider t ($t \geq 1$) consecutive iterations of CubeHash. We define the function $H = \text{Compress}(M, V)$ with an $8bt$ -bit message $M = M^0 || \dots || M^{t-1}$, a 1024-bit initial value V and a $(1024 - 8b)$ -bit output H . The initial value V is used to initialize the 32-word internal state of CubeHash. Each M^i is a b -byte message block. We start from the initialized internal state and update it in t iterations. That is, in t iterations the t message blocks M^0, \dots, M^{t-1} are sequentially processed in order to transform the internal state into a final value. The output H is then the last $128 - b$ bytes of the final internal state value which is ready to absorb the $(t + 1)$ -st message block (the 32-word internal state is interpreted as a 128-byte vector).

Our goal is to find collisions for this Compress function. In the next section we explain how collisions can be constructed for CubeHash itself.

4.3 Collision Construction

We are planning to construct collision pairs (M', M'') for CubeHash- r/b which are of the form $M' = M^{\text{pre}} || M || M^t || M^{\text{suf}}$ and $M'' = M^{\text{pre}} || M \oplus \Delta || M^t \oplus \Delta^t || M^{\text{suf}}$. Here, M^{pre} is the common prefix of the colliding pairs whose length in bytes is a multiple of b , M^t is one message block of b bytes and M^{suf} is the common suffix of the colliding pairs whose length is arbitrary. The message prefix M^{pre} is chosen for randomizing the initial value V . More precisely, V is the content of the internal state after processing the message prefix M^{pre} . For this value of V , $(M, M \oplus \Delta)$ is a collision pair for the compression function, *i.e.* $\text{Compress}(M, V) = \text{Compress}(M \oplus \Delta, V)$. Remember that a collision for the Compress indicates collision over the last $128 - b$ bytes of the internal state. The message blocks M^t and $M^t \oplus \Delta^t$ are used to get rid of the difference in the first b bytes of the internal state. The difference Δ^t is called the *erasing block difference* and is computed as follows. When we evaluate the Compress with inputs (M, V) and $(M \oplus \Delta, V)$, Δ^t is the difference in the first b bytes of the final internal state values.

Once we find message prefix M^{pre} , message M and difference Δ , any message pairs (M', M'') of the above-mentioned form is a collision for CubeHash for *any* message block M^t and *any* message suffix M^{suf} . We find the difference Δ using the linearization method of Section 2 to applied to CubeHash in the next section. Then, M^{pre} and M are found by finding a preimage of zero under the Condition function as explained in Section 3. Algorithm 4 in the extended version of this article [9] shows how CubeHash Condition function can be implemented in practice for a given differential path.

4.4 Linear Differentials for CubeHash-r/b

As we explained in Section 2 the linear transformation $\text{Compress}_{\text{lin}}$ can be identified by a matrix $\mathcal{H}_{h \times m}$. We are interested in Δ 's such that $\mathcal{H}\Delta = 0$ and such that the differential trails have high probability. For CubeHash-r/b with t iterations, $\Delta = \Delta^0 || \dots || \Delta^{t-1}$ and \mathcal{H} has size $(1024 - 8b) \times 8bt$, see Section 4.2. This matrix *suffers from having low rank*. This enables us to find low weight vectors of the kernel. We then hope that they are also good candidates for providing highly probable trails, see Section 2.2. Assume that this matrix has rank $(8bt - \tau)$, $\tau \geq 0$, signifying existence of $2^\tau - 1$ nonzero solutions to $\mathcal{H}\Delta = 0$. To find a low weight nonzero Δ , we use the following method.

The rank of \mathcal{H} being $(8bt - \tau)$ shows that the solutions can be expressed by identifying τ variables as free and expressing the rest in terms of them. Any choice for the free variables uniquely determines the remaining $8bt - \tau$ variables, hence providing a unique member of the kernel. We choose a set of τ free variables at random. Then, we set one, two, or three of the τ free variables to bit value 1, and the other $\tau - 1$, or $\tau - 2$ or $\tau - 3$ variables to bit value 0 with the hope to get a Δ providing a high probability differential path. We have made exhaustive search over all $\tau + \binom{\tau}{2} + \binom{\tau}{3}$ possible choices for all $b \in \{1, 2, 3, 4, 8, 16, 32, 48, 64\}$ and $r \in \{1, 2, 3, 4, 5, 6, 7, 8\}$ in order to find the best characteristics. Table 1 includes the ordered pair (t, y) , *i.e.* the corresponding number of iterations and the $-\log_2$ probability (number of bit conditions) of the best raw probability path we found. For most of the cases, the best characteristic belongs to the minimum value of t for which $\tau > 0$. There are a few exceptions to consider which are starred in Table 1. For example in the CubeHash-3/4 case, while for $t = 2$ we have $\tau = 4$ and $y = 675$, by increasing the number of iterations to $t = 4$, we get $\tau = 40$ and a better characteristic with $y = 478$. This may hold for other cases as well since we only increased t until our program terminated in a reasonable time. We would like to emphasize that since we are using linear differentials, the erasing block difference Δ^t only depends on the difference Δ , see Section 4.3.

Table 1. The values of (t, y) for the differential path with the best found raw probability

$r \setminus b$	1	2	3	4	8	12	16	32	48	64
1	(14, 1225)	(8, 221)*	(4, 46)	(4, 32)	(4, 32)	-	-	-	-	-
2	(7, 1225)	(4, 221)*	(2, 46)	(2, 32)	(2, 32)	-	-	-	-	-
3	(16, 4238)*	(6, 1881)	(4, 798)	(4, 478)*	(4, 478)*	(4, 400)*	(4, 400)*	(4, 400)*	(3, 364)*	(2, 65)
4	(8, 2614)	(3, 964)	(2, 195)	(2, 189)	(2, 189)	(2, 156)	(2, 156)	(2, 156)	(2, 130)	(2, 130)
5	(18, 10221)*	(8, 4579)	(4, 2433)	(4, 1517)	(4, 1517)	(4, 1244)	(4, 1244)	(4, 1244)	(4, 1244)*	(2, 205)
6	(10, 4238)	(3, 1881)	(2, 798)	(2, 478)	(2, 478)	(2, 400)	(2, 400)	(2, 400)	(2, 351)	(2, 351)
7	(14, 13365)	(8, 5820)	(4, 3028)	(4, 2124)	(4, 2124)	(4, 1748)	(4, 1748)	(4, 1748)	(4, 1748)*	(2, 447)
8	(4, 2614)	(4, 2614)	(2, 1022)	(2, 1009)	(2, 1009)	(2, 830)	(2, 830)	(2, 830)	(2, 637)	(2, 637)

Second preimage attacks on CubeHash. Any differential path with raw probability greater than 2^{-512} can be considered as a (theoretical) second preimage attack on CubeHash with 512-bit digest size. In Table 1 the entries which do not correspond to a successful second preimage attack, *i.e.* $y > 512$, are shown in gray, whereas the others have been highlighted. For example, our differential path for CubeHash-6/4 with raw probability 2^{-478} indicates that by only one hash evaluation we can produce a second preimage with probability 2^{-478} . Alternatively, it can be stated that for a fraction of 2^{-478} messages we can easily provide a second preimage. The list of differential trails for highlighted entries can be found in the extended version [9].

4.5 Collision Attacks on CubeHash Variants

Although Table 1 includes our best found differential paths with respect to raw probability or equivalently second preimage attack, when it comes to freedom degrees use for collision attack, these trails might not be the optimal ones. In other words, for a specific r and b , there might be another differential path which is worse in terms of raw probability but is better regarding the collision attack complexity if we use some freedom degrees speedup. As an example, for CubeHash-3/48 with the path which has raw probability 2^{-364} , using our method of Section 3 the time complexity can be reduced to about $2^{58.9}$ (partial) evaluation of its condition function. However, there is another path with raw probability 2^{-368} which has time complexity of about $2^{53.3}$ (partial) evaluation of its condition function. Table 2 shows the best paths we found regarding the reduced complexity of the collision attack using our method of Section 3. While most of the paths are still the optimal ones with respect to the raw probability, the starred entries indicate the ones which invalidate this property. Some of the interesting differential paths for starred entries in Table 2 are given in the extended version [9].

Table 3 shows the reduced time complexities of collision attack using our method of Section 3 for the differential paths of Table 2. To construct the dependency table, we have analyzed the Condition function at byte level, see Remark 1. The time complexities are in logarithm 2 basis and might be improved if the dependency table is analyzed at a bit level instead. The complexity unit is (partial) evaluation of their respective Condition function. We remind that the full evaluation of a Condition function corresponding to a t -iteration differential path is almost the same as application of t iterations (rt rounds) of CubeHash. We emphasize that the complexities are independent of digest size. All the complexities which are less than $2^{c/2}$ can be considered as a successful collision attack if the hash size is bigger than c bits. The complexities bigger than 2^{256} have been shown in gray as they are worse than birthday attack, considering 512-bit digest size. The successfully attacked instances have been highlighted.

The astute reader should realize that the complexities of Table 3 correspond to the optimal threshold value, see Section 3.2. Refer to the extended version [9] to see the effect of the threshold value on the complexity.

Practice versus theory. We provided a framework which is handy in order to analyze many hash functions in a generic way. In practice, the optimal threshold value may be a little different from the theoretical one. Moreover, by slightly playing with the neighboring bits in the suggested partitioning corresponding to a given threshold value

Table 2. The values of (t, y) for the differential path with the best found total complexity (Table 3) includes the reduced complexities using our method of Section 3

$r \setminus b$	1	2	3	4	8	12	16	32	48	64
1	(14, 1225)	(8, 221)	(4, 46)	(4, 32)	(4, 32)	–	–	–	–	–
2	(7, 1225)	(4, 221)	(2, 46)	(2, 32)	(2, 32)	–	–	–	–	–
3	(16, 4238)	(6, 1881)	(4, 798)	(4, 478)	(4, 478)	(4, 400)	(4, 400)	(4, 400)	(3, 368)*	(2, 65)
4	(8, 2614)	(3, 964)	(2, 195)	(2, 189)	(2, 189)	(2, 156)	(2, 156)	(2, 156)	(2, 134)*	(2, 134)*
5	(18, 10221)	(8, 4579)	(4, 2433)	(4, 1517)	(4, 1517)	(4, 1250)*	(4, 1250)*	(4, 1250)*	(4, 1250)*	(2, 205)
6	(10, 4238)	(3, 1881)	(2, 798)	(2, 478)	(2, 478)	(2, 400)	(2, 400)	(2, 400)	(2, 351)	(2, 351)
7	(14, 13365)	(8, 5820)	(4, 3028)	(4, 2124)	(4, 2124)	(4, 1748)	(4, 1748)	(4, 1748)	(4, 1748)	(2, 455)*
8	(4, 2614)	(4, 2614)	(2, 1022)	(2, 1009)	(2, 1009)	(2, 830)	(2, 830)	(2, 830)	(2, 655)*	(2, 655)*

(Algorithm 2), we may achieve a partitioning which is more suitable for applying the attacks. In particular, Table 3 contains the theoretical complexities for different CubeHash instances under the assumption that the Condition function behaves ideally with respect to the first issue discussed in Section 3.2. In practice, deviation from this assumption increases the effective complexity. For particular instances, more simulations need to be done to analyze the potential non-randomness effects in order to give a more exact estimation of the practical complexity.

According to Section 4.3 for a given linear difference Δ , we need to find message prefix M^{pre} and conforming message M for collision construction. Our backtracking (tree-based) search implementation of Algorithm 1 for CubeHash-3/64 finds M^{pre} and M in 2^{21} (median complexity) instead of the $2^{9.4}$ of Table 3. The median decreases to 2^{17} by backtracking three steps at each depth instead of one, see Section 3.2. For CubeHash-4/48 we achieve the median complexity $2^{30.4}$ which is very close to the theoretical value $2^{30.7}$ of Table 3. Collision examples for CubeHash-3/64 and CubeHash-4/48 can be found in the extended paper [9]. Our detailed analysis of CubeHash variants shows that the practical complexities for all of them except 3-round CubeHash are very close to the theoretical values of Table 3. We expect the practical complexities for CubeHash instances with three rounds to be slightly bigger than the given theoretical numbers. For detailed comments we refer to the extended paper [9].

Comparison with the previous results. The first analysis of CubeHash was proposed by Aumasson et al. [3] in which the authors showed some non-random properties for several versions of CubeHash. A series of collision attacks on CubeHash-1/b and CubeHash-2/b for large values of b were announced by Aumasson [1] and Dai [12].

Table 3. Theoretical \log_2 complexities of improved collision attacks with freedom degrees use at byte level for the differential paths of Table 2

$r \setminus b$	1	2	3	4	8	12	16	32	48	64
1	1121.0	135.1	24.0	15.0	7.6	–	–	–	–	–
2	1177.0	179.1	27.0	17.0	7.9	–	–	–	–	–
3	4214.0	1793.0	720.0	380.1	292.6	153.5	102.0	55.6	53.3	9.4
4	2598.0	924.0	163.0	138.4	105.3	67.5	60.7	54.7	30.7	28.8
5	10085.0	4460.0	2345.0	1397.0	1286.0	946.0	868.0	588.2	425.0	71.7
6	4230.0	1841.0	760.6	422.1	374.4	260.4	222.6	182.1	147.7	144.0
7	13261.0	5709.0	2940.0	2004.0	1892.0	1423.0	1323.0	978.0	706.0	203.0
8	2606.0	2590.0	982.0	953.0	889.0	699.0	662.0	524.3	313.0	304.4

Collision attacks were later investigated deeply by Brier and Peyrin [8]. Our results improve on all existing ones as well as attacking some untouched variants.

5 Generalization

In sections 2 and 3 we considered modular-addition-based compression functions which use only modular additions and linear transformations. Moreover, we concentrated on XOR approximation of modular additions in order to linearize the compression function. This method is however quite general and can be applied to a broad class of hash constructions, covering many of the existing hash functions. Additionally, it lets us consider other linear approximations as well. We view a compression function $H = \text{Compress}(M, V) : \{0, 1\}^m \times \{0, 1\}^v \rightarrow \{0, 1\}^h$ as a binary finite state machine (FSM). The FSM has an internal state which is consecutively updated using message M and initial value V . We assume that FSM operates as follows, and we refer to such Compress functions as *binary-FSM-based*. The concept can also cover non-binary fields.

The internal state is initially set to zero. Afterwards, the internal state is sequentially updated in a limited number of steps. The output value H is then derived by truncating the final value of the internal state to the specified output size. At each step, the internal state is updated according to one of these *two* possibilities: either the whole internal state is updated as an affine transformation of the current internal state, M and V , or *only one* bit of the internal state is updated as a *nonlinear* Boolean function of the current internal state, M and V . Without loss of generality, we assume that all of the nonlinear updating Boolean functions (NUBF) have zero constant term (*i.e.* the output of zero vector is zero) and none of the involved variables appear as a pure linear term (*i.e.* changing any input variable does not change the output bit with certainty). This assumption, coming from the simple observation that we can integrate constants and linear terms in an affine updating transformation (AUT), is essential for our analysis. Linear approximations of the FSM can be achieved by replacing AUTs with linear transformations by ignoring the constant terms and NUBFs with linear functions of their arguments. Similar to Section 2 this gives us a linearized version of the compression function which we denote by $\text{Compress}_{\text{lin}}(M, V)$. As we are dealing with differential cryptanalysis, we take the notation $\text{Compress}_{\text{lin}}(M) = \text{Compress}_{\text{lin}}(M, 0)$. The argument given in Section 2 is still valid: elements of the kernel of the linearized compression function (*i.e.* Δ 's *s.t.* $\text{Compress}_{\text{lin}}(\Delta) = 0$) can be used to construct differential trails.

Let n_{nl} denote the total number of NUBFs in the FSM. We count the NUBFs by starting from zero. We introduce four functions $\Lambda(M, V)$, $\Phi(\Delta)$, $\Lambda^\Delta(M, V)$ and $\Gamma(\Delta)$ all of output size n_{nl} bits. To define these functions, consider the two procedures which implement the FSMs of $\text{Compress}(M, V)$ and $\text{Compress}_{\text{lin}}(\Delta)$. Let the Boolean function g^k , $0 \leq k < n_{\text{nl}}$, stand for the k -th NUBF and denote its linear approximation as in $\text{Compress}_{\text{lin}}$ by g_{lin}^k . Moreover, denote the input arguments of the Boolean functions g^k and g_{lin}^k in the FSMs which compute $\text{Compress}(M, V)$ and $\text{Compress}_{\text{lin}}(\Delta)$ by the vectors x^k and δ^k , respectively. Note that δ^k is a function of Δ whereas x^k depends on M and V . The k -th bit of $\Gamma(\Delta)$, $\Gamma_k(\Delta)$, is set to one iff the argument of the k -th linearized NUBF is not the all-zero vector, *i.e.* $\Gamma_k(\Delta) = 1$ iff $\delta^k \neq 0$. We then define $\Lambda_k(M, V) = g^k(x^k)$, $\Phi_k(\Delta) = g_{\text{lin}}^k(\delta^k)$ and $\Lambda_k^\Delta(M, V) = g^k(x^k \oplus \delta^k)$. We can then present the following proposition. The proof is given in the full version paper [9].

Proposition 2. *Let Compress be a binary-FSM-based compression function. For any message difference Δ , let $\{i_0, \dots, i_{y-1}\}$, $0 \leq i_0 < i_1 < \dots < i_{y-1} < n_{\text{nl}}$ be the positions of 1's in the vector $\Gamma(\Delta)$ where $y = \text{wt}(\Gamma(\Delta))$. We define the condition function $Y = \text{Condition}_\Delta(M, V)$ where the j -th bit of Y is computed as*

$$Y_j = A_{i_j}(M, V) \oplus A_{i_j}^\Delta(M, V) \oplus \Phi_{i_j}(\Delta). \quad (6)$$

Then, if Δ is in the kernel of $\text{Compress}_{\text{lin}}$, $\text{Condition}_\Delta(M, V) = 0$ implies that the pair $(M, M \oplus \Delta)$ is a collision for Compress with the initial value V .

Remark 2. The modular-addition-based compression functions can be implemented as binary-FSM-based compression by considering one bit memory for the carry bit. All the NUBFs for this FSM are of the form $g(x, y, z) = xy \oplus xz \oplus yz$. The XOR approximation of modular addition in Section 2 corresponds to approximating all the NUBFs g by the zero function, i.e. $g_{\text{lin}}(x, y, z) = 0$. It is straightforward to show that $A_k(M, V) = g(\mathbf{A}_k, \mathbf{B}_k, \mathbf{C}_k)$ and $\Phi_k(\Delta) = g_{\text{lin}}(\alpha_k, \beta_k, 0)$. We then deduce that $\Gamma_k(\Delta) = \alpha_k \vee \beta_k \vee 0$ and $A_k^\Delta(M, V) = g(\mathbf{A}_k \oplus \alpha_k, \mathbf{B}_k \oplus \beta_k, \mathbf{C}_k \oplus 0)$. As a result we get

$$\begin{aligned} Y_j &= A_{i_j}(M, V) \oplus A_{i_j}^\Delta(M, V) \oplus \Phi_{i_j}(\Delta) \\ &= (\alpha_{i_j} \oplus \beta_{i_j})\mathbf{C}_{i_j} \oplus \alpha_{i_j}\mathbf{B}_{i_j} \oplus \beta_{i_j}\mathbf{A}_{i_j} \oplus \alpha_{i_j}\beta_{i_j} \end{aligned} \quad (7)$$

whenever $\alpha_{i_j} \vee \beta_{i_j} = 1$; this agrees with equation (5). Refer to the extended version [9] for more details and to see how other linear approximations could be used.

6 Application to MD6

MD6 [23], designed by Rivest et al., is a SHA-3 candidate that provides security proofs regarding some differential attacks. The core part of MD6 is the function f which works with 64-bit words and maps 89 input words (A_0, \dots, A_{88}) into 16 output words $(A_{16r+73}, \dots, A_{16r+88})$ for some integer r representing the number of rounds. Each round is composed of 16 steps. The function f is computed based on the following recursion

$$A_{i+89} = L_{r_i, l_i}(S_i \oplus A_i \oplus (A_{i+71} \wedge A_{i+68}) \oplus (A_{i+58} \wedge A_{i+22}) \oplus A_{i+72}), \quad (8)$$

where S_i 's are some publicly known constants and L_{r_i, l_i} 's are some known simple linear transformations. The 89-word input of f is of the form $Q||U||W||K||B$ where Q is a known 15-word constant value, U is a one-word node ID, W is a one-word control word, K is an 8-word key and B is a 64-word data block. For more details about function f and the mode of operation of MD6, we refer to the submission document [23]. We consider the compression function $H = \text{Compress}(M, V) = f(Q||U||W||K||B)$ where $V = U||W||K$, $M = B$ and H is the 16-word compressed value. Our goal is to find a collision $\text{Compress}(M, V) = \text{Compress}(M', V)$ for arbitrary value of V . We later explain how such collisions can be translated into collisions for the MD6 hash function.

According to our model (Section 5), MD6 can be implemented as an FSM which has $64 \times 16r$ NUBFs of the form $g(x, y, z, w) = x \cdot y \oplus z \cdot w$. Remember that

² In the MD6 document [23], C and L_{r_i, l_i} are respectively denoted by V and g_{r_i, l_i} .

the NUBFs must not include any linear part or constant term. We focus on the case where we approximate all NUBFs with the zero function. This corresponds to ignoring the AND operations in equation (8). This essentially says that in order to compute $\text{Compress}_{\text{lin}}(\Delta) = \text{Compress}_{\text{lin}}(\Delta, 0)$ for a 64-word $\Delta = (\Delta_0, \dots, \Delta_{63})$, we map $(A'_0, \dots, A'_{24}, A'_{25}, \dots, A'_{88}) = 0 \parallel \Delta = (0, \dots, 0, \Delta_0, \dots, \Delta_{63})$ into the 16 output words $(A'_{16r+73}, \dots, A'_{16r+88})$ according to the linear recursion

$$A'_{i+89} = L_{r_i, l_i}(A'_i \oplus A'_{i+72}). \tag{9}$$

For a given Δ , the function Γ is the concatenation of 16r words $A'_{i+71} \vee A'_{i+68} \vee A'_{i+58} \vee A'_{i+22}$, $0 \leq i \leq 16r - 1$. Therefore, the number of bit conditions equals

$$y = \sum_{i=0}^{16r-1} \text{wt}(A'_{i+71} \vee A'_{i+68} \vee A'_{i+58} \vee A'_{i+22}). \tag{10}$$

Note that this equation compactly integrates cases 1 and 2 given in section 6.9.3.2 of [23] for counting the number of active AND gates. Algorithm 3 in the extended version of this article [9] shows how the Condition function is implemented using equations (6), (8) and (9).

Using a similar linear algebraic method to the one used in Section 4.4 for CubeHash, we have found the collision difference of equation (11) for $r = 16$ rounds with a raw probability $p_\Delta = 2^{-90}$. In other words, Δ is in the kernel of $\text{Compress}_{\text{lin}}$ and the condition function has $y = 90$ output bits. Note that this does not contradict the proven bound in [23]: one gets at least 26 active AND gates.

$$\Delta_i = \begin{cases} \text{F6D164597089C40E} & i = 2 \\ 2000000000000000 & i = 36 \\ 0 & 0 \leq i \leq 63, i \neq 2, 36 \end{cases} \tag{11}$$

In order to efficiently find a conforming message pair for this differential path we need to analyze the dependency table of its condition function. Referring to our notations in Section 3.2, our analysis of the dependency table of function $\text{Condition}_\Delta(M, 0)$ at word level (units of $u = 64$ bits) shows that the partitioning of the condition function is as in Table 4 for threshold value $th = 0$. For this threshold value clearly $p_i = 0$. The optimal values for q_i 's (computed according to the complexity analysis of the same section) are also given in Table 4, showing a total attack complexity of $2^{30.6}$ (partial) condition function evaluation³. By analyzing the dependency table with smaller units the complexity may be subject to reduction.

A collision example for $r = 16$ rounds of f can be found in the full version [9]. Our 16-round colliding pair provides near collisions for $r = 17, 18$ and 19 rounds, respectively, with **63, 144** and **270** bit differences over the 1024-bit long output of f . Refer to [9] to see how collisions for reduced-round f can be turned into collisions for reduced-round MD6 hash function. The original MD6 submission [23] mentions inversion of the function f up to a dozen rounds using SAT solvers. Some slight nonrandom behavior of the function f up to 33 rounds has also been reported [17].

³ By masking M_{38} and M_{55} respectively with 092E9BA68F763BF1 and DFFBFF7FEFFDFBFB after random setting, the 35 condition bits of the first three steps are satisfied for free, reducing the complexity to $2^{30.0}$ instead.

Table 4. Input and output partitionings of the Condition function of MD6 with $r = 16$ rounds

i	\mathcal{M}_i	\mathcal{Y}_i	q_i	q'_i
0	–	\emptyset	0	0
1	$\{M_{38}\}$	$\{Y_1, \dots, Y_{29}\}$	29	0
2	$\{M_{55}\}$	$\{Y_{43}, \dots, Y_{48}\}$	6	0
3	$\{M_0, M_5, M_{46}, M_{52}, M_{54}\}$	$\{Y_0\}$	1	0
4	$\{M_j j = 3, 4, 6, 9, 21, 36, 39, 40, 42, 45, 49, 50, 53, 56, 57\}$	$\{Y_{31}, \dots, Y_{36}\}$	6	0
5	$\{M_{41}, M_{51}, M_{58}, M_{59}, M_{60}\}$	$\{Y_{30}, Y_{51}\}$	2	0
6	$\{M_j j = 1, 2, 7, 8, 10, 11, 12, 17, 18, 20, 22, 24, 25, 26, 29, 33, 34, 37, 43, 44, 47, 48, 61, 62, 63\}$	$\{Y_{52}, \dots, Y_{57}\}$	6	0
7	$\{M_{27}\}$	$\{Y_{37}, \dots, Y_{42}\}$	6	0
8	$\{M_{13}, M_{16}, M_{23}\}$	$\{Y_{50}\}$	1	0
9	$\{M_{35}\}$	$\{Y_{49}\}$	1	0
10	$\{M_{14}, M_{15}, M_{19}, M_{28}\}$	$\{Y_{58}, Y_{61}\}$	2	0
11	$\{M_{30}, M_{31}, M_{32}\}$	$\{Y_{59}, Y_{60}, Y_{62}, \dots, Y_{89}\}$	30	0

7 Conclusion

We presented a framework for an in-depth study of linear differential attacks on hash functions. We applied our method to reduced round variants of CubeHash and MD6, giving by far the best known collision attacks on these SHA-3 candidates. Our results may be improved by considering start-in-the-middle attacks if the attacker is allowed to choose the initial value of the internal state.

Acknowledgment. The second author has been supported in part by European Commission through the ICT programme under contract ICT-2007-216676 ECRYPT II, and the third author by GEBERT RÜF STIFTUNG, project no. GRS-069/07. The authors would like to thank the reviewers of Asiacrypt 2009, Deian Stefan, Martijn Stam, Jean-Philippe Aumasson and Dag Arne Osvik for their helpful comments.

References

1. Aumasson, J.-P.: Collision for CubeHash-2/120 – 512. NIST mailing list, December 4 (2008), <http://ehash.iaik.tugraz.at/uploads/a/a9/Cubehash.txt>
2. Aumasson, J.-P., Fischer, S., Khazaei, S., Meier, W., Rechberger, C.: New Features of Latin Dances: Analysis of Salsa, ChaCha, and Rumba. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 470–488. Springer, Heidelberg (2008)
3. Aumasson, J.-P., Meier, W., Naya-Plasencia, M., Peyrin, T.: Inside the hypercube. In: Boyd, C., González Nieto, J. (eds.) ACISP 2009. LNCS, vol. 5594, pp. 202–213. Springer, Heidelberg (2009)
4. Bernstein, D.J.: CubeHash specification (2.b.1). Submission to NIST SHA-3 competition
5. Bernstein, D.J.: CubeHash parameter tweak: 16 times faster, <http://cubehash.cr.yp.to/submission/>
6. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Radiogatan, a belt-and-mill hash function. Presented at Second Cryptographic Hash Workshop, Santa Barbara (August 2006)
7. Biham, E., Chen, R.: Near-Collisions of SHA-0. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 290–305. Springer, Heidelberg (2004)
8. Brier, E., Peyrin, T.: Cryptanalysis of CubeHash. Applied Cryptography and Network Security. In: Abdalla, M., Pointcheval, D., Fouque, P.-A., Vergnaud, D. (eds.) ACNS 2009. LNCS, vol. 5536, pp. 354–368. Springer, Heidelberg (2009)

9. Brier, E., Khazaei, S., Meier, W., Peyrin, T.: Linearization Framework for Collision Attacks: Application to CubeHash and MD6 (Extended Version). In Cryptology ePrint Archive, Report 2009/382, <http://eprint.iacr.org/2009/382>
10. Canteaut, A., Chabaud, F.: A new algorithm for finding minimum-weight words in a linear code: application to McEliece's cryptosystem and to narrow-sense BCH codes of length 511. *IEEE Transactions on Information Theory* 44(1), 367–378 (1998)
11. Chabaud, F., Joux, A.: Differential Collisions in SHA-0. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 56–71. Springer, Heidelberg (1998)
12. Dai, W.: Collisions for CubeHash-1/45 and CubeHash-2/89 (2008), <http://www.cryptopp.com/sha3/cubehash.pdf>
13. eBASH: ECRYPT Benchmarking of All Submitted Hashes, <http://bench.cr.yp.to/ebash.html>
14. Fuhr, T., Peyrin, T.: Cryptanalysis of Radiogatun. In: Dunkelman, O. (ed.) FSE 2009. LNCS, vol. 5665, pp. 122–138. Springer, Heidelberg (2009)
15. Indestege, S., Preneel, B.: Practical collisions for EnRUPt. In: Dunkelman, O. (ed.) FSE 2009. LNCS, vol. 5665, pp. 246–259. Springer, Heidelberg (2009)
16. 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)
17. Khovratovich, D.: Nonrandomness of the 33-round MD6. Presented at the rump session of FSE 2009 (2009), Slides: <http://fse2009rump.cr.yp.to/>
18. Klima, V.: Tunnels in Hash Functions: MD5 Collisions Within a Minute. ePrint archive (2006), <http://eprint.iacr.org/2006/105.pdf>
19. Lipmaa, H., Moriai, S.: Efficient Algorithms for Computing Differential Properties of Addition. In: Matsui, M. (ed.) FSE 2001. LNCS, vol. 2355, pp. 336–350. Springer, Heidelberg (2002)
20. Manuel, S., Peyrin, T.: Collisions on SHA-0 in One Hour. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 16–35. Springer, Heidelberg (2008)
21. Naito, Y., Sasaki, Y., Shimoyama, T., Yajima, J., Kunihiko, 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)
22. National Institute of Science and Technology. Announcing request for candidate algorithm nominations for a new cryptographic hash algorithm (SHA-3) family. Federal Register, 72(112) (November 2007)
23. Rivest, R.L., Agre, B., Bailey, D.V., Crutchfield, C., Dodis, Y., Fleming, K.E., Khan, A., Krishnamurthy, J., Lin, Y., Reyzin, L., Shen, E., Sukha, J., Sutherland, D., Tromer, E., Yin, Y.L.: The MD6 hash function — a proposal to NIST for SHA-3. Submission to NIST SHA-3 competition (2008)
24. Peyrin, T.: Cryptanalysis of Grindahl. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 551–567. Springer, Heidelberg (2007)
25. Pramstaller, N., Rechberger, C., Rijmen, V.: Exploiting Coding Theory for Collision Attacks on SHA-1. In: Smart, N.P. (ed.) Cryptography and Coding 2005. LNCS, vol. 3796, pp. 78–95. Springer, Heidelberg (2005)
26. Rijmen, V., Oswald, E.: Update on SHA-1. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 58–71. Springer, Heidelberg (2005)
27. 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)

Preimages for Step-Reduced SHA-2

Kazumaro Aoki¹, Jian Guo^{2,*}, Krystian Matusiewicz³, Yu Sasaki^{1,4},
and Lei Wang⁴

¹ NTT Information Sharing Platform Laboratories, NTT Corporation
3-9-11 Midori-cho, Musashino-shi, Tokyo, 180-8585 Japan

{aoki.kazumaro,sasaki.yu}@lab.ntt.co.jp

² Division of Mathematical Sciences

School of Physical and Mathematical Sciences
Nanyang Technological University, Singapore

guojian@ntu.edu.sg

³ Department of Mathematics

Technical University of Denmark, Denmark

K.Matusiewicz@mat.dtu.dk

⁴ University of Electro-Communications

1-5-1 Choufugaoka, Choufu-shi, Tokyo, 182-8585 Japan

wanglei@ice.uec.ac.jp

Abstract. In this paper, we present preimage attacks on up to 43-step SHA-256 (around 67% of the total 64 steps) and 46-step SHA-512 (around 57.5% of the total 80 steps), which significantly increases the number of attacked steps compared to the best previously published preimage attack working for 24 steps. The time complexities are $2^{251.9}$, 2^{509} for finding pseudo-preimages and $2^{254.9}$, $2^{511.5}$ compression function operations for full preimages. The memory requirements are modest, around 2^6 words for 43-step SHA-256 and 46-step SHA-512. The pseudo-preimage attack also applies to 43-step SHA-224 and SHA-384. Our attack is a meet-in-the-middle attack that uses a range of novel techniques to split the function into two independent parts that can be computed separately and then matched in a birthday-style phase.

Keywords: SHA-256, SHA-512, hash, preimage attack, meet-in-the-middle.

1 Introduction

Cryptographic hash functions are important building blocks of many secure systems. SHA-1 and SHA-2 (SHA-224, SHA-256, SHA-384, and SHA-512) [1] are hash functions standardized by the National Institute of Standards and Technology (NIST) and widely used all over the world. However, a collision attack on SHA-1 has been discovered recently by Wang *et al.* [2]. Since the structure of SHA-2 is similar to SHA-1 and they are both heuristic designs with no known

* This work was done while visiting Technical University of Denmark and was partly supported by a DCAMM grant.

security guarantees or reductions, an attack on SHA-2 might be discovered in the future too. To avoid a situation when all FIPS standardized functions would be broken, NIST is currently conducting a competition to determine a new hash function standard called SHA-3 [3]. From the engineering viewpoint, migration from SHA-1 to SHA-3 will take a long time. SHA-2 will take an important role during that transitional period. Hence, rigorous security evaluation of SHA-2 using the latest analytic techniques is important.

NIST requires SHA-3 candidates of n -bit hash length to satisfy a several security properties [3], first and foremost

- Preimage resistance of n bits,
- Second-preimage resistance of $n - k$ bits for any message shorter than 2^k blocks,
- Collision resistance of $n/2$ bits.

NIST claims that the security of each candidate is evaluated in the environment where they are tuned so that they run as fast as SHA-2 [4]. It seems that NIST tries to evaluate each candidate by comparing it with SHA-2. However, the security of SHA-2 is not well understood yet. Hence, the evaluation of the security of SHA-2 with respect to the security requirements for SHA-3 candidates is also important as it may influence our perspective on the SHA-3 speed requirements.

SHA-256 and SHA-512 consist of 64 steps and 80 steps, respectively. The first analysis of SHA-2 with respect to collision resistance was described by Mendel *et al.* [5], which presented the collision attack on SHA-2 reduced to 19 steps. After that, several researches have improved the result. In particular, the work by Nikolić and Biryukov improved the collision techniques [6]. The best collision attacks so far are the ones proposed by Indestege *et al.* [7] and Sanadhya and Sarkar [8], both describing collision attacks for 24 steps. The only analysis of preimage resistance we are aware of is a recent attack on 24 steps of SHA-2 due to Isobe and Shibutani [9].

One may note the work announced at the rump session by Yu and Wang [10], which claimed to have found a non-randomness property of SHA-256 reduced to 39 steps. Since the non-randomness property is not included in the security requirements for SHA-3, we do not discuss it in this paper. In summary, the current best attacks on SHA-2 with respect to the security requirements for SHA-3 work for only 24 steps.

After Saarinen [11] and Leurent [12] showed examples of meet-in-the-middle preimage attacks, the techniques for such preimage attacks have been developed very rapidly. Attacks based on the concept of meet-in-the-middle have been reported for various hash functions, for example MD5 [13], SHA-1, HAVAL [14], and so on [15,16,17,18]. The meet-in-the-middle preimage attack is also applied to recently designed hash function ARIRANG [19], which is one of SHA-3 candidates, by Hong *et al.* [20]. However, due to the complex message schedule in SHA-2, these recently developed techniques have not been applied to SHA-2 yet.

Our contribution. We propose preimage attacks on 43-step SHA-256 and 46-step SHA-512 which drastically increase the number of attacked steps compared

to the previous preimage attack on 24 steps. We first explain various attack techniques for attacking SHA-2. We then explain how to combine these techniques to maximize the number of attacked steps. It is interesting that more steps of SHA-512 can be attacked than of SHA-256 with so-called partial-fixing technique proposed by Aoki and Sasaki [15]. This is due to the difference of the word size as functions σ and Σ mix 32-bit variables in SHA-256 more rapidly than in the case of double-size variables in SHA-512.

Our attacks are meet-in-the-middle. We first consider the application of the previous meet-in-the-middle techniques to SHA-2. We then analyse the message expansion of SHA-2 by considering all previous techniques and construct the attack by finding new independent message-word partition, which is the fundamental part of this attack.

Our attacks and a comparison with other results are summarized in Table 1.

Table 1. Comparison of preimage attacks on reduced SHA-2

Reference		Target	Steps	Complexity		Memory (approx.)
				Pseudo-preimage	Preimage	
Ours	Section 7	SHA-224	43	$2^{219.9}$	-	2^6 words
[9]		SHA-256	24	2^{240}	2^{240}	$2^{16} \cdot 64$ bits
Ours	Section 5	SHA-256	42	$2^{245.3}$	$2^{251.7}$	2^{12} words
Ours	Section 5	SHA-256	43	$2^{251.9}$	$2^{254.9}$	2^6 words
Ours	Section 7	SHA-384	43	2^{366}	-	2^{19} words
[9]		SHA-512	24	2^{480}	2^{480}	not given
Ours	Section 6	SHA-512	42	2^{488}	2^{501}	2^{27} words
Ours	Section 6	SHA-512	46	2^{509}	$2^{511.5}$	2^6 words

Outline. In Section 2, we briefly describe SHA-2. Section 3 gives an overview of the meet-in-the-middle preimage attack. In Section 4, we describe all techniques of our preimage attack. Then Sections 5 and 6 explain how these techniques can be applied together to mount an attack on SHA-256 and SHA-512, respectively. In Section 7, we put some remark on our attack. Section 8 concludes this paper.

2 SHA-2 Specification

Description of SHA-256. In this section we describe SHA-256, consult [1] for full details. SHA-256 adopts the Merkle-Damgård structure [21, Algorithm 9.25]. The message string is first padded with a single “1” bit, appropriate number of zero bits and then 64-bit length of the original message so that the length of the padded message is a multiple of 512 bits and then divided into 512-bit blocks, $(M_0, M_1, \dots, M_{N-1})$ where $M_i \in \{0, 1\}^{512}$.

The hash value h_N is computed by iteratively using the compression function CF, which takes a 512-bit message block and a 256-bit chaining variable as the input and yields an updated 256-bit chaining variable as the output,

$$\begin{cases} h_0 \leftarrow IV, \\ h_{i+1} \leftarrow CF(h_i, M_i) \quad (i = 0, 1, \dots, N - 1), \end{cases} \quad (1)$$

where IV is a constant value defined in the specification.

The compression function is based on the Davies-Meyer mode [21, Algorithm 9.42]. It consists of a message expansion and a data processing. Let \gg^x and \ggg^x denote the x -bit right shift and rotation, respectively. First, the message block is expanded by the message expansion function,

$$W_i \leftarrow \begin{cases} m_i & \text{for } 0 \leq i < 16, \\ \sigma_1(W_{i-2}) + W_{i-7} + \sigma_0(W_{i-15}) + W_{i-16} & \text{for } 16 \leq i < 64. \end{cases} \quad (2)$$

where $(m_0, m_1, \dots, m_{15}) \leftarrow M_i$ ($m_j \in \{0, 1\}^{32}$) and “+” denotes addition modulo $2^{\text{word-size}}$. In SHA-256 the word size is 32 bits. Functions $\sigma_0(X)$ and $\sigma_1(X)$ are defined as

$$\begin{aligned} \sigma_0(X) &\leftarrow (X \ggg^7) \oplus (X \ggg^{18}) \oplus (X \ggg^3), \\ \sigma_1(X) &\leftarrow (X \ggg^{17}) \oplus (X \ggg^{19}) \oplus (X \ggg^{10}). \end{aligned} \quad (3)$$

where “ \oplus ” stands for bitwise XOR operation.

Let us use p_j to denote a 256-bit value consisting of the concatenation of eight words $A_j, B_j, C_j, D_j, E_j, F_j, G_j$ and H_j . The data processing computes h_{i+1} as follows.

$$\begin{cases} p_0 \leftarrow h_i, \\ p_{j+1} \leftarrow R_j(p_j, W_j), \quad (j = 0, 1, \dots, 63) \\ h_{i+1} \leftarrow h_i + p_{64}, \end{cases} \quad (4)$$

Step function R_j is defined as follows

$$\begin{cases} T_1^{(j)} \leftarrow H_j + \Sigma_1(E_j) + \text{Ch}(E_j, F_j, G_j) + K_j + W_j, \\ T_2^{(j)} \leftarrow \Sigma_0(A_j) + \text{Maj}(A_j, B_j, C_j), \\ A_{j+1} \leftarrow T_1^{(j)} + T_2^{(j)}, \quad B_{j+1} \leftarrow A_j, \quad C_{j+1} \leftarrow B_j, \quad D_{j+1} \leftarrow C_j, \\ E_{j+1} \leftarrow D_j + T_1^{(j)}, \quad F_{j+1} \leftarrow E_j, \quad G_{j+1} \leftarrow F_j, \quad H_{j+1} \leftarrow G_j. \end{cases} \quad (5)$$

Above, K_j is a constant, different for each step, and the following functions are used

$$\begin{aligned} \text{Ch}(X, Y, Z) &\leftarrow (X \vee Y) \oplus ((\neg X) \vee Z), \\ \text{Maj}(X, Y, Z) &\leftarrow (X \vee Y) \oplus (X \vee Z) \oplus (Y \vee Z), \\ \Sigma_0(X) &\leftarrow (X \ggg^2) \oplus (X \ggg^{13}) \oplus (X \ggg^{22}), \\ \Sigma_1(X) &\leftarrow (X \ggg^6) \oplus (X \ggg^{11}) \oplus (X \ggg^{25}). \end{aligned} \quad (6)$$

where \neg means bitwise negation of the word.

Description of SHA-512. The structure of SHA-512 is basically the same as SHA-256. In SHA-512, the word size is 64 bits, double of SHA-256, hence, the message-block size is 1024 bits and the size of chaining variable p_j is 512 bits.

The compression function has 80 steps. Rotation numbers in $\sigma_0, \sigma_1, \Sigma_0$, and Σ_1 are different from those used in SHA-256, which are shown below.

$$\begin{aligned}
 \sigma_0(X) &\leftarrow (X \ggg^1) \oplus (X \ggg^8) \oplus (X \ggg^7), \\
 \sigma_1(X) &\leftarrow (X \ggg^{19}) \oplus (X \ggg^{61}) \oplus (X \ggg^6), \\
 \Sigma_0(X) &\leftarrow (X \ggg^{28}) \oplus (X \ggg^{34}) \oplus (X \ggg^{39}), \\
 \Sigma_1(X) &\leftarrow (X \ggg^{14}) \oplus (X \ggg^{18}) \oplus (X \ggg^{41}).
 \end{aligned}
 \tag{7}$$

3 Overview of the Meet-in-the-Middle Preimage Attack

A preimage attack on a narrow-pipe Merkle-Damgård hash function is usually based on a pseudo-preimage attack on its underlying compression function, where a pseudo-preimage is a preimage of the compression function with an appropriate padding. Many compression functions adopt Davies-Meyer mode, which computes $E_u(v) \oplus v$, where u is the message, v is the intermediate hash value and E is a block cipher.

First we recall the attack strategy on a compression function, which has been illustrated in Fig. 1. Denote by h the given target hash value. The high-level description of the attack for the simplest case is as follows.

1. Divide the key u of the block cipher E into two *independent* parts: u_1 and u_2 . Hereafter, independent parts are called “chunks” and independent inputs u_1 and u_2 are called “neutral words”.
2. Randomly determine the other input value v of the block cipher E .
3. Carry out the forward calculation utilizing v and all possible values of u_1 , and store all the obtained intermediate values in a table T_F .
4. Carry out the backward calculation utilizing $h \oplus v$ and all possible values of u_2 , and store all the intermediate values in a table T_B .
5. Check whether there exists a collision between T_F and T_B . If a collision exists, a pseudo-preimage of h has been generated. Otherwise, go to Step 2.

The main novelty of the meet-in-the-middle preimage attacks is, by utilizing independence of u_1 and u_2 of the key input, transforming the problem of finding a preimage of h to the problem of finding a collision on the intermediate values, which has a much lower complexity than the former one. Suppose there

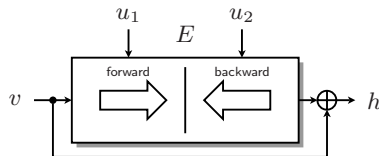


Fig. 1. Meet-in-the-middle attack strategy on a Davies-Meyer compression function $E_u(v) \oplus v$

are 2^t possible values for each of u_1 and u_2 . Using 2^t compression function computations, the attacker obtains 2^t elements in *each* of T_F and T_B . The collision probability is roughly 2^{2t-n} , where n is the bit length of h , much better than the probability 2^{t-n} of finding a preimage by a brute force search with complexity 2^t .

4 The List of Attack Techniques

This section describes the list of techniques used in the attack. Some of them were used before in previous meet-in-the-middle attacks [15,18,16]. We explain them here first and then in Sections 5 and 6, we show how to combine them in an attack on SHA-2.

4.1 Splice-and-Cut

The meet-in-the-middle attack starts with dividing the key input into two independent parts. The idea of *splice-and-cut* is based on the observation made in [15] that the last and first steps of the block cipher E in Davies-Meyer mode can be regarded as consecutive by considering the feed-forward operation.

This allows the attacker to choose any step as the starting step of the meet-in-the-middle, which helps with finding more suitable independent chunks.

This technique can find only pseudo-preimages of the given hash value instead of preimages. However, pseudo-preimages can be converted to preimages with a conversion algorithm explained below.

4.2 Converting Pseudo-preimages to Preimages

In x -bit iterated hash functions, a pseudo-preimage attack with complexity 2^y , $y < x - 2$ can be converted to a preimage attack with complexity of $2^{\frac{x+y}{2}+1}$ [21, Fact9.99]. The idea is applying the unbalanced meet-in-the-middle attack with generating $2^{(x-y)/2}$ pseudo-preimages and generating $2^{(x+y)/2}$ 1-block chaining variables starting from IV.

4.3 Partial-Matching

The example in Fig. 1 is the simplest and optimistic case. In fact, in the previous attacks, the key input cannot be divided into just two independent chunks. Usually besides the two independent chunks u_1 and u_2 , there is another part, which depends on both u_1 and u_2 . Hence, the stored intermediate values in T_F and T_B are ones at different steps. This raises a problem: how the values in T_F and T_B can be compared. However, many hash functions, including SHA-2, have Unbalanced Feistel Network structure, where the intermediate values will only be updated partially at one step. This means that a part of the intermediate values does not change during several steps and the attacker can check the match of two values partially.

Consider SHA-2, assume one chunk produces the value of p_j and the other chunk produces the value of p_{j+s} . The attacker wants to efficiently check whether or not p_j and p_{j+s} match without the knowledge of $W_j, W_{j+1}, \dots, W_{j+s-1}$. In SHA-2, the maximum number of s is 7.

Assume the value of $p_{j+7} = A_{j+7} \parallel B_{j+7} \parallel \dots \parallel H_{j+7}$ is known and W_{j+6} is unknown. By backward computation, we can obtain the values of $A_{j+6}, B_{j+6}, \dots, G_{j+6}$. This is because $A_{j+6}, B_{j+6}, C_{j+6}, E_{j+6}, F_{j+6}$, and G_{j+6} are just copies of corresponding values in p_{j+7} and D_{j+6} is computed as follows.

$$D_{j+6} \leftarrow E_{j+7} - (A_{j+7} - (\Sigma_0(B_{j+7}) + \text{Maj}(B_{j+7}, C_{j+7}, D_{j+7}))). \quad (8)$$

By repeating the similar computation, in the end, A_j is computed from p_{j+7} without the knowledge of $W_j, W_{j+1}, \dots, W_{j+6}$. Note that this technique was already used (but not explicitly named) in [9].

4.4 Partial-Fixing

This is an extension of the partial-matching technique that considers *parts* of registers of the internal state. It increases the number of steps that can exist between two independent chunks. Assume that the attacker is carrying out the computation using u_1 and he is facing a step whose key input depends on both u_1 and u_2 . Because the computation cannot go ahead without the knowledge of u_2 , the chunk for u_1 must stop at this step. The *partial-fixing* technique is partially fixing the values of u_1 and u_2 so that we can obtain partial knowledge even if the full computation depends on both u_1 and u_2 .

The partial-fixing technique for SHA-2 has not been considered previously. Assume we can fix the lower x bits of the message word in each step. Under this assumption, 1 step can be partially computed easily. Let us consider the step function of SHA-2 in the forward direction. Equations using W_j is as follows.

$$\begin{cases} T_1^{(j)} \leftarrow H_j + \Sigma_1(E_j) + \text{Ch}(E_j, F_j, G_j) + K_j + W_j, \\ A_{j+1} \leftarrow T_1^{(j)} + T_2^{(j)}, \quad E_{j+1} \leftarrow D_j + T_1^{(j)}. \end{cases} \quad (9)$$

If the lower x bits of W_j are fixed, the lower x bits of A_{j+1} (and E_{j+1}) can be computed independently of the upper $32 - x$ bits of W_j . Let us consider to skip another step in forward direction. The equation for A_{j+2} is as follows:

$$A_{j+2} \leftarrow T_1^{(j+1)} + \Sigma_0(A_{j+1}) + \text{Maj}(A_{j+1}, B_{j+1}, C_{j+1}). \quad (10)$$

We know only the lower x bits on A_{j+1} . Hence, we can compute Maj function for only the lower x bits. How about the Σ_0 function? We analysed the relationship of the number of consecutive fixed bits from LSB in the input and output of $\sigma_0, \sigma_1, \Sigma_0$, and Σ_1 . The results are summarized in Table 2.

From Table 2, if x is large enough, we can compute the lower $x - 22$ bits of A_{j+2} in SHA-256 and the lower $x - 39$ bits in SHA-512, though the number of known bits is greatly reduced after the Σ_0 function. This fact also implies

Table 2. Relationship of number of consecutive fixed bits from LSB in input and output of σ and Σ

	SHA-256				SHA-512			
	Σ_0	Σ_1	σ_0	σ_1	Σ_0	Σ_1	σ_0	σ_1
Input	x	x	x	x	x	x	x	x
output	$x - 22$	$x - 25$	$x - 18$	$x - 19$	$x - 39$	$x - 41$	$x - 8$	$x - 61$

When x agrees with the word size, the output is x . When the number described in the output is negative, the output is 0.

that we cannot obtain the value of A_{j+3} since the number of fixed bits will be always 0. In the end, the partial-fixing technique can be applied for up to 2 steps in forward direction. Similarly, we considered the partial-fixing technique in backward, and found that it can be applied up to 6 steps.

However we have another problem in the first assumption; the lower x bits of each message word can be fixed. This is difficult to achieve because the fixed bits in message words are mixed by the σ function in the message expansion. In fact, we could apply the partial-fixing technique for computing only 1 step in forward, and only 2 steps in backward for SHA-256. However, in SHA-512, the bit-mixing speed of σ is relatively slow due to the double word size. In fact, we could compute 2 steps in forward, and 6 steps in backward. Finally, 10 steps in total can be skipped by the partial-matching and partial-fixing techniques for SHA-256, and 15 steps for SHA-512. (These numbers of steps are explained in Sections 5 and 6)

4.5 Indirect-Partial-Matching

This is another extension of partial-matching. Consider the intermediate values in T_F and T_B . We can express them as functions of u_1 and u_2 , respectively. If the next message word used in forward direction can be expressed as $\psi_1(u_1) + \psi_2(u_2)$ and computation of chaining register at the matching point does not destroy this relation (because the message word is also added), the matching point can still be expressed as a sum of two independent functions of u_1, u_2 , e.g. $\psi_F(u_1) + \xi_F(u_2)$. Similarly, we can express the matching point from backward as $\psi_B(u_1) + \xi_B(u_2)$, and we are to find match. Now, instead of finding a match directly, we can compute $\psi_F(u_1) - \psi_B(u_1)$ in forward direction and $\xi_B(u_2) - \xi_F(u_2)$ in backward direction independently and find a match.

In case of SHA-2, it is possible to extend the 7-step partial-matching to 9-step indirect-partial-matching by inserting one step just before and after the partial matching.

Note this technique can be combined with partial-fixing technique by applying them in order: partial-fixing, partial-matching and indirect-partial-matching. However, there are some constraints that need to be satisfied, such as the independence of message word used in indirect-partial-matching, while we need to be able to compute enough bits at the matching point in order to carry out the partial-matching efficiently.

4.6 Initial Structure

In some cases, the two independent chunks u_1 and u_2 will overlap with each other. The typical example is that the order of the input key of E is $u_1u_2u_1u_2$. This creates a problem: how should the attacker carry out the forward and backward computations independently. The *Initial Structure* technique was proposed by [16] to solve such a problem. Previous attacks usually set a certain step as the starting step, then randomly determine the intermediate value at that step, and carry out the independent computations. However, the initial structure technique sets all the steps of u_2u_1 in the middle of $u_1u_2u_1u_2$ together as the starting point. Denote the intermediate values at the beginning and last step of u_2u_1 as I_1 and I_2 respectively. For each possible value of u_1 , the attacker can derive a corresponding value I_1 . Similarly, for each possible value of u_2 , the attacker can derive a corresponding value I_2 . Moreover, any pair (I_1, u_1) and (I_2, u_2) can be matched at the steps of u_2u_1 of $u_1u_2u_1u_2$. Thus, the attacker can carry out independent computations utilizing (I_1, u_1) and (I_2, u_2) .

Initial structure for SHA-2 makes use of the absorption property of the function $\text{Ch}(x, y, z) = xy \oplus (-x)z$. If x is $\mathbf{1}$ (all bits are 1), then $\text{Ch}(\mathbf{1}, y, z) = y$ which means z does not affect the result of Ch function in this case; similarly when x is $\mathbf{0}$ (all bits are 0), y does not affect the result. When we want to control partial output (few bits), we need to fix the corresponding bits of x instead of all bits of x .

We consider 4 consecutive step functions, i.e. from step i to step $i + 3$. We show that, under certain conditions, we can move the last message word W_{i+3} to step i and move W_i to step $i + 1$ while keeping the final output after step $i + 3$ unchanged.

Assume we want to transfer upwards a message word W_{i+3} . Due to the absorption property of Ch, we can move W_{i+3} to step $i + 2$ (adding it to register G_{i+2}) if all the bits of E_{i+2} are fixed to 1. This is illustrated in Fig. 2 (left). Similarly, we can further move W_{i+3} to step $i + 1$ (adding it to register F_{i+1}) if all the bits of E_{i+1} are 0. Then, we still can move it upwards by transferring it to register E_i after step transformation in step i .

The same principle applies if we want to transfer only part of the register W_{i+3} . If l most significant bits (MSB) of W_{i+3} are arbitrary and the rest is set to zero (to avoid interference with addition on least significant bits), we need to fix l MSB of E_{i+2} to one and l MSB of E_{i+1} to zero.

As l MSB of E_{i+1} need to be 0, we need to use l MSB of W_i to satisfy this requirement. This reduces the space of W_i to 2^{32-l} . Similarly, we need to choose those W_i that fix l MSB of E_{i+2} to one. This is possible because changing the value of W_i influences the state of register E_{i+2} through Σ_1 at step $i + 1$. We experimentally checked that changing W_i generates changes in E_{i+2} that are sufficiently close to uniformly distributed. Satisfying additional constraints on l bits further reduces the space of W_i to 2^{32-2l} .

The important thing to note here is that if we fix the values of F_{i+1} , G_{i+1} and of the sum $D_{i+1} + H_{i+1}$ we can precompute the set of good values for W_i and store them in a table. Then, we can later recall them at negligible cost.

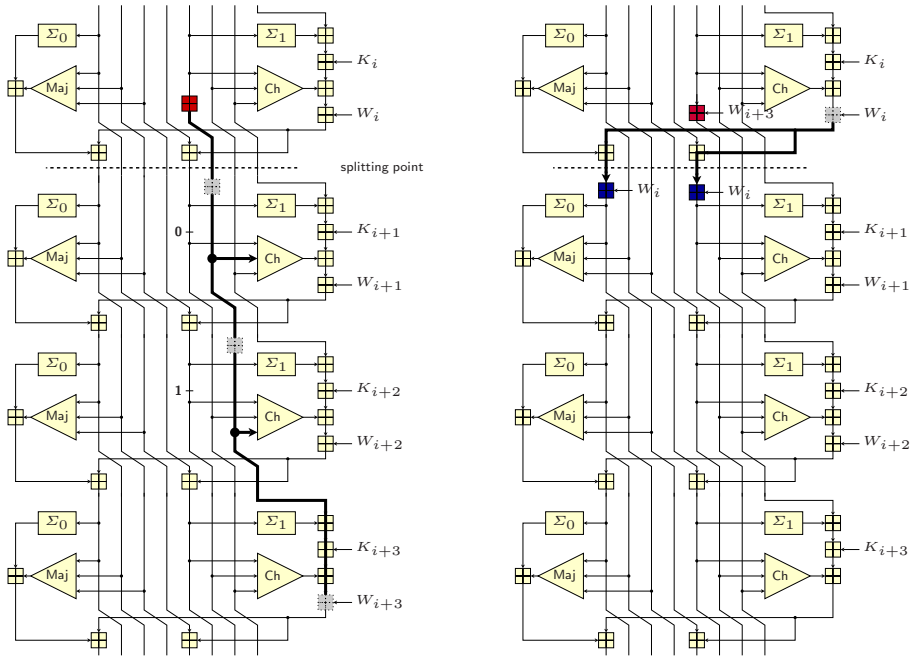


Fig. 2. Initial structure for SHA-2 allows to move the addition of W_{i+3} upwards provided that the Ch functions absorb the appropriate inputs (left); move W_i one step downwards (right)

On the other hand, message word W_i can be moved to step $i + 1$ with no constraint, as shown in Fig. 2 (right).

This procedure essentially swaps the order of words W_i and W_{i+3} .

4.7 Two-Way Expansion

Message expansion usually works in such a way that some consecutive several messages can determine the rest. For SHA-2, any consecutive 16 message words can determine the rest since the message expansion is a bijective mapping. This enables us to control any intermediate 16 message words and then expand the rest in both ways. This technique gives us more freedom of choices of neutral words, and extends the number of steps for the two chunks a lot. Note that the maximum number of consecutive steps for the two chunks is 30 for SHA-2. Since the message expansion is a bijective mapping, no matter which neutral word is chosen, it must be used to compute at least one of the any consecutive 16 message words. So each chunk of consecutive steps is of length at most 15.

4.8 Message Compensation

For some choice of neutral words, two chunks are not able to achieve the optimal length. By forcing some of the other message words to cancel the change introduced by neutral words, the optimal or near-optimal length could be achieved.

Combining the initial structure, two-way expansion and message compensation techniques, we are able to find two chunks of length 33. We choose to control on $\{W_z, \dots, W_{z+15}\}$, for some z which we will determine later. We choose W_{z+5} and W_{z+8} as neutral words. We show the first chunk $\{W_{z-10}, \dots, W_{z+4}, W_{z+8}\}$ to be independent from W_{z+5} and second chunk $\{W_{z+5}, W_{z+6}, W_{z+7}, W_{z+9}, \dots, W_{z+22}\}$ to be independent from W_{z+8} . Note that W_{z+8} is “moved” to first chunk by method explained in initial structure. For forward direction, we need to show $\{W_{z-10}, \dots, W_{z-1}\}$ are independent from W_{z+5} when they are expanded from $\{W_z, \dots, W_{z+15}\}$.

$$W_{z-1} = W_{z+15} - \sigma_1(W_{z+13}) - W_{z+8} - \sigma_0(W_z) , \tag{11}$$

$$W_{z-2} = W_{z+14} - \sigma_1(W_{z+12}) - W_{z+7} - \sigma_0(W_{z-1}) , \tag{12}$$

$$W_{z-3} = W_{z+13} - \sigma_1(W_{z+11}) - W_{z+6} - \sigma_0(W_{z-2}) , \tag{13}$$

$$W_{z-4} = W_{z+12} - \sigma_1(W_{z+10}) - \mathbf{W}_{z+5} - \sigma_0(W_{z-3}) , \tag{14}$$

$$W_{z-5} = W_{z+11} - \sigma_1(W_{z+9}) - W_{z+4} - \sigma_0(W_{z-4}) , \tag{15}$$

$$W_{z-6} = W_{z+10} - \sigma_1(W_{z+8}) - W_{z+3} - \sigma_0(W_{z-5}) , \tag{16}$$

$$W_{z-7} = W_{z+9} - \sigma_1(W_{z+7}) - W_{z+2} - \sigma_0(W_{z-6}) , \tag{17}$$

$$W_{z-8} = W_{z+8} - \sigma_1(W_{z+6}) - W_{z+1} - \sigma_0(W_{z-7}) , \tag{18}$$

$$W_{z-9} = W_{z+7} - \sigma_1(\mathbf{W}_{z+5}) - W_z - \sigma_0(W_{z-8}) , \tag{19}$$

$$W_{z-10} = W_{z+6} - \sigma_1(W_{z+4}) - W_{z-1} - \sigma_0(W_{z-9}) . \tag{20}$$

We note that W_{z+5} is used in (19) and (14), we compensate them by using W_{z+7} and W_{z+12} . By “compensating” we mean making the equation value independent from W_{z+5} by forcing $W_{z+7} - \sigma_1(W_{z+5}) = C$ (C is some constant, we use 0 for simplicity) and $W_{z+12} - W_{z+5} = C$. W_{z+7} is also used in (17), however we can use W_{z+9} to compensate for it, i.e. set $W_{z+9} = \sigma_1(W_{z+7}) = \sigma_1^2(W_{z+5})$. Then W_{z+9} and W_{z+12} are used in steps above, so we continue this recursively and finally have the following constraints that ensure the proper compensation of values of W_{z+5} .

$$\begin{aligned} W_{z+7} &= \sigma_1(W_{z+5}) , \\ W_{z+9} &= \sigma_1^2(W_{z+5}) , \\ W_{z+11} &= \sigma_1^3(W_{z+5}) , \\ W_{z+13} &= \sigma_1^4(W_{z+5}) , \\ W_{z+15} &= \sigma_1^5(W_{z+5}) , \\ W_{z+12} &= W_{z+5} , \\ W_{z+14} &= 2\sigma_1(W_{z+5}) . \end{aligned} \tag{21}$$

The second chunk is independent from W_{z+8} automatically without any compensation. The 33-step two-chunk is valid regardless of the choice of z as long as $z > 10$. To simplify the notation, we use W_j, \dots, W_{j+32} to denote the two chunks, then W_{j+15} and W_{j+18} are the two neutral words. We reserve the final choice of j for later to pick the one that allows to attack the most steps, as described later.

5 Preimage Attack against 43 Steps SHA-256

5.1 Number of Attacked Steps

The attack on SHA-256 uses 33-step two-chunk W_j, \dots, W_{j+32} explained in Section 4. Hence, in forward direction, p_{j+33} can be computed independently of the other chunk and in backward direction, p_j can be computed independently of the other chunk. We extend the number of attacked steps as much as possible with partial-fixing (PF) and indirect-partial-matching (IPM) techniques.

Forward computation of A_{j+34} : The equation for A_{j+34} is as follows.

$$\begin{cases} A_{j+34} = \Sigma_0(A_{j+33}) + \text{Maj}(A_{j+33}, B_{j+33}, C_{j+33}) + H_{j+33} \\ \quad + \Sigma_1(E_{j+33}) + \text{Ch}(E_{j+33}, F_{j+33}, G_{j+33}) + K_{j+33} + W_{j+33}, \\ W_{j+33} = \sigma_1(W_{j+31}) + W_{j+26} + \sigma_0(W_{j+18}) + W_{j+17} \end{cases}$$

We can use either PF or IPM to compute A_{j+34} . If we use PF, we fix the lower l bits of W_{j+18} , which is a neutral word for the other chunk. According to Table 2, this fixes the lower $l - 18$ bits of $\sigma_0(W_{j+18})$. Finally, the lower $l - 18$ bits of A_{j+34} can be computed. If we use IPM, we describe A_{j+34} as a sum of functions of each neutral words i.e. $A_{j+34} = \psi_F(W_{j+15}) + \xi_F(W_{j+18})$. From the above equations, they can be easily done. Note that IPM is more efficient than PF with respect to only computing A_{j+34} because IPM does not need to fix a part of neutral word.

Forward computation of A_{j+35} : The equation for A_{j+35} is as follows.

$$\begin{cases} A_{j+35} = \Sigma_0(A_{j+34}) + \text{Maj}(A_{j+34}, B_{j+34}, C_{j+34}) + \dots + W_{j+34}, \\ W_{j+34} = \sigma_1(W_{j+32}) + W_{j+27} + \sigma_0(W_{j+19}) + W_{j+18} \end{cases}$$

Neither PF nor IPM can compute A_{j+35} . If we used PF for A_{j+34} , only the lower $l - 18$ bits are known. This makes all bits of A_{j+35} unknown after the computation of $\Sigma_0(A_{j+34})$. If we used IPM, A_{j+34} is described as a sum of two independent functions. However, because Σ_0 consists of XOR of three self-rotations, it seems difficult to describe $\Sigma_0(A_{j+34})$ as a sum of two independent functions.

In summary, we can skip only 1 step in forward. In this case, using IPM is more efficient than using PF.

Backward computation of H_{j-1} : The equation for H_{j-1} is as follows.

$$\begin{cases} H_{j-1} = A_j - (\Sigma_0(B_j) + \text{Maj}(B_j, C_j, D_j)) \\ \quad - \Sigma_1(F_j) - \text{Ch}(F_j, G_j, H_j) - K_{j-1} - W_{j-1}, \\ W_{j-1} = W_{j+15} - \sigma_1(W_{j+13}) - W_{j+8} + \sigma_0(W_j) \end{cases}$$

We can use either PF or IPM to compute H_{j-1} . If we use PF, we fix the lower l bits of W_{j+15} , and then, the lower l bits of H_{j-1} can be computed. If we use IPM, we describe H_{j-1} as a sum of functions of each neutral word.

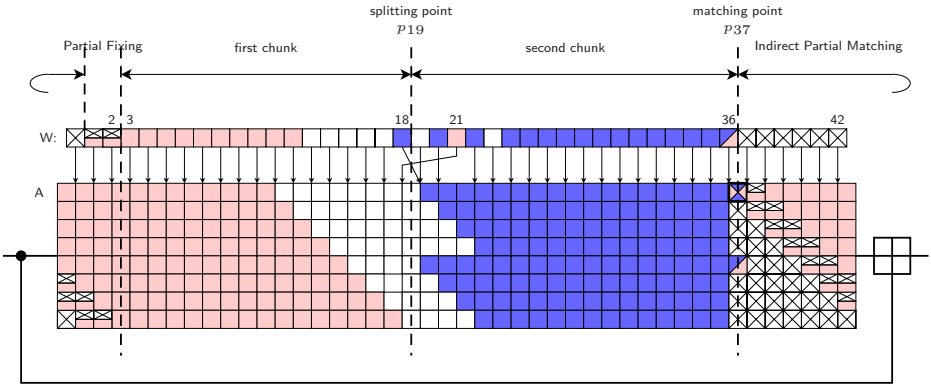


Fig. 3. Separation of chunks and dependencies of state words for SHA-256

Backward computation of H_{j-2} : The equation for H_{j-2} is as follows.

$$\begin{cases} H_{j-2} = A_{j-1} - (\Sigma_0(B_{j-1}) + \text{Maj}(B_{j-1}, C_{j-1}, D_{j-1})) \\ \quad - \Sigma_1(F_{j-1}) - \text{Ch}(F_{j-1}, G_{j-1}, H_{j-1}) - K_{j-2} - W_{j-2}, \\ W_{j-2} = W_{j+14} - \sigma_1(W_{j+12}) - W_{j+7} + \sigma_0(W_{j-1}) \end{cases}$$

We can use PF to compute H_{j-2} but cannot use IPM. To describe $\text{Ch}(F_{j-1}, G_{j-1}, H_{j-1})$ and $\sigma_0(W_{j-1})$ as a sum of two independent functions seems difficult. If we used PF for H_{j-1} , we can obtain the lower l bits of $\text{Ch}(F_{j-1}, G_{j-1}, H_{j-1})$ and lower $l - 18$ bits of $\sigma_0(W_{j-1})$. Finally, we can compute the lower $l - 18$ bits of H_{j-2} .

By the similar analysis, we confirmed that we cannot compute H_{j-3} . In summary, we can skip 2 steps in backward with PF which fixes the lower $l, l > 18$ bits of W_{j+15} .

The attack uses 33-step two-chunk W_j, \dots, W_{j+32} including 4-step initial structure. Apply PF for W_{j-1} and W_{j-2} , and apply IPM for W_{j+34} . Finally, 43 steps are attacked by skipping additional 7 steps using partial-matching technique.

36 steps (W_{j-2} to W_{j+34}) must be located sequentially. We have several options for j . We choose $j = 3$ for the following two purposes; (1) W_{13}, W_{14} , and W_{15} can be freely chosen to satisfy message padding rules, (2) pseudo-preimage attack on SHA-224 is possible (explained in Section 7).

We need to fix the lower $l + 18$ bits of W_{18} to fix the lower l bits of W_2 by PF. Besides, we lose half of remaining freedom to construct 4-step initial structure. Hence, we choose l to balance $l - 18$ and $\frac{32-l}{2}$, i.e. we choose $l = 23$.

The overview of the separation of chunks is shown in Fig. 3. \square denotes variables depending only on W_{21} ; \blacksquare denotes variables depending only on W_{18} ; \blacklozenge and \blacktriangleright denote registers that can be expressed as a sum modulo 2^{32} of two independent functions of neutral variables W_{18} and W_{21} ; \boxtimes denotes registers with few bits depending only on W_{21} ; \boxplus denotes registers depending on both W_{18} and W_{21} in a complicated way.

5.2 Attack Procedure

1. Randomly choose the values for internal chaining variable p_{19} (after the movement of message words by initial structure) and message word W_{19} . Randomly fix the lower 23 bits of W_{18} . By using the remaining 9 free bits of W_{18} , find 2^5 values on average that correctly construct the 4-step initial structure, and store them in the table T_W . Let us call this an initial table-preparation.
2. Randomly choose message words not related to initial structure and neutral words, i.e. $W_{13}, W_{14}, W_{15}, W_{16}, W_{17}, W_{23}$. Let us call this an initial configuration.
3. For all 2^5 possible W_{18} in T_W , compute the corresponding $W_{20}, W_{22}, W_{24}, W_{25}, W_{26}, W_{27}, W_{28}$ as shown in equations (21). Compute forward and find $\psi_F(W_{18})$. Store the pairs $(W_{18}, \psi_F(W_{18}))$ in a list L_F .
4. For all 2^4 possible values (the lower 4 bits) of W_{21} , compute backward and find $\xi_F(W_{21})$, which is $\sigma_0(W_{21})$ in this attack, and the lower 4 bits of A_{37} .
5. Compare the lower 4 bits of $A_{37} - \sigma_0(W_{21})$ and the lower 4 bits of $\psi_F(W_{18})$ stored in L_F .
6. If a match is found, compute $A_{37}, B_{37}, \dots, H_{37}$ with the corresponding W_{18} and W_{21} and check whether results from both directions match each other. If they do, output p_0 and W_0, \dots, W_{15} as a pseudo-preimage.
7. Repeat steps 2 – 6 for all possible choices of $W_{13}, W_{16}, W_{17}, W_{21}$. Note, the MSB of W_{13} is fixed to 1 to satisfy message padding. Hence, we have 2^{127} freedom for this step.
8. If no freedom remains in step 7, repeat steps 1 – 7.
9. Repeat steps 1 – 8 2^4 times to obtain 2^4 pseudo-preimages. Then, convert them to a preimage according to [21, Fact.9.99].

5.3 Complexity Estimation

We assume the complexity for 1 step function and 1-step message expansion is $\frac{1}{43}$ compression function operation of 43-step SHA-256. We also assume that the speed of memory access is negligible compared to computation time for step function and message expansion. Complexity for step 1 is 2^9 and use a memory of 2^5 words. Complexity for step 2 is negligible. In step 3, we compute $p_{j+1} \leftarrow R_j(p_j, W_j)$ for $j = 18, 19, \dots, 36$ and corresponding message expansion. Hence, the complexity is $2^5 \frac{19}{43}$. We use a memory of $2^5 \times 2$ words. Similarly, in step 4, we compute $p_j \leftarrow R_{j+1}^{-1}(p_{j+1}, W_j)$ for $j = 20, 19, \dots, 2$ and 6 more steps for partial-fixing and partial-matching. Hence, the complexity is $2^4 \frac{25}{43}$. In step 5, we compare the match of lower 4 bits of $2^9 (= 2^4 \cdot 2^5)$ items. Hence, 2^5 results will remain. Complexity for step 6 is $2^5 \frac{8}{43}$ and the probability that all other bits match is 2^{-252} . Hence, the number of remaining pair becomes $2^{-247} (= 2^5 \cdot 2^{-252})$. So far, the complexity from step 2 to 6 is $2^5 \frac{19}{43} + 2^4 \frac{25}{43} + 2^5 \frac{8}{43} = 2^5 \frac{39.5}{43} \approx 2^{4.878}$. In step 7, this is repeated 2^{127} times and its complexity is $2^{131.878}$. Step 8 is computed 2^{120} times. This takes $2^{120} \cdot (2^9 + 2^{131.878}) \approx 2^{251.9}$. This is the complexity of the pseudo-preimage attack on SHA-256 43-steps. Finally, at Step 9, preimages are

found with a complexity of $2^{1+(251.878+256)/2} = 2^{254.939} \approx 2^{254.9}$. The required memory for finding a pseudo-preimage is 2^5 words and $2^5 \times 2$ words in Steps [11](#) and [13](#), which is $2^5 \times 3$ words. For finding a preimage, we need to store $2^{1.9}$ pseudo-preimages for unbalanced meet-in-the-middle. This requires a memory of $2^{1.9} \times 24$ words.

5.4 Attack on 42 Steps SHA-256

When we attack 42 steps, We use 1-step IPM instead of 2-step PF in backward. This allows the attacker to use more message freedom. We choose $l = 10$ so that l and $\frac{32-l}{2}$ are balanced. Because each chunk has at least 10 free bits, the complexity for finding pseudo-preimages is approximately $2^{246} (= 2^{256} \cdot 2^{-10})$. The precise evaluation is listed in Table [11](#).

6 Preimage Attack against 46 Steps SHA-512

6.1 Basic Strategy for SHA-512

For SHA-512, we can attack more steps than SHA-256 by using PF. This occurs by the following two properties;

- Message-word size of SHA-512 is bigger than that of SHA-256. Hence, the bit-mixing speed of σ and Σ functions are slower than SHA-256.
- The choice of three rotation numbers for the σ_0 function is very biased.

To consider the above, we determine to use the message freedom available to the attacker for applying PF as much as possible.

Construction of the 4-step initial structure explained in Section [4](#) consumes a lot of message freedom. Therefore, we do not use the 4-step initial structure for SHA-512. Construction of the 3-step initial structure also needs a lot of message freedom. On the other hand, 2-step initial structure does not consume any message freedom because we do not have to control Ch functions. Finally, in our attack, we use a 31-step two-chunk including 2-step initial structure. Because construction of 2-step initial structure is much simpler than that of 4-step initial structure, we omit the detailed explanation of the construction.

6.2 Chunk Separation

The 31 message words we use are W_j to W_{j+30} . We apply the 2-step initial structure for W_{j+15} and W_{j+16} , hence the neutral words for the first chunk is W_{j+16} and for the second chunk is W_{j+15} . Whenever we change the value of W_{j+16} , we change the values of $W_{j+7}, W_{j+6}, \dots, W_j$ by message compensation technique so that the change does not impact to the second chunk. Similarly, whenever we change W_{j+15} , we change $W_{j+17}, W_{j+19}, W_{j+21}, W_{j+22}, \dots, W_{j+30}$. Finally, W_j to W_{j+30} can form the 31-step two-chunks.

6.3 Partial-Fixing Technique

We skip 6 steps in backward and 2 steps in forward by PF. Namely, we need to partially compute $W_{j-1}, W_{j-2}, \dots, W_{j-6}$ independently of W_{j+15} , and partially compute W_{j+31} and W_{j+32} independently of W_{j+16} . The equations for these message words are as follows.

$$\begin{aligned} \underline{W_{j-1}_l} &= \underline{W_{j+15}_l} - \sigma_1(W_{j+13}) - W_{j+8} + \sigma_0(W_j), \\ \underline{W_{j-2}_{l-8}} &= \underline{W_{j+14}} - \sigma_1(W_{j+12}) - W_{j+7} + \sigma_0(\underline{W_{j-1}_l}), \\ \underline{W_{j-3}_{l-16}} &= \underline{W_{j+13}} - \sigma_1(W_{j+11}) - W_{j+6} + \sigma_0(\underline{W_{j-2}_{l-8}}), \\ \underline{W_{j-4}_{l-24}} &= \underline{W_{j+12}} - \sigma_1(W_{j+10}) - W_{j+5} + \sigma_0(\underline{W_{j-3}_{l-16}}), \\ \underline{W_{j-5}_{l-32}} &= \underline{W_{j+11}} - \sigma_1(W_{j+9}) - W_{j+4} + \sigma_0(\underline{W_{j-4}_{l-24}}), \\ \underline{W_{j-6}_{l-40}} &= \underline{W_{j+10}} - \sigma_1(W_{j+8}) - W_{j+3} + \sigma_0(\underline{W_{j-5}_{l-32}}), \\ \\ \underline{W_{j+31}_{l-8}} &= \sigma_1(W_{j+29}) + W_{j+24} + \sigma_0(\underline{W_{j+16}_l}) + W_{j+15}, \\ \underline{W_{j+32}_l} &= \sigma_1(W_{j+30}) + W_{j+25} + \sigma_0(W_{j+17}) + \underline{W_{j+16}_l}. \end{aligned}$$

Remember Table 2. If the lower l bits of input of σ_0 is fixed, we can compute the lower $l - 8$ bits of its output. In backward, if we fix the lower l bits of W_{j+15} , the lower l bits of W_{j-1} , the lower $l - 8$ bits of W_{j-2} , the lower $l - 16$ bits of W_{j-3} , the lower $l - 24$ bits of W_{j-4} , the lower $l - 32$ bits of W_{j-5} , and the lower $l - 40$ bits of W_{j-6} can become independent of the second chunk. This results in computing the lower l bits of H_{j-1} , the lower $l - 8$ bits of H_{j-2} , the lower $l - 16$ bits of H_{j-3} , the lower $l - 41$ bits of H_{j-4} , the lower $l - 49$ bits of H_{j-5} , and the lower $l - 57$ bits of H_{j-6} . Note that we also need to consider Σ_1 to compute H_{j-4} , H_{j-5} , and H_{j-6} . If we fix the lower l bits of W_{j+16} , the lower $l - 8$ bits of W_{j+31} , and the lower l bits of W_{j+32} can become independent of the first chunk. This results in computing the lower $l - 8$ bits of A_{j+32} , and the lower $l - 47$ bits of A_{j+33} .

Therefore, if we choose $l = 60$, we can match the lower 3 bits of H_{j-6} and 13 bits of A_{j+33} after we skip 7 steps by the partial-matching technique.

6.4 Attack Overview

The attack uses 31-step two-chunk W_j, \dots, W_{j+30} including 2-step initial structure. Apply PF for $W_{j-1}, W_{j-2}, \dots, W_{j-6}$, and W_{j+31}, W_{j+32} . Finally, 46 steps are attacked by skipping additional 7 steps using partial-matching technique.

39 steps (W_{j-6} to W_{j+32}) must be located sequentially. Because $W_{j+8}, W_{j+9}, W_{j+10}, W_{j+11}, W_{j+12}, W_{j+13}, W_{j+14}, W_{j+18}, W_{j+20}$ are the message words we fix in advance, we choose $j = 6$ so that W_{14} and W_{15} can be chosen to satisfy message padding rules. The MSB of W_{13} can also be satisfied. In this chunk separation, W_{j+7} can be described as $W_{j+7} = \text{Const} - W_{j+16}$, where Const is a chosen fixed value and the lower l bits of W_{j+16} are fixed. If we fix Const and the MSB of W_{j+16} to 0 and some value, respectively, and choose the lower l bits of W_{j+16} so that the MSB of $-W_{j+16}$ does not change for all active bits of W_{j+16} , we can always fix the MSB of W_{j+7} .

The number of free bits in W_{j+16} is 3. ($l = 60$ but we fix the MSB for satisfying padding for W_{13} .) The number of free bits in W_{j+15} is 4. Results from both chunks are compared with 3 bits. Therefore, the final complexity of pseudo-preimage attack is approximately 2^{509} . This is converted to a preimage attack whose complexity is approximately $2^{511.5}$. For finding pseudo-preimages, this attack needs to store 2^3 items. Hence, the required memory is $2^3 \times 9$ words. For finding preimages, we need to store $2^{1.5}$ pseudo-preimages for unbalanced meet-in-the-middle. This requires a memory of $2^{1.5} \times 24$ words.

6.5 Attack on 42 Steps SHA-512

When we attack 42 steps, we stop using 1-step PF in forward and 3-step PF in backward. We choose $l = 40$. Because each chunk has at least 24 free bits, the complexity for finding pseudo-preimages is approximately $2^{488} (= 2^{512} \cdot 2^{-24})$. The precise evaluation is listed in Table [11](#).

7 Remarks

7.1 Length of Preimages

The preimages are of at least two blocks, last block is used to find pseudo-preimages and the second last block links to the input chaining of last block. Two block preimages is only possible if we can preset the message words used for encoding the length (m_{14} and m_{15} for SHA-2) of last block according to the padding and length encoding rules. In our case, this can be done in the first step of the algorithm. On the other hand, we can leave m_{14} and m_{15} as random, later we can still resolve the length using expandable messages [\[22\]](#).

7.2 SHA-224 and SHA-384

Our attack on 43 steps SHA-256 can also produce pseudo-preimages for SHA-224 by using the approach by Sasaki [\[23\]](#). In our attack, we match 4-bits of A_{37} which is essentially equivalent to G_{43} . Then, we repeat the attack until other registers randomly match i.e. we wait until $A_{43}, B_{43}, \dots, F_{43}$, and H_{43} randomly match. In SHA-224, the value of H_{43} is discarded in the output. Hence, we do not have to care the match of H_{43} , which results in decreasing the complexity by 2^{32} bits. Hence, pseudo-preimages of SHA-224 can be computed with a complexity of $2^{219.9} (= 2^{251.9} \cdot 2^{-32})$. Note, this cannot be converted to a preimage attack on SHA-224 because the size of intermediate chaining variable is 256 bits.

If we apply our attack on SHA-512 to SHA-384, W_{13}, W_{14} , and W_{15} will depend on neutral words. Hence, we cannot confirm 46 steps SHA-384 can be attacked or not because of padding problem. However, 43 steps SHA-384 can be attacked by using the same chunk as SHA-256. By considering the difference of word size and application of PF, we can optimize the complexity by choosing $l = 27$ so that $l - 8$ and $\frac{64-l}{2}$ are balanced.

7.3 Multi-preimages and Second-Preimages

We note that the method converting pseudo-preimage to preimages can be further extended to find multi-preimages. We find first k block multi-collisions [24], then follow the expandable message [22] to link to the final block. This gives 2^k multi-preimages with additional $k2^{n/2}$ computations, which is negligible when k is much smaller than $2^{(n-t)/2}$ (t denotes number of bits for each chunk, refer to Section 3). We need additional $128k$ bytes of memory to store the k block multi-collisions. Furthermore, most of the message words are randomly chosen, this attack naturally gives second preimages with high probability. Above multi-preimages are most probably multi-second preimages.

8 Conclusions

In this paper, we presented preimage attacks on 43 steps SHA-256 and 46 steps SHA-512. The time complexity of the attack for 43-step SHA-256 is $2^{254.9}$ and it requires $2^5 \cdot 3$ words of memory. The time complexity of the attack for 46-step SHA-512 is $2^{511.5}$ and it requires $2^3 \cdot 9$ words of memory. The number of attacked steps is greatly improved from the best previous attack, in other words, the security margin of SHA-256 and SHA-512 is greatly reduced. Because SHA-256 and SHA-512 have 64 and 80 steps, respectively, they are currently secure.

An open question worth investigating would be to see if the current attacks may still be improved. Perhaps finding $15 + 4 + 15$ pattern of chunks with 4-step initial structure in the middle or using better partial-fixing technique that would utilize middle bits of the message word would extend the attacks.

The preimage attack we presented creates a very interesting situation for SHA-2 when a preimage attack, covering 43 or 46 steps, is much better than the best known collision attack, with only 24 steps. Our attack does not convert to collision attack because of the complexity above the birthday bound. However, we believe that the existence of such a preimage attack suggests that a collision attack of similar length could be also possible. In that light, the problem of finding collisions for reduced variants of SHA-256 definitely deserves more attention.

Acknowledgements

We would like to thank Lars R. Knudsen, Christian Rechberger, and the anonymous reviewers for the helpful comments. Jian Guo was supported by the Singapore Ministry of Education under Research Grant T206B2204. Krystian Matusiewicz was supported by grant 274-07-0246 from the Danish Research Council for Technology and Production Sciences.

References

1. U.S. Department of Commerce, National Institute of Standards and Technology: Secure Hash Standard (SHS) (Federal Information Processing Standards Publication 180-3) (2008),
http://csrc.nist.gov/publications/fips/fips180-3/fips180-3_final.pdf

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(212) Friday, November 2, 2007 Notices (2007), http://csrc.nist.gov/groups/ST/hash/documents/FR_Notice_Nov07.pdf
4. U.S. Department of Commerce, National Institute of Standards and Technology: NIST's Plan for Handling Tunable Parameters. Presentation by Souradyuti Paul at The First SHA-3 Candidate Conference (February 2009), <http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/Feb2009/>
5. Mendel, F., Pramstaller, N., Rechberger, C., Rijmen, V.: Analysis of step-reduced SHA-256. In: Robshaw, M.J.B. (ed.) FSE 2006. LNCS, vol. 4047, pp. 126–143. Springer, Heidelberg (2006)
6. Nikolić, I., Biryukov, A.: Collisions for step-reduced SHA-256. In: [25], pp. 1–15
7. Indestege, S., Mendel, F., Preneel, B., Rechberger, C.: Collisions and other non-random properties for step-reduced SHA-256. In: [26], pp. 276–293
8. Sanadhya, S.K., Sarkar, P.: New collision attacks against up to 24-step SHA-2 (extended abstract). In: Rijmen, V., Das, A., Chowdhury, D.R. (eds.) INDOCRYPT 2008. LNCS, vol. 5365, pp. 91–103. Springer, Heidelberg (2008)
9. Isobe, T., Shibutani, K.: Preimage attacks on reduced Tiger and SHA-2. In: Dunkelman, O. (ed.) FSE 2009. LNCS, vol. 5665, pp. 139–155. Springer, Heidelberg (2009)
10. Yu, H., Wang, X.: Non-randomness of 39-step SHA-256 (2008), <http://www.iacr.org/conferences/eurocrypt2008v/index.html>
11. Saarinen, M.J.O.: A meet-in-the-middle collision attack against the new FORK-256. In: Srinathan, K., Pandu Rangan, C., Yung, M. (eds.) INDOCRYPT 2007. LNCS, vol. 4859, pp. 10–17. Springer, Heidelberg (2007)
12. Leurent, G.: MD4 is not one-way. In: [25], pp. 412–428
13. Rivest, R.L.: Request for Comments 1321: The MD5 Message Digest Algorithm. The Internet Engineering Task Force (1992), <http://www.ietf.org/rfc/rfc1321.txt>
14. Zheng, Y., Pieprzyk, J., Seberry, J.: HAVAL — one-way hashing algorithm with variable length of output. In: Seberry, J., Zheng, Y. (eds.) AUSCRYPT 1992. LNCS, vol. 718, pp. 83–104. Springer, Heidelberg (1993)
15. Aoki, K., Sasaki, Y.: Preimage attacks on one-block MD4, 63-step MD5 and more. In: [26], pp. 103–119
16. 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 (2009)
17. 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)
18. Sasaki, Y., Aoki, K.: Preimage attacks on 3, 4, and 5-pass HAVAL. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 253–271. Springer, Heidelberg (2008)
19. Chang, D., Hong, S., Kang, C., Kang, J., Kim, J., Lee, C., Lee, J., Lee, J., Lee, S., Lee, Y., Lim, J., Sung, J.: ARIRANG. NIST home page: http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/submissions_rnd1.html
20. Hong, D., Kim, W.H., Koo, B.: Preimage attack on ARIRANG. Cryptology ePrint Archive, Report 2009/147 (2009), <http://eprint.iacr.org/2009/147>
21. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of applied cryptography. CRC Press, Boca Raton (1997)

22. 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)
23. Sasaki, Y.: Meet-in-the-middle attacks using output truncation in 3-pass HAVAL. In: Samarati, P., Yung, M., Martinelli, F. (eds.) ISC 2009. LNCS, vol. 5735, pp. 79–94. Springer, Heidelberg (2009)
24. Joux, A.: Multicollisions in iterated hash functions. Application to cascaded constructions. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 306–316. Springer, Heidelberg (2004)
25. Nyberg, K. (ed.): FSE 2008. LNCS, vol. 5086. Springer, Heidelberg (2008)
26. Avanzi, R., Keliher, L., Sica, F. (eds.): SAC 2008. LNCS, vol. 5381. Springer, Heidelberg (2009)

Fiat-Shamir with Aborts: Applications to Lattice and Factoring-Based Signatures

Vadim Lyubashevsky*

Department of Computer Science
Tel-Aviv University
Tel Aviv 69978, Israel
vlyubash@cs.ucsd.edu

Abstract. We demonstrate how the framework that is used for creating efficient number-theoretic ID and signature schemes can be transferred into the setting of lattices. This results in constructions of the most efficient to-date identification and signature schemes with security based on the worst-case hardness of problems in ideal lattices. In particular, our ID scheme has communication complexity of around 65,000 bits and the length of the signatures produced by our signature scheme is about 50,000 bits. All prior lattice-based identification schemes required on the order of millions of bits to be transferred, while all previous lattice-based signature schemes were either stateful, too inefficient, or produced signatures whose lengths were also on the order of millions of bits. The security of our identification scheme is based on the hardness of finding the approximate shortest vector to within a factor of $\tilde{O}(n^2)$ in the standard model, while the security of the signature scheme is based on the same assumption in the random oracle model. Our protocols are very efficient, with all operations requiring $\tilde{O}(n)$ time.

We also show that the technique for constructing our lattice-based schemes can be used to improve certain number-theoretic schemes. In particular, we are able to shorten the length of the signatures that are produced by Girault's factoring-based digital signature scheme ([\[10\]\[11\]\[31\]](#)).

1 Introduction

The appeal of building cryptographic primitives based on the hardness of lattice problems began with the seminal work of Ajtai who showed that one-way functions could be built with security based on the worst-case hardness of certain lattice problems [\[2\]](#). Unfortunately, cryptographic primitives that were built with this very strong security property were extremely inefficient for practical applications. For example, evaluating one-way and collision-resistant hash functions

* Supported by the Israel Science Foundation and by a European Research Council (ERC) Starting Grant. Some of the work in this paper was performed while the author was a student at the University of California, San Diego.

required $\tilde{O}(n^2)$ time and space [2,24], and in public-key cryptosystems, the keys were on the order of megabytes [3,33,34,28] (also see [25] for concrete parameter proposals for the scheme in [34]). Therefore some new ideas were required in order to make provably-secure lattice-based primitives a realistic alternative to ones based on number-theory.

A promising approach for improving efficiency is to use lattices that possess extra algebraic structure, and it is precisely this extra structure that makes the NTRU cryptosystem [14] (which unfortunately does not have a proof of security) very efficient in practice. A step in the direction of building *provably-secure* lattice-based primitives was taken by Micciancio [23], who showed that one could build *efficient* ($\tilde{O}(n)$ evaluation time) one-way functions with security based on the worst-case instances of problems pertaining to cyclic lattices (cyclic lattices are lattices that correspond to ideals in the ring $\mathbb{Z}[\mathbf{x}]/\langle \mathbf{x}^n - 1 \rangle$). This result was later extended to give constructions of collision-resistant hash functions by either restricting the domain [29] or changing the ring [18] in Micciancio's scheme. These works then led to constructions and implementations of collision-resistant hash functions [20] with security based on worst-case problems in lattices corresponding to ideals in $\mathbb{Z}[\mathbf{x}]/\langle \mathbf{x}^n + 1 \rangle$ whose performance was comparable to the performance of ad-hoc hash functions that are currently in use today. And because there is a very close connection between collision-resistant hash functions and more sophisticated primitives such as ID schemes and digital signatures, it was very natural to ask whether these primitives also had efficient lattice-based constructions. There has been some recent work in this direction, which we will now describe.

Lyubashevsky and Micciancio constructed a one-time signature in which signing and verification can be performed in time $\tilde{O}(n)$ [19]. Using standard techniques, the one-time signature can be transformed into a full-fledged signature scheme using a signature-tree [21,22] with only an additional work factor of $O(\log n)$. While this combination results in a very theoretically-appealing scheme where all the operations take time $\tilde{O}(n)$, it does require the use of a tree, which is a somewhat unwanted feature in practice. Another signature scheme was proposed by Gentry et al. in [9]. Their signature scheme follows the hash-and-sign paradigm, and when instantiated with algebraic lattices [37], verification takes time $\tilde{O}(n)$, but $\tilde{O}(n^4)$ time is needed to do the signing (it is plausible that the signing time could be reduced to $\tilde{O}(n^2)$ with a more careful analysis).

A different way of constructing digital signature schemes is to first construct an identification scheme of a certain form and then convert it to a signature scheme using the Fiat-Shamir transform [7,32,11]. The identification schemes of Micciancio and Vadhan [26], Lyubashevsky [17], and Kawachi et al. [15] can all be instantiated such that the secret and public keys are of size $\tilde{O}(n)$, and the entire interaction takes $\tilde{O}(n)$ time as well. While these constructions seem essentially optimal, they contain a common inefficiency. The ID schemes all have the form of standard commit-challenge-response protocols (see Figure 1, for an example of one where Y is the commitment, c is the challenge, and z is the response), and the inefficiency lies in the fact that for each challenge bit,

the response consists of $\tilde{O}(n)$ bits. Since the security of the protocol is directly connected to the number of challenge bits sent by the verifier, it means that for every bit of security, $\tilde{O}(n)$ bits need to be transmitted. Theoretically, this does not cause a problem because one only needs $\omega(\log n)$ bits of security in order for the protocol to be considered secure against polynomial-time adversaries, and then the total running time of the above protocols is still $\tilde{O}(n)$. But in practice, this is a rather unsatisfactory solution because one wants some concrete security guarantee, say 80 bits, and then the communication complexity of the ID scheme will be about 80 times larger (the size of the signature in the derived signature scheme would be 160 times larger) than possibly necessary. This is in sharp contrast to number-theoretic ID schemes where the response of the prover is longer than the challenge by only a small factor.

What allows number-theoretic ID schemes like Schnorr [35], GQ [13], Girault [10], Okamoto [27], etc. to be so “compact” is that the challenge string in these protocols is not treated as a sequence of independent 0’s and 1’s, but instead the entire string is interpreted as an integer from a certain domain. This can be done because there is a lot of underlying algebraic structure upon which these schemes are built. On the other hand, lattices do not seem to have as much algebraic underpinning, and so the schemes based on them are very combinatorial in nature which is why the challenge strings are treated simply as a sequence of independent challenges much like in generic zero-knowledge proofs for NP. The main accomplishment of the current work is to show how to exploit the limited algebraic structure of ideal lattices in order to use the challenge bits collectively rather than individually, which ends up greatly improving the practical efficiency of lattice-based identification and signature schemes.

1.1 Contributions and Comparisons

Lattice-based constructions. We construct a lattice-based ID scheme in which the challenge string is treated as a polynomial in a certain ring, and one correct response to it from the prover is enough for authentication. The caveat is that some constant fraction of the time, the prover cannot respond to the challenge from the verifier and must abort the protocol. The result of this is that the “commit” and “challenge” steps of the ID scheme now must be repeated several times to ensure that a valid prover is accepted with some decent probability. But using standard techniques, one can significantly shorten the length of the “commit” part of the protocol, and because of the structure of our scheme, the challenge can always be the same. Therefore the number of transmitted bits is dominated by the length of the single “response”.

Even more optimizations are possible when converting the ID scheme into a signature scheme using the Fiat-Shamir transform. In the resulting signature scheme, there is of course no longer any interaction until the signer outputs the signature. And therefore there is no need for the signer to output the attempts in which he failed to sign (which correspond to the times he couldn’t answer the challenge in the ID scheme). So while the failures do cost time, the length of the final signature is as short as it would have been if the signer only attempted

to sign once and succeeded. And because the probability of failure is a small constant ($\approx 2/3$), we only expect to repeat the signature protocol 3 times before succeeding.

All operations in our scheme take time $\tilde{O}(n)$ and we prove that the ID and signature schemes are secure based on the worst-case hardness of approximating the shortest vector to within a factor of $\tilde{O}(n^2)$ in lattices corresponding to ideals in the ring $\mathbb{Z}[\mathbf{x}]/\langle \mathbf{x}^n + 1 \rangle$ (the security of the signature scheme is in the random oracle model). Compared to previous works, our asymptotic hardness assumption is the same as that in [19,17] (although the scheme of [19] is secure in the standard model), but is worse than that in [26,9,37] (where the factor is $\tilde{O}(n^{1.5})$) and in [15] (where the factor is $\tilde{O}(n)$).

Based on the work of Gama and Nguyen [8] who worked out the effectiveness of current state-of-the-art lattice reduction algorithms, we present some concrete parameters with which our schemes can be instantiated. On the low end, the outputted signatures are about 50000 bits in length (the ID scheme requires about 65000 bits to be transmitted). While the scheme of [15] has better asymptotic security, the response to each challenge bit seems to require at least 10000 bits. So if we would like the challenge to be 160 bits for security purposes, the response (and therefore the signature size) will be over a million bits. The signature schemes of [26] and [17] look like they would have their signatures be about 160 times longer than ours (the ID schemes would require communications that are about 80 times longer), again because the responses are done separately to every challenge bit. So even though our ID and signature schemes have worse asymptotic security, their structure makes them much more practically efficient.

At this point it is not possible to give an accurate comparison of our signature scheme to the hash-and-sign signature schemes [9,37] because no concrete parameters were given for those schemes. But independent of the signature sizes, our scheme will still have the advantage in that signing can be done in time $\tilde{O}(n)$ rather than $\tilde{O}(n^4)$.

The signature length of the one-time signature in [19] may actually be a little shorter than in our scheme, but this advantage is lost when the one-time signature gets converted to a general stateless signature scheme. If a signature tree is used in the conversion, then the signature length may go up by a factor of the tree depth, which would make it much less efficient. On the other hand, one could build a hash tree using any collision-resistant hash function, and then the signatures would only increase by the product of the tree depth and the hash function output. If the scheme is to be completely stateless and support about 2^{60} signatures, and we use SHA-256 as the hash function, then the length of the one-time signature in [19] would increase by about 15,000 bits, which would make it somewhat longer than our signature. The similarity between the signature sizes of our scheme and the scheme in [19] is no coincidence, and we further discuss the relationship between the two schemes in Section 1.2.

Factoring-based signatures. We show that the ideas used to construct our lattice-based digital signature can also be used for shortening the length of some number-theoretic schemes. The signature scheme originally proposed by Girault

[10], and analyzed in [11,31] is a scheme whose security, in the random oracle model, is based on the hardness of factorization. What is particularly attractive about it is that if the signer can do some pre-computing before receiving the message, then signing can be done with just one random oracle query, one multiplication, and one addition over the integers (no modular reduction is required). We show how to reduce the length of the signature in an instantiation of the scheme due to Pointcheval [31] from 488 bits to 425.

1.2 Techniques

There is a pattern that emerges when looking at constructions of certain ID and signature schemes based on the hardness of factoring and discrete log. The informal chain of reductions from the hard problem to the signature scheme looks as follows:

$$\text{Hard Problem} \leq \text{CRHF} \leq \text{One-time signature} \leq \text{ID scheme} \leq \text{Signature}$$

For example, finding collisions in the hash function $h(x) = g^x \bmod N$ implies being able to factor N . This can be converted into a one-time signature with the secret key being some pair of integers (x, y) , public keys being $h(x), h(y)$, and the signature of a message c being $xc + y$. The one-time signature can then be converted into an ID scheme by simply picking a new y every time (Figure 1) and c now being a challenge chosen by the verifier. The ID scheme can then be converted to a signature scheme by using the Fiat-Shamir transform which replaces the verifier with a random oracle (Figure 5) [10,11,31]. The same idea can be used with the hash function $h(x_1, x_2) = (g_1^{x_1} g_2^{x_2} \bmod p)$, in which finding collisions implies solving the discrete log problem. The ID and signature schemes resulting from that hash function are due to Okamoto [27].

It turns out that a somewhat similar approach can be used to build lattice-based primitives as well. The works of [29,18], showed a reduction from the *worst-case* problem of finding short vectors in algebraic lattices to finding collisions in hash functions. The work of [19] can be viewed as a transformation of the hash function to a one-time signature, and this current work can then be seen as the continuation of this chain of reductions where the one-time signature of [19] is converted into an ID scheme and then into a signature scheme.

But what prevents the techniques used in number-theoretic schemes to be directly extended to lattice-based ones, is that lattices allow for much less algebraic structure. For example, the domains in number-theoretic hash functions are rings, while in lattice-based ones, the domain is just a subset of a ring (in particular, those elements in the ring that have small Euclidean norm) that is neither closed under addition nor multiplication. This is very related to the fact that the factoring and discrete log problems can be reduced to finding an element in the kernel of some homomorphic function, while finding short vectors in lattices reduces to the problem of finding *small* elements in the kernel of a homomorphism. This difference is what seems to give lattice problems resistance against polynomial-time quantum algorithms that solve factoring and discrete log [36], but at the same time it also hinders constructions of lattice-based primitives.

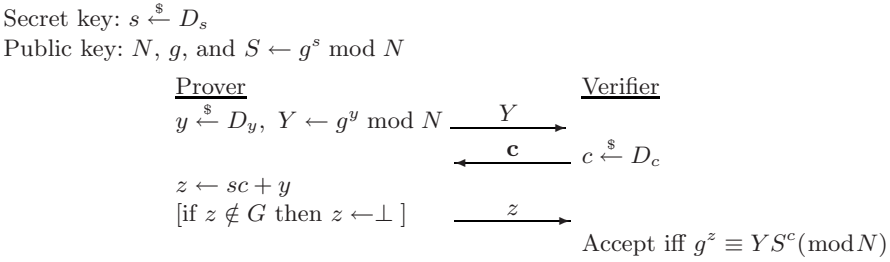


Fig. 1. Factoring-Based Identification Schemes. The parameters for this scheme are in Figure 6. The line in [] is only performed in the aborting version of the scheme.

In overcoming this limitation, the one-time signature of [19] had to leak parts of its secret key. While it wasn't a problem in that setting because the secret key is only used once, in ID schemes the same secret key is used over and over, and so leaking a part of the secret key every time would result in complete insecurity. In this paper, we solve this difficulty by using an aborting technique that was introduced in [17]. The idea behind aborting is that the prover can elect to abort the protocol in order to protect some information about his secret key (mainly, the protocol needs to remain witness-indistinguishable). In this work, we are able to relax the conditions that were needed for witness-indistinguishability in [17], and this allows us to construct much more efficient lattice-based protocols as well as extend the technique to other contexts, such as the factoring-based scheme described in Section 1.1. We essentially show that all that is needed for the aborting technique to be applicable is a collision-resistant homomorphic hash function that has small elements in its kernel. We believe that this technique can find further applications.

1.3 Intuition for Aborting

Understanding where aborting might be useful is best accomplished with an example. Consider the protocol in Figure 1 (for this discussion, it is not necessary to understand why the protocol works), which has the form of a typical 3-round commit-challenge-response ID scheme. The secret key is some s and the public key is $h(s)$ where h is a function that happens to be $h(s) = g^s \bmod N$ in our example. In the first step of the protocol, the prover picks a parameter y , and sends $h(y)$, to the verifier. The verifier picks a random "challenge" c , and sends it to the prover. The third step of the protocol consists of a response of the prover to the challenge. This response must somehow use the secret key, and in our example, the secret key s is multiplied by c and then added to y . Notice that sending sc without adding it to y would completely reveal s , and so the job of y is to somehow mask the exact value of sc . If the operation sc takes place in some finite group, then a natural idea for masking would be to pick y uniformly at random from that group. The intuition is that if nothing about y is known, then the value $y + sc$ is also completely random (of course, something is known

about y when the prover sends $h(y)$ to the verifier, but we gloss over that here). And this is exactly what is done in well-known ID schemes such as Schnorr [35], GQ [13], Okamoto [27], etc..

But sometimes it is infeasible to pick y uniformly at random from the group. In Girault's ID-scheme [10,11,31] (Figure 1), the multiplication sc is performed over the integers, which is an infinite group. A way to do masking in this scheme is to pick a y in a range that is much larger than the range of sc . So for example, if $0 \leq sc \leq R$, then one could pick a random y from the range $[0, \dots, 2^{64}R]$. Then, with very high probability, the value of $sc + y$ will be in $[R, \dots, 2^{64}R]$, in which case it will be impossible to determine anything about sc if nothing is known about y .

In constructing our lattice-based ID scheme, the same difficulty is encountered as in Girault's scheme, except we do not have the luxury of picking y (or something analogous to y in the lattice-based scheme) from such a large range because doing so would require us to make a much stronger complexity assumption which would significantly decrease the efficiency of the protocol (we would have to assume that it is hard to find a super-polynomial approximation of the shortest vector instead of just an $\tilde{O}(n^2)$ approximation). Our solution is to instead pick y from a much smaller set, something analogous to $[0, \dots, 2R]$, but only reveal $sc + y$ if it falls into the range $[R, \dots, 2R]$. If the range is picked carefully and the function h is a homomorphism that has "small" elements in its kernel, then one can show that if the prover only reveals values in this range and aborts otherwise, the protocol will be perfectly witness-indistinguishable. The witness-indistinguishability is then used to prove security of the protocol by showing that a forger can be used for extracting collisions in h .

The same technique can also be applied to Girault's scheme. Notice that if we pick y uniformly at random from the range $[0, \dots, 2R]$ instead of from $[0, \dots, 2^{64}R]$, the length of $sc + y$ will be 63 bits shorter. We point out that our aborting factoring-based ID scheme in Figure 1 which uses this idea is actually worse than the corresponding non-aborting one because the savings gained by shortening $sc + y$ are lost in case the prover has to abort and the ID protocol has to be repeated. But the advantage of aborting does show itself when the ID protocol is converted into a signature scheme using the Fiat-Shamir paradigm (Figure 5). In a signature scheme, there is no interaction, and therefore there is no need for the signer to ever include the aborted signing attempts into the final signature. So if the signer needs to abort, he simply reruns the protocol until he gets a signature in the correct range. The end result is that the eventual signature is shorter than it would have been in schemes such as [10,11,31] where the signer does not have the option to abort.

2 Preliminaries

2.1 Notation

We will denote vectors by bold letters. For convenience, vectors of vectors will be denoted by a bold letter with a hat. For example, if $\mathbf{a}_1, \mathbf{a}_2$ are elements of

\mathbb{Z}^n , then we can write $\hat{\mathbf{a}} = (\mathbf{a}_1, \mathbf{a}_2)$. The ℓ_∞ norm of \mathbf{a} is written as $\|\mathbf{a}\|_\infty$, and $\|\hat{\mathbf{a}}\|_\infty$ for $\hat{\mathbf{a}} = (\mathbf{a}_1, \dots, \mathbf{a}_m)$ is defined as $\max_i(\|\mathbf{a}_i\|_\infty)$. If S is a set, then $a \stackrel{\$}{\leftarrow} S$ means that a is chosen uniformly at random from S . All logarithms are assumed to be base 2.

2.2 Lattices and Algebra

An integer lattice Λ is a subgroup of \mathbb{Z}^n . The approximate Shortest Vector Problem ($\text{SVP}_\gamma(\Lambda)$) asks to find a vector \mathbf{v} in Λ such that $\|\mathbf{v}\|_\infty$ is no more than γ times larger than the vector in Λ with the smallest ℓ_∞ norm. In this work, we will be interested in lattices that exhibit an additional algebraic property – in particular, they correspond to ideals in the ring $\mathbb{Z}[\mathbf{x}]/\langle \mathbf{x}^n + 1 \rangle$. We will say that a lattice Λ is an $(\mathbf{x}^n + 1)$ -cyclic lattice if for every vector $(v_0, \dots, v_{n-2}, v_{n-1}) \in \Lambda$, the vector $(-v_{n-1}, v_0, \dots, v_{n-2})$ is also in Λ . If we look at the vectors as polynomials (i.e. $(v_0, \dots, v_{n-2}, v_{n-1})$ as $v_0 + \dots + v_{n-2}\mathbf{x}^{n-2} + v_{n-1}\mathbf{x}^{n-1}$), then an $(\mathbf{x}^n + 1)$ -cyclic lattice is an ideal in $\mathbb{Z}[\mathbf{x}]/\langle \mathbf{x}^n + 1 \rangle$ because in this ring,

$$(v_0 + \dots + v_{n-2}\mathbf{x}^{n-2} + v_{n-1}\mathbf{x}^{n-1}) \cdot \mathbf{x} = -v_{n-1} + v_0\mathbf{x} + \dots + v_{n-2}\mathbf{x}^{n-1}.$$

The ring that will be most important to us throughout the paper is the ring $\mathbb{Z}_p[\mathbf{x}]/\langle \mathbf{x}^n + 1 \rangle$ where p is some odd positive integer. The elements in $\mathbb{Z}_p[\mathbf{x}]/\langle \mathbf{x}^n + 1 \rangle$ will be represented by polynomials of degree $n - 1$ having coefficients in the range $[-\frac{p-1}{2}, \frac{p-1}{2}]$. Throughout the paper, we will treat polynomials in $\mathbb{Z}_p[\mathbf{x}]/\langle \mathbf{x}^n + 1 \rangle$ and vectors in \mathbb{Z}^n as the same data type. So when, for example, we talk of multiplying two vectors, we actually mean converting the vectors to polynomials and then multiplying the polynomials in $\mathbb{Z}_p[\mathbf{x}]/\langle \mathbf{x}^n + 1 \rangle$. Similarly, the norm¹ of a polynomial is just the norm of the corresponding vector. It’s not hard to see that for polynomials $\mathbf{v}, \mathbf{w} \in \mathbb{Z}_p[\mathbf{x}]/\langle \mathbf{x}^n + 1 \rangle$, the following relation holds:

$$\|\mathbf{v}\mathbf{w}\|_\infty \leq \|\mathbf{v}\|_\infty \|\mathbf{w}\|_1 \leq n\|\mathbf{v}\|_\infty \|\mathbf{w}\|_\infty$$

$(\mathbf{x}^n + 1)$ -cyclic lattices are a particular class of lattices that received attention because one can construct efficient and provably secure cryptographic primitives based on the hardness of finding approximate short vectors in these lattices [18,29,19,20]. The main reason for this efficiency is that the multiplication of two polynomials in $\mathbb{Z}_p[\mathbf{x}]/\langle \mathbf{x}^n + 1 \rangle$ can be done in time $\tilde{O}(n)$ using the Fast Fourier Transform. While the results in this paper can be applied to lattices that correspond to ideals in other rings, it would only unnecessarily complicate matters because the ring $\mathbb{Z}[\mathbf{x}]/\langle \mathbf{x}^n + 1 \rangle$ seems to be the most useful theoretically and in practice.

While a lot is known about the complexity of SVP in general lattices, very little is known about this problem when restricted to ideal lattices. Nevertheless, the problem is related to some problems in algebraic number theory (see [18,30])

¹ This is a slight abuse of the word *norm*. Because of the reduction modulo p , it’s not true that for any integer α we have $\|\alpha\mathbf{a}\|_\infty = |\alpha|\|\mathbf{a}\|_\infty$, but it still holds true that $\|\mathbf{a} + \mathbf{b}\|_\infty \leq \|\mathbf{a}\|_\infty + \|\mathbf{b}\|_\infty$ and $\|\alpha\mathbf{a}\|_\infty \leq |\alpha|\|\mathbf{a}\|_\infty$.

that do not have any efficient solution. And it seems that the currently best lattice algorithms are unable to take advantage of the extra structure provided by ideal lattices. Therefore, it still seems that solving SVP_γ takes time $2^{O(n)}$ when $\gamma = n^{O(1)}$ [16,4].

2.3 Lattice-Based Collision-Resistant Hash Function

Let R be the ring $\mathbb{Z}_p[\mathbf{x}]/\langle \mathbf{x}^n + 1 \rangle$. We define the following family of hash functions:

Definition 1. For any integer m and $D \subseteq R$, the function family $\mathcal{H}(R, D, m)$ mapping D^m to R is defined as

$$\mathcal{H}(R, D, m) = \{h_{\hat{\mathbf{a}}} : \hat{\mathbf{a}} \in R^m\}, \text{ where for any } \hat{\mathbf{z}} \in D^m, h_{\hat{\mathbf{a}}}(\hat{\mathbf{z}}) = \hat{\mathbf{a}} \cdot \hat{\mathbf{z}}$$

That is, if $\hat{\mathbf{a}} = (\mathbf{a}_1, \dots, \mathbf{a}_m)$ and $\hat{\mathbf{z}} = (\mathbf{z}_1, \dots, \mathbf{z}_m)$, then $h_{\hat{\mathbf{a}}}(\hat{\mathbf{z}}) = \mathbf{a}_1 \mathbf{z}_1 + \dots + \mathbf{a}_m \mathbf{z}_m$ where all the operations are performed in the ring $\mathbb{Z}_p[\mathbf{x}]/\langle \mathbf{x}^n + 1 \rangle$. It's not hard to see that the hash functions in $\mathcal{H}(R, D, m)$ satisfy the following two properties for any $\hat{\mathbf{y}}, \hat{\mathbf{z}} \in R^m$ and $\mathbf{c} \in R$:

$$h(\hat{\mathbf{y}} + \hat{\mathbf{z}}) = h(\hat{\mathbf{y}}) + h(\hat{\mathbf{z}}) \tag{1}$$

$$h(\hat{\mathbf{y}}\mathbf{c}) = h(\hat{\mathbf{y}})\mathbf{c} \tag{2}$$

The collision problem $\text{Col}(h, D)$ is defined as follows:

Definition 2. Given an element $h \in \mathcal{H}(R, D, m)$, the collision problem $\text{Col}(h, D)$, where $D \subseteq R$, asks to find two distinct elements $\hat{\mathbf{z}}, \hat{\mathbf{z}}' \in D$ such that $h(\hat{\mathbf{z}}) = h(\hat{\mathbf{z}}')$.

In [18], it was shown that when D is some restricted domain, solving the $\text{Col}(h, D)$ problem for random $h \in \mathcal{H}(R, D, m)$ is as hard as solving SVP_γ for any $(\mathbf{x}^n + 1)$ -cyclic lattice.

Theorem 1. Let $R = \mathbb{Z}_p[\mathbf{x}]/\langle \mathbf{x}^n + 1 \rangle$ be a ring where n is any power of 2, and define $D = \{\mathbf{y} \in R : \|\mathbf{y}\|_\infty \leq d\}$ for some integer d . Let $\mathcal{H}(R, D, m)$ be a hash function family as in Definition 1 such that $m > \frac{\log p}{\log 2d}$ and $p \geq 4dmn^{1.5} \log n$. If there is a polynomial-time algorithm that solves the $\text{Col}(h, D)$ problem for a random $h \in \mathcal{H}(R, D, m)$ with some non-negligible probability, then there is a polynomial-time algorithm that can solve $\text{SVP}_\gamma(\Lambda)$ for every $(\mathbf{x}^n + 1)$ -cyclic lattice Λ , where $\gamma = 16dmn \log^2 n$.

2.4 Cryptographic Definitions

Digital Signatures. We recall the definitions of signature schemes and what it means for a signature scheme to be secure.

Definition 3. A signature scheme consists of a triplet of polynomial-time (possibly probabilistic) algorithms (G, S, V) such that for every pair of outputs (s, v) of $G(1^n)$ and any n -bit message m ,

$$\Pr[V(v, m, S(s, m)) = 1] = 1$$

where the probability is taken over the randomness of algorithms S and V .

In the above definition, G is called the key-generation algorithm, S is the signing algorithm, V is the verification algorithm, and s and v are, respectively, the signing and verification keys.

A signature scheme is said to be secure if there is only a negligible probability that any forger, after seeing signatures of messages of his choosing, can sign a message whose signature he has not already seen [12].

Definition 4. A signature scheme (G, S, V) is said to be secure if for every polynomial-time (possibly randomized) forger \mathcal{F} , the probability that after seeing the public key and $\{(\mu_1, S(s, \mu_1)), \dots, (\mu_q, S(s, \mu_q))\}$ for any q messages μ_i of its choosing (where q is polynomial in n), \mathcal{F} can produce $(\mu \neq \mu_i, \sigma)$ such that $V(v, \mu, \sigma) = 1$, is negligibly small. The probability is taken over the randomness of G , S , V , and \mathcal{F} .

In the standard security definition of a signature scheme, the forger should not be able to produce a signature of a new message. A stronger notion of security, called *strong unforgeability* requires that in addition to the above, a forger shouldn't even be able to come up with a different signature for a message whose signature he has already seen. The schemes presented in this paper satisfy this stronger notion of unforgeability.

Identification Schemes. An identification scheme consists of a key-generation algorithm and a description of an interactive protocol between a prover, possessing the secret key, and verifier possessing the corresponding public key. In general, it is required that the verifier accepts the interaction with a prover who behaves honestly with probability one, but this definition can be relaxed so that sometimes an honest prover is not accepted with some small probability.

The standard active attack model against identification schemes proceeds in two phases [5]. In the first phase, the adversary interacts with the prover in an effort to obtain some information. In the second stage, the adversary plays the role of the prover and tries to make a verifier accept the interaction. We remark that in the second stage, the adversary no longer has access to the honest prover. The adversary succeeds if he is able to make an honest verifier accept with some non-negligible probability.

Witness-Indistinguishability. We will only define the concept of witness-indistinguishability in a way that pertains to our application and we refer the reader to [6] for the more general definition. For convenience, we will use the notation from the identification protocol in Figure 1. An identification scheme

is said to be perfectly witness-indistinguishable if for any public key S , and any two valid secret keys s, s' (i.e. $s, s' \in D_s$ and $g^s \bmod N = g^{s'} \bmod N = S$), the view of any (possibly malicious) verifier in the interaction where the prover uses s has the exact same distribution as the view where the prover uses s' . In other words, it is impossible for the verifier to figure out which of the valid secret keys the prover is using to authenticate himself.

3 Lattice-Based Constructions

In this section, we present our lattice-based identification (Figure 3) and signature (Figure 4) schemes. In Figure 2 we define all the parameters that will appear in this section as well as give some concrete instantiations. The parameter κ controls the size of the domain from which the challenges/signatures come from. In order to have soundness error of at most 2^{-80} , this parameter must be set such that the size of this domain is 2^{160} . The parameter p is chosen such that every public key has a very high probability of having multiple corresponding secret keys associated with it. The free parameters n, m , and σ need to be set in a way so that it is computationally infeasible find collisions in the underlying hash function family $\mathcal{H}(R, D, m)$.

The last two lines of the above table deal with the practical cryptanalysis of our signature scheme. The last line of the table specifies the length of the shortest vector in a certain lattice defined by our signature scheme that can be found in practice, while the line above that specifies the length of the vector that needs to be found in order to forge a signature. See Section 3.3 for more details.

Parameter	Definition	Sample Instantiations			
n	integer that is a power of 2	512	512	512	1024
m	any integer	4	5	8	8
σ	any integer	127	2047	2047	2047
κ	integer s.t. $2^\kappa \binom{n}{\kappa} \geq 2^{160}$	24	24	24	21
p	integer $\approx (2\sigma + 1)^m \cdot 2^{-\frac{128}{n}}$	$2^{31.7}$	$2^{59.8}$	$2^{95.8}$	$2^{95.9}$
R	ring $\mathbb{Z}_p[\mathbf{x}]/(\mathbf{x}^n + 1)$				
D	$\{\mathbf{g} \in R : \ \mathbf{g}\ _\infty \leq mn\sigma\kappa\}$				
D_s	$\{\mathbf{g} \in R : \ \mathbf{g}\ _\infty \leq \sigma\}$				
D_c	$\{\mathbf{g} \in R : \ \mathbf{g}\ _1 \leq \kappa\}$				
D_y	$\{\mathbf{g} \in R : \ \mathbf{g}\ _\infty \leq mn\sigma\kappa\}$				
G	$\{\mathbf{g} \in R : \ \mathbf{g}\ _\infty \leq mn\sigma\kappa - \sigma\kappa\}$				
Signature Size	$\approx mn \log(2mn\sigma\kappa)$ bits	49000	72000	119000	246000
Public Key Size	$\approx n \log p$ bits	16000	31000	49000	98000
Secret Key Size	$\approx mn \log(2\sigma + 1)$ bits	16000	31000	49000	98000
Hash Function Size	$\approx mn \log p$ bits	65000	153000	392000	786000
Length of vector needed to break signature		$2^{23.5}$	$2^{27.9}$	$2^{28.6}$	$2^{29.4}$
Length of shortest vector that can be found		$2^{25.5}$	$2^{36.7}$	$2^{47.6}$	$2^{69.4}$

Fig. 2. Lattice-Based Schemes' Parameter Definitions and Sample Instantiations

3.1 Identification Scheme

The secret key of the prover, denoted $\widehat{\mathbf{s}}$, consists of a set of m polynomials from the set D_s which are picked uniformly at random. The public key of the prover consists of a hash function h which is picked randomly from the family $\mathcal{H}(R, D, m)$, and the polynomial $\mathbf{S} = h(\widehat{\mathbf{s}})$. We point out that it is not necessary for every prover to have a distinct h . If trusted randomness is available, then everyone can share the same random h which considerably lowers the public key size because the hash function h can be hard-coded into the signing and verification algorithms.

In the first step of the protocol, the prover picks a random $\widehat{\mathbf{y}} \in D_y^m$, and “commits” to it by sending $\mathbf{Y} = h(\widehat{\mathbf{y}})$ to the verifier. The verifier then picks a random challenge \mathbf{c} from D_c and sends it to the prover. The prover then computes $\widehat{\mathbf{z}} = \widehat{\mathbf{s}}\mathbf{c} + \widehat{\mathbf{y}}$. If this result falls into the range G^m , the prover sends it to the verifier. Otherwise, he aborts the protocol. Upon receiving $\widehat{\mathbf{z}}$, the verifier accepts the interaction if $\widehat{\mathbf{z}} \in G^m$ and $h(\widehat{\mathbf{z}}) = \mathbf{S}\mathbf{c} + \mathbf{Y}$. Using the homomorphic properties of h (see (I) and (II)), we see that $h(\widehat{\mathbf{s}}\mathbf{c} + \widehat{\mathbf{y}}) = \mathbf{S}\mathbf{c} + \mathbf{Y}$, and so an honest prover who does not abort will always be accepted.

Proving the soundness and completeness of the protocol is done using the following series of steps:

1. Show that an honest prover is accepted with probability $1/e$.
2. Show that the ID scheme is perfectly witness-indistinguishable.
3. Show that with probability $1 - 2^{-128}$, for a randomly-picked $\widehat{\mathbf{s}} \in D_s^m$, there is another $\widehat{\mathbf{s}}' \in D_s^m$ such that $h(\widehat{\mathbf{s}}) = h(\widehat{\mathbf{s}}')$.
4. Show how to extract a collision in h from an adversary who succeeds in breaking the protocol

Step 1 shows that the completeness of the protocol is $1/e$. We will explain how to increase this number later. Step 2 is essentially the main part of the proof, which shows that for every pair of possible secret keys $\widehat{\mathbf{s}}, \widehat{\mathbf{s}}'$ such that $\mathbf{S} = h(\widehat{\mathbf{s}}) = h(\widehat{\mathbf{s}}')$, no adversarial verifier can determine which secret key is being used by the prover. The reason for this is that we have set up the parameters so that for every secret key $\widehat{\mathbf{s}} \in D_s^m$, every challenge $\mathbf{c} \in D_c$, and every response $\widehat{\mathbf{z}} \in G^m$, the value of $\widehat{\mathbf{z}} - \widehat{\mathbf{s}}\mathbf{c}$ is in D_y . This implies that having seen the history $(\mathbf{Y}, \mathbf{c}, \widehat{\mathbf{z}})$, it is impossible to tell whether the secret key was $\widehat{\mathbf{s}}$ and we picked a masking parameter $\widehat{\mathbf{y}}$, or the secret key was $\widehat{\mathbf{s}}'$ and we picked the masking parameter $\widehat{\mathbf{y}}' = \widehat{\mathbf{z}} - \widehat{\mathbf{s}}'\mathbf{c} = \widehat{\mathbf{y}} + \widehat{\mathbf{s}}\mathbf{c} - \widehat{\mathbf{s}}'\mathbf{c} = \widehat{\mathbf{y}} + (\widehat{\mathbf{s}} - \widehat{\mathbf{s}}')\mathbf{c}$ because $h(\widehat{\mathbf{s}}) = h(\widehat{\mathbf{s}}') = \mathbf{S}$ and $h(\widehat{\mathbf{y}}) = h(\widehat{\mathbf{y}}') = \mathbf{Y}$.

To make the claim in step 2 non-vacuous, we need to show that for a randomly picked secret key, there is indeed a high probability that another secret key exists which produces the same public key. This is done in step 3.

In step 4, we show how to use a successful adversary to solve the $\text{Col}(h, D)$ problem for a random $h \in \mathcal{H}(R, D, m)$. Given a random $h \in \mathcal{H}(R, D, m)$, we pick a random secret key $\widehat{\mathbf{s}}$ and publish the public keys h and $\mathbf{S} = h(\widehat{\mathbf{s}})$. In the first stage of the attack, the adversary plays the role of the verifier, and we are able to perfectly play the part of the prover because we know the secret key. In the

Private key: $\widehat{\mathbf{s}} \xleftarrow{\$} D_s^m$

Public key: $h \xleftarrow{\$} \mathcal{H}(R, D, m), \mathbf{S} \leftarrow h(\widehat{\mathbf{s}})$

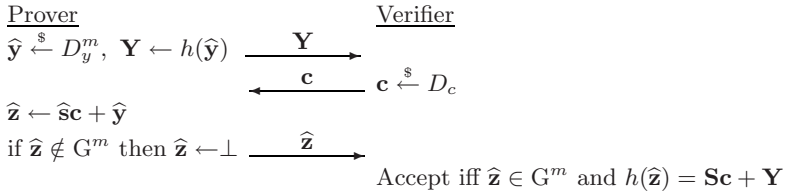


Fig. 3. Lattice-Based Identification Scheme

second stage when the adversary attempts to impersonate the prover, we receive his commitment, and send a random challenge $\mathbf{c} \in D_c$. After he responds with $\widehat{\mathbf{z}}$, we rewind and pick another random challenge $\mathbf{c}' \in D_c$, to which the adversary will respond with $\widehat{\mathbf{z}}'$. The responses of the adversary and our knowledge of the secret key allow us to obtain the equation $h(\widehat{\mathbf{z}} - \widehat{\mathbf{s}}\mathbf{c}) = h(\widehat{\mathbf{z}}' - \widehat{\mathbf{s}}\mathbf{c}')$. By our choice of parameters, both $\widehat{\mathbf{z}} - \widehat{\mathbf{s}}\mathbf{c}$ and $\widehat{\mathbf{z}}' - \widehat{\mathbf{s}}\mathbf{c}'$ are in D , and because of the witness-indistinguishability of the protocol, the adversary cannot know our exact secret key. Therefore with probability at least $1/2$, $\widehat{\mathbf{z}} - \widehat{\mathbf{s}}\mathbf{c}$ and $\widehat{\mathbf{z}}' - \widehat{\mathbf{s}}\mathbf{c}'$ will be distinct and we have a collision for h . Thus an adversary who can break the ID scheme can be used to solve $\text{Col}(h, D)$ for random $h \in \mathcal{H}(R, D, m)$, and by Theorem 1, this implies finding the approximate short vector in all $(\mathbf{x}^n + 1)$ -cyclic lattices.

Theorem 2. *If the identification scheme in Figure 3 is insecure against active attacks for the parameters in Table 2, then there is polynomial-time algorithm that can solve $\text{SVP}_\gamma(\Lambda)$ for $\gamma = \tilde{O}(n^2)$ for every lattice Λ corresponding to an ideal in the ring $\mathbb{Z}[\mathbf{x}]/\langle \mathbf{x}^n + 1 \rangle$.*

Notice that the ID scheme is not quite satisfactory because a valid prover is only accepted with probability $1/e$. This means that the scheme may have to be repeated several times until the prover succeeds. Because we showed that the scheme is witness-indistinguishable, the repetitions can be performed in parallel, and the witness-indistinguishability property will still be preserved [6]. So the straight-forward way to modify the ID scheme would be, for example, to pick 30 different $\widehat{\mathbf{y}}_i$'s and send the $\mathbf{Y}_i = h(\widehat{\mathbf{y}}_i)$ to the verifier. Then the verifier will send 30 challenges, and the prover replies to the first one of these challenges that he can. This would result in a protocol where the honest prover is accepted with probability about $1 - 2^{-20}$.

But there are some significant improvements that can be made. First of all, the verifier needs to send only one challenge, rather than one challenge for every commitment (this is because we show that for every challenge \mathbf{c} , the probability of aborting is equal over the random choice of $\widehat{\mathbf{y}}$). And secondly, we can use a standard trick to shorten the length of every \mathbf{Y}_i , which will result in large savings in our protocol because the length of each \mathbf{Y} is approximately $n \log p$ bits, which could be as large as 100,000 bits! Instead of sending \mathbf{Y} , we can send $H(\mathbf{Y})$ where H is any collision resistant hash function. Unlike with h , we will not need H to

Signing Key: $\widehat{\mathbf{s}} \xleftarrow{\$} D_s^m$
 Verification Key: $h \xleftarrow{\$} \mathcal{H}(R, D, m), \mathbf{S} \leftarrow h(\widehat{\mathbf{s}})$
 Random Oracle: $H : \{0, 1\}^* \rightarrow D_c$

<p> Sign($\mu, h, \widehat{\mathbf{s}}$) 1: $\widehat{\mathbf{y}} \xleftarrow{\\$} D_y^m$ 2: $\mathbf{e} \leftarrow H(h(\widehat{\mathbf{y}}), \mu)$ 3: $\widehat{\mathbf{z}} \leftarrow \widehat{\mathbf{s}}\mathbf{e} + \widehat{\mathbf{y}}$ 4: if $\widehat{\mathbf{z}} \notin G^m$, then goto step 1 5: output $(\widehat{\mathbf{z}}, \mathbf{e})$ </p>	<p> Verify($\mu, \widehat{\mathbf{z}}, \mathbf{e}, h, \mathbf{S}$) 1: Accept iff $\widehat{\mathbf{z}} \in G^m$ and $\mathbf{e} = H(h(\widehat{\mathbf{z}}) - \mathbf{S}\mathbf{e}, \mu)$ </p>
--	---

Fig. 4. Lattice-Based Signature Scheme

have any algebraic properties like (1) and (2), so H could be a cryptographic hash function such as SHA or an efficient lattice-based hash function from [20] whose output is about 512 bits. So sending 30 $H(\mathbf{Y})$'s will only require about 15,000 bits in total. In this modified protocol, the verifier's challenge and the prover's reply remain the same as in the old protocol. But to authenticate the prover, the verifier checks whether $\widehat{\mathbf{z}} \in G^m$ and that $H(h(\widehat{\mathbf{z}}) - \mathbf{S}\mathbf{e})$ is equal to some $H(\mathbf{Y})$ sent by the prover in the first step [2]. It can be shown that an adversary who breaks this protocol can be used to find a collision either in H or in h . We will give more details in the full version of the paper.

3.2 Signature Scheme

Our signature scheme is presented in Figure 4. The public and secret keys are just like in the ID scheme. To sign a message μ , we pick a random $\widehat{\mathbf{y}}$ and compute $\mathbf{e} = H(h(\widehat{\mathbf{y}}), \mu)$ and send $(\widehat{\mathbf{z}}, \mathbf{e})$ as the signature only if $\widehat{\mathbf{z}}$ is in the set G^m . Otherwise we repeat the procedure until $\widehat{\mathbf{z}}$ ends up in G^m . The probability that we succeed in getting $\widehat{\mathbf{z}}$ to be in G^m on any particular try is the same as the probability that the ID scheme in Figure 3 doesn't send \perp , which is $1/e$. So we expect to repeat the signing procedure less than 3 times to get a signature.

The witness-indistinguishability of the signature scheme follows directly from the witness indistinguishability of the ID scheme because the challenge is now simply generated by a random oracle rather than by the verifier. The proof of security of the signature scheme uses the forking lemma [32] to obtain two signatures from a forger that use the same random oracle query. Then using the same ideas as in the security proof of the ID scheme, it can be shown how to use these signatures to obtain a solution to the $\text{Col}(h, D)$ problem for a random $h \in \mathcal{H}(R, D, m)$.

Theorem 3. *If the signature scheme in Figure 4 for the parameters in Table 2 is not strongly unforgeable, then there is a polynomial-time algorithm that can solve $\text{SVP}_\gamma(\Lambda)$ for $\gamma = \tilde{O}(n^2)$ for every lattice Λ corresponding to an ideal in the ring $\mathbb{Z}[\mathbf{x}]/\langle \mathbf{x}^n + 1 \rangle$.*

² One could lower the communication complexity even further by combining the 30 hashes into a hash tree.

3.3 Concrete Parameters

The security of our ID (and signature) scheme depends on its soundness and the hardness of finding collisions in hash functions from a certain family. As mentioned earlier, we set the parameters κ and p such that the soundness error is at most 2^{-80} . We now discuss how to set the remaining parameters so that finding collisions in the resulting hash function is infeasible with the techniques known today. For this, we will use the work of [8], who showed that, given a reasonable amount of time, algorithms for finding short vectors in random lattices produce a vector that is no smaller than 1.01^n times the shortest vector of the lattice.

We showed that an adversary who succeeds in forging a signature can be used to find a collision in a hash function chosen randomly from $\mathcal{H}(R, D, m)$. This is equivalent to finding “short” vectors a certain lattice which we will now define. For a polynomial $\mathbf{a} \in \mathbb{Z}_p[\mathbf{x}]/\langle \mathbf{x}^n + 1 \rangle$, let $Rot(\mathbf{a})$ be the $n \times n$ matrix whose i^{th} column is the polynomial $\mathbf{a}\mathbf{x}^i$, and let \mathbf{A} be the $n \times nm$ matrix $\mathbf{A} = [Rot(\mathbf{a}_1) || Rot(\mathbf{a}_2) || \dots || Rot(\mathbf{a}_m)]$ where \mathbf{a}_i are the polynomials which define the hash function h . If we define the lattice $\Lambda_p^\perp(\mathbf{A}) = \{\mathbf{u} \in \mathbb{Z}^{mn} : \mathbf{A}\mathbf{u} = 0 \pmod{p}\}$, then finding a vector $\mathbf{u} \in \Lambda_p^\perp(\mathbf{A})$ whose ℓ_∞ norm is at most $2mn\sigma\kappa$ is equivalent to finding a collision in $h \in \mathcal{H}(R, D, m)$.

The random lattices on which the experiments of [8] were run differ from $\Lambda_p^\perp(\mathbf{A})$, but in [25], experiments were run on lattices that are very similar [3] to $\Lambda_p^\perp(\mathbf{A})$ which obtained the same results as [8]. Furthermore, it was shown in [25] that it is inefficient to try to find a short vector in $\Lambda_p^\perp(\mathbf{A})$ by using all its mn dimensions. Rather, one should only use the first $\sqrt{n \log p / \log 1.01}$ dimensions and zero out the others. This results in a vector whose ℓ_2 length is $\min\{p, 2^{2\sqrt{n \log p / \log 1.01}}\}$, and whose ℓ_∞ norm is at least

$$\min\{p, 2^{2\sqrt{n \log p / \log 1.01}} \cdot (n \log p / \log 1.01)^{-1/4}\} \tag{3}$$

Since solving the $Col(h, D)$ problem is equivalent to finding an element $\hat{\mathbf{y}}$ such that $h(\hat{\mathbf{y}}) = 0$ and $\|\hat{\mathbf{y}}\|_\infty \leq 2mn\sigma\kappa$, we want to make sure that when we set our parameters, the value of $2mn\sigma\kappa$ is smaller than the value in (3). In the instantiation of the scheme that produces a signature of length approximately 49000 bits, the value of $2mn\sigma\kappa$ is around $2^{23.5}$, while the value of the shortest vector (in the ℓ_∞ norm) that can be found according to (3) is around $2^{25.5}$ (see the last two lines of the table in Figure 2).

We hope that our work provides further motivation for studying lattice-reduction algorithms for lattices of the form $\Lambda_p^\perp(\mathbf{A})$, which also happen to be central to the cryptanalysis of other lattice-based schemes such as [20,19,15].

³ The lattices in [25] were just like $\Lambda_p^\perp(\mathbf{A})$, except each entry of \mathbf{A} was chosen uniformly at random modulo p . Since the currently best lattice-reduction algorithms don't “see” the algebraic structure of the lattice, it is very reasonable to assume that their performance will be the same on our lattices and the lattices in [25]. Of course it's possible that a different algorithm that has yet to be discovered will be able to use the algebraic structure of \mathbf{A} to achieve better results.

4 Factoring-Based Constructions

We now present a modification of a signature scheme presented in [31] whose security is based on the hardness of factoring. We will need the following two definitions from [31].

Definition 5. A prime p is said to be α -strong if $p = 2r + 1$ where r is an integer whose prime factors are all greater than α .

Definition 6. Let $N = pq$, where p and q are primes. Then an element $g \in \mathbb{Z}_N^*$ is said to be an **asymmetric basis** if the parity of $\text{ord}(g)$ in \mathbb{Z}_p^* differs from the parity of $\text{ord}(g)$ in \mathbb{Z}_q^* .

Both schemes are presented in Figure 5 (our scheme only differs from that in [31] by the addition of line 4), and the parameters in [31] as well as our modified parameters are presented in Figure 6. We point out that the scheme of [31] is a variant of Girault’s scheme [10], and our technique of shortening the signature length would apply equally well to all its variants [10,31,11] as well as to the blind signature constructed in [31].

The signature of a message μ consists of the pair (z, e) . The length of z in the non-aborting version of the protocol has length $k + k' + \log \sigma = 360$, while in our protocol the length is $k + 1 + \log \sigma = 297$. The savings are essentially due to the fact that we can pick y in a much smaller range, and the fact that we are allowed to abort keeps the scheme secure.

If in step 4, z is not in G , then the signing procedure has to be repeated. It can be shown that this happens with probability $1/2$. So we expect to run the signing protocol twice for every signature. But if we assume that off-line computations (i.e. computations before receiving the message) are free, then we can change the protocol so that we expect to compute just one extra random oracle query over the non-aborting signature scheme. The way to do this is to always keep several y_i and $g^{y_i} \bmod N$ stored along with the ranges that e would have to fall into so that $se + y_i \in G$ (the range is just $(G - y_i)/s$). Then when we are asked to sign a message μ , we compute $e = H(g^{y_i} \bmod N, \mu)$ and then check

Secret Key: $s \xleftarrow{\$} D_s$

Public Key: N, g , and $S \leftarrow g^s \bmod N$

Random Oracle: $H : \{0, 1\}^* \rightarrow D_c$

Sign(μ, N, g, s)

1: $y \xleftarrow{\$} D_y$

2: $e \leftarrow H(g^y \bmod N, \mu)$

3: $z \leftarrow se + y$

4: [if $z \notin G$, then goto step 1]

5: output (z, e)

Verify(μ, z, e, N, g, S)

1: Accept iff $e = H(g^z S^{-e} \bmod N, \mu)$

Fig. 5. Factoring-Based Signature Schemes. Line 4 is only executed in the aborting scheme.

	Without Aborting	With Aborting
k	128	
N	1024-bit product of two 2^k -strong primes	
g	asymmetric basis in \mathbb{Z}_N^* such that $ord(g)$ has 160 bits	
σ	2^{168}	
D_c	$\{0, \dots, 2^k\}$	
D_s	$\{0, \dots, \sigma\}$	
k'	64	-
D_y	$\{0, \dots, 2^{k+k'} \cdot \sigma\}$	$\{0, \dots, 2^{k+1} \cdot \sigma\}$
G	-	$\{2^k \cdot \sigma, \dots, 2^{k+1} \cdot \sigma\}$
Signature Size (bits)	488	425

Fig. 6. Factoring-Based Scheme’s Variable Definitions

whether it’s in the valid range of y_1 . If it is, then we compute $sc + y_1$ and output it. If it’s not, then we recompute e using y_2 , and so on. The important thing to note is that we only compute $sc + y_i$ once, and we still expect to succeed after two tries. As an added bonus, we only use up one y_i per message, since the y_i that “didn’t work” can be safely tried for the next message.

Theorem 4. *An adversary who breaks the aborting signature scheme in T steps can be used to factor N in $poly(T)$ steps.*

Acknowledgements. I am very grateful to Daniele Micciancio for suggesting many substantial improvements and simplifications to an earlier version of this work. I would also like to thank the anonymous referees for their useful suggestions.

References

1. Abdalla, M., An, J.H., Bellare, M., Namprempre, C.: From identification to signatures via the Fiat-Shamir transform: Minimizing assumptions for security and forward-security. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 418–433. Springer, Heidelberg (2002)
2. Ajtai, M.: Generating hard instances of lattice problems. In: STOC (1996)
3. Ajtai, M., Dwork, C.: A public-key cryptosystem with worst-case/average-case equivalence. In: STOC (1997); An improved version is described in ECCC 2007
4. Ajtai, M., Kumar, R., Sivakumar, D.: A sieve algorithm for the shortest lattice vector problem. In: STOC, pp. 601–610 (2001)
5. Feige, U., Fiat, A., Shamir, A.: Zero-knowledge proofs of identity. J. Cryptology 1(2), 77–94 (1988)
6. Feige, U., Shamir, A.: Witness indistinguishable and witness hiding protocols. In: STOC, pp. 416–426 (1990)
7. 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)
8. Gama, N., Nguyen, P.Q.: Predicting lattice reduction. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 31–51. Springer, Heidelberg (2008)

9. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices, and new cryptographic constructions. In: STOC (2008)
10. Girault, M.: An identity-based identification scheme based on discrete logarithms modulo a composite number. In: Damgård, I.B. (ed.) EUROCRYPT 1990. LNCS, vol. 473, pp. 481–486. Springer, Heidelberg (1991)
11. Girault, M., Poupard, G., Stern, J.: On the fly authentication and signature schemes based on groups of unknown order. *J. Cryptology* 19(4), 463–487 (2006)
12. Goldwasser, S., Micali, S., Rivest, R.: A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.* 17(2), 281–308 (1988)
13. Guillou, L., Quisquater, J.J.: A "paradoxical" identity-based signature scheme resulting from zero-knowledge. In: Goldwasser, S. (ed.) CRYPTO 1988. LNCS, vol. 403, pp. 216–231. Springer, Heidelberg (1990)
14. Hoffstein, J., Pipher, J., Silverman, J.H.: NTRU: A ring-based public key cryptosystem. In: Buhler, J.P. (ed.) ANTS 1998. LNCS, vol. 1423, pp. 267–288. Springer, Heidelberg (1998)
15. 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)
16. Lenstra, A.K., Lenstra Jr., H.W., Lovasz, L.: Factoring polynomials with rational coefficients. *Mathematische Annalen* (261), 513–534 (1982)
17. Lyubashevsky, V.: Lattice-based identification schemes secure under active attacks. In: *Public Key Cryptography*, pp. 162–179 (2008)
18. 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)
19. Lyubashevsky, V., Micciancio, D.: Asymptotically efficient lattice-based digital signatures. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 37–54. Springer, Heidelberg (2008)
20. Lyubashevsky, V., Micciancio, D., Peikert, C., Rosen, A.: SWIFFT: a modest proposal for FFT hashing. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 54–72. Springer, Heidelberg (2008)
21. Merkle, R.: A digital signature based on a conventional encryption function. In: Pomerance, C. (ed.) CRYPTO 1987. LNCS, vol. 293, pp. 369–378. Springer, Heidelberg (1988)
22. Merkle, R.: A certified digital signature. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 218–238. Springer, Heidelberg (1990)
23. Micciancio, D.: Generalized compact knapsacks, cyclic lattices, and efficient one-way functions. *Computational Complexity* 16(4), 365–411 (2007)
24. Micciancio, D., Regev, O.: Worst-case to average-case reductions based on Gaussian measures. *SIAM J. on Computing* 37(1), 267–302 (2007)
25. Micciancio, D., Regev, O.: Lattice-based cryptography. In: Bernstein, D., Buchmann, J. (eds.) *Post-quantum Cryptography*. Springer, Heidelberg (2009)
26. 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)
27. 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)
28. Peikert, C.: Public-key cryptosystems from the worst-case shortest vector problem. In: STOC (2009)

29. Peikert, C., Rosen, A.: Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 145–166. Springer, Heidelberg (2006)
30. Peikert, C., Rosen, A.: Lattices that admit logarithmic worst-case to average-case connection factors. In: STOC (2007)
31. Pointcheval, D.: The composite discrete logarithm and secure authentication. In: Imai, H., Zheng, Y. (eds.) PKC 2000. LNCS, vol. 1751, pp. 113–128. Springer, Heidelberg (2000)
32. Pointcheval, D., Stern, J.: Security arguments for digital signatures and blind signatures. *J. Cryptology* 13(3), 361–396 (2000)
33. Regev, O.: New lattice-based cryptographic constructions. *J. ACM* 51(6), 899–942 (2004)
34. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: STOC (2005)
35. Schnorr, C.P.: Efficient signature generation by smart cards. *J. Cryptology* 4(3), 161–174 (1991)
36. Shor, P.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.* 26(5), 1484–1509 (1997)
37. Stehle, D., Steinfeld, R., Tanaka, K., Xagawa, K.: Efficient public-key encryption based on ideal lattices. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 617–635. Springer, Heidelberg (2009)

Efficient Public Key Encryption Based on Ideal Lattices

(Extended Abstract)

Damien Stehlé^{1,2}, Ron Steinfeld², Keisuke Tanaka³, and Keita Xagawa³

¹ CNRS/Department of Mathematics and Statistics,
University of Sydney NSW 2006, Australia

² Centre for Advanced Computing - Algorithms and Cryptography,
Department of Computing, Macquarie University, NSW 2109, Australia

³ Department of Mathematical and Computing Sciences,
Tokyo Institute of Technology, Japan

Abstract. We describe public key encryption schemes with security provably based on the worst case hardness of the approximate Shortest Vector Problem in some structured lattices, called ideal lattices. Under the assumption that the latter is exponentially hard to solve even with a quantum computer, we achieve CPA-security against subexponential attacks, with (quasi-)optimal asymptotic performance: if n is the security parameter, both keys are of bit-length $\tilde{O}(n)$ and the amortized costs of both encryption and decryption are $\tilde{O}(1)$ per message bit. Our construction adapts the trapdoor one-way function of Gentry *et al.* (STOC'08), based on the Learning With Errors problem, to structured lattices. Our main technical tools are an adaptation of Ajtai's trapdoor key generation algorithm (ICALP'99) and a re-interpretation of Regev's quantum reduction between the Bounded Distance Decoding problem and sampling short lattice vectors.

1 Introduction

Lattice-based cryptography has been rapidly developing in the last few years, inspired by the breakthrough result of Ajtai in 1996 [Ajt96], who constructed a one-way function with average-case security provably related to the worst-case complexity of hard lattice problems. The attractiveness of lattice-based cryptography stems from its provable security guarantees, well studied theoretical underpinnings, simplicity and potential efficiency (Ajtai's one-way function is a matrix-vector multiplication over a small finite field), and also the apparent security against quantum attacks. The main complexity assumption is the hardness of approximate versions of the Shortest Vector Problem (SVP). The $\text{GapSVP}_{\gamma(n)}$ problem consists in, given a lattice of dimension n and a scalar d , replying YES if there exists a non-zero lattice vector of norm $\leq d$ and NO if all non-zero lattice vectors have norm $\geq \gamma(n)d$. The complexity of $\text{GapSVP}_{\gamma(n)}$ increases with n , but decreases with $\gamma(n)$. Although the latter is believed to be exponential in n for any

polynomial $\gamma(n)$, minimizing the degree of $\gamma(n)$ is very important in practice, to allow the use of a practical dimension n for a given security level.

LATTICE-BASED PUBLIC KEY ENCRYPTION. The first provably secure lattice-based cryptosystem was proposed by Ajtai and Dwork [3], and relied on a variant of GapSVP in arbitrary lattices (it is now known to also rely on GapSVP [19]). Subsequent works proposed more efficient alternatives [33,30,9,28]. The current state of the art [9,28] is a scheme with public/private key length $\tilde{O}(n^2)$ and encryption/decryption throughput of $\tilde{O}(n)$ bit operations per message bit. Its security relies on the quantum worst-case hardness of GapSVP $_{\tilde{O}(n^{1.5})}$ in arbitrary lattices. The security can be de-quantumized at the expense of both increasing $\gamma(n)$ and decreasing the efficiency, or relying on a new and less studied problem [28]. In parallel to the provably secure schemes, there have also been heuristic proposals [11,12]. In particular, unlike the above schemes which use unstructured random lattices, the NTRU encryption scheme [12] exploits the properties of *structured* lattices to achieve high efficiency with respect to key length ($\tilde{O}(n)$ bits) and encryption/decryption cost ($\tilde{O}(1)$ bit operation per message bit). Unfortunately, its security remains heuristic and it was an important open challenge to provide a provably secure scheme with comparable efficiency.

PROVABLY SECURE SCHEMES FROM IDEAL LATTICES. Micciancio [20] introduced the class of structured *cyclic* lattices, which correspond to ideals in polynomial rings $\mathbb{Z}[x]/(x^n - 1)$, and presented the first provably secure one-way function based on the worst-case hardness of the restriction of $\mathcal{Poly}(n)$ -SVP to cyclic lattices. (The problem γ -SVP consists in computing a non-zero vector of a given lattice, whose norm is no more than γ times larger than the norm of a shortest non-zero lattice vector.) At the same time, thanks to its algebraic structure, this one-way function enjoys high efficiency comparable to the NTRU scheme ($\tilde{O}(n)$ evaluation time and storage cost). Subsequently, Lyubashevsky and Micciancio [17] and independently Peikert and Rosen [29] showed how to modify Micciancio's function to construct an efficient and provably secure collision resistant hash function. For this, they introduced the more general class of *ideal* lattices, which correspond to ideals in polynomial rings $\mathbb{Z}[x]/f(x)$. The collision resistance relies on the hardness of the restriction of $\mathcal{Poly}(n)$ -SVP to ideal lattices (called $\mathcal{Poly}(n)$ -Ideal-SVP). The average-case collision-finding problem is a natural computational problem called Ideal-SIS, which has been shown to be as hard as the worst-case instances of Ideal-SVP. Provably secure efficient signature schemes from ideal lattices have also been proposed [18,15,16,14], but constructing efficient provably secure public key encryption from ideal lattices was an interesting open problem.

OUR RESULTS. We describe the first provably CPA-secure public key encryption scheme whose security relies on the hardness of the worst-case instances of $\tilde{O}(n^2)$ -Ideal-SVP against subexponential quantum attacks. It achieves asymptotically optimal efficiency: the public/private key length is $\tilde{O}(n)$ bits and the amortized encryption/decryption cost is $\tilde{O}(1)$ bit operations per message bit (encrypting $\tilde{\Omega}(n)$ bits at once, at a $\tilde{O}(n)$ cost). Our security assumption is

that $\tilde{O}(n^2)$ -Ideal-SVP cannot be solved by any subexponential time quantum algorithm, which is reasonable given the state-of-the-art lattice algorithms [36]. Note that this is stronger than standard public key cryptography security assumptions. On the other hand, contrary to most of public key cryptography, lattice-based cryptography allows security against subexponential quantum attacks. Our main technical tool is a re-interpretation of Regev’s quantum reduction [33] between the Bounded Distance Decoding problem (BDD) and sampling short lattice vectors. Also, by adapting Ajtai’s trapdoor generation algorithm [2] (or more precisely its recent improvement by Alwen and Peikert [5]) to structured ideal lattices, we are able to construct efficient provably secure trapdoor signatures, ID-based identification schemes, CCA-secure encryption and ID-based encryption. We think these techniques are very likely to find further applications.

Most of the cryptosystems based on general lattices [33,30,31,9,28] rely on the average-case hardness of the *Learning With Errors* (LWE) problem introduced in [33]. Our scheme is based on a structured variant of LWE, that we call Ideal-LWE. We introduce novel techniques to circumvent two main difficulties that arise from the restriction to ideal lattices. Firstly, the previous cryptosystems based on unstructured lattices all make use of Regev’s worst-case to average-case classical reduction [33] from BDD to LWE (this is the *classical step* in the quantum reduction of [33] from SVP to LWE). This reduction exploits the unstructured-ness of the considered lattices, and does not seem to carry over to the structured lattices involved in Ideal-LWE. In particular, the probabilistic independence of the rows of the LWE matrices allows to consider a single row in [33, Cor. 3.10]. Secondly, the other ingredient used in previous cryptosystems, namely Regev’s reduction [33] from the computational variant of LWE to its decisional variant, also seems to fail for Ideal-LWE: it relies on the probabilistic independence of the columns of the LWE matrices.

Our solution to the above difficulties avoids the *classical step* of the reduction from [33] altogether. Instead, we use the *quantum step* to construct a new quantum average-case reduction from SIS (the unstructured variant of Ideal-SIS) to LWE. It also works from Ideal-SIS to Ideal-LWE. Combined with the known reduction from worst-case Ideal-SVP to average-case Ideal-SIS [17], we obtain a quantum reduction from Ideal-SVP to Ideal-LWE. This shows the hardness of the computational variant of Ideal-LWE. Because we do not obtain the hardness of the decisional variant, we use a generic hardcore function to derive pseudorandom bits for encryption. This is why we need to assume the exponential hardness of SVP. The encryption scheme follows as an adaptation of [9, Sec. 7.1].

The main idea of our new quantum reduction from Ideal-SIS to Ideal-LWE is a re-interpretation of Regev’s quantum step in [33]. The latter was presented as a worst-case quantum reduction from sampling short lattice vectors in a lattice L to solving BDD in the dual lattice \hat{L} . We observe that this reduction is actually stronger: it is an average-case reduction which works given an oracle for BDD in \hat{L} with a normally distributed error vector. Also, as pointed out in [9], LWE can be seen as a BDD with a normally distributed error in a certain lattice whose dual is essentially the SIS lattice. This leads to our SIS to LWE reduction. Finally

we show how to apply it to reduce Ideal-SIS to Ideal-LWE – this involves a probabilistic lower bound for the minimum of the Ideal-LWE lattice. We believe our new SIS to LWE reduction is of independent interest. Along with [22], it provides an alternative to Regev’s quantum reduction from GapSVP to LWE. Ours is weaker because the derived GapSVP factor increases with the number of LWE samples, but it has the advantage of carrying over to the ideal case. Also, when choosing practical parameters for lattice-based encryption (see, e.g., [23]), it is impractical to rely on the worst-case hardness of SVP. Instead, the practical average-case hardness of LWE is evaluated based on the best known attack which consists in solving SIS. Our reduction justifies this heuristic by showing that it is indeed necessary to (quantumly) break SIS in order to solve LWE.

ROAD-MAP. We provide some background in Section 2. Section 3 shows how to hide a trapdoor in the adaptation of SIS to ideal lattices. Section 4 contains the new reduction between SIS and LWE. Finally, in Section 5, we present our CPA-secure encryption scheme and briefly describe other cryptographic constructions.

NOTATION. Vectors will be denoted in bold. We denote by $\langle \cdot, \cdot \rangle$ and $\| \cdot \|$ the inner product and the Euclidean norm. We denote by $\rho_s(\mathbf{x})$ (resp. ν_s) the standard n -dimensional Gaussian function (resp. distribution) with center $\mathbf{0}$ and variance s , i.e., $\rho_s(\mathbf{x}) = \exp(-\pi\|\mathbf{x}\|^2/s^2)$ (resp. $\nu_s(\mathbf{x}) = \rho_s(\mathbf{x})/s^n$). We use the notations $\tilde{O}(\cdot)$ and $\tilde{\Omega}(\cdot)$ to hide poly-logarithmic factors. If D_1 and D_2 are two probability distributions over a discrete domain E , their statistical distance is $\Delta(D_1, D_2) = \frac{1}{2} \sum_{x \in E} |D_1(x) - D_2(x)|$. If a function f over a countable domain E takes non-negative real values, its sum over an arbitrary $F \subseteq E$ will be denoted by $f(F)$. If q is a prime number, we denote by \mathbb{Z}_q the field of integers modulo q . We denote by Ψ_s the reduction modulo q of ν_s .

2 Reminders and Background Results on Lattices

We refer to [21] for a detailed introduction to the computational aspects of lattices. In the present section, we remind the reader very quickly some fundamental properties of lattices that we will need. We then introduce the so-called ideal lattices, and finally formally define some computational problems.

Euclidean lattices. An n -dimensional lattice L is the set of all integer linear combinations of some linearly independent vectors $\mathbf{b}_1, \dots, \mathbf{b}_n \in \mathbb{R}^n$, i.e., $L = \sum \mathbb{Z}\mathbf{b}_i$. The \mathbf{b}_i ’s are called a basis of L . The i th minimum $\lambda_i(L)$ is the smallest r such that L contains i linearly independent vectors of norms $\leq r$. We let $\lambda_1^\infty(L)$ denote the first minimum of L with respect to the infinity norm. If $B = (\mathbf{b}_1, \dots, \mathbf{b}_n)$ is a basis, we define its norm by $\|B\| = \max \|\mathbf{b}_i\|$ and its fundamental parallelepiped by $P(B) = \{\sum_i c_i \mathbf{b}_i \mid \mathbf{c} \in [0, 1]^n\}$. Given a basis B for lattice L and a vector $\mathbf{c} \in \mathbb{R}^n$, we define $\mathbf{c} \bmod L$ as the unique vector in $P(B)$ such that $\mathbf{c} - (\mathbf{c} \bmod L) \in L$ (the basis being implicit). For any lattice L and any $s > 0$, the sum $\rho_s(L)$ is finite. We define the lattice Gaussian distribution by $D_{L,s}(\mathbf{b}) = \frac{\rho_s(\mathbf{b})}{\rho_s(L)}$, for any $\mathbf{b} \in L$. If L is a lattice, its dual \hat{L} is the lattice $\{\hat{\mathbf{b}} \in \mathbb{R}^n \mid \forall \mathbf{b} \in L, \langle \hat{\mathbf{b}}, \mathbf{b} \rangle \in \mathbb{Z}\}$. We will use the following results.

Lemma 1 ([29, Lemma 2.11] and [27, Lemma 3.5]). *For any x in an n -dimensional lattice L and $s \geq 2\sqrt{\ln(10n)/\pi}/\lambda_1^\infty(\widehat{L})$, we have $D_{L,s}(x) \leq 2^{-n+1}$.*

Lemma 2 ([22, Lemma 2.10]). *Given an n -dimensional lattice L , we have $\Pr_{x \sim D_{L,s}}[\|x\| > s\sqrt{n}] \leq 2^{-n+1}$.*

Ideal lattices. Ideal lattices are a subset of lattices with the computationally interesting property of being related to polynomials via structured matrices. The n -dimensional vector-matrix product costs $\widetilde{O}(n)$ arithmetic operations instead of $O(n^2)$. Let $f \in \mathbb{Z}[x]$ a monic degree n polynomial. For any $g \in \mathbb{Q}[x]$, there is a unique pair (q, r) with $\deg(r) < n$ and $g = qf + r$. We denote r by $g \bmod f$ and identify r with the vector $\mathbf{r} \in \mathbb{Q}^n$ of its coefficients. We define $\text{rot}_f(\mathbf{r}) \in \mathbb{Q}^{n \times n}$ as the matrix whose rows are the $x^i r(x) \bmod f(x)$'s, for $0 \leq i < n$. We extend that notation to the matrices A over $\mathbb{Q}[x]/f$, by applying rot_f component-wise. Note that $\text{rot}_f(g_1)\text{rot}_f(g_2) = \text{rot}_f(g_1g_2)$ for any $g_1, g_2 \in \mathbb{Q}[x]/f$. The strengths of our cryptographic constructions depend on the choice of f . Its quality is quantified by its expansion factor (we adapt the definition of [17] to the Euclidean norm):

$$\text{EF}(f, k) = \max \left\{ \frac{\|g \bmod f\|}{\|g\|} \mid g \in \mathbb{Z}[x] \setminus \{0\} \text{ and } \deg(g) \leq k(\deg(f) - 1) \right\},$$

where we identified the polynomial $g \bmod f$ (resp. g) with the coefficients vector. Note that if $\deg(g) < n$, then $\|\text{rot}_f(g)\| \leq \text{EF}(f, 2) \cdot \|g\|$. We will concentrate on the polynomials $x^{2^k} + 1$, although most of our results are more general. We recall some basic properties of $x^{2^k} + 1$ (see [7] for the last one).

Lemma 3. *Let $k \geq 0$ and $n = 2^k$. Then $f(x) = x^n + 1$ is irreducible in $\mathbb{Q}[x]$. Its expansion factor is $\leq \sqrt{2}$. Also, for any $g = \sum_{i < n} g_i x^i \in \mathbb{Q}[x]/f$, we have $\text{rot}_f(g)^T = \text{rot}_f(\bar{g})$ where $\bar{g} = g_0 - \sum_{1 \leq i < n} g_{n-i} x^i$. Furthermore, if q is a prime such that $2n \mid (q - 1)$, then f has n linear factors in $\mathbb{Z}_q[x]$. Finally, if $k \geq 2$ and q is a prime with $q \equiv 3 \pmod 8$, then $f = f_1 f_2 \bmod q$ where each f_i is irreducible in $\mathbb{Z}_q[x]$ and can be written $f_i = x^{n/2} + t_i x^{n/4} - 1$ with $t_i \in \mathbb{Z}_q$.*

Let I be an ideal of $\mathbb{Z}[x]/f$, i.e., a subset of $\mathbb{Z}[x]/f$ closed under addition and multiplication by any element of $\mathbb{Z}[x]/f$. It corresponds to a sublattice of \mathbb{Z}^n . An f -ideal lattice is a sublattice of \mathbb{Z}^n that corresponds to an ideal $I \subseteq \mathbb{Z}[x]/f$.

Hard lattice problems. The most famous lattice problem is SVP. Given a basis of a lattice L , it aims at finding a shortest vector in $L \setminus \{\mathbf{0}\}$. It can be relaxed by asking for a non-zero vector that is no longer than $\gamma(n)$ times a solution to SVP, for a prescribed function $\gamma(\cdot)$. The best polynomial time algorithm [435] solves γ -SVP only for a slightly subexponential γ . When γ is polynomial in n , then the most efficient algorithm [4] has an exponential worst-case complexity both in time and space. If we restrict the set of input lattices to ideal lattices, we obtain the problem Ideal-SVP (resp. γ -Ideal-SVP), which is implicitly parameterized by a sequence of polynomials f of growing degrees. No algorithm is known to perform non-negligibly better for Ideal-SVP than for SVP. It is believed that no subexponential quantum algorithm solves the computational variants of SVP

or Ideal-SVP in the worst case. These worst-case problems can be reduced to the following average-case problems, introduced in [1] and [9].

Definition 1. *The Small Integer Solution problem with parameters $q(\cdot)$, $m(\cdot)$, $\beta(\cdot)$ ($\text{SIS}_{q,m,\beta}$) is as follows: Given n and a matrix G sampled uniformly in $\mathbb{Z}_{q(n)}^{m(n) \times n}$, find $\mathbf{e} \in \mathbb{Z}^{m(n)} \setminus \{\mathbf{0}\}$ such that $\mathbf{e}^T G = \mathbf{0} \pmod{q(n)}$ (the modulus being taken component-wise) and $\|\mathbf{e}\| \leq \beta(n)$. The Ideal Small Integer Solution problem with parameters q, m, β and f ($\text{Ideal-SIS}_{q,m,\beta}^f$) is as follows: Given n and m polynomials g_1, \dots, g_m chosen uniformly and independently in $\mathbb{Z}_q[x]/f$, find $e_1, \dots, e_m \in \mathbb{Z}[x]$ not all zero such that $\sum_{i \leq m} e_i g_i = 0$ in $\mathbb{Z}_q[x]/f$ and $\|\mathbf{e}\| \leq \beta$, where \mathbf{e} is the vector obtained by concatenating the coefficients of the e_i 's.*

The above problems can be interpreted as lattice problems. If $G \in \mathbb{Z}_q^{m \times n}$, then the set $G^\perp = \{\mathbf{b} \in \mathbb{Z}^m \mid \mathbf{b}^T G = \mathbf{0} \pmod{q}\}$ is an m -dimensional lattice and solving SIS corresponds to finding a short non-zero vector in it. Similarly, Ideal-SIS consists in finding a small non-zero element in the $\mathbb{Z}[x]/f$ -module $M^\perp(\mathbf{g}) = \{\mathbf{b} \in (\mathbb{Z}[x]/f)^m \mid \langle \mathbf{b}, \mathbf{g} \rangle = 0 \pmod{q}\}$, where $\mathbf{g} = (g_1, \dots, g_m)$. It can be seen as a lattice problem by applying the rot_f operator. Note that the m of SIS is n times larger than the m of Ideal-SIS. Lyubashevsky and Micciancio [17] reduced Ideal-SVP to Ideal-SIS. The approximation factors in [17] are given in terms of the infinity norm. For our purposes, it is more natural to use the Euclidean norm. To avoid losing a \sqrt{n} factor by simply applying the norm equivalence formula, we modify the proof of [17]. We also adapt it to handle the case where the Ideal-SIS solver has a subexponentially small success probability, at the cost of an additional factor of $\tilde{O}(\sqrt{n})$ in the SVP approximation factor.

Theorem 1. *Suppose that f is irreducible over \mathbb{Q} . Let $m = \text{Poly}(n)$ and $q = \tilde{O}(\text{EF}(f, 3)\beta m^2 n)$ be integers. A polynomial-time (resp. subexponential-time) algorithm solving $\text{Ideal-SIS}_{q,m,\beta}^f$ with probability $1/\text{Poly}(n)$ (resp. $2^{-o(n)}$) can be used to solve γ -Ideal-SVP in polynomial-time (resp. subexponential-time) with $\gamma = \tilde{O}(\text{EF}^2(f, 2)\beta mn^{1/2})$ (resp. $\gamma = \tilde{O}(\text{EF}^2(f, 2)\beta mn)$).*

The problem LWE is dual to SIS in the sense that if $G \in \mathbb{Z}_q^{m \times n}$ is the SIS-matrix, then LWE involves the dual of the lattice G^\perp . We have $\widehat{G^\perp} = \frac{1}{q}L(G)$ where $L(G) = \{\mathbf{b} \in \mathbb{Z}^m \mid \exists \mathbf{s} \in \mathbb{Z}_q^n, G\mathbf{s} = \mathbf{b} \pmod{q}\}$.

Definition 2. *The Learning With Errors problem with parameters q, m and a distribution χ on $\mathbb{R}/[0, q)$ ($\text{LWE}_{q,m;\chi}$) is as follows: Given n , a matrix $G \in \mathbb{Z}_q^{m \times n}$ sampled uniformly at random and $G\mathbf{s} + \mathbf{e} \in (\mathbb{R}/[0, q))^n$, where $\mathbf{s} \in \mathbb{Z}_q^n$ is chosen uniformly at random and the coordinates of $\mathbf{e} \in (\mathbb{R}/[0, q))^m$ are independently sampled from χ , find \mathbf{s} . The Ideal Learning With Errors problem with parameters q, m , a distribution χ on $\mathbb{R}/[0, q)$ and f ($\text{Ideal-LWE}_{m,q;\chi}^f$) is the same as above, except that $G = \text{rot}_f(\mathbf{g})$ with \mathbf{g} chosen uniformly in $(\mathbb{Z}_q[x]/f)^m$.*

We will use the following results on the LWE and Ideal-LWE lattices.

Lemma 4. *Let n, m and q be integers with q prime, $m \geq 5n \log q$ and $n \geq 10$. Then for all but a fraction $\leq q^{-n}$ of the G 's in $\mathbb{Z}_q^{m \times n}$, we have $\lambda_1^\infty(L(G)) \geq q/4$ and $\lambda_1(L(G)) \geq 0.07\sqrt{mq}$.*

Lemma 5. *Let n, m and q be integers with $q = 3 \pmod 4$ prime and $m \geq 41 \log q$ and $n = 2^k \geq 32$. Then for all but a fraction $\leq q^{-n}$ of the \mathbf{g} 's in $(\mathbb{Z}_q[x]/f)^m$, we have $\lambda_1^\infty(L(\text{rot}_f(\mathbf{g}))) \geq q/4$ and $\lambda_1(L(\text{rot}_f(\mathbf{g}))) \geq 0.017\sqrt{mnq}$.*

3 Hiding a Trapdoor in Ideal-SIS

In this section we show how to hide a trapdoor in the problem Ideal-SIS. Ajtai [2] showed how to simultaneously generate a (SIS) matrix $A \in \mathbb{Z}_q^{m \times n}$ and a (trapdoor) basis $S = (\mathbf{s}_1, \dots, \mathbf{s}_m) \in \mathbb{Z}^{m \times m}$ of the lattice $A^\perp = \{\mathbf{b} \in \mathbb{Z}^m : \mathbf{b}^T A = \mathbf{0} \pmod q\}$, with the following properties:

1. The distribution of A is close to the uniform distribution over $\mathbb{Z}_q^{m \times n}$.
2. The basis vectors $\mathbf{s}_1, \dots, \mathbf{s}_m$ are short.

Recently, Alwen and Peikert [5] improved Ajtai's construction in the sense that the created basis has shorter vectors: $\|S\| = \tilde{O}(n \log q)$ with $m = \Omega(n \log q)$ and overwhelming probability and $\|S\| = O(\sqrt{n \log q})$ with $m = \Omega(n \log^2 q)$. We modify both constructions to obtain a trapdoor generation algorithm for the problem Ideal-SIS, with a resulting basis whose norm is as small as the one of [5].

Before describing the construction, we notice that the construction of [5] relies on the Hermite Normal Form (HNF), but that here there is no Hermite Normal Form for the rings under scope. We circumvent this issue by showing that except in negligibly rare cases we may use a matrix which is HNF-like.

Theorem 2. *There exists a probabilistic polynomial time algorithm with the following properties. It takes as inputs n, σ, r , an odd prime q , and integers m_1, m_2 . It also takes as input a degree n polynomial $f \in \mathbb{Z}[x]$ and random polynomials $\mathbf{a}_1 \in (\mathbb{Z}_q[x]/f)^{m_1}$. We let $f = \prod_{i \leq t} f_i$ be the factorization of f over \mathbb{Z}_q . We let $\kappa = \lceil 1 + \log q \rceil$, $\Delta = \left(\prod_{i \leq t} \left(1 + \left(\frac{q}{3^r} \right)^{\deg f_i} \right) - 1 \right)^{1/2}$ and $m = m_1 + m_2$. The algorithm succeeds with probability $\geq 1 - p$ over \mathbf{a}_1 , where $p = (1 - \prod_{i \leq t} (1 - q^{-\deg f_i}))^\sigma$. When it does, it returns $\mathbf{a} = \begin{pmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \end{pmatrix} \in (\mathbb{Z}_q[x]/f)^m$ and a basis S of the lattice $\text{rot}_f(\mathbf{a})^\perp$, such that:*

1. The distance to uniformity of \mathbf{a} is at most $p + m_2 \Delta$.
2. The quality of S is as follows:
 - If $m_1 \geq \max\{\sigma, \kappa, r\}$ and $m_2 \geq \kappa$, then $\|S\| \leq \text{EF}(f, 2) \cdot \sqrt{2} \kappa r^{1/2} n^{3/2}$. Additionally, $\|S\| \leq \text{EF}(f, 2) \sqrt{3a\kappa r} \cdot n$ with probability $1 - 2^{-a + O(\log nm_1 r)}$ for a super-logarithmic function $a = a(n) = \omega(\log n)$.
 - If $m_1 \geq \max\{\sigma, \kappa, r\}$ and $m_2 \geq \kappa m_1$, then $\|S\| \leq \text{EF}(f, 2) (4\sqrt{nr} + 3)$.
3. In particular, for $f = x^{2^k} + 1$ with $k \geq 2$ and a prime q with $q \equiv 3 \pmod 8$, the following holds:

- We can set $\sigma = 1$ and $r = \lceil 1 + \log_3 q \rceil$. Then, the error probability is $p = q^{-\Omega(n)}$ and the parameter Δ is $2^{-\Omega(n)}$.
- If $m_1, m_2 \geq \kappa$, then $\|S\| \leq \sqrt{6\kappa r} \cdot n = O(\sqrt{an} \log q)$ with probability $1 - 2^{-a + O(\log nm_1 r)}$ for a super-logarithmic function $a = a(n) = \omega(\log n)$.
- If $m_1 \geq \kappa$ and $m_2 \geq \kappa m_1$, then $\|S\| \leq \sqrt{2}(4\sqrt{nr} + 3) = O(\sqrt{n \log q})$.

In the rest of this section, we only describe the analog of the second construction of Alwen and Peikert, i.e., the case $m_2 \geq \kappa m_1$, due to lack of space.

3.1 A Trapdoor for Ideal-SIS

We now construct the trapdoor for Ideal-SIS. More precisely, we want to simultaneously construct a uniform $\mathbf{a} \in \mathcal{R}^m$ with $\mathcal{R} = \mathbb{Z}_q[x]/f$, and a small basis S of the lattice A^\perp where $A = \text{rot}_f(\mathbf{a})$. For this, it suffices to find a basis of the module $M^\perp(\mathbf{a}) = \{\mathbf{y} \in \mathcal{R}_0^m \mid \langle \mathbf{y}, \mathbf{a} \rangle \equiv 0 \pmod q\}$, with $\mathcal{R}_0 = \mathbb{Z}[x]/f$.

The principle of the design. In the following, for two matrices X and Y , $[X|Y]$ denotes the concatenation of the columns of X followed by Y and $[X; Y]$ denotes the concatenation of the rows of X and the rows of Y .

We mainly follow the Alwen-Peikert construction. Let $m_1 \geq \sigma, r$. Let us assume that we generate random polynomials $A_1 = [a_1, \dots, a_{m_1}]^T \in \mathcal{R}^{m_1 \times 1}$. We will construct a random matrix $A_2 \in \mathcal{R}^{m_2 \times 1}$ with a structured matrix $S \in \mathcal{R}_0^{m \times m}$ such that $SA = 0$ and S is a basis of the module $M^\perp(\mathbf{a})$, where $A = [A_1; A_2]$. We first construct an HNF-like basis F of the module $M^\perp(\mathbf{a})$ with A . Next, we construct a unimodular matrix Q such that $S = QF$ is a short basis of the module. More precisely, S has the following form:

$$S = \begin{bmatrix} V & P \\ D & B \end{bmatrix} = \underbrace{\begin{bmatrix} -I_{m_1} & P \\ 0 & B \end{bmatrix}}_Q \cdot \underbrace{\begin{bmatrix} H & 0 \\ U & I_{m_2} \end{bmatrix}}_F.$$

Note that, by setting B lower triangular with diagonal coefficients equal to 1, the matrix Q is unimodular.

In this design principle, we want $FA = 0$. Hence, we should set

$$HA_1 = 0 \text{ and } A_2 = -UA_1.$$

Notice that, in order to prove that F is a basis of A^\perp , it suffices to show that H is a basis of A_1^\perp . The first equation is satisfied by setting H be an HNF-like matrix (see below). By setting $U = G + R$, with G to be defined later on and R a random matrix, we have that A_2 is almost uniformly random in \mathcal{R} by Micciancio’s regularity lemma (Lemma 6). More precisely, the i -th row of R is chosen from $(\{-1, 0, 1\}^n)^r \times (\{0\}^n)^{m_1-r}$.

Lemma 6 (Adapted from [20, Th. 4.2]). *Let \mathbb{F} be a finite field and $f \in \mathbb{F}[x]$ be monic and of degree $n > 0$. Let R be the ring $\mathbb{F}[x]/f$. Let $D \subseteq \mathbb{F}$ and $r > 0$. For $a_1, \dots, a_r \in R$, we denote by $H(a_1, \dots, a_r)$ the random variable $\sum_{i \leq r} b_i a_i \in$*

R where the b_i 's are degree $< n$ polynomials with coefficients chosen independently and uniformly in D . If U_1, \dots, U_r are independent uniform random variables in R , then the statistical distance to uniformity of $(U_1, \dots, U_r, H(U_1, \dots, U_r))$ is below:

$$\frac{1}{2} \sqrt{\prod_{i \leq t} \left(1 + \left(\frac{|\mathbb{F}|}{|D|^r} \right)^{\deg f_i} \right) - 1},$$

where $f = \prod_{i \leq t} f_i$ is the factorization of f over \mathbb{F} .

We show below how to choose P and G such that $PG = H - I_{m_1}$. With this relation, the design principle form of S therefore implies that $V = -H + P(G + R) = PR - I_{m_1}$, and $D = B(G + R)$. Our constructions for P, G, B also ensure that P, B and BG have ‘small’ entries so that S has ‘small’ entries.

A construction of H without HNF. We start with how to construct H for $A_1 = [a_1, \dots, a_{m_1}]^T \in \mathcal{R}^{m_1 \times 1}$. Since $m_1 \geq \max\{\sigma, \kappa, r\}$, we have $a_{i^*} \in \mathcal{R}^*$ for some index i^* with probability at least $1 - p$, where \mathcal{R}^* denotes the set of invertible elements of \mathcal{R} . For now, we set $i^* = 1$ for simplicity. Using this a_{i^*} , we can construct an HNF-like matrix H : the first row is $q\mathbf{e}_1$ and the i -th row is $h_i\mathbf{e}_1 + \mathbf{e}_i$ for $i = 2, \dots, m_1$, where \mathbf{e}_i is a row vector in $\mathcal{R}_0^{m_1}$ such that the i -th element is 1 and others are 0, and $h_i = -a_i \cdot a_1^{-1} \bmod q$ such that $h_i \in [0, q)^n$. Let \mathbf{h}_i denote the i -th row of H . By the definition of H , $H \cdot A_1 \equiv 0 \bmod q$. Thus, each row vector \mathbf{h}_i is in $M^\perp(\mathbf{a}_1)$, where $\mathbf{a}_1 = A_1$. It is obvious that $\mathbf{h}_1, \dots, \mathbf{h}_{m_1}$ are linearly independent over \mathcal{R}_0 . Hence, we need to only show that H is indeed the basis of $M^\perp(\mathbf{a}_1)$, but this is a routine work.

Next, we consider the case where $i^* \neq 1$. In this case, we swap rows 1 and i^* of A_1 so that $a_1 \in \mathcal{R}^*$, and call it A'_1 . Applying the method above, we get a basis H' of $A_1^\perp(A'_1)$. By swapping columns 1 and i^* and rows 1 and i^* of H' , we get a basis H of $A_1^\perp(A_1)$. In the following, we denote by i^* the index i such that $a_i \in \mathcal{R}^*$ and $h_{i,i} = q$. Note that our strategy fails if there is no index i such that $a_i \in \mathcal{R}^*$: this is not an issue, as this occurs only with small probability.

Preliminaries of the construction. Hereafter, we set $W = BG$. We often use the matrix $T_\kappa = (t_{i,j}) \in \mathcal{R}_0^{\kappa \times \kappa}$, where $t_{i,i} = 1, t_{i+1,i} = -2$, and all other $t_{i,j}$'s are 0. Notice that the i -th row of T_κ^{-1} is $(2^{i-1}, 2^{i-2}, \dots, 1, 0, \dots, 0) \in \mathcal{R}_0^\kappa$.

3.2 An Analogue to the Second Alwen-Peikert Construction

The idea of the second construction in [5] is to have G contain the rows of $H - I_{m_1}$. This helps decrease the norms of the rows of P and V . To do so, we define $B = \text{diag}(T_\kappa, \dots, T_\kappa, I_{m_2 - m_1\kappa})$. Note that $B^{-1} = \text{diag}(T_\kappa^{-1}, \dots, T_\kappa^{-1}, I_{m_2 - m_1\kappa})$.

Let \mathbf{h}'_j denote the j -th row of $H - I_{m_1}$. Let $W = [W_1; W_2; \dots; W_{m_1}; 0]$, where $W_j = [\mathbf{w}_{j,\kappa}; \dots; \mathbf{w}_{j,1}] \in \mathcal{R}_0^{\kappa \times m_1}$. We compute the $\mathbf{w}_{j,k}$'s such that $\mathbf{h}'_j = \sum_k 2^{k-1} \cdot \mathbf{w}_{j,k}$ and the components of all $\mathbf{w}_{j,k}$'s are polynomials with coefficients in $\{0, 1\}$.

By this construction, $T_\kappa^{-1} \cdot W_j$ contains \mathbf{h}'_j in the last row. Then, $G = B^{-1} \cdot W$ contains rows \mathbf{h}'_j for $j = 1, \dots, m_1$. The matrix $P = [\mathbf{p}_1; \dots; \mathbf{p}_{m_1}]$ picks all rows $\mathbf{h}'_1, \dots, \mathbf{h}'_{m_1}$ in G by setting $\mathbf{p}_j = \mathbf{e}_{\kappa j} \in \mathcal{R}_0^{m_2}$.

The norm of S is $\max\{\|S_1\|, \|S_2\|\}$, where $S_1 = [V|P]$ and $S_2 = [D|B]$. For simplicity, we only consider the case where $f = x^n + 1$. In the general case, the bound on $\|S\|$ involves an extra $\text{EF}(f, 2)$ factor.

We have that $\|BG\|^2 = \|W\|^2 \leq n$, since the entries of \mathbf{h}'_j are all 0 except one which is either $h_{i^*,j}$ or $q - 1$. Hence, we obtain that

$$\|S_2\|^2 \leq \|D\|^2 + \|B\|^2 \leq (3\sqrt{nr} + \sqrt{n})^2 + 5 \leq (4\sqrt{nr} + 3)^2.$$

It is obvious that $\|P\| \leq 1$. Additionally, we have that $\|PR\|^2 \leq nr$. Therefore:

$$\|S_1\|^2 \leq \|V\|^2 + \|P\|^2 \leq (\sqrt{nr} + 1)^2 + 1 \leq (\sqrt{nr} + 2)^2,$$

which completes the proof of Theorem 2. □

4 From LWE to SIS

We show that any efficient algorithm solving LWE with some non-negligible probability may be used by a quantum machine to efficiently solve SIS with non-negligible probability. A crucial property of the reduction is that the matrix underlying the SIS and LWE instances is preserved. This allows the reduction to remain valid while working on Ideal-SIS and Ideal-LWE.

Theorem 3. *Let q, m, n be integers, and $\alpha \in (0, 1)$ with $n \geq 32$, $\text{Poly}(n) \geq m \geq 5n \log q$ and $\alpha < \min\left(\frac{1}{10\sqrt{\ln(10m)}}, 0.006\right)$. Suppose that there exists an algorithm that solves $\text{LWE}_{m,q;\Psi_{\alpha q}}$ in time T and with probability $\varepsilon \geq 4m \exp\left(-\frac{\pi}{4\alpha^2}\right)$. Then there exists a quantum algorithm that solves $\text{SIS}_{m,q;\frac{\sqrt{m}}{2\alpha}}$ in time $\text{Poly}(T, n)$ and with probability $\frac{\varepsilon^3}{64} - O(\varepsilon^5) - 2^{-\Omega(n)}$. The result still holds when replacing LWE by Ideal-LWE^f and SIS by Ideal-SIS^f, for $f = x^n + 1$ with $n = 2^k \geq 32$, $m \geq 41 \log q$ and $q \equiv 3 \pmod 8$.*

When $\alpha = O(1/\sqrt{n})$, the reduction applies even to a subexponential algorithm for LWE (with success probability $\varepsilon = 2^{-o(n)}$), transforming it into a subexponential quantum algorithm for SIS (with success probability $\varepsilon = 2^{-o(n)}$). The reduction works also for larger $\alpha = O(1/\sqrt{\log n})$, but in this case only applies to polynomial algorithms for LWE (with success probability $\varepsilon = \Omega(1/\text{Poly}(n))$).

The reduction is made of two components. First, we argue that an algorithm solving LWE provides an algorithm that solves a certain bounded distance decoding problem, where the error vector is normally distributed. In a second step, we show that Regev’s quantum algorithm [32, Lemma 3.14] can use such an algorithm to construct small solutions to SIS.

4.1 From LWE to BDD

An algorithm solving LWE allows us to solve, for certain lattices, a variation of the Bounded Distance Decoding problem. In that variation of BDD, the error vector is sampled according to a specified distribution.

Definition 3. *The problem BDD_χ with parameter distribution $\chi(\cdot)$ is as follows: Given an n -dimensional lattice L and a vector $\mathbf{t} = \mathbf{b} + \mathbf{e}$ where $\mathbf{b} \in L$ and \mathbf{e} is distributed according to $\chi(n)$, the goal is to find \mathbf{b} . We say that a randomized algorithm \mathcal{A} solves BDD_χ for a lattice L with success probability $\geq \varepsilon$ if, for every $\mathbf{b} \in L$, on input $\mathbf{t} = \mathbf{b} + \mathbf{e}$, algorithm \mathcal{A} returns \mathbf{b} with probability $\geq \varepsilon$ over the choice of \mathbf{e} and the randomness of \mathcal{A} .*

For technical reasons, our reduction will require a randomized BDD_χ algorithm whose behaviour is independent of the solution vector \mathbf{b} , even when the error vector is fixed. This is made precise below.

Definition 4. *A randomized algorithm \mathcal{A} solving BDD_χ for lattice L is said to be strongly solution-independent (SSI) if, for every fixed error vector \mathbf{e} , the probability (over the randomness of \mathcal{A}) that, given input $\mathbf{t} = \mathbf{b} + \mathbf{e}$ with $\mathbf{b} \in L$, algorithm \mathcal{A} returns \mathbf{b} is independent of \mathbf{b} .*

We show that if we have an algorithm that solves $\text{LWE}_{m,q;\Psi_{\alpha q}}$, then we can construct an algorithm solving $\text{BDD}_{\nu_{\alpha q}}$ for some lattices. Moreover, the constructed BDD algorithm is SSI.

Lemma 7. *Let q, m, n be integers and $\alpha \in (0, 1)$, with $m, \log q = \text{Poly}(n)$. Suppose that there exists an algorithm \mathcal{A} that solves $\text{LWE}_{m,q;\Psi_{\alpha q}}$ in time T and with probability $\varepsilon \geq 4m \exp(-\frac{\pi}{4\alpha^2})$. Then there exists $\mathcal{S} \subseteq \mathbb{Z}_q^{m \times n}$ of proportion $\geq \varepsilon/2$ and an SSI algorithm \mathcal{A}' such that if $G \in \mathcal{S}$, algorithm \mathcal{A}' solves $\text{BDD}_{\nu_{\alpha q}}$ for $L(G)$ in time $T + \text{Poly}(n)$ and with probability $\geq \varepsilon/4$.*

Proof. If $G \in \mathbb{Z}_q^{m \times n}$ and $\mathbf{s} \in \mathbb{Z}_q^n$ are sampled uniformly and if the coordinates of \mathbf{e} are sampled according to $\Psi_{\alpha q}$, then \mathcal{A} finds \mathbf{s} with probability $\geq \varepsilon$ over the choices of G, \mathbf{s} and \mathbf{e} and a string w of internal random bits. This implies that there exists a subset \mathcal{S} of the G 's of proportion $\geq \varepsilon/2$ such that for any $G \in \mathcal{S}$, algorithm \mathcal{A} succeeds with probability $\geq \varepsilon/2$ over the choices of \mathbf{s}, \mathbf{e} and w . For any $G \in \mathcal{S}$, we have $\Pr_{\mathbf{s}, \mathbf{e}, w}[\mathcal{A}(G\mathbf{s} + \mathbf{e}, w) = \mathbf{s}] \geq \varepsilon/2$.

On input $\mathbf{t} = \mathbf{b} + \mathbf{e}$, algorithm \mathcal{A}' works as follows: it samples \mathbf{s} uniformly in \mathbb{Z}_q^n ; it computes $\mathbf{t}' = \mathbf{t} + A\mathbf{s}$, which is of the form $\mathbf{t}' = G\mathbf{s}' + q\mathbf{k} + \mathbf{e}$, where $\mathbf{k} \in \mathbb{Z}^m$; it calls \mathcal{A} on $\mathbf{t}' \bmod q$ and finds \mathbf{s}' (with probability $\geq \varepsilon/2$); it then computes $\mathbf{e}' = \mathbf{t}' - G\mathbf{s}' \bmod q$ and returns $\mathbf{t} - \mathbf{e}'$. Suppose that \mathcal{A} succeeds, i.e., we have $\mathbf{s} = \mathbf{s}'$. Then $\mathbf{e}' = \mathbf{e} \bmod q$. Using the standard tail bound on the continuous Gaussian and the lower bound on ε we obtain that \mathbf{e} has a component of magnitude $\geq q/2$ with probability $\leq m \exp(-\pi/(2\alpha)^2) \leq \varepsilon/4$. The algorithm thus succeeds with probability $\geq \varepsilon/2 - \varepsilon/4 = \varepsilon/4$. □

We now show that an algorithm solving $\text{BDD}_{\nu_{\alpha q}}$ can be used to solve a quantized version of it. This quantization is required for the quantum part of our reduction.

The intuition behind the proof is that the discretization grid is so fine (the parameter R can be chosen extremely large) that at the level of the grid the distribution ν_s looks constant.

Lemma 8. *Let $s > 0$ and L be an n -dimensional. Suppose that there exists an SSI algorithm \mathcal{A} that solves BDD_{ν_s} for L in time T and with probability ε . Then there exists an R , whose bit-length is polynomial in $T, n, |\log s|$ and the bit-size of the given basis of L , and an SSI algorithm \mathcal{A}' that solves $\text{BDD}_{D_{L/R,s}}$ within a time polynomial in $\log R$ and with probability $\geq \varepsilon - 2^{-\Omega(n)}$.*

At this point, we have an R of bit-length polynomial in $T, n, |\log \alpha|$ and an SSI algorithm \mathcal{B} with run-time polynomial in $\log R$ that solves $\text{BDD}_{D_{L(G)/R,\alpha q}}$, for any G in a subset $\mathcal{S} \subseteq \mathbb{Z}_q^{m \times n}$ of proportion $\geq \varepsilon/2$, with probability $\geq \varepsilon/4 - 2^{-\Omega(n)}$ over the random choices of e and the internal randomness w . In the following we assume that on input $t = b + e$, algorithm \mathcal{B} outputs e when it succeeds, rather than b . We implement \mathcal{B} quantumly as follows: the quantum algorithm \mathcal{B}_Q maps the state $|e\rangle |b + e\rangle |w\rangle$ to the state $|e - \mathcal{B}(b + e, w)\rangle |b + e\rangle |w\rangle$.

4.2 A New Interpretation of Regev’s Quantum Reduction

We first recall Regev’s quantum reduction [32, Lemma 3.14]. It uses a randomized BDD oracle \mathcal{B}^{wc} that finds the closest vector in a given lattice L to a given target vector, as long as the target is within a prescribed distance $d < \frac{\lambda_1(L)}{2}$ of L (as above, we assume that \mathcal{B}^{wc} returns the error vector). It returns a sample from the distribution $D_{\hat{L}, \frac{\sqrt{n}}{\sqrt{2d}}}$. We implement oracle \mathcal{B}^{wc} as a quantum oracle \mathcal{B}_Q^{wc} as above. We assume \mathcal{B}_Q^{wc} accepts random inputs of length ℓ .

1. Set R to be a large constant and build a quantum state which is within ℓ_2 distance $2^{-\Omega(n)}$ of the normalized state corresponding to $\sum_{w \in \{0,1\}^\ell} \sum_{x \in \frac{L}{R}, \|x\| < d} \rho_{\frac{d}{\sqrt{n}}}(x) |x \bmod L\rangle |w\rangle$.
2. Apply the BDD oracle \mathcal{B}_Q^{wc} to the above state to remove the entanglement and obtain a state which is within ℓ_2 distance $2^{-\Omega(n)}$ of the normalized state corresponding to $\sum_{x \in \frac{L}{R}, \|x\| < d} \rho_{\frac{d}{\sqrt{n}}}(x) |0\rangle |x \bmod L\rangle |w\rangle$.
3. Apply the quantum Fourier transform over \mathbb{Z}_R^n to the second register to obtain a state that is within ℓ_2 distance $2^{-\Omega(n)}$ of the normalized state corresponding to $\sum_{x \in \hat{L}, \|x\| < \frac{n}{d}} \rho_{\frac{\sqrt{n}}{d}}(x) |x \bmod (R \cdot \hat{L})\rangle$.
4. Measure the latter to obtain a vector $\hat{b} \bmod R \cdot \hat{L}$. Using Babai’s algorithm [6], recover \hat{b} and output it. Its distribution is within statistical distance $2^{-\Omega(n)}$ of $D_{\hat{L}, \frac{\sqrt{n}}{\sqrt{2d}}}$.

We now replace the perfect oracle \mathcal{B}_Q^{wc} by an imperfect one.

Lemma 9. *Suppose we are given an n -dimensional lattice L , parameters $R > 2^{2n} \lambda_n(L)$ and $s < \frac{\lambda_1(L)}{2\sqrt{2n}}$, and an SSI algorithm \mathcal{B} that solves $\text{BDD}_{D_{\frac{L}{R},s}}$ for L with run-time T and success probability ε . Then there exists a quantum algorithm \mathcal{R}*

which outputs a vector $\widehat{\mathbf{b}} \in \widehat{L}$ whose distribution is within distance $1 - \varepsilon^2/2 + O(\varepsilon^4) + 2^{-\Omega(n)}$ of $D_{\widehat{L}, \frac{1}{2s}}$. It finishes in time polynomial in $T + \log R$.

Proof. The quantum algorithm \mathcal{R} is Regev’s algorithm above with parameter $d = \sqrt{2ns} < \frac{\lambda_1(L)}{2}$, where \mathcal{B}_Q^{wc} is replaced by the quantum implementation \mathcal{B}_Q of \mathcal{B} . We just saw that if the $\text{BDD}_{D_{L/R,s}}$ oracle was succeeding with probability $1 - 2^{-\Omega(n)}$, then the output vector $\widehat{\mathbf{b}}$ would follow a distribution whose statistical distance to $D_{\widehat{L}, \frac{1}{2s}}$ would be $2^{-\Omega(n)}$. To work around the requirement that the oracle succeeds with overwhelming probability, we use the notion of trace distance between two quantum states, which is an adaptation of the statistical distance (see [25], Ch. 9)). The trace distance between two (pure) quantum states $|t_1\rangle$ and $|t_2\rangle$ is $\delta(|t_1\rangle, |t_2\rangle) = \sqrt{1 - |\langle t_1 | t_2 \rangle|^2}$. Its most important property is that for any generalized measurement (POVM), if D_1 (resp. D_2) is the resulting probability distribution when starting from $|t_1\rangle$ (resp. $|t_2\rangle$) then $\Delta(D_1, D_2) \leq \delta(|t_1\rangle, |t_2\rangle)$. Let $|t_1\rangle$ denote the state at the end of Step 2 of Regev’s algorithm when we use \mathcal{B}^{wc} , and let $|t_2\rangle$ denote the state that we obtain at the end of Step 2 when we use \mathcal{B} . We upper bound $\delta(|t_1\rangle, |t_2\rangle)$ as follows.

Since $\mathcal{B}^{wc}(\mathbf{x} \bmod L, w) = \mathbf{x}$ for $\|\mathbf{x}\| < d$, we have that $|t_1\rangle$ is within ℓ_2 distance (and hence trace distance) $2^{-\Omega(n)}$ of the normalized state

$$|t'_1\rangle = 2^{-\ell/2} \sum_{w \in \{0,1\}^\ell} \sum_{\mathbf{x} \in \frac{L}{R}} \sqrt{D_{L/R,s}^d(\mathbf{x})} |\mathbf{0}\rangle |\mathbf{x} \bmod L\rangle |w\rangle,$$

where $D_{L/R,s}^d$ denotes the normalized distribution obtained by truncating $D_{L/R,s}$ to vectors of norm $< d$. On the other hand, for the imperfect oracle \mathcal{B} , we have that $|t_2\rangle$ is within trace distance $2^{-\Omega(n)}$ of the normalized state

$$|t'_2\rangle = 2^{-\ell/2} \sum_{w \in \{0,1\}^\ell} \sum_{\mathbf{x} \in \frac{L}{R}} \sqrt{D_{L/R,s}^d(\mathbf{x})} |\mathbf{x} - \mathcal{B}(\mathbf{x} \bmod L, w)\rangle |\mathbf{x} \bmod L\rangle |w\rangle.$$

Let $S_{\mathcal{B}} = \{(\mathbf{x}, w) \in \frac{L}{R} \times \{0,1\}^\ell \mid \|\mathbf{x}\| < d \text{ and } \mathcal{B}(\mathbf{x} \bmod L, w) = \mathbf{x}\}$. Notice that, if $(\mathbf{x}, w) \notin S_{\mathcal{B}}$, the states $|\mathbf{x} - \mathcal{B}(\mathbf{x} \bmod L, w)\rangle |\mathbf{x} \bmod L\rangle |w\rangle$ and $|\mathbf{0}\rangle |\mathbf{x}' \bmod L\rangle |w'\rangle$ are orthogonal for all (\mathbf{x}', w') . Furthermore, if $(\mathbf{x}, w) \in S_{\mathcal{B}}$, the states $|\mathbf{0}\rangle |\mathbf{x} \bmod L\rangle |w\rangle$ and $|\mathbf{0}\rangle |\mathbf{x}' \bmod L\rangle |w'\rangle$ are orthogonal for all $(\mathbf{x}', w') \neq (\mathbf{x}, w)$ with $\|\mathbf{x}'\| < d$, because the mapping $\mathbf{x} \mapsto \mathbf{x} \bmod L$ is 1-1 over \mathbf{x} of norm $< d < \lambda_1(L)/2$. It follows that $|\langle t'_1 | t'_2 \rangle| = \sum_{(\mathbf{x}, w) \in S_{\mathcal{B}}} 2^{-\ell} D_{L/R,s}^d(\mathbf{x})$. Hence, $|\langle t'_1 | t'_2 \rangle|$ is equal to the probability p that $\mathcal{B}(\mathbf{x} \bmod L, w) = \mathbf{x}$, over the choices of \mathbf{x} from the distribution $D_{L/R,s}^d$ and w uniformly random in $\{0,1\}^\ell$. By Lemma 2, using the fact that $d > \sqrt{ns}$, we have $p \geq \widehat{p} - 2^{-\Omega(n)}$, where \widehat{p} is the corresponding probability when \mathbf{x} is sampled from $D_{L/R,s}$. Finally, we have $\widehat{p} = \sum_{\mathbf{x}} D_{L/R,s}(\mathbf{x}) \Pr_w[\mathcal{B}(\mathbf{x} \bmod L, w) = \mathbf{x}]$. By the strong solution-independence of \mathcal{B} , we have $\Pr_w[\mathcal{B}(\mathbf{x} \bmod L, w) = \mathbf{x}] = \Pr_w[\mathcal{B}(\mathbf{b} + \mathbf{x}, w) = \mathbf{x}]$ for any fixed $\mathbf{b} \in L$. Therefore, \widehat{p} is the success probability of \mathcal{B} in solving $\text{BDD}_{D_{L/R,s}}$, so $\widehat{p} \geq \varepsilon$ by assumption. Overall, we conclude that $\delta(|t_1\rangle, |t_2\rangle) \leq \sqrt{1 - \varepsilon^2 + 2^{-\Omega(n)}}$, and hence the output of \mathcal{R} is within statistical distance $1 - \varepsilon^2/2 + O(\varepsilon^4) + 2^{-\Omega(n)}$ of $D_{\widehat{L}, \frac{1}{2s}}$, as claimed. \square

To prove Theorem 3, we apply Lemma 9 to the lattices $L(G)$ for $G \in \mathcal{S}$, with algorithm \mathcal{B} . For that, we need to ensure that the hypothesis $\alpha q < \frac{\lambda_1(L(G))}{2\sqrt{2m}}$ is satisfied. From Lemma 4 (resp. Lemma 5 in the case of Ideal-LWE), we know that with probability $1 - 2^{-\Omega(n)}$ over the choice of G in $\mathbb{Z}_q^{m \times n}$, we have $\lambda_1^\infty(L(G)) \geq \frac{q}{4}$ and $\lambda_1(L(G)) \geq 0.07\sqrt{mq}$. For such ‘good’ G ’s, the hypothesis $\alpha q < \frac{\lambda_1(L(G))}{2\sqrt{2m}}$ is satisfied, since $\alpha < 0.006$. The set \mathcal{S}' of the G ’s in \mathcal{S} for which that condition is satisfied represents a proportion $\geq \varepsilon/2 - 2^{-\Omega(n)}$ of $\mathbb{Z}_q^{m \times n}$. Suppose now that $G \in \mathcal{S}'$. Lemma 9 shows that we can find a vector $\mathbf{s} \in G^\perp = q\widehat{L(G)}$ that follows a distribution whose distance to $D_{G^\perp, \frac{1}{2\alpha}}$ is $\Delta = 1 - \frac{\varepsilon^2}{32} + O(\varepsilon^4) + 2^{-\Omega(n)}$. Thanks to Lemmas 1 and 2 (since $G \in \mathcal{S}$ and $\alpha \leq 1/(10\sqrt{\ln(10m)})$, the hypothesis of Lemma 1 is satisfied), we have that with probability $\geq 1 - 2^{-\Omega(n)} - \Delta = \frac{\varepsilon^2}{32} - O(\varepsilon^4) - 2^{-\Omega(n)}$, the returned \mathbf{s} is a non-zero vector of G^\perp whose norm is $\leq \frac{\sqrt{m}}{2\alpha}$. Multiplying by the probability $\geq \varepsilon/2 - 2^{-\Omega(n)}$ that $G \in \mathcal{S}'$ gives the claimed success probability and completes the proof of Theorem 4. \square

5 Cryptographic Applications

We now use the results of Sections 3 and 4 to construct efficient cryptographic primitives based on ideal lattices. This includes the first provably secure lattice-based public-key encryption scheme with asymptotically optimal encryption and decryption computation costs of $\tilde{O}(1)$ bit operations per message bit.

5.1 Efficient Public-Key Encryption Scheme

Our scheme is constructed in two steps. Firstly, we use the LWE mapping $(\mathbf{s}, \mathbf{e}) \mapsto G \cdot \mathbf{s} + \mathbf{e} \pmod q$ as an injective trapdoor one-way function, with the trapdoor being the full-dimensional set of vectors in G^\perp from Section 3, and the one-wayness being as hard as Ideal-SIS (and hence Ideal-SVP) by Theorem 3. This is an efficient ideal lattice analogue of some trapdoor functions presented in [9,28] for arbitrary lattices. Secondly, we apply the Goldreich-Levin hardcore function based on Toeplitz matrices [10, Sec. 2.5] to our trapdoor function, and XOR the message with the hardcore bits to obtain a semantically secure encryption. To obtain the $\tilde{O}(1)$ amortized bit complexity per message bit, we use $\tilde{\Omega}(n)$ hardcore bits, which induces a subexponential loss in the security reduction.

Our trapdoor function family **ld-Trap** is defined in Figure 1. For security parameter $n = 2^k$, we fix $f(x) = x^n + 1$ and $q = \text{Poly}(n)$ a prime satisfying $q \equiv 3 \pmod 8$. From Lemma 3, it follows that f splits modulo q into two irreducible factors of degree $n/2$. We set $\sigma = 1$, $r = 1 + \log_3 q = \tilde{O}(1)$ and $m = (\lceil \log q \rceil + 1)\sigma + r = \tilde{O}(1)$. We define $\mathcal{R} = \mathbb{Z}_q[x]/f$. The following lemma ensures the correctness of the scheme (this is essentially identical to [28, Sec. 4.1]) and asserts that the evaluation and inversion functions can be implemented efficiently.

- **Generating a function with trapdoor.** Run the algorithm from Theorem 2 using $f = x^n + 1, n, q, r, \sigma, m$ as inputs. Suppose it succeeds. It returns $\mathbf{g} \in (\mathbb{Z}_q[x]/f)^m$ (function index) and a trapdoor full-rank set S of linearly independent vectors in $\text{rot}_f(\mathbf{g})^\perp \subseteq \mathbb{Z}_q^{mn \times mn}$ with $\|S\| \leq \sqrt{2}(4\sqrt{nr} + 3) =: L$ (we have $L = \tilde{O}(\sqrt{n})$).
- **Function evaluation.** Given function index \mathbf{g} , we define the trapdoor function $h_{\mathbf{g}} : \mathbb{Z}_q^n \times \mathbb{Z}_q^{mn} \rightarrow \mathbb{Z}_q^{mn}$ as follows. On input \mathbf{s} uniformly random in \mathbb{Z}_q^n and $\mathbf{e} \in \mathbb{Z}_q^{mn}$ sampled from $\overline{\Psi}_{\alpha q}$ (defined as the rounding of $\Psi_{\alpha q}$ to the closest integer vector), we compute and return: $\mathbf{c} = h_{\mathbf{g}}(\mathbf{s}, \mathbf{e}) := \text{rot}_f(\mathbf{g}) \cdot \mathbf{s} + \mathbf{e} \bmod q$.
- **Function inversion.** Given $\mathbf{c} = h_{\mathbf{g}}(\mathbf{s}, \mathbf{e})$ and trapdoor S , compute $\mathbf{d} = S^T \cdot \mathbf{c} \bmod q$ and $\mathbf{e}' = S^{-T} \cdot \mathbf{d}$ (in \mathbb{Q}). Compute $\mathbf{u} = \mathbf{c} - \mathbf{e}' \bmod q$ and $\mathbf{s}' = (\text{rot}_f(\mathbf{g}_1))^{-1} \cdot \mathbf{u}_1 \bmod q$, where \mathbf{u}_1 consists of the first n coordinates of \mathbf{u} . Return $(\mathbf{s}', \mathbf{e}')$.

Fig. 1. The trapdoor function family **ld-Trap**

Lemma 10. *Let $q > 2\sqrt{mn}L$ and $\alpha = o(1/(L\sqrt{\log n}))$. Then for any $\mathbf{s} \in \mathcal{R}$ and for \mathbf{e} sampled from $\overline{\Psi}_{\alpha q}$, the inversion algorithm recovers (\mathbf{s}, \mathbf{e}) with probability $1 - n^{-\omega(1)}$ over the choice of \mathbf{e} . Furthermore, the evaluation and inversion algorithms for $h_{\mathbf{g}}$ can be implemented with run-time $\tilde{O}(n)$.*

The one-wayness of **ld-Trap** is equivalent to the hardness of $\text{LWE}_{m,q;\overline{\Psi}_{\alpha q}}$. Furthermore, an instance of $\text{LWE}_{m,q;\overline{\Psi}_{\alpha q}}$ can be efficiently converted by rounding to an instance of $\text{LWE}_{m,q;\overline{\Psi}_{\alpha q}}$. This proves Lemma 11.

Lemma 11. *Any attacker against the one-wayness of **ld-Trap** (with parameters m, α, q) with run-time T and success probability ε provides an algorithm for $\text{LWE}_{m,q;\overline{\Psi}_{\alpha q}}$ with run-time T and success probability ε .*

By combining our trapdoor function with the GL hardcore function [10, Sec. 2.5] we get the encryption scheme of Figure 2.

- **Key generation.** For security parameter n , run the generation algorithm of **ld-Trap** to get an $h_{\mathbf{g}}$ and a trapdoor S . We can view the first component of the domain of $h_{\mathbf{g}}$ as a subset of $\mathbb{Z}_2^{\ell_I}$ for $\ell_I = O(n \log q) = \tilde{O}(n)$. Generate $\mathbf{r} \in \mathbb{Z}_2^{\ell_I + \ell_M}$ uniformly and define the Toeplitz matrix $M_{GL} \in \mathbb{Z}_2^{\ell_M \times \ell_i}$ (allowing fast multiplication [26]) whose i th row is $[r_i, \dots, r_{\ell_I + i - 1}]$. The public key is (\mathbf{g}, \mathbf{r}) and the secret key is S .
- **Encryption.** Given ℓ_M -bit message M with $\ell_M = n/\log n = \tilde{\Omega}(n)$ and public key (\mathbf{g}, \mathbf{r}) , sample (\mathbf{s}, \mathbf{e}) with $\mathbf{s} \in \mathbb{Z}_q^n$ uniform and \mathbf{e} sampled from $\overline{\Psi}_{\alpha q}$, and evaluate $C_1 = h_{\mathbf{g}}(\mathbf{s}, \mathbf{e})$. Compute $C_2 = M \oplus (M_{GL} \cdot \mathbf{s})$, where the product $M_{GL} \cdot \mathbf{s}$ is computed over \mathbb{Z}_2 , and \mathbf{s} is viewed as a string over $\mathbb{Z}_2^{\ell_I}$. Return the ciphertext (C_1, C_2) .
- **Decryption.** Given ciphertext (C_1, C_2) and secret key (S, \mathbf{r}) , invert C_1 to compute (\mathbf{s}, \mathbf{e}) such that $h_{\mathbf{g}}(\mathbf{s}, \mathbf{e}) = C_1$, and return $M = C_2 \oplus (M_{GL} \cdot \mathbf{s})$.

Fig. 2. The semantically secure encryption scheme **ld-Enc**

Theorem 4. *Any IND-CPA attacker against **ld-Enc** with run-time T and success probability $1/2 + \varepsilon$ provides an algorithm for Ideal-LWE $_{m,q;\overline{\Psi}_{\alpha q}}^f$ with run-time $O(2^{3\ell_M} n^3 \varepsilon^{-3} \cdot T)$ and success probability $\Omega(2^{-\ell_M} n^{-1} \cdot \varepsilon)$.*

Proof. The attacker can be converted to a GL hardcore function distinguisher that, given $C_1 = h_g(\mathbf{s}, \mathbf{e})$, M_{GL} , and ℓ_M bit string z , for \mathbf{s} sampled uniformly in \mathbb{Z}_q^n , \mathbf{e} sampled from $\tilde{\Psi}_{\alpha q}$, and M_{GL} constructed as in the key generation procedure, distinguishes whether z is uniformly random (independent of \mathbf{s} and \mathbf{e}) or $z = M_{GL} \cdot \mathbf{s}$. It has run-time T and advantage ε . The result follows by applying Lemma 2.5.8, Proposition 2.5.7 and Proposition 2.5.3 in [10]. Note that we do not need to give the vector \mathbf{e} additionally to \mathbf{s} as input to the GL function, as \mathbf{e} is uniquely determined once \mathbf{s} is given (with overwhelming probability). \square

By using Lemma 10 and Theorems 1, 3 and 4, we get our main result.

Corollary 1. *Any IND-CPA attacker against encryption scheme Id-Enc with run-time $2^{o(n)}$ and success probability $1/2 + 2^{-o(n)}$ provides a quantum algorithm for $\tilde{O}(n^2)$ -Ideal-SVP with $f(x) = x^n + 1$ and $n = 2^k$, with run-time $2^{o(n)}$ and overwhelming success probability. Furthermore, the scheme Id-Enc encrypts and decrypts $\tilde{\Omega}(n)$ bits within $\tilde{O}(n)$ bit operations, and its keys have $\tilde{O}(n)$ bits.*

5.2 Further Applications

Our results have several other applications, adapting various known constructions for unstructured lattices to ideal lattices, as summarised below.

CCA2-secure encryption. Peikert [28] derived a CCA2-secure encryption scheme from the non-structured variant of the trapdoor function family Id-Trap from Figure 1, using the framework of [31,34] for building a CCA2-secure scheme from a collection of injective trapdoor functions that is secure under correlated product (i.e., one-wayness is preserved if several functions are evaluated on the same input). The approach of [28] can be applied to Id-Trap, using the equality between Ideal-LWE $_{km}$ and the product of k instances of Ideal-LWE $_m$, multiple hardcore bits as in Id-Enc, and instantiating the required strongly unforgeable signature with the Ideal-SVP-based scheme of [18]. By choosing $k = \tilde{O}(n)$ (the bit-length of the verification key in [18]) and $\alpha = \tilde{O}(n^{-3/2})$, we obtain a CCA2-secure scheme that encrypts $\tilde{\Omega}(n)$ bits within $\tilde{O}(n^2)$ bit operations and whose security relies on the exponential quantum hardness of $\tilde{O}(n^4)$ -Ideal-SVP.

Trapdoor signatures. Gentry *et al.* [9] give a construction of a trapdoor signature (in the random oracle model) from any family of collision-resistant preimage sampleable functions (PSFs). They show how to sample preimages of $f_G(\mathbf{x}) = \mathbf{x}^T G$, where $G \in \mathbb{Z}_q^{m \times n}$, using a full-dimensional set of short vectors in G^\perp . By applying this to $G = \text{rot}_f(\mathbf{g})$ and using the trapdoor generation algorithm from Section 3, we obtain a PSF whose collision resistance relies on Ideal-SIS, and hence Ideal-SVP, and thus a structured variant of the trapdoor signature scheme of [9], with $\tilde{O}(n)$ verification time and signature length.

ID-based identification. From lattice-based signatures, we derive ID-based identification (IBI) and ID-based signature (IBS). Applying the standard strategy, we construct lattice-based IBI schemes as follows: The master generates a

key pair of a lattice-based signature scheme, say (G, S) ; Each user obtains from the master a short vector \mathbf{e} such that $\mathbf{e}^T G = H(id)$, where H is a random oracle; The prover proves to the verifier that he/she has a short vector \mathbf{e} through the Micciancio-Vadhan protocol [24]. This combination yields concurrently secure IBI schemes based on $\tilde{O}(n^2)$ -SVP and $\tilde{O}(n^2)$ -Ideal-SVP in the random oracle model. As the MV protocol is witness indistinguishable, we can use the Fiat-Shamir heuristic [8] and derive lattice-based IBS schemes.

ID-based encryption (IBE). It is shown in [9] that the unstructured variant of the above trapdoor signature can be used as the identity key extraction for an IBE scheme. This requires a ‘dual’ version of **Id-Enc**, in which the public key is of the form (\mathbf{g}, u) , where $u = H(id)$ is the hashed identity, and the secret key is the signature of id , i.e., a short preimage of u under $f_{\mathbf{g}}(\mathbf{x}) = \mathbf{x}^T \text{rot}_f(\mathbf{g})$. We construct the ‘dual’ encryption as (C_1, C_2) where $C_1 = h_{\mathbf{g}}(\mathbf{s}, \mathbf{e})$ and $C_2 = T_{\ell}(\text{rot}_f(u) \cdot \mathbf{s}) + M$, where $M \in \mathbb{Z}_q^{\ell}$ contains the message and $T_{\ell}(\text{rot}_f(u) \cdot \mathbf{s})$ denotes the first ℓ coordinates of $\text{rot}_f(u) \cdot \mathbf{s} \bmod q$. By adapting the results of [13], we show that $T_{\ell}(\text{rot}_f(u) \cdot \mathbf{s})$ is an exponentially-secure generic hardcore function for uniform $u \in \mathbb{Z}_q^n$, when $\ell = o(n)$. This allows us to prove the IND-CPA security of the resulting IBE scheme based on the hardness of Ideal-SVP.

Acknowledgements. We thank Chris Peikert and Oded Regev for helpful discussions. The first author was partly supported by the LaRedA ANR grant, the second author by a Macquarie University Research Fellowship (MQRF) and ARC Discovery Grant DP0987734, and the fourth author by KAKENHI 19-55201.

References

1. Ajtai, M.: Generating hard instances of lattice problems (extended abstract). In: Proceedings of STOC 1996, pp. 99–108. ACM, 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. Ajtai, M., Dwork, C.: A public-key cryptosystem with worst-case/average-case equivalence. In: Proceedings of STOC 1997, pp. 284–293. ACM, New York (1997)
4. Ajtai, M., Kumar, R., Sivakumar, D.: A sieve algorithm for the shortest lattice vector problem. In: Proceedings of STOC 2001, pp. 601–610. ACM, New York (2001)
5. Alwen, J., Peikert, C.: Generating shorter bases for hard random lattices. In: STACS 2009. LNCS. Springer, Heidelberg (2009)
6. Babai, L.: On Lovász’ lattice reduction and the nearest lattice point problem. *Combinatorica* 6, 1–13 (1986)
7. Blake, I.F., Gao, S., Mullin, R.C.: Explicit factorization of $x^{2^k} + 1$ over F_p with prime $p \equiv 3 \pmod{4}$. *App. Alg. in Eng., Comm. and Comp.* 4, 89–94 (1992)
8. 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)
9. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Proceedings of STOC 2008, pp. 197–206. ACM, New York (2008)

10. Goldreich, O.: Foundations of Cryptography. Basic Applications, vol. II. Cambridge University Press, Cambridge (2001)
11. Goldreich, O., Goldwasser, S., Halevi, S.: Public-key cryptosystems from lattice reduction problems. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 112–131. Springer, Heidelberg (1997)
12. Hoffstein, J., Pipher, J., Silverman, J.H.: NTRU: a ring-based public key cryptosystem. In: Buhler, J.P. (ed.) ANTS 1998. LNCS, vol. 1423, pp. 267–288. Springer, Heidelberg (1998)
13. Holenstein, T., Maurer, U., Sjödin, J.: Complete classification of bilinear hard-core functions. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 73–91. Springer, Heidelberg (2004)
14. 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)
15. 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)
16. Lyubashevsky, V.: Towards Practical Lattice-Based Cryptography. PhD thesis, University of California, San Diego (2008)
17. 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)
18. Lyubashevsky, V., Micciancio, D.: Asymptotically efficient lattice-based digital signatures. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 37–54. Springer, Heidelberg (2008)
19. Lyubashevsky, V., Micciancio, D.: On bounded distance decoding, unique shortest vectors, and the minimum distance problem. In: Halevi, S. (ed.) Crypto 2009. LNCS, vol. 5677, pp. 450–461. Springer, Heidelberg (2009)
20. Micciancio, D.: Generalized compact knapsacks, cyclic lattices, and efficient one-way functions. Computational Complexity 16(4), 365–411 (2007)
21. Micciancio, D., Goldwasser, S.: Complexity of lattice problems: a cryptographic perspective. Kluwer Academic Press, Dordrecht (2002)
22. Micciancio, D., Regev, O.: Worst-case to average-case reductions based on gaussian measures. SIAM J. Comput. 37(1), 267–302 (2007)
23. Micciancio, D., Regev, O.: Lattice-based Cryptography. In: Post-Quantum Cryptography. Springer, Heidelberg (2008)
24. 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)
25. Nielsen, M.A., Chuang, I.L.: Quantum Computation and Quantum Information. Cambridge University Press, Cambridge (2000)
26. Pan, V.Y.: Structured matrices and polynomials, unified superfast algorithms. Springer and Birkhäuser (2001)
27. Peikert, C.: Limits on the hardness of lattice problems in ℓ_p norms. Computational Complexity 2(17), 300–351 (2008)
28. Peikert, C.: Public-key cryptosystems from the worst-case shortest vector problem. In: Proceedings of STOC 2009, pp. 333–342. ACM, New York (2009)
29. Peikert, C., Rosen, A.: Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 145–166. Springer, Heidelberg (2006)

30. Peikert, C., Vaikuntanathan, V., Waters, B.: A framework for efficient and composable oblivious transfer. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 554–571. Springer, Heidelberg (2008)
31. Peikert, C., Waters, B.: Lossy trapdoor functions and their applications. In: Proceedings of STOC 2008, pp. 187–196. ACM, New York (2008)
32. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. Extended version of [33], May 2 (2009), <http://www.cs.tau.ac.il/~odedr/>
33. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Proceedings of STOC 2005, pp. 84–93. ACM, New York (2005)
34. Rosen, A., Segev, G.: Chosen-ciphertext security via correlated products. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 419–436. Springer, Heidelberg (2009)
35. Schnorr, C.P.: A hierarchy of polynomial lattice basis reduction algorithms. *Theor. Comput. Sci* 53, 201–224 (1987)
36. Schnorr, C.P.: Hot topics of LLL and lattice reduction. In: The Proceedings of the LLL+25 conference (to appear, 2009)

Smooth Projective Hashing and Password-Based Authenticated Key Exchange from Lattices

Jonathan Katz^{1,*} and Vinod Vaikuntanathan²

¹ University of Maryland

jkatz@cs.umd.edu

² IBM Research

vinodv@alum.mit.edu

Abstract. We describe a public-key encryption scheme based on lattices — specifically, based on the hardness of the *learning with error* (LWE) problem — that is secure against chosen-ciphertext attacks while admitting (a variant of) smooth projective hashing. This encryption scheme suffices to construct a protocol for password-based authenticated key exchange (PAKE) that can be proven secure based on the LWE assumption in the standard model. We thus obtain the first PAKE protocol whose security relies on a lattice-based assumption.

1 Password-Based Authenticated Key Exchange

Protocols for password-based authenticated key exchange (PAKE) enable two users to generate a common, cryptographically-strong key based on an initial, low-entropy, shared secret (i.e., a password). The difficulty in this setting is to prevent *off-line* dictionary attacks where an adversary exhaustively enumerates potential passwords on its own, attempting to match the correct password to observed protocol executions. Roughly, a PAKE protocol is “secure” if off-line attacks are of no use and the best attack is an *on-line* dictionary attack where an adversary must actively try to impersonate an honest party using each possible password. On-line attacks of this sort are inherent in the model of password-based authentication; more importantly, they can be detected by the server as failed login attempts and (at least partially) defended against.

Due to the widespread use of passwords, a significant amount of research has focused on designing PAKE protocols. Early work [13] (see also [14]) considered a “hybrid” model where users share public keys in addition to a password. In the more challenging “password-only” setting clients and servers are required to share *only* a password. Bellare and Merritt [4] initiated research in this direction, and presented a PAKE protocol with heuristic arguments for its security. It was not until several years later that formal models for PAKE were developed [3,5,11], and provably secure PAKE protocols were shown in the random oracle/ideal cipher models [3,5,18].

* Work done while visiting IBM. Research supported by NSF grants #0627306 and #0716651, and NSF CAREER award #0447075.

Goldreich and Lindell [11] constructed the first PAKE protocol without random oracles, and their approach remains the only one for the plain model where there is no additional setup. Unfortunately, their protocol is inefficient in terms of communication, computation, and round complexity. (Nguyen and Vadhan [19] show efficiency improvements, but achieve a weaker notion of security. In any case, their protocol is similarly impractical.) The Goldreich-Lindell protocol also does not tolerate concurrent executions by the same party.

Katz, Ostrovsky, and Yung [17] demonstrated the first *efficient* PAKE protocol with a proof of security in the standard model; extensions and improvements of this protocol were given in [9,6,16,8]. In contrast to the work of Goldreich and Lindell, these protocols are secure even under concurrent executions by the same party. On the other hand, these protocols all require a *common reference string* (CRS). While this may be less appealing than the “plain model,” reliance on a CRS does not appear to be a serious drawback in the context of PAKE since the CRS can be hard-coded into the protocol implementation. A different PAKE protocol in the CRS model is given by Jiang and Gong [15].

PAKE based on lattices? Cryptographic primitives based on lattices are appealing because of known worst-case/average-case connections between lattice problems, as well as because several lattice problems are currently immune to quantum attacks. Also, the best-known algorithms for several lattice problems require exponential time (in contrast to sub-exponential algorithms for, e.g., factoring). None of the existing PAKE constructions (in either the random oracle or standard models), however, can be instantiated with lattice-based assumptions.¹ The barrier to constructing a lattice-based PAKE protocol using the KOY/GL approach [17,9] is that this approach requires a CCA-secure encryption scheme (more generally, a non-malleable commitment scheme) with an associated *smooth projective hash system* [7,9]. (See Section 2.) Until recently, the existence of CCA-secure encryption schemes based on lattices (even ignoring the additional requirement of smooth projective hashing) was open. Peikert and Waters [22] gave the first constructions of CCA-secure encryption based on lattices, but the schemes they propose are not readily amenable to the smooth projective hashing requirement. Subsequent constructions [24,20,12] do not immediately support smooth projective hashing either.

1.1 Our Results

Building on ideas of [24,20,12], we show a new construction of a CCA-secure public-key encryption scheme based on the hardness of the *learning with error* (LWE) problem [23]. We then demonstrate (a variant of) a smooth projective hash system for our scheme. This is the most technically difficult aspect of our work, and is of independent interest as the first construction of a smooth projective hash system (for a conjectured hard-on-average language) based on lattice

¹ To the best of our knowledge this includes the protocol of Goldreich and Lindell [11], which requires a one-to-one one-way function on an infinite domain (in addition to oblivious transfer, which can be based on lattice assumptions [21]).

assumptions. (Instantiating the smooth projective hash framework using lattice assumptions is stated as an open question in [21].) Finally, we show that our encryption scheme can be plugged into a modification of the Katz-Ostrovsky-Yung/Gennaro-Lindell framework [17,9] to give a PAKE protocol based on the LWE assumption.

Organization of the paper. In Section 2 we define a variant of smooth projective hashing (SPH) that we call *approximate* SPH. We then show in Section 3 that a CCA-secure encryption scheme having an approximate SPH system suffices for our desired application to PAKE.

The main technical novelty of our paper is in the sections that follow. In Section 4 we review the LWE problem and some preliminaries. As a prelude to our main construction, we show in Section 5 a CPA-secure encryption scheme based on the LWE problem, with an associated approximate SPH system. In Section 6 we describe how to extend this initial scheme to obtain CCA-security.

Throughout the paper, we denote the security parameter by n .

2 Approximate Smooth Projective Hash Functions

Smooth projective hash functions were introduced by Cramer and Shoup [7]; we follow (and adapt) the treatment of Gennaro and Lindell [9], who extend the original definition. Rather than aiming for utmost generality, we tailor the definitions to our eventual application.

Roughly speaking, the differences between our definition and that of Gennaro-Lindell are as follows. (This discussion assumes familiarity with [9]; for the reader not already familiar with that work, a self-contained description is given below.) In [9] there are sets X and $L \subset X$; *correctness* is guaranteed for $x \in L$, while *smoothness* is guaranteed for $x \in X \setminus L$. Here, we require only *approximate* correctness, and moreover only for elements in a subset $\bar{L} \subseteq L$. Details follow.

Fix a CCA-secure (labeled) public-key encryption scheme ($\text{Gen}, \text{Enc}, \text{Dec}$) and an efficiently recognizable message space \mathcal{D} (which will correspond to the dictionary of passwords in our application to PAKE). We assume the encryption scheme defines a notion of *ciphertext validity* such that (1) validity of a ciphertext (with respect to pk) can be determined efficiently using pk alone, and (2) all honestly generated ciphertexts are valid. We also assume no decryption error.

For the rest of the discussion, fix a key pair (pk, sk) as output by $\text{Gen}(1^n)$ and let \mathcal{C} denote the set of valid ciphertexts with respect to pk . Define sets X , $\{\bar{L}_m\}_{m \in \mathcal{D}}$, and \bar{L} as follows. First, set

$$X = \{(\text{label}, C, m) \mid \text{label} \in \{0, 1\}^n; C \in \mathcal{C}; m \in \mathcal{D}\}.$$

Next, for $m \in \mathcal{D}$ let $\bar{L}_m = \{(\text{label}, \text{Enc}_{pk}(\text{label}, m), m) \mid \text{label} \in \{0, 1\}^n\} \subset X$; i.e., \bar{L}_m is the set of honestly generated encryptions of m (using any label). Let $\bar{L} = \cup_{m \in \mathcal{D}} \bar{L}_m$. Finally, define

$$L_m = \{(\text{label}, C, m) \mid \text{label} \in \{0, 1\}^n; \text{Dec}_{sk}(\text{label}, C) = m\},$$

and set $L = \cup_{m \in \mathcal{D}} L_m$. (Recall we assume no decryption error, and so L_m depends only on pk .) Note that $\bar{L}_m \subseteq L_m$ for all m . Furthermore, for any ciphertext C and label $\in \{0, 1\}^n$ there is at most one $m \in \mathcal{D}$ for which $(\text{label}, C, m) \in L$.

Approximate smooth projective hash functions. An *approximate smooth projective hash function* is a collection of keyed functions $\{H_k : X \rightarrow \{0, 1\}^n\}_{k \in K}$, along with a *projection function* $\alpha : K \times (\{0, 1\}^* \times \mathcal{C}) \rightarrow S$, satisfying notions of (approximate) *correctness* and *smoothness*:

Approximate correctness: If $x = (\text{label}, C, m) \in \bar{L}$ then the value of $H_k(x)$ is approximately determined by $\alpha(k, \text{label}, C)$ and x (in a sense we will make precise below).

Smoothness: If $x \in X \setminus L$ then the value of $H_k(x)$ is statistically close to uniform given $\alpha(k, \text{label}, C)$ and x (assuming k was chosen uniformly in K).

We stress that, in contrast to [9], we require nothing for $x \in L \setminus \bar{L}$; furthermore, even for $x \in \bar{L}$ we require only approximate correctness. We highlight also that, as in [9], the projection function α should be a function of label, C only.

Formally, an $\varepsilon(n)$ -approximate smooth projective hash function is defined by a sampling algorithm that, given pk , outputs $(K, G, \mathbb{H} = \{H_k : X \rightarrow \{0, 1\}^n\}_{k \in K}, S, \alpha : K \times (\{0, 1\}^* \times \mathcal{C}) \rightarrow S)$ such that:

1. There are efficient algorithms for (1) sampling a uniform $k \in K$, (2) computing $H_k(x)$ for all $k \in K$ and $x \in X$, and (3) computing $\alpha(k, \text{label}, C)$ for all $k \in K$ and $(\text{label}, C) \in \{0, 1\}^* \times \mathcal{C}$.
2. For $x = (\text{label}, C, m) \in \bar{L}$ the value of $H_k(x)$ is approximately determined by $\alpha(k, \text{label}, C)$, relative to the Hamming metric. Specifically, let $\text{Ham}(a, b)$ denote the Hamming distance of two strings $a, b \in \{0, 1\}^n$. Then there is an efficient algorithm H' that takes as input $s = \alpha(k, \text{label}, C)$ and $\bar{x} = (\text{label}, C, m, r)$ for which $C = \text{Enc}_{pk}(\text{label}, m; r)$ and satisfies:

$$\Pr[\text{Ham}(H_k(x), H'(s, \bar{x})) \geq \varepsilon \cdot n] = \text{negl}(n),$$

where the probability is taken over choice of k .

3. For any $x = (\text{label}, C, m) \in X \setminus L$, the following two distributions have statistical distance negligible in n :

$$\{k \leftarrow K; s = \alpha(k, \text{label}, C) : (s, H_k(x))\}$$

and

$$\{k \leftarrow K; s = \alpha(k, \text{label}, C); v \leftarrow \{0, 1\}^n : (s, v)\}.$$

3 A PAKE Protocol from Approximate SPH

We use the standard definition of security for PAKE [3][17][9].

Here, we show that a modification of the Gennaro-Lindell framework [9] can be used to construct a PAKE protocol from any CCA-secure encryption scheme that has associated with it an approximate smooth projective hash function as

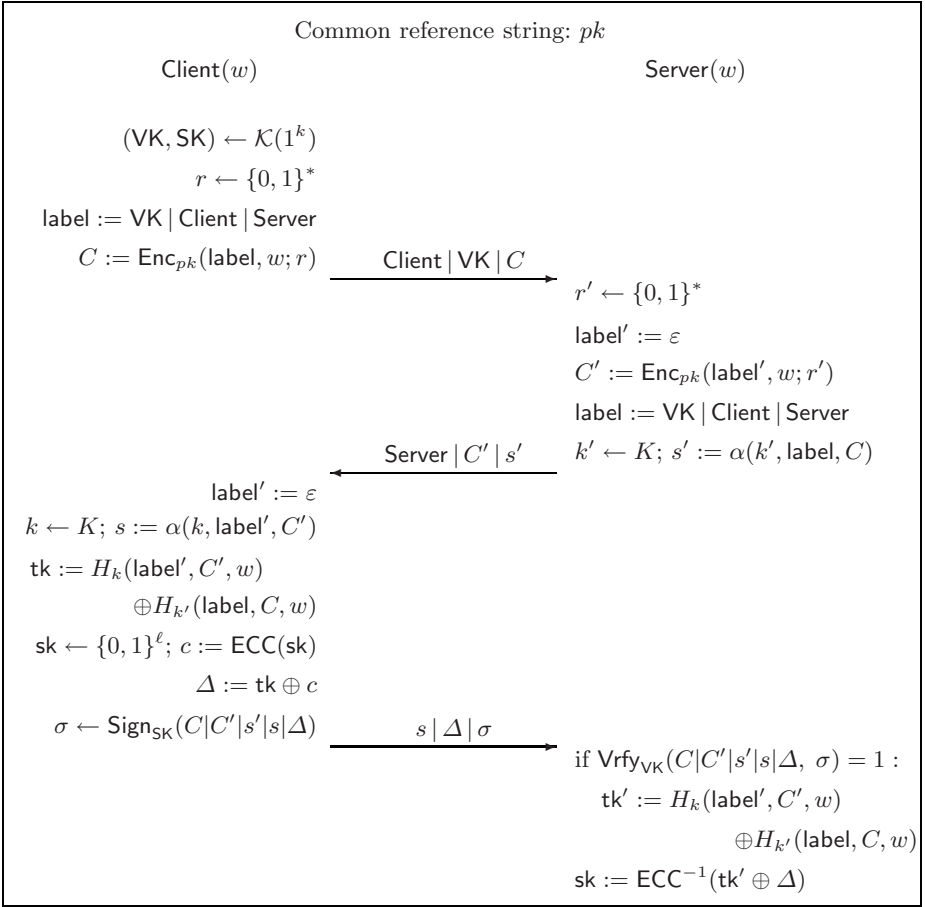


Fig. 1. A 3-round PAKE protocol. The common session key is sk.

defined in Section 2. A high-level overview of the protocol is given in Figure 1; a more detailed discussion follows.

Setup. We assume a common reference string is established before any executions of the protocol take place. The common reference string consists of a public key pk for a CCA-secure encryption scheme (Gen, Enc, Dec) that has an associated ϵ -approximate smooth projective hash system $(K, G, \mathbb{H} = \{H_k : X \rightarrow \{0, 1\}^n\}_{k \in K}, S, \alpha : K \times (\{0, 1\}^* \times \mathcal{C}) \rightarrow S)$. We stress that no parties in the system need to hold the secret key corresponding to pk .

Protocol execution. We now describe an execution of the protocol between an honest client Client and server Server, holding common password w . To begin, the client runs a key-generation algorithm \mathcal{K} for a one-time signature scheme to generate verification key VK and corresponding secret (signing) key SK. The

client sets $\text{label} := \text{VK}|\text{Client}|\text{Server}$ and then encrypts the password w using this label to obtain ciphertext C . It then sends the message $\text{Client}|\text{VK}|C$ to the server.

Upon receiving the initial message $\text{Client}|\text{VK}|C$ from the client, the server computes its own encryption of the password using label $\text{label}' = \varepsilon$, resulting in a ciphertext C' . The server also chooses a random hash key $k' \leftarrow K$ and then computes the projection $s' := \alpha(k', \text{label}, C)$. It sends C' and s' to the client.

After receiving the second protocol message from the server, the client chooses a random hash key $k \leftarrow K$ and computes the projection $s := \alpha(k, \text{label}', C')$. At this point it computes a *temporary* session key $\text{tk} = H_k(\text{label}', C', w) \oplus H_{k'}(\text{label}, C, w)$, where $H_k(\text{label}', C', w)$ is computed using the known hash key k , and $H_{k'}(\text{label}, C, w)$ is computed using the randomness r that was used to generate C . (Recall that C is an honestly generated encryption of w .) Up to this point, the protocol follows the Gennaro-Lindell framework exactly. As will become clear, however, the server will not be able to recover tk but will instead only recover some value tk' that is *close* to tk ; the rest of the client's computation is aimed at repairing this defect.

The client chooses a random session key $\text{sk} \in \{0, 1\}^\ell$ for some ℓ to be specified. Let $\text{ECC} : \{0, 1\}^\ell \rightarrow \{0, 1\}^n$ be an error-correcting code correcting a 2ε -fraction of errors. The client computes $c := \text{ECC}(\text{sk})$ and sets $\Delta := \text{tk} \oplus c$. Finally, it signs $C|C'|s'|s|\Delta$ and sends s, Δ , and the resulting signature σ to the server.

The server verifies σ in the obvious way and rejects if the signature is invalid. Otherwise, the server computes a temporary session key tk' analogously to the way the client did: that is, the server sets $\text{tk}' = H_k(\text{label}', C', w) \oplus H_{k'}(\text{label}, C, w)$, where $H_{k'}(\text{label}, C, w)$ is computed using the hash key k' known to the server, and $H_k(\text{label}', C', w)$ is computed using the randomness r' that was used to generate C' . (Recall that C' is an honestly generated encryption of w .) Finally, the server computes $\text{sk} := \text{ECC}^{-1}(\text{tk}' \oplus \Delta)$.

Correctness. We now argue that, in an honest execution of the protocol, the client and server compute matching session keys with all but negligible probability. Approximate correctness of the smooth projective hash function implies that $H_k(\text{label}, C, w)$ as computed by the client is within Hamming distance εn from $H_k(\text{label}, C, w)$ as computed by the server, except with negligible probability. The same holds for $H_{k'}(\text{label}', C', w)$. Thus, with all but negligible probability we have $\text{Ham}(\text{tk}, \text{tk}') \leq 2\varepsilon \cdot n$. Assuming this is the case we have

$$\text{Ham}(\text{tk}' \oplus \Delta, c) = \text{Ham}(\text{tk}' \oplus \Delta, \text{tk} \oplus \Delta) \leq 2\varepsilon \cdot n,$$

and so $\text{ECC}^{-1}(\text{tk}' \oplus \Delta) = \text{ECC}^{-1}(c) = \text{sk}$.

Security. The proof of security of the protocol follows [1749] closely; we sketch the main ideas. First, as in [1749], we note that for a passive adversary (i.e., one that simply observes interactions between the server and the client), the shared session-key is pseudorandom. This is simply because the transcript of each interaction consists of semantically-secure encryptions of the password w and the projected keys of the approximate SPH system.

It remains to deal with active (man-in-the-middle) adversaries that modify the messages sent from the client to the server and back. The crux of our proof,

as in [17,9], is a combination of the following two observations (for concreteness, consider an adversary that interacts with a client instance holding password w).

- By the CCA-security of the encryption scheme, the probability that the adversary can construct a new ciphertext that decrypts to the client’s password w is at most $q/|\mathcal{D}| + \text{negl}(n)$, where q is the number of on-line attacks and \mathcal{D} is the password dictionary.
- If the adversary sends the client a ciphertext that does not decrypt to the client’s password, then the session-key computed by the client is statistically close to uniform conditioned on the adversary’s view.

We defer a complete proof to the full version.

Recalling the definitions from Section 2, note that correctness of the protocol relies on (approximate) correctness for honestly generated encryptions of the correct password (i.e., for $x \in \bar{L}$), whereas security requires smoothness for ciphertexts that do not decrypt to the correct password (i.e., for $x \notin L$).

4 The Learning with Errors Problem

The “learning with errors” (LWE) problem was introduced by Regev [23] as a generalization of the “learning parity with noise” problem. For positive integers n and $q \geq 2$, a vector $\mathbf{s} \in \mathbb{Z}_q^n$, and a probability distribution χ on \mathbb{Z}_q , let $A_{\mathbf{s},\chi}$ be the distribution obtained by choosing a vector $\mathbf{a} \in \mathbb{Z}_q^n$ uniformly at random and a noise term $x \leftarrow \chi$, and outputting $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + x) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$.

For an integer $q = q(n)$ and an error distribution $\chi = \chi(n)$ over \mathbb{Z}_q , the learning with errors problem $\text{LWE}_{q,\chi}$ is defined as follows: Given access to an oracle that outputs (polynomially many) samples from $A_{\mathbf{s},\chi}$ for a uniformly random $\mathbf{s} \in \mathbb{Z}_q^n$, output \mathbf{s} with noticeable probability. The decisional variant of the LWE problem, denoted $\text{distLWE}_{q,\chi}$, is to distinguish samples chosen according to $A_{\mathbf{s},\chi}$ for a uniformly random $\mathbf{s} \in \mathbb{Z}_q^n$ from samples chosen according to the uniform distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$. Regev [23] shows that for $q = \text{poly}(n)$ prime, the LWE and distLWE problems are polynomially equivalent.

Gaussian error distributions. For any $r > 0$, the density function of a one-dimensional Gaussian distribution over \mathbb{R} is given by $D_r(x) = 1/r \cdot \exp(-\pi(x/r)^2)$. In this work we always use a *truncated* Gaussian distribution, i.e., the Gaussian distribution D_r whose support is restricted to x such that $|x| < r\sqrt{n}$. The truncated and non-truncated distributions are statistically close, and we drop the word “truncated” from now on. For $\beta > 0$, define $\bar{\Psi}_\beta$ to be the distribution on \mathbb{Z}_q obtained by drawing $y \leftarrow D_\beta$ and outputting $\lfloor q \cdot y \rfloor \pmod q$. We write $\text{LWE}_{q,\beta}$ as an abbreviation for $\text{LWE}_{q,\bar{\Psi}_\beta}$.

We also define the *discrete Gaussian distribution* $D_{\mathbb{Z}^m,r}$ over the integer lattice \mathbb{Z}^m , which assigns probability proportional to $\prod_{i \in [m]} D_r(e_i)$ to each $\mathbf{e} \in \mathbb{Z}^m$. It is possible to efficiently sample from $D_{\mathbb{Z}^m,r}$ for any $r > 0$ [10].

Evidence for the hardness of $\text{LWE}_{q,\beta}$ follows from results of Regev [23], who gave a *quantum* reduction from approximating certain problems on n -dimensional

lattices in the worst case to within $\tilde{O}(n/\beta)$ factors to solving $\text{LWE}_{q,\beta}$ for dimension n , subject to the condition that $\beta \cdot q \geq 2\sqrt{n}$. Recently, Peikert [20] also gave a related *classical* reduction for similar parameters. For our purposes, we note that the $\text{LWE}_{q,\beta}$ problem is believed to be hard (given the state-of-the-art in lattice algorithms) for any polynomial q and inverse-polynomial β (subject to the above condition).

Matrix notation for LWE. In this paper, we view all our vectors as column vectors. At times, we find it convenient to describe the LWE problem $\text{LWE}_{q,\beta}$ using a compact matrix notation: find \mathbf{s} given $(\mathbf{A}, \mathbf{A}\mathbf{s} + \mathbf{x})$, where $\mathbf{A} \leftarrow \mathbb{Z}_q^{m \times n}$ is chosen uniformly and $\mathbf{x} \leftarrow \overline{\Psi}_\beta^m$. We also use similar notation for the decision version distLWE .

Connection to lattices. The LWE problem can be thought of as a “bounded-distance decoding problem” on a particular kind of m -dimensional lattice defined by the matrix \mathbf{A} . Specifically, define the lattice

$$\Lambda(\mathbf{A}) = \{\mathbf{y} \in \mathbb{Z}^m : \exists \mathbf{s} \in \mathbb{Z}^n \text{ s.t. } \mathbf{y} \equiv \mathbf{A}^T \mathbf{s} \pmod{q}\}.$$

The LWE problem can then be restated as: given \mathbf{y} which is the sum of a lattice point $\mathbf{A}\mathbf{s}$ and a short “noise vector” \mathbf{x} , find the “closest” lattice vector \mathbf{s} . One can show that as long as \mathbf{x} is short (say, $\|\mathbf{x}\| < q/16$), there is a unique closest vector to \mathbf{y} (see, e.g., [10]).

4.1 Some Supporting Lemmas

We present two technical lemmas regarding the LWE problem that will be used to prove smoothness of our (approximate) SPH systems in Sections 5.2 and 6.2.

If $m \geq n \log q$, the lattice $\Lambda(\mathbf{A})$ is quite sparse. In fact, we expect most vectors $\mathbf{z} \in \mathbb{Z}_q^m$ to be far from $\Lambda(\mathbf{A})$. The first lemma (originally shown in [23]) formalizes this intuition.

Let $\text{dist}(\mathbf{z}, \Lambda(\mathbf{A}))$ denote the distance of the vector \mathbf{z} from the lattice $\Lambda(\mathbf{A})$. The lemma shows that for most matrices $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$, the fraction of vectors $\mathbf{z} \in \mathbb{Z}_q^m$ that are “very close” to $\Lambda(\mathbf{B})$ is “very small”. The proof is by probabilistic method, and appears in the full version.

Lemma 1. *Let n, q, m be integers such that $m \geq n \log q$. For all but a negligible fraction of matrices \mathbf{A} ,*

$$\Pr_{\mathbf{z} \leftarrow \mathbb{Z}_q^m} [\text{dist}(\mathbf{z}, \Lambda(\mathbf{A})) \leq \sqrt{q}/4] \leq q^{-(m+n)/2}.$$

Fix a number $r > 0$, and let $\mathbf{e} \leftarrow D_{\mathbb{Z}^m, r}$ be drawn from the discrete Gaussian distribution over the integer lattice \mathbb{Z}^m . If the vector \mathbf{z} is (close to) a linear combination of the columns of \mathbf{A} , then given $\mathbf{e}^T \mathbf{A}$ one can (approximately) predict $\mathbf{e}^T \mathbf{z}$. The second lemma shows a converse of this statement when r is large enough. Namely, it says that if \mathbf{z} and all its non-zero multiples are far from the lattice $\Lambda(\mathbf{A})$, then $\mathbf{e}^T \mathbf{A}$ does not give any information about $\mathbf{e}^T \mathbf{z}$. In other words, given $\mathbf{e}^T \mathbf{A}$ (where $\mathbf{e} \leftarrow D_{\mathbb{Z}^m, r}$ for a large enough r) $\mathbf{e}^T \mathbf{z}$ is

statistically close to random. This lemma was first shown in [10], and was used in the construction of an oblivious transfer protocol in [21].

More formally, for a matrix $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ and a vector $\mathbf{z} \in \mathbb{Z}_q^m$, let $\Delta_r(\mathbf{A}, \mathbf{z})$ denote the statistical distance between the uniform distribution on \mathbb{Z}_q^{n+1} and the distribution of $(\mathbf{e}^T \mathbf{A}, \mathbf{e}^T \mathbf{z})$, where $\mathbf{e} \leftarrow D_{\mathbb{Z}^m, r}$. Then,

Lemma 2. [10, Lemma 6.3] *Let $r \geq \sqrt{q} \cdot \omega(\sqrt{\log n})$. Then for most matrices $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$, the following is true: if $\mathbf{z} \in \mathbb{Z}_q^m$ is such that for all non-zero $a \in \mathbb{Z}_q$, $\text{dist}(az, \Lambda(\mathbf{A})) \geq \sqrt{q}/4$, then $\Delta_r(\mathbf{A}, \mathbf{z}) \leq \text{negl}(n)$.*

5 Approximate Smooth Projective Hashing from Lattices

As a warmup to our main result we first construct a CPA-secure encryption scheme with an approximate SPH system. The main ideas in our final construction are already present here.

5.1 A CPA-Secure Encryption Scheme

The encryption scheme we use is a variant of the scheme presented in [10,20], and is based on the hardness of the LWE problem. We stress that the novelty of this work is in constructing an approximate SPH system for this scheme.

We begin by describing a basic encryption scheme having decryption time exponential in the message length.² We then modify the scheme so that decryption can be done in polynomial time.

The message space is \mathbb{Z}_q^ℓ for some integers q, ℓ . In the basic encryption scheme, the public key consists of a matrix $\mathbf{B} \in \mathbb{Z}_q^{m \times n}$, along with $\ell + 1$ vectors $\mathbf{u}_0, \dots, \mathbf{u}_\ell \in \mathbb{Z}_q^m$. To encrypt a message $\mathbf{w} = (w_1, \dots, w_\ell) \in \mathbb{Z}_q^\ell$ the sender chooses a uniformly random vector $\mathbf{s} \leftarrow \mathbb{Z}_q^n$ and an error vector $\mathbf{x} \leftarrow \overline{\Psi}_\beta^m$. The ciphertext is

$$\mathbf{y} = \mathbf{B}\mathbf{s} + \left(\mathbf{u}_0 + \sum_{i=1}^{\ell} w_i \cdot \mathbf{u}_i\right) + \mathbf{x} \in \mathbb{Z}_q^m$$

The scheme is CPA-secure, since the $\text{distLWE}_{q,\beta}$ assumption implies that the ciphertext is pseudorandom.

The ciphertext produced by the encryption algorithm is a vector \mathbf{y} such that $\mathbf{y} - (\mathbf{u}_0 + \sum_{i=1}^{\ell} w_i \cdot \mathbf{u}_i)$ is “close” to the lattice $\Lambda(\mathbf{B})$ (the exact definition of “close” depends on the error parameter β). Decrypting a ciphertext is done by finding (via exhaustive search over the message space) a message \mathbf{w} for which $\mathbf{y} - (\mathbf{u}_0 + \sum_{i=1}^{\ell} w_i \cdot \mathbf{u}_i)$ is “close” to $\Lambda(\mathbf{B})$, using the following trapdoor structure first discovered by Ajtai [1], and later improved by Alwen and Peikert [2].

² Interestingly, for our eventual application to PAKE a CCA-secure version of this scheme would suffice since the scheme has the property that it is possible to efficiently tell whether a given ciphertext is an encryption of a given message (and this is all that is needed to prove security for the protocol in Section 3).

Lemma 3 ([12]). Fix integers $q \geq 2$ and $m \geq 4n \log^2 q$. There is a PPT algorithm $\text{TrapSamp}(1^n, q, m)$ that outputs matrices $\mathbf{B} \in \mathbb{Z}_q^{m \times n}$ and $\mathbf{T} \in \mathbb{Z}^{m \times m}$ such that the distribution of \mathbf{B} is statistically close to the uniform distribution over $\mathbb{Z}_q^{m \times n}$, and there is an algorithm $\text{BDDsolve}(\mathbf{T}, \cdot)$ that takes as input a vector $\mathbf{z} \in \mathbb{Z}^m$ and does the following:

- if there is a vector $\mathbf{s} \in \mathbb{Z}^m$ such that $\text{dist}(\mathbf{z}, \mathbf{B}\mathbf{s} \pmod{q}) \leq \sqrt{q}/4$, then the output is \mathbf{s} .
- if for every vector $\mathbf{s} \in \mathbb{Z}^m$, $\text{dist}(\mathbf{z}, \mathbf{B}\mathbf{s}) > \sqrt{q}/4$, then the output is \perp .

Proof. \mathbf{T} is a full-rank matrix such that (a) each row of \mathbf{t}_i has bounded ℓ_2 norm, i.e., $\|\mathbf{t}_i\| \leq 4\sqrt{m}$, and (b) $\mathbf{T}\mathbf{B} = 0 \pmod{q}$. [12] showed how to sample a pair (\mathbf{B}, \mathbf{T}) such that \mathbf{B} is statistically close to uniform and \mathbf{T} has the above properties.

Given such a matrix \mathbf{T} and a vector $\mathbf{z} \in \mathbb{Z}^m$, $\text{BDDsolve}(\mathbf{T}, \mathbf{z})$ works as follows:

- first, compute $\mathbf{z}' = q \cdot \mathbf{T}^{-1} \cdot \lfloor (\mathbf{T} \cdot \mathbf{z}) / q \rfloor \pmod{q}$.
- Compute (using Gaussian elimination) a vector $\mathbf{s} \in \mathbb{Z}_q^n$ such that $\mathbf{z}' = \mathbf{B}\mathbf{s}$ (if such exists; else, output \perp).
- If $\text{dist}(\mathbf{z}, \mathbf{B}\mathbf{s}) \leq \sqrt{q}/4$, then output \mathbf{s} else output \perp .

First, if $\mathbf{z} = \mathbf{B}\mathbf{s} + \mathbf{x}$ for some $\mathbf{s} \in \mathbb{Z}_q^n$ and $\mathbf{x} \in \mathbb{Z}_q^m$ such that $\|\mathbf{x}\| \leq \sqrt{q}/4$, then the procedure above computes

$$\mathbf{z}' = q \cdot \mathbf{T}^{-1} \cdot \lfloor (\mathbf{T} \cdot (\mathbf{B}\mathbf{s} + \mathbf{x})) / q \rfloor \pmod{q} = \mathbf{B}\mathbf{s} \pmod{q}$$

This is because each co-ordinate of $\mathbf{T}\mathbf{x}$ has magnitude at most $\|\mathbf{T}\| \cdot \|\mathbf{x}\| \leq 4\sqrt{m} \cdot \sqrt{q}/4 \ll q$, and consequently,

$$\lfloor (\mathbf{T} \cdot (\mathbf{B}\mathbf{s} + \mathbf{x})) / q \rfloor = \lfloor (\mathbf{T} \cdot \mathbf{B}\mathbf{s}) / q \rfloor = \mathbf{T} \cdot (\mathbf{B}\mathbf{s}) / q$$

where the final equality is because $\mathbf{T}\mathbf{B} = 0 \pmod{q}$.

Finally, if $\text{dist}(\mathbf{z}, \mathbf{A}(\mathbf{B})) > \sqrt{q}/4$, then the last line of the procedure above causes the output to be \perp always. □

We now modify the decryption algorithm in two ways. The first of these modifications ensures that the decryption algorithm runs in polynomial time, and the second is needed for our approximate SPH system.

First, to avoid the exponential dependence of the decryption time on the message length, we modify the encryption scheme by letting the public key contain the matrix $\mathbf{A} = [\mathbf{B}|\mathbf{U}]$, where the columns of $\mathbf{U} \in \mathbb{Z}_q^{m \times (\ell+1)}$ are the vectors $\mathbf{u}_0, \dots, \mathbf{u}_\ell$. The secret-key is a trapdoor for the entire matrix \mathbf{A} (as opposed to just \mathbf{B} as in the previous description). The ciphertext from the previous description can then be written as

$$\mathbf{y} = \mathbf{A}^T \begin{pmatrix} \mathbf{s} \\ 1 \\ \mathbf{w} \end{pmatrix} + \mathbf{x} \in \mathbb{Z}_q^m$$

and decryption uses the BDDsolve procedure from Lemma 3 to recover the vector $(\mathbf{s}, 1, \mathbf{m})$. The crucial point is that, during key generation, the receiver can generate the matrix \mathbf{A} along with an appropriate trapdoor for decryption.

Secondly, we relax the decryption algorithm so that it finds an $a \in \mathbb{Z}_q$ and a message \mathbf{w} for which $a(\mathbf{y} - (\mathbf{u}_0 + \sum_{i=1}^{\ell} w_i \cdot \mathbf{u}_i))$ is “close” to $\Lambda(\mathbf{B})$. This modified decryption algorithm correctly decrypts the ciphertexts generated by Enc (which corresponds to the case $a = 1$), but it also decrypts ciphertexts that would never be output by Enc. This modification to the decryption algorithm enables us to prove smoothness for the approximate SPH system.

Parameters. Let n be the security parameter, and $\ell = n$ be the message length. The parameters of the system are a prime $q = q(n, \ell)$, a positive integer $m = m(n, \ell)$, and a Gaussian error parameter $\beta = \beta(n, \ell) \in (0, 1]$ that defines a distribution $\bar{\Psi}_\beta$. For concrete instantiations of these parameters, see Theorem 1.

We now describe the scheme:

Key generation. Choose a matrix $\mathbf{A} \in \mathbb{Z}_q^{m \times (n+\ell+1)}$ together with the trapdoor \mathbf{T} by running $(\mathbf{A}, \mathbf{T}) \leftarrow \text{TrapSamp}(1^m, 1^{n+\ell+1}, q)$, where TrapSamp is as described in Lemma 3. Let the public key be \mathbf{A} and the secret-key is \mathbf{T} .

Encryption. To encrypt the message $\mathbf{w} \in \mathbb{Z}_q^\ell$ with respect to a public key as above, the sender chooses $\mathbf{s} \leftarrow \mathbb{Z}_q^n$ uniformly at random, and an error vector $\mathbf{x} \leftarrow \bar{\Psi}_\beta^{mn}$. The ciphertext is

$$\mathbf{y} = \mathbf{A} \cdot \begin{pmatrix} \mathbf{s} \\ 1 \\ \mathbf{w} \end{pmatrix} + \mathbf{x} \pmod{q}$$

Decryption. The decryption algorithm works as below.

```

for  $a = 1$  to  $q - 1$  do
    Compute  $\begin{pmatrix} \mathbf{s} \\ a' \\ \mathbf{w} \end{pmatrix} \leftarrow \text{BDDsolve}(\mathbf{T}, a\mathbf{y})$ 
    if  $a' = a$  then
        output  $\mathbf{w}/a$  and stop
    else try the next value of  $a$ 
end
If the above fails for all  $a$ , output  $\perp$ 
    
```

Theorem 1. *Let n, ℓ, m, q, β be chosen such that $m \geq 4(n + \ell) \log q$ and $\beta < 1/(2 \cdot m^2 n \cdot \omega(\sqrt{\log n}))$. Then the scheme above is a CCA-secure encryption scheme assuming the hardness of $\text{distLWE}_{n,m,q,\beta}$.*

5.2 An Approximate SPH System

Fix a public key $\mathbf{A} \in \mathbb{Z}_q^{m \times (n+\ell+1)}$ for the system (where we write $\mathbf{A} = [\mathbf{B}|\mathbf{U}]$, as usual), and a dictionary $\mathcal{D} \stackrel{\text{def}}{=} \mathbb{Z}_q^\ell$. Sets X, \bar{L}_m and L_m are defined in Section 2.

(For our purposes, all vectors $\mathbf{y} \in \mathbb{Z}_q^m$ are valid ciphertexts). Let r be such that

$$\sqrt{q} \cdot \omega(\sqrt{\log n}) \leq r \leq \varepsilon / (8 \cdot mn^2 \cdot \beta).$$

(Looking ahead, we remark that the upper bound on r will be used for correctness, and the lower bound will be used for smoothness.)

A key for the SPH system is a k -tuple of vectors $(\mathbf{e}_1, \dots, \mathbf{e}_k)$ where each $\mathbf{e}_i \leftarrow D_{\mathbb{Z}^m, r}$ is drawn independently from the discrete Gaussian distribution. The reader may want to keep in mind the inverse relationship between the parameters r and β : the larger the error parameter β in the encryption scheme, the smaller the discrete-Gaussian radius r (and vice versa).

1. The projection set $S \stackrel{\text{def}}{=} (\mathbb{Z}_q^n)^k$. For a key $(\mathbf{e}_1, \dots, \mathbf{e}_k) \in (\mathbb{Z}_q^m)^k$, the projection is $\alpha(\mathbf{e}_1, \dots, \mathbf{e}_k) = (\mathbf{u}_1, \dots, \mathbf{u}_k)$, where $\mathbf{u}_i = \mathbf{B}^T \mathbf{e}_i$.
2. We now define the smooth projective hash function $\mathcal{H} = \{H_k\}_{k \in K}$. On input a key $(\mathbf{e}_1, \dots, \mathbf{e}_k) \in K$ and a ciphertext $\mathbf{c} = (\text{label}, \mathbf{y}, \mathbf{m})$, the hash function is computed as follows. First compute

$$z_i = \mathbf{e}_i^T \left[\mathbf{y} - \mathbf{U} \cdot \begin{pmatrix} 1 \\ \mathbf{m} \end{pmatrix} \right] \in \mathbb{Z}_q.$$

Treat z_i as a number in $[-(q-1)/2 \dots (q-1)/2]$ and output $b_1 \dots b_k \in \{0, 1\}^k$ where

$$b_i = \begin{cases} 0 & \text{if } z_i < 0 \\ 1 & \text{if } z_i > 0 \end{cases}.$$

3. On input a projected key $(\mathbf{u}_1, \dots, \mathbf{u}_k) \in S$, a ciphertext $\mathbf{c} = (\text{label}, \mathbf{y}, \mathbf{m})$ and a witness $\mathbf{s} \in \mathbb{Z}_q^n$ for the ciphertext, the hash function is computed as $H_{\mathbf{u}}(\mathbf{c}, \mathbf{s}) = b_1 \dots b_k$ where

$$b_i = \begin{cases} 0 & \text{if } \mathbf{u}_i^T \mathbf{s} < 0 \\ 1 & \text{if } \mathbf{u}_i^T \mathbf{s} > 0 \end{cases}.$$

Theorem 2. *Let the parameters n, ℓ, m, q, β be as in Theorem 1, and r be as above. Then, $\mathcal{H} = \{H_k\}_{k \in K}$ is an ε -approximate smooth projective hash system.*

Proof. Clearly, the following procedures can all be done in polynomial time: (1) sampling a uniform key for the hash function $(\mathbf{e}_1, \dots, \mathbf{e}_k) \leftarrow (D_{\mathbb{Z}^m, r})^k$, (2) computing the hash function H on input the key $(\mathbf{e}_1, \dots, \mathbf{e}_k)$ and a ciphertext \mathbf{c} , (3) computing the projection-key $\alpha(\mathbf{e}_1, \dots, \mathbf{e}_k)$, and (4) computing the hash function given the projected key $(\mathbf{u}_1, \dots, \mathbf{u}_k)$, a ciphertext \mathbf{c} , and a witness \mathbf{s} for the ciphertext \mathbf{c} .

Approximate correctness. We now show ε -approximate correctness. Consider any $(\text{label}, \mathbf{y}, \mathbf{m}) \in \bar{L}$, i.e., where \mathbf{y} is a ciphertext produced by the encryption algorithm on input the message \mathbf{m} . This means that \mathbf{y} can be written as

$$\mathbf{y} = \mathbf{B} \cdot \mathbf{s} + \mathbf{U} \cdot \begin{pmatrix} 1 \\ \mathbf{m} \end{pmatrix} + \mathbf{x} \pmod{q} \tag{1}$$

where $\|\mathbf{x}\| \leq \beta q \cdot \sqrt{mn}$ (recall we work with truncated Gaussians).

We first show that for each $i \in [k]$, the values z_i (computed using the key) and $\mathbf{s}^T \mathbf{u}_i$ (computed using the projected key) are “close”. More precisely, we show that $|z_i - \mathbf{u}_i^T \mathbf{s}| \leq \varepsilon/2 \cdot (q/4)$. This follows because

$$|z_i - \mathbf{u}_i^T \mathbf{s}| = |(\mathbf{e}_i^T (\mathbf{B}\mathbf{s} + \mathbf{x}) - \mathbf{u}_i^T \mathbf{s})| = |\mathbf{e}_i^T \mathbf{x}|, \tag{2}$$

where the first equality uses the fact that \mathbf{y} can be written as in Equation (1), and the second uses the fact that $\mathbf{u}_i = \mathbf{e}_i^T \mathbf{B}$. Now, $|\mathbf{e}_i^T \mathbf{x}| \leq \|\mathbf{e}_i\| \cdot \|\mathbf{x}\| \leq (r\sqrt{mn}) \cdot (\beta q\sqrt{mn}) < \varepsilon/2 \cdot q/4$.

Each \mathbf{u}_i is statistically close to uniform, by an application of the leftover hash lemma; in particular, this means that $\mathbf{s}^T \mathbf{u}_i \in \mathbb{Z}_q$ is uniformly random.³ Let b_i be the i^{th} bit of $H_{(\mathbf{e}_1, \dots, \mathbf{e}_k)}(\mathbf{c})$ and b'_i be the i^{th} bit of $H'_{(\mathbf{u}_1, \dots, \mathbf{u}_k)}(\mathbf{c}, \mathbf{s})$. Using Equation (2), we see that the probability that $b_i \neq b'_i$ (over the randomness of \mathbf{e}_i) is at most $\varepsilon/2$. Thus, by a Chernoff bound, the Hamming distance between $H_{(\mathbf{e}_1, \dots, \mathbf{e}_k)}(\mathbf{c})$ and $H'_{(\mathbf{u}_1, \dots, \mathbf{u}_k)}(\mathbf{c}, \mathbf{s})$ is at most εk with overwhelming probability. This shows approximate correctness.

Smoothness. Consider any $(\text{label}, \mathbf{y}, \mathbf{m}) \in X \setminus L$. By definition of L , this means that the decryption algorithm, on input $(\text{label}, \mathbf{y}, \mathbf{m})$ and *any* possible secret key sk , does not output \mathbf{m} . In other words, the decryption algorithm outputs either \perp , or a message $\mathbf{m}' \neq \mathbf{m}$. Define

$$\mathbf{z} := \mathbf{y} - \mathbf{U} \cdot \begin{pmatrix} 1 \\ \mathbf{m} \end{pmatrix} \text{ and } \mathbf{z}' := \mathbf{y} - \mathbf{U} \cdot \begin{pmatrix} 1 \\ \mathbf{m}' \end{pmatrix}.$$

We will show that for every non-zero $a \in \mathbb{Z}_q$, $a\mathbf{z}$ is far from the $\Lambda(\mathbf{B})$. More precisely, we will show that for every non-zero $a \in \mathbb{Z}_q$,

$$\text{dist}(a\mathbf{z}, \Lambda(\mathbf{B})) \geq \sqrt{q}/4.$$

An application of Lemma 2 then shows that for every $i \in [k]$, the pair $(\mathbf{e}_i^T \mathbf{B}, \mathbf{e}_i^T \mathbf{z})$ is statistically close to the uniform distribution over \mathbb{Z}_q^{n+1} .

Let us analyze the two cases:

- The output of the decryption algorithm is \perp . In particular, this means that for every $a \in [1 \dots q - 1]$, the vector $a\mathbf{z}$ is far from $\Lambda(\mathbf{B})$.
- The output of the decryption algorithm is a message $\mathbf{m}' \neq \mathbf{m}$. This could happen only if there is an $a' \in \mathbb{Z}_q$ such that $a'\mathbf{z}'$ is close to the lattice $\Lambda(\mathbf{B})$. Suppose, for contradiction, that $a\mathbf{z}$ is close to $\Lambda(\mathbf{B})$ as well. The claim below shows that this cannot happen with high probability over the random choice of \mathbf{U} . Thus, with high probability, $a\mathbf{z}$ is far from $\Lambda(\mathbf{B})$.

Claim. The following event happens with negligible probability over the uniformly random choice of $\mathbf{U} \in \mathbb{Z}_q^{m \times \ell}$: there exist numbers $a, a' \in \mathbb{Z}_q$, vectors $\mathbf{m} \neq \mathbf{m}' \in \mathbb{Z}_q^\ell$ and a vector $\mathbf{y} \in \mathbb{Z}_q^m$ s.t.

$$\text{dist}(a\mathbf{z}, \Lambda(\mathbf{B})) \leq \sqrt{q}/4 \text{ and } \text{dist}(a'\mathbf{z}', \Lambda(\mathbf{B})) \leq \sqrt{q}/4.$$

³ This holds only for $\mathbf{s} \neq \mathbf{0}$. We omit consideration of this technical issue for the purposes of this paper.

Proof. Fix some $a, a' \in \mathbb{Z}_q$, $\mathbf{m} \neq \mathbf{m}' \in \mathbb{Z}_q^\ell$ and $\mathbf{y} \in \mathbb{Z}_q^m$. We first observe that since the vectors $\begin{pmatrix} 1 \\ \mathbf{m} \end{pmatrix}$ and $\begin{pmatrix} 1 \\ \mathbf{m}' \end{pmatrix}$ are linearly independent and \mathbf{U} is uniformly random, the vectors $a\mathbf{z}$ and $a'\mathbf{z}'$ are uniformly random and (statistically) independent. Applying Lemma 1, we get that

$$\begin{aligned} \Pr_{\mathbf{U} \in \mathbb{Z}_q^{m \times \ell}} [\text{dist}(a\mathbf{z}, \Lambda(\mathbf{B})) \sqrt{q}/4 \text{ and } \text{dist}(a'\mathbf{z}', \Lambda(\mathbf{B})) \leq \sqrt{q}/4] \\ \leq (q^{-m/2} \cdot \text{negl}(n))^2 = q^{-m} \cdot \text{negl}(n). \end{aligned}$$

Now, an application of union bound shows that the required probability is at most $q^2 \cdot q^{2\ell} \cdot q^m \cdot (q^{-m} \cdot \text{negl}(n))$, which is negligible in n . \square

This completes the proof of Theorem 2. \square

6 A CCA-Secure Encryption Scheme Based on Lattices

In this section we describe a CCA-secure encryption scheme, along with an approximate SPH system, based on the hardness of the LWE problem. The CCA-secure encryption scheme builds on the CPA-secure encryption scheme described in Section 5.1, and the SPH system is the same as the one from Section 5.2 with a few modifications.

6.1 A CCA-Secure Encryption Scheme

The encryption scheme is similar to the schemes in [20,12] (which, themselves, are instantiations of the general construction of Rosen and Segev [24]). The main difference between [20,12] and our scheme is the relaxed notion of decryption, which we already use in the CPA-secure construction in Section 5.1. A formal description of the scheme follows.

Parameters. Let n be the security parameter, and $\ell = \text{poly}(n)$ be the message length. The parameters of the system are a prime $q = q(n, \ell)$, an integer $m = m(n, \ell) \in \mathbb{Z}^+$, and a Gaussian error parameter $\beta = \beta(n, \ell) \in (0, 1]$ that defines a distribution $\bar{\Psi}_\beta$. For concrete instantiations of these parameters, see Theorem 3.

Key generation. For $i \in [n]$ and $b \in \{0, 1\}$, choose $2n$ matrices $\mathbf{A}_{i,b} \leftarrow \mathbb{Z}_q^{m \times (n+\ell+1)}$ together with short bases $\mathbf{S}_{i,b} \in \mathbb{Z}^{m \times m}$ for $\Lambda^\perp(\mathbf{A}_{i,b})$. More precisely, let

$$(\mathbf{A}_{i,b}, \mathbf{S}_{i,b}) \leftarrow \text{TrapSamp}(1^m, 1^{n+\ell+1}, q),$$

where TrapSamp is as described in Lemma 3. Output the public and secret keys

$$pk = \{\mathbf{A}_{i,0}, \mathbf{A}_{i,1}\}_{i \in [n]} \text{ and } sk = \{\mathbf{S}_{1,0}, \mathbf{S}_{1,1}\}.$$

(Note that the receiver does not use the trapdoors for $i > 1$ and so the $\{\mathbf{A}_{i,b}\}_{i > 1}$ could, in fact, simply be chosen at random.)

Encryption. To encrypt the message $\mathbf{w} \in \mathbb{Z}_q^\ell$ with respect to a public key as above, the sender first generates a key pair $(\text{VK}, \text{SK}) \leftarrow \text{SigKeyGen}(1^n)$ for a one-time signature scheme; let $\text{VK} = \text{VK}_1, \dots, \text{VK}_n$ denote the bits of the verification key. Define the matrix \mathbf{A}_{VK} as

$$\mathbf{A}_{\text{VK}} = \begin{bmatrix} \mathbf{A}_{1, \text{VK}_1} \\ \vdots \\ \mathbf{A}_{n, \text{VK}_n} \end{bmatrix}.$$

Choose $\mathbf{s} \leftarrow \mathbb{Z}_q^n$ uniformly at random, and choose an error vector $\mathbf{x} \leftarrow \overline{\Psi}_\beta^{mn}$. The ciphertext is $(\text{VK}, \mathbf{y}, \sigma)$ where

$$\mathbf{y} = \mathbf{A}_{\text{VK}} \cdot \begin{pmatrix} \mathbf{s} \\ 1 \\ \mathbf{w} \end{pmatrix} + \mathbf{x} \pmod{q}$$

and $\sigma = \text{Sign}_{\text{SK}}(\mathbf{y})$.

Decryption. To decrypt a ciphertext $(\text{VK}, \mathbf{y}, \sigma)$, first verify that σ is a correct signature on \mathbf{y} and output \perp if not. Otherwise, parse \mathbf{y} into n consecutive blocks $\mathbf{y}_1, \dots, \mathbf{y}_n$, where $\mathbf{y}_i \in \mathbb{Z}_q^m$. Then,

```

for  $a = 1$  to  $q - 1$  do
    Compute  $\mathbf{t} := \begin{pmatrix} \mathbf{s} \\ a' \\ \mathbf{w} \end{pmatrix} \leftarrow \text{BDDsolve}(\mathbf{T}_{1, \text{VK}_1}, a\mathbf{y})$ 
    if  $a' = a$  then
        if  $\|\mathbf{A}_{i, \text{VK}_i} \cdot \mathbf{t} - a\mathbf{y}_i\| \leq \sqrt{q}/4$  for all  $i \in [n]$  then
            output  $\mathbf{w}/a$  and stop
        else try the next value of  $a$ 
    end
If the above fails for all  $a$ , output  $\perp$ 

```

Theorem 3. *Let n, ℓ, m, q, β be such that $m \geq 4(n + \ell) \log^2 q$ and $\beta < 1/(2 \cdot m^2 n \cdot \omega(\sqrt{\log n}))$. Then, the scheme above is a CCA-secure encryption scheme assuming the hardness of $\text{distLWE}_{n, m, q, \beta}$.*

The proof of correctness is similar to that of the CPA-secure encryption scheme. CCA-security follows from the ideas of [20,12]. As we observed, the main change between our encryption scheme and the one in [20,12] is that the decryption algorithm tries to decrypt “all multiples of the ciphertext”. We defer the details of the proof to the full version.

6.2 An Approximate SPH System

Fix a public key $\{\mathbf{A}_{i,0}, \mathbf{A}_{i,1}\}_{i \in [n]}$, and a password dictionary $\mathcal{D} \stackrel{\text{def}}{=} \mathbb{Z}_q^\ell$. The main difference from the presentation in Section 5.2 is in the definition of ciphertext validity: now, a labeled ciphertext $(\text{label}, \text{VK}, \mathbf{y}, \sigma)$ is defined to be valid

if $\text{Verify}_{\text{VK}}(\text{label}||\mathbf{y}, \sigma) = \text{accept}$. Clearly, all honestly generated ciphertexts are valid and this condition can be checked in polynomial time. We define the sets X , $\overline{L}_{\mathbf{m}}$, and $L_{\mathbf{m}}$ for $\mathbf{m} \in \mathcal{D}$ exactly as in Section 2.

As in Section 5.2, a hash key is a k -tuple of vectors $(\mathbf{e}_1, \dots, \mathbf{e}_k)$ where each $\mathbf{e}_i \leftarrow D_{\mathbb{Z}^m, r}$ is drawn independently from the discrete Gaussian distribution. The projection function and the hash computation are the same, except that here they use the matrices \mathbf{B}_{VK} and \mathbf{U}_{VK} respectively (instead of \mathbf{B} and \mathbf{U} in Section 5.2). In particular, this means that the projection function depends on the ciphertext (as allowed by the definition of an approximate SPH). The proof of the theorem below follows analogously to that of Theorem 2; we defer the proof to the full version of this paper.

Theorem 4. *Let $m \geq 4(n + \ell) \log q$, $\beta < 1/(2 \cdot m^2 n \cdot \omega(\sqrt{\log n}))$ and r be such that*

$$\sqrt{q} \cdot \omega(\sqrt{\log n}) \leq r \leq \varepsilon / (8 \cdot mn^2 \cdot \beta).$$

Then $\mathcal{H} = \{H_k\}_{k \in K}$ is an ε -approximate smooth projective hash system.

References

1. 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)
2. Alwen, J., Peikert, C.: Generating shorter bases for hard random lattices. In: STACS 2009. Dagstuhl Seminar Proceedings, vol. 09001, pp. 75–86. Schloss Dagstuhl (2009), <http://drops.dagstuhl.de/portals/STACS09/>
3. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated key exchange secure against dictionary attacks. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 139–155. Springer, Heidelberg (2000)
4. Bellare, S.M., Merritt, M.: Encrypted key exchange: Password-based protocols secure against dictionary attacks. In: IEEE Symposium on Security & Privacy, pp. 72–84. IEEE, Los Alamitos (1992)
5. Boyko, V., MacKenzie, P.D., Patel, S.: Provably secure password-authenticated key exchange using Diffie-Hellman. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 156–171. Springer, Heidelberg (2000)
6. Canetti, R., Halevi, S., Katz, J., Lindell, Y., MacKenzie, P.D.: Universally composable password-based key exchange. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 404–421. Springer, Heidelberg (2005)
7. Cramer, R., Shoup, V.: Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 45–64. Springer, Heidelberg (2002)
8. Gennaro, R.: Faster and shorter password-authenticated key exchange. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 589–606. Springer, Heidelberg (2008)
9. Gennaro, R., Lindell, Y.: A framework for password-based authenticated key exchange. ACM Trans. Information and System Security 9(2), 181–234 (2006)
10. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: 40th Annual ACM Symposium on Theory of Computing (STOC), pp. 197–206. ACM Press, New York (2008)

11. Goldreich, O., Lindell, Y.: Session-key generation using human passwords only. *Journal of Cryptology* 19(3), 241–340 (2006)
12. Goldwasser, S., Vaikuntanathan, V.: Correlation-secure trapdoor functions and CCA-secure encryption from lattices (manuscript, 2009)
13. Gong, L., Lomas, T.M.A., Needham, R.M., Saltzer, J.H.: Protecting poorly chosen secrets from guessing attacks. *IEEE J. Selected Areas in Communications* 11(5), 648–656 (1993)
14. Halevi, S., Krawczyk, H.: Public-key cryptography and password protocols. *ACM Trans. Information and System Security* 2(3), 230–268 (1999)
15. Jiang, S., Gong, G.: Password based key exchange with mutual authentication. In: Handschuh, H., Hasan, M.A. (eds.) *SAC 2004*. LNCS, vol. 3357, pp. 267–279. Springer, Heidelberg (2004)
16. Katz, J., MacKenzie, P.D., Taban, G., Gligor, V.D.: Two-server password-only authenticated key exchange. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) *ACNS 2005*. LNCS, vol. 3531, pp. 1–16. Springer, Heidelberg (2005)
17. Katz, J., Ostrovsky, R., Yung, M.: Efficient password-authenticated key exchange using human-memorable passwords. In: Pfitzmann, B. (ed.) *EUROCRYPT 2001*. LNCS, vol. 2045, pp. 475–494. Springer, Heidelberg (2001)
18. MacKenzie, P.D., Patel, S., Swaminathan, R.: Password-authenticated key exchange based on RSA. In: Okamoto, T. (ed.) *ASIACRYPT 2000*. LNCS, vol. 1976, pp. 599–613. Springer, Heidelberg (2000)
19. Nguyen, M.-H., Vadhan, S.: Simpler session-key generation from short random passwords. *Journal of Cryptology* 21(1), 52–96 (2008)
20. 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)
21. Peikert, C., Vaikuntanathan, V., Waters, B.: A framework for efficient and composable oblivious transfer. In: Wagner, D. (ed.) *CRYPTO 2008*. LNCS, vol. 5157, pp. 554–571. Springer, Heidelberg (2008); A full version, containing additional results, <http://eprint.iacr.org/2007/348.pdf>
22. Peikert, C., Waters, B.: Lossy trapdoor functions and their applications. In: Ladner, R.E., Dwork, C. (eds.) 40th Annual ACM Symposium on Theory of Computing (STOC), pp. 187–196. ACM Press, New York (2008)
23. 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)
24. Rosen, A., Segev, G.: Chosen-ciphertext security via correlated products. In: Reingold, O. (ed.) *TCC 2009*. LNCS, vol. 5444, pp. 419–436. Springer, Heidelberg (2009)

PSS Is Secure against Random Fault Attacks

Jean-Sébastien Coron and Avradip Mandal

University of Luxembourg

Abstract. A fault attack consists in inducing hardware malfunctions in order to recover secrets from electronic devices. One of the most famous fault attack is Bellcore’s attack against RSA with CRT; it consists in inducing a fault modulo p but not modulo q at signature generation step; then by taking a gcd the attacker can recover the factorization of $N = pq$. The Bellcore attack applies to any encoding function that is deterministic, for example FDH. Recently, the attack was extended to *randomized* encodings based on the ISO/IEC 9796-2 signature standard. Extending the attack to other randomized encodings remains an open problem.

In this paper, we show that the Bellcore attack cannot be applied to the PSS encoding; namely we show that PSS is provably secure against random fault attacks in the random oracle model, assuming that inverting RSA is hard.

Keywords: Probabilistic Signature Scheme, Provable Security, Fault Attacks, Bellcore Attack.

1 Introduction

RSA [14] is still the most widely used signature scheme in practical applications. To sign a message m with RSA, the signer first applies an encoding function μ to m , and then computes the signature $\sigma = \mu(m)^d \bmod N$. The signature is verified by checking that $\sigma^e = \mu(m) \bmod N$. For efficiency reasons RSA signatures are often computed using the Chinese Remainder Theorem (CRT); in this case the signature is first computed modulo p and q separately:

$$\sigma_p = m^d \bmod p, \quad \sigma_q = m^d \bmod q$$

and then σ_p and σ_q are combined by CRT to form the signature σ .

Boneh, DeMillo and Lipton showed that RSA signatures computed with CRT can be vulnerable to fault attacks [3]. If the attacker can induce a fault when σ_q is computed while keeping the computation of σ_p correct, one obtains:

$$\sigma_p = m^d \bmod p, \quad \sigma_q \neq m^d \bmod q$$

and the resulting faulty signature σ satisfies

$$\sigma^e = m \bmod p, \quad \sigma^e \neq m \bmod q.$$

Therefore, given one faulty signature σ , the attacker can recover the factorization of N by computing $\gcd(\sigma^e - m \bmod N, N) = p$. This attack actually applies to any deterministic RSA encoding, e.g. Full Domain Hash (FDH) [2] with $\sigma = H(m)^d \bmod N$.

More generally, the attack applies to any probabilistic scheme where the random used to generate the signature is sent along with the signature, e.g. as in the Probabilistic Full Domain Hash (PFDH) encoding [6] where the signature is $\sigma \| r$ with $\sigma = H(m \| r)^d \bmod N$. In that case, given the faulty value of σ and knowing r , the attacker can still factor N by computing $\gcd(\sigma^e - H(m \| r) \bmod N, N) = p$.

However, if the random r is not given to the attacker along with the signature σ then the Bellcore attack is thwarted. This is the case for signatures of the form $\sigma = \mu(m, r)^d \bmod N$ where the random r is only recovered when verifying the signature, as in PSS [2]. To recover r one needs a *correct* signature; from a faulty signature, the attacker cannot retrieve r nor infer $\mu(m, r)$ in order to compute $\gcd(\sigma^e - \mu(m, r) \bmod N, N) = p$, unless r is short enough to be guessed by exhaustive search. Note that obtaining another correct signature for m would not help the attacker since with high probability a different random r' would be used to generate this signature.

Recently, it was shown how to extend Bellcore’s attack to a large class of randomized RSA encoding schemes [7]. The extended attack was illustrated with the ISO/IEC 9796-2 standard [11]. ISO/IEC 9796-2 is originally a deterministic encoding scheme but often used in combination with message randomization, as in the EMV standard [8]. The ISO/IEC 9796-2 encoded message has the form

$$\mu(m) = 6A_{16} \| m[1] \| H(m) \| BC_{16}$$

where $m = m[1] \| m[2]$ is split into two parts. The authors of [7] showed that if the randomness introduced into $m[1]$ is not too large (e.g. less than 160 bits for a 2048-bit RSA modulus), then a single faulty signature allows to factor N as in the original Bellcore attack. The attack is based on Coppersmith’s technique for finding small roots of polynomial equations [5], which is based on the LLL algorithm [12].

However, extending the attack to other randomized RSA signatures remains an open problem. In particular, it is natural to ask whether the Bellcore attack could apply to PSS [2], the most popular RSA-based signature scheme. In this paper, we show that the Bellcore attack cannot be extended to PSS; namely we show that PSS is provably secure against random fault attacks in the random oracle model, assuming that inverting RSA is hard.

More precisely, we consider an extended model of security in which the attacker, in addition to the regular signing oracle, has access to a faulty signature oracle; that is, the attacker can request faulty signatures either modulo p or modulo q . For a faulty signature modulo q , the signer first generates the correct value modulo p :

$$\sigma_p = \mu(m, r)^d \bmod p$$

but generates a random σ_q modulo q . With CRT the signer then computes σ' such that $\sigma' = \sigma_p \bmod p$ and $\sigma' = \sigma_q \bmod q$, and returns the faulty signature

σ' to the adversary. Our result is that PSS is still secure under this extended notion of security, in the random oracle model, assuming that inverting RSA is hard.

2 Security Model

We recall the definition of a signature scheme.

Definition 1 (signature scheme). *A signature scheme $(\text{Gen}, \text{Sign}, \text{Verify})$ is defined as follows:*

- *The key generation algorithm Gen is a probabilistic algorithm which given 1^k , outputs a pair of matching public and private keys, (pk, sk) .*
- *The signing algorithm Sign takes the message M to be signed, the public key pk and the private key sk , and returns a signature $x = \text{Sign}_{sk}(M)$. The signing algorithm may be probabilistic.*
- *The verification algorithm Verify takes a message M , a candidate signature x' and pk . It returns a bit $\text{Verify}_{pk}(M, x')$, equal to one if the signature is accepted, and zero otherwise. We require that if $x \leftarrow \text{Sign}_{sk}(M)$, then $\text{Verify}_{pk}(M, x) = 1$.*

In the existential unforgeability under an adaptive chosen message attack scenario, the forger can dynamically obtain signatures of messages of his choice and attempts to output a valid forgery. A *valid forgery* is a message/signature pair (M, x) such that $\text{Verify}_{pk}(M, x) = 1$ whereas the signature of M was never requested by the forger.

In the following, we consider an extended model of security in which the attacker, in addition to the regular signing oracle, has access to a faulty signature oracle; that is, the attacker can request faulty signatures either modulo p or modulo q . For a faulty signature modulo q , the signer first generates the correct value modulo p :

$$\sigma_p = \mu(m, r)^d \pmod{p}$$

and generates a random σ_q modulo q . With CRT the signer then computes σ' such that $\sigma' = \sigma_p \pmod{p}$ and $\sigma' = \sigma_q \pmod{q}$, and returns the faulty signature σ' to the adversary. This is actually equivalent to first computing a correct signature σ :

$$\sigma = \mu(m, r)^d \pmod{N}$$

and then generating a random u modulo q and computing the faulty signature:

$$\sigma' = \sigma + u \cdot p \pmod{N}$$

Formally, we consider the following scenario between a challenger and an attacker. Our scenario applies to any RSA based signature scheme in which a signature σ is computed as $\sigma = \mu(m, r)^d \pmod{N}$ for some (randomized) encoding function $\mu(m, r)$.

Setup: the challenger generates an RSA modulus $N = p \cdot q$, a public exponent e such that $\gcd(e, \phi(N)) = 1$ and a private exponent d such that $e \cdot d = 1 \pmod{\phi(N)}$. The challenger sends (N, e) to the adversary.

Queries: the adversary can make regular signature queries to the challenger. In this case, given a message m , the challenger generates a random r and output the (correct) signature:

$$\sigma = \mu(m, r)^d \pmod{N}$$

Additionally, the attacker can make faulty signature queries. For every such query, the attacker specifies whether the fault should be modulo p or modulo q . For a faulty signature modulo q , the challenger first generates a random r and computes the correct signature:

$$\sigma = \mu(m, r)^d \pmod{N}$$

Then the challenger generates a random u modulo q , and computes:

$$\sigma' = \sigma + u \cdot p \pmod{N}$$

and sends σ' to the attacker. The challenger proceeds similarly if a faulty signature modulo p is requested.

Forgery: eventually the attacker must output a forgery, that is a message signature pair (m, x) such that $\text{Verify}_{pk}(m, x) = 1$ whereas the signature of m was never requested by the forger, neither as a regular signature query nor in a faulty signature query.

This completes the description of the attack scenario. As usual, we say that a signature scheme is (t, ε) -secure if no adversary running in time t can output a forgery with probability better than ε .

The PSS scheme was proven secure in the random oracle model [1], and our security proof with faulty signatures is also in the random oracle model. It is well known that a security proof in the random oracle model does not necessarily imply that a scheme is secure in the real world (see [4]). Although it is always better to have a security proof in the standard model, we think that it is still better to have a proof in the random oracle model than no proof at all.

2.1 Why Random Faults?

In our security model we have assumed that when a faulty signature σ' is obtained, it has the uniform distribution modulo p (or modulo q). This could be seen as a very strong assumption; namely in practice the faults might have a completely non-random distribution. Consider for example a fault attack inducing the values of the registers to be set to zero. This gives $\sigma_p = 0$ and recovering p is then straightforward: simply compute $\gcd(\sigma', N) = p$. To prevent from this attack we could assume that when a fault occurs the value σ_p still has enough min-entropy.

In the following we argue that 1) the random fault assumption is almost unavoidable if we want to obtain a security proof and 2) such assumption might actually be reasonable in practice.

Assume that a fault gives a random $\sigma_p \pmod p$ but with the k most significant bits set to 0, for some small integer k . That is, the attacker can obtain a list of faulty signatures σ'_i such that the corresponding $\sigma'_{i,p} = \sigma'_i \pmod p$ satisfy:

$$0 \leq \sigma'_{i,p} < \frac{p}{2^k} \tag{1}$$

for all $1 \leq i \leq n$, where n is the number of faulty signatures. We show how to recover p , using an attack similar to [13]. With LLL [12], the attacker computes a short vector (u_1, \dots, u_n) such that:

$$\sum_{i=1}^n u_i \cdot \sigma'_i = 0 \pmod N$$

This implies:

$$\sum_{i=1}^n u_i \cdot \sigma'_{i,p} = 0 \pmod p$$

Since from (I) the $\sigma'_{i,p}$ are small modulo p , if the u_i 's are small enough, then the equality will hold not only modulo p but also over \mathbb{Z} :

$$\sum_{i=1}^n u_i \cdot \sigma'_{i,p} = 0$$

This gives a vector (u_1, \dots, u_n) that is orthogonal in \mathbb{Z} to the unknown vector $(\sigma'_{1,p}, \dots, \sigma'_{n,p})$. It is shown in [13] that by generating sufficiently many such vectors, one can recover the unknown vector $(\sigma'_{1,p}, \dots, \sigma'_{n,p})$ and eventually p .

Note that this attack applies to any RSA-based signature scheme with CRT, not only to PSS. This attack shows it is *not* enough for σ_p to have min-entropy, as only a few bits of entropy loss compared to the uniform distribution enable to recover p . Therefore, if we want to obtain a security proof, it seems necessary to assume that σ_p is uniformly distributed modulo p .

Actually the random fault assumption might be reasonable in practice. Namely to prevent probing attacks, the data being transmitted in the memory bus inside the micro-processor is usually encrypted. Therefore, the content of a register after a fault attack could still be some encrypted value, so it can be reasonable to model this register value as uniformly random.

3 PSS Is Secure against Random Fault Attacks

3.1 The PSS Scheme

We recall the definition of the PSS scheme [2]. The scheme uses three hash functions $h : \{0, 1\}^* \rightarrow \{0, 1\}^{k_1}$, $g_1 : \{0, 1\}^{k_1} \rightarrow \{0, 1\}^{k_0}$ and $g_2 : \{0, 1\}^{k_1} \rightarrow \{0, 1\}^{k-k_0-k_1-1}$, where k , k_0 and k_1 are parameters.

Key Generation: generate a k -bit RSA modulus $N = pq$, and a random exponent $e \in \mathbb{Z}_{\phi(N)}^*$. Generate d such that $e \cdot d = 1 \pmod{\phi(N)}$. The public-key is (N, e) ; the private key is (N, d) .

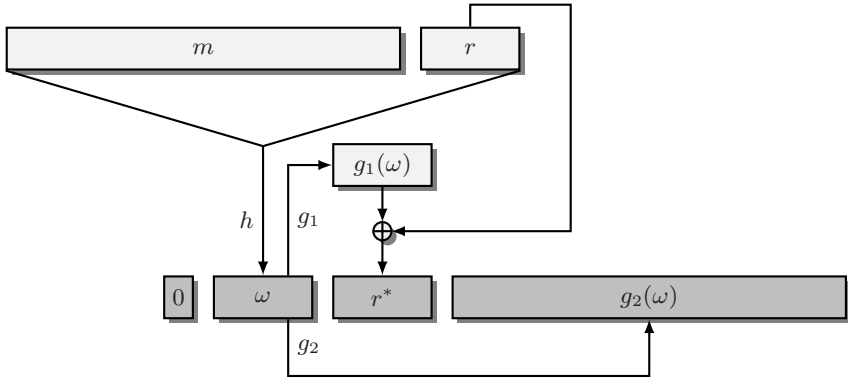


Fig. 1. PSS: the components of the image $y = 0\|\omega\|r^*\|g_2(\omega)$ are darkened. The signature of m is $y^d \bmod N$.

Signature generation: given a message m , do the following:

1. $r \leftarrow \{0, 1\}^{k_0}$
2. $\omega \leftarrow h(m\|r)$
3. $r^* \leftarrow g_1(\omega) \oplus r$
4. $y \leftarrow 0\|\omega\|r^*\|g_2(\omega)$
5. Return $\sigma = y^d \bmod N$

Signature Verification: given a message m and a signature σ , do the following:

1. Let $y = \sigma^e \bmod N$
2. Parse y as $0\|\omega\|r^*\|\gamma$. If the parsing fails return 0.
3. $r \leftarrow r^* \oplus g_1(\omega)$
4. If $h(m\|r) = \omega$ and $g_2(\omega) = \gamma$ return 1.
5. else return 0.

3.2 Security Proof

We first give an intuition of the proof. We denote by $\mu(m, r)$ the PSS encoding scheme, that is $\mu(m, r) = 0\|\omega\|r^*\|g_2(\omega)$ where $\omega = h(m\|r)$ and $r^* = g_1(\omega) \oplus r$.

We receive as input a challenge (N, e, η) and we must output $\eta^d \bmod N$. In the original PSS security proof [2], when receiving a signature query, the simulator generates a random α modulo N such that $\alpha^e \bmod N$ can be written as $0\|\omega\|s\|t$. The simulator generates a random r of k_0 bits. Then it lets $h(m, r) = \omega$, $g_1(\omega) = s \oplus r$ and $g_2(\omega) = t$. Therefore we have that $\mu(m, r) = (\alpha^e \bmod N)$. The simulator can then return α as a signature for m . When receiving a hash query for $h(m, r)$, the simulator generates a random α modulo N such that $\eta \cdot \alpha^e$ can be written as $0\|\omega\|s\|t$; it then proceeds as previously. In this case we have $\mu(m, r) = (\eta \cdot \alpha^e \bmod N)$. Therefore a forgery for $\mu(m, r)$ enables to compute $\eta^d \bmod N$.

One can see that if there is no collision on the randoms r used for signature generation, and no collision on the values ω , then the simulation is perfect. Then

given a forgery σ' for some message m' , with high probability we have that $\mu(m', r') = (\eta \cdot \alpha^e \bmod N)$ for some known α . Therefore from $\sigma' = \mu(m', r')^d \bmod N$ one can compute $\eta^d \bmod N$ as required and solve the RSA challenge.

In our extended model of security, we must additionally simulate a faulty signature oracle. To do this, one could first generate as previously a random α modulo N such that $\alpha^e \bmod N$ can be written as $0\|\omega\|s\|t$. The simulator generates a random r of k_0 bits. Then it lets $h(M, r) = \omega$, $g_1(\omega) = s \oplus r$ and $g_2(\omega) = t$, so that again $\mu(m, r) = (\alpha^e \bmod N)$. Then instead of returning the correct signature α , the simulator could generate a random u modulo q , and output the faulty signature:

$$\alpha' = \alpha + u \cdot p \bmod N \tag{2}$$

Obviously our simulator cannot do this, because it does not know the prime factors p and q . Instead we show that the distribution of α' is statistically close to uniform in \mathbb{Z}_N ; therefore, the simulator can simply return a random $\alpha' \in \mathbb{Z}_N$.

Since RSA is a permutation, instead of considering the distribution of α' , one can consider the distribution of $y' = \alpha'^e \bmod N$. From (2) we have:

$$y' = y + v \cdot p \bmod N$$

where v is uniformly distributed modulo q and y is uniformly distributed in $[0, 2^{k-1}[$. The following lemma shows that the distribution of y' is statistically close to uniform in \mathbb{Z}_N .

Lemma 1. *Let $N = pq$ be a k -bit modulus where p and q are $k/2$ -bit, and let y be a random integer such that $0 \leq y < 2^{k-1}$. Let v be a random integer modulo q . Then the distribution of $y' = y + v \cdot p \bmod N$ is ϵ -statistically close to uniform modulo N , with $\epsilon = \frac{4}{2^{k/2}}$*

Proof. We consider a fixed $a \in \mathbb{Z}_N$ and we provide an estimate of $\Pr[y' = a]$. For this we consider the solutions of the equation:

$$a \equiv y + v \cdot p \bmod N \tag{3}$$

We have that for every $v \in [0, q)$, there exists a unique $y \in [0, N[$ which satisfies the above relation. However we are only interested in the y 's in the range $[0, 2^{k-1}[$. We have that for each $i \in [1, q]$, the pair:

$$(v = q - i, y = a + ip \bmod N)$$

is a solution of (3) iff

$$a + ip \bmod N < 2^{k-1} \tag{4}$$

Depending on the choice of a , there are actually either $\lfloor \frac{2^{k-1}}{p} \rfloor$ or $\lfloor \frac{2^{k-1}}{p} \rfloor + 1$ many i values which satisfy relation (4). Hence there are $\lfloor \frac{2^{k-1}}{p} \rfloor$ or $\lfloor \frac{2^{k-1}}{p} \rfloor + 1$ many solutions to congruence (3) such that $y < 2^{k-1}$. Since y and v are random integers in the range $[0, 2^{k-1})$ and $[0, q)$ respectively, this gives:

$$\left\lfloor \frac{2^{k-1}}{p} \right\rfloor \cdot \frac{1}{2^{k-1}} \cdot \frac{1}{q} \leq \Pr[y' = a] \leq \left(\left\lfloor \frac{2^{k-1}}{p} \right\rfloor + 1 \right) \cdot \frac{1}{2^{k-1}} \cdot \frac{1}{q}$$

We write $\lfloor \frac{2^{k-1}}{p} \rfloor = c$, which gives $p \cdot c < 2^{k-1} < p \cdot c + p$. We obtain:

$$\begin{aligned} \Pr[y' = a] &\geq \frac{c}{2^{k-1}q} = \frac{1}{N} \cdot \frac{pc}{2^{k-1}} = \frac{1}{N} \cdot \left(1 - \frac{2^{k-1} - pc}{2^{k-1}}\right) \\ &> \frac{1}{N} \cdot \left(1 - \frac{p}{2^{k-1}}\right) \quad (\text{as } 2^{k-1} < pc + p) \\ &> \frac{1}{N} \cdot \left(1 - \frac{2p}{N}\right) \quad (\text{as } 2^{k-1} > \frac{N}{2}) \\ &= \left(1 - \frac{2}{q}\right) \cdot \frac{1}{N} \end{aligned}$$

Similarly, we have:

$$\Pr[y' = a] \leq \left(1 + \frac{2}{q}\right) \cdot \frac{1}{N}$$

This gives:

$$\left(1 - \frac{2}{q}\right) \cdot \frac{1}{N} \leq \Pr[y' = a] \leq \left(1 + \frac{2}{q}\right) \cdot \frac{1}{N}$$

for all $a \in [0, N)$. This implies that the distribution of y' is $\frac{4}{2^{k/2}}$ -statistically close to uniform modulo N as $q > 2^{k/2-1}$. □

Lemma □ shows that it is sufficient for our simulator to return a random α' modulo N as the faulty signature. In other words, instead of first generating a random $y \in [0, 2^{k-1})$, then a random v modulo q , then $y' = y + v \cdot p$ and finally $\alpha' = y'^d \pmod N$, the simulator can simply output a random α' modulo N , and such output will be statistically indistinguishable from a faulty signature.

However to this faulty signature α' corresponds a correct signature α such that:

$$\alpha = \alpha' - u \cdot p \pmod N$$

where u is randomly distributed modulo q . Equivalently letting $y' = \alpha'^e \pmod N$ there exists a corresponding value y with:

$$y = y' - v \cdot p \pmod N \tag{5}$$

where v is randomly distributed modulo q such that y can be written as:

$$y = 0 \parallel \omega \parallel s \parallel t = \mu(m, r)$$

This implicitly defines $h(m, r) = \omega$, $g_1(\omega) = s \oplus r$ and $g_2(\omega) = t$ for the simulation of random oracles h , g_1 and g_2 .

Since our simulator does not know p , it cannot compute y in equation (5) and therefore our simulator does not know the corresponding values of ω , s and t ; therefore our simulator cannot answer the corresponding h queries, g_1 queries and g_2 queries if such queries are made by the attacker. Intuitively for h -queries it is sufficient that the set of r values is exponentially large; for this the parameter k_0 must be large enough. For g_1 and g_2 queries we must show

that the adversary has a negligible probability of querying ω . This is shown in the following lemma: we show that given a faulty signature α' (or equivalently $y' = \alpha'^e \pmod N$) the distribution of ω has enough variability, if the parameter k_1 is sufficiently large. This implies that ω does not need to be computed, and therefore the factorization of N is not needed for our simulation.

Lemma 2. *Let $N = pq$ be a k -bit modulus where p and q are $k/2$ -bit, and let y be a random integer such that $0 \leq y < 2^{k-1}$. Let v be a random integer modulo q , and let $y' = y + v \cdot p \pmod N$. Write $y = 0\|\omega\|x$ where ω is k_1 -bit and x is $k - k_1 - 1$ bits. Given y' , for any ω' of k_1 -bit we have:*

$$\Pr[\omega = \omega' | y'] \leq \frac{8}{2^{\min(k_1, k/2)}}$$

Proof. We have that:

$$\Pr[\omega = \omega' | y'] = \frac{\#\{(y, v) \text{ pairs, s.t. } y' = y + v \cdot p \pmod N \text{ and } y = 0\|\omega'\|x\}}{\#\{(y, v) \text{ pairs, s.t. } y' = y + v \cdot p \pmod N \text{ and } 0 \leq y < 2^{k-1}\}}$$

For a fixed v , the value $y \pmod N$ gets fixed by the relation $y' = y + v \cdot p \pmod N$. Moreover at least $\lfloor \frac{q}{2} \rfloor$ of the possible v values give $y \pmod N$ in the desired range between 0 and 2^{k-1} . Hence the denominator of the above fraction can be lower bounded by $\lfloor \frac{q}{2} \rfloor$.

We have that for a fixed y' , the value of y is fixed modulo p ; hence for a fixed ω' with $y = 0\|\omega'\|x$, the value of x is also fixed modulo p . As x is $k - k_1 - 1$ -bit, over \mathbb{Z} there can be at most $\lceil \frac{2^{k-k_1-1}}{p} \rceil$ many possible x values. Hence the numerator of the above fraction can be upper bounded by $\lceil \frac{2^{k-k_1-1}}{p} \rceil$.

Hence we have,

$$\Pr[\omega = \omega' | y'] \leq \frac{\lceil \frac{2^{k-k_1-1}}{p} \rceil}{\lfloor \frac{q}{2} \rfloor} < \frac{\frac{2^{k-k_1-1}}{2^{k/2-1}} + 1}{2^{k/2-2}} = \frac{2^{k-k_1-1} + 2^{k/2-1}}{2^{k-3}} < \frac{8}{2^{\min(k_1, k/2)}}$$

□

Formally, we obtain the following theorem:

Theorem 1. *Assume that no algorithm can invert RSA in time t' with probability better than ε' . Then the signature scheme $\text{PSS}[k_0, k_1]$ is $(t, q_h, q_g, q_s, q_{f_s}, \varepsilon)$ secure, where*

$$\begin{aligned} t(k) &= t'(k) - [q_s(k) + q_g(k) + q_h(k) + 1] \cdot k_0 \cdot \Theta(k^3) \\ \varepsilon(k) &= \varepsilon'(k) + (q_s + q_{f_s} + 1) \cdot (q_s + q_{f_s} + q_h) \cdot 2^{-k_0} + 8 \cdot q_g \cdot q_{f_s} \cdot 2^{-\min(k_1, k/2)} \\ &\quad + (q_h + q_s + q_{f_s}) \cdot (q_h + q_g + q_s + q_{f_s} + 1) \cdot 2^{-k_1} \\ &\quad + q_h \cdot q_{f_s} \cdot 2^{-k_0} + 4 \cdot q_{f_s} \cdot 2^{-k/2} \end{aligned}$$

Here the attacker can make at most q_h, q_g, q_s, q_{f_s} number of h queries, g queries, signature queries and fault signature queries respectively.

Proof. We use a simulator which behaves in exactly same way as in original PSS security proof [2], in addition it answers fault queries with a uniformly random integer modulo N . Now if the attacker is successful against our simulator then we break the RSA challenge (N, e, η) as in the original paper.

We must show that any attacker which is successful against the original attack scenario will be successful against our simulator. For that, we use a sequence of games. We start with Game_0 , which is exactly the attack scenario, which requires to know the factorization of N . Then we progressively modify the game, so that eventually knowledge of the factorization of N is not needed anymore. We denote by S_i the event that the attacker succeeds in Game_i .

Game_0 : this is the attack scenario. We answer signature queries as specified in the signature generation algorithm, using the private exponent d . We simulate the faulty signature queries by first generating a correct signature σ and then computing $\sigma' = \sigma + u \cdot p \pmod N$ for a random u modulo q . In the following for simplicity we only consider faulty signatures modulo q ; faulty signatures modulo p are simulated in exactly the same way.

Game_1 : we abort if there is a collision for ω at Step 2 of the signature generation algorithm, or if the random r used during signature generation has already appeared before. We call this event A_1 . More precisely event A_1 happens if one of the following is true:

- The random r used in a signature oracle or faulty signature oracle query collides with either 1) the r used in a previous signature oracle or faulty signature oracle query or 2) the r used in a previous h oracle query.
- The h function output in a signature oracle or faulty signature oracle query collides with either 1) the h function outputs in previous signature oracle or faulty signature oracle queries or 2) with a previous h oracle query output or 3) a previous g oracle query input.
- The h oracle query output collides with either 1) a h function output in previous signature oracle or faulty signature oracle query or 2) a previous h oracle query output or 3) a previous g oracle query input.

We obtain:

$$\Pr[A_1] \leq (q_s + q_{fs}) \cdot (q_s + q_{fs} + q_h) \cdot 2^{-k_0} + (q_h + q_s + q_{fs}) \cdot (q_h + q_g + q_s + q_{fs}) \cdot 2^{-k_1}$$

and:

$$|\Pr[S_1] - \Pr[S_0]| \leq \Pr[A_1]$$

Game_2 : we construct a similar simulator as in the original PSS security proof [2]; however to deal with faulty signature queries we continue to use the factorization of N .

The simulator receives as input a challenge η and must output $\eta^d \pmod N$. When receiving a signature query, the simulator generates a random α modulo N such that $\alpha^e \pmod N$ can be written as $0\|\omega\|s\|t$. The simulator generates a random r of k_0 bits. Then it lets $h(m, r) = \omega$, $g_1(\omega) = s \oplus r$ and $g_2(\omega) = t$.

When receiving a hash query for $h(m, r)$, the challenger generates a random α modulo N such that $\eta \cdot \alpha^e \pmod N$ can be written as $0\|\omega\|s\|t$; it then defines $h(m, r) = \omega$, $g_1(\omega) = s \oplus r$ and $g_2(\omega) = t$ as previously. The queries to g_1 and g_2 are simulated by returning a random value for every new input.

To simulate the faulty signature oracle, one first generates as above a random α modulo N such that $\alpha^e \pmod N$ can be written as $0\|\omega\|s\|t$. The simulator generates a random r of k_0 bits. Then it lets $h(m, r) = \omega$, $g_1(\omega) = s \oplus r$ and $g_2(\omega) = t$. Then instead of returning α , the simulator generates a random u modulo q , and outputs:

$$\alpha' = \alpha + u \cdot p \pmod N \tag{6}$$

In **Game₂** we abort as in **Game₁**, and additionally in the following case: while generating a random α modulo N such that $\alpha^e \pmod N$ can be written as $0\|\omega\|s\|t$ during signature or faulty signature queries (and similarly for $h(m, r)$ queries), we stop after trying $k_0 + 1$ times. This adds $(q_h + q_s + q_{fs}) \cdot 2^{-k_0}$ in the error term:

$$|\Pr[S_2] - \Pr[S_1]| \leq (q_h + q_s + q_{fs}) \cdot 2^{-k_0}$$

Game₃: we abort if the attacker makes a query for $g(\omega)$ where ω was used in a faulty signature for message m and random r , while the attacker has not made a query to $h(m, r)$ before. We define this event as A_3 . As all the query answers are simulated independently, from Lemma [2](#) this gives:

$$|\Pr[S_3] - \Pr[S_2]| \leq \Pr[A_3] \leq q_g \cdot q_{fs} \cdot \frac{8}{2^{\min(k_1, k/2)}}$$

Game₄: we abort if the attacker makes a query for $h(m, r)$ where r was used to generate a faulty signature with ω , while the attacker has not made a query before to $g(\omega)$. In this case the attacker's view is independent from r , which gives:

$$|\Pr[S_4] - \Pr[S_3]| \leq q_h \cdot q_{fs} \cdot 2^{-k_0}$$

Game₅: we abort if the attacker makes a query for $h(m, r)$ where r was used to generate a faulty signature, or if the attacker makes a query for $g(\omega)$ where ω was used in a faulty signature. **Game₅** is the same as **Game₄** since for a faulty signature m with random r and ω , either the attacker starts with a $h(m, r)$ query or it starts with a $g(\omega)$ query.

$$\Pr[S_5] = \Pr[S_4]$$

Game₆: we change the way the faulty signature oracle is simulated. Instead of first generating α and then α' as in equation [\(6\)](#), we first generate a uniformly random α' and then a random u modulo q such that $\alpha^e \pmod N$ can be written as $0\|\omega\|s\|t$. From Lemma [1](#) we have:

$$|\Pr[S_6] - \Pr[S_5]| \leq q_{fs} \cdot \frac{4}{2^{k/2}}$$

Game₇: since we do not answer the queries for $h(m, r)$ where r was used to generate a faulty signature, and the queries for $g(\omega)$ where ω was used in a

faulty signature, we do not need to compute ω . Therefore, we do not need to compute a random u modulo q such that $\alpha^e \pmod N$ can be written as $0\|\omega\|s\|t$. Therefore we do not need to know the factorization of N anymore, and we have:

$$\Pr[S_7] = \Pr[S_6]$$

Finally, if the adversary outputs a forgery with probability at least ε in **Game**₀, then the adversary must output a forgery with probability at least $\varepsilon - |\Pr[S_7] - \Pr[S_0]|$ in **Game**₇. As in the original PSS security proof, from this forgery we can solve the RSA challenge with probability at least:

$$\varepsilon' = \varepsilon - |\Pr[S_7] - \Pr[S_0]| - 2^{-k_1}$$

Combining the previous inequalities, we get (6). □

4 PSS-R Is Secure against Fault Attacks

In PSS-R or PSS with message recovery the goal is to save bandwidth such that the message is recoverable from the signature; hence it is not necessary to send the message separately.

4.1 The PSS-R Scheme

We recall the definition of the PSS-R scheme [2]. The scheme uses three hash functions $h : \{0, 1\}^* \rightarrow \{0, 1\}^{k_1}$, $g_1 : \{0, 1\}^{k_1} \rightarrow \{0, 1\}^{k_0}$ and $g_2 : \{0, 1\}^{k_1} \rightarrow \{0, 1\}^{k-k_0-k_1-1}$, where k , k_0 and k_1 are the parameters.

Key Generation: generate a k -bit RSA modulus $N = pq$, and a random exponent $e \in \mathbb{Z}_{\phi(N)}^*$. Generate d such that $e \cdot d = 1 \pmod{\phi(N)}$. The public-key is (N, e) ; the private key is (N, d) .

Signature generation: given a message m , do the following:

1. $r \leftarrow \{0, 1\}^{k_0}$
2. $\omega \leftarrow h(M\|r)$
3. $r^* \leftarrow g_1(\omega) \oplus r$
4. $m^* \leftarrow g_2(\omega) \oplus m$
5. $y \leftarrow 0\|\omega\|r^*\|m^*$
6. Return $\sigma = y^d \pmod N$

Message Recovery: given a signature σ , do the following:

1. Let $y = \sigma^e \pmod N$
2. Parse y as $0\|\omega\|r^*\|m^*$. If the parsing fails return REJECT.
3. $r \leftarrow r^* \oplus g_1(\omega)$
4. $m \leftarrow m^* \oplus g_2(\omega)$
5. If $h(m\|r) = \omega$ return m .
6. else return REJECT.

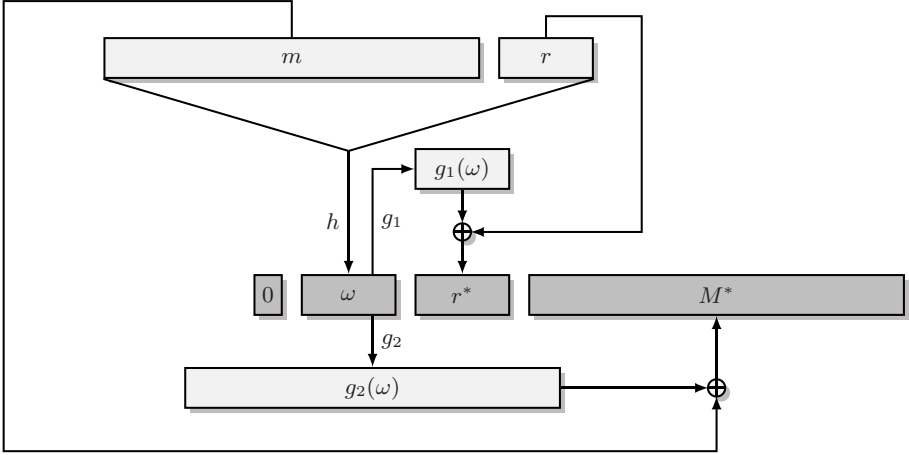


Fig. 2. PSS-R: Components of image $y = 0\|\omega\|r^*\|M^*$ are darkened. The signature of M is $y^d \bmod N$.

4.2 Security Proof

Theorem 2. Assume that no algorithm can invert RSA in time t' with probability better than ε' . Then the signature scheme $\text{PSS-R}[k_0, k_1]$ is $(t, q_h, q_g, q_s, q_{fs}, \varepsilon)$ secure, where:

$$\begin{aligned}
 t(k) &= t'(k) - [q_s(k) + q_g(k) + q_h(k) + 1] \cdot k_0 \cdot \Theta(k^3) \\
 \varepsilon(k) &= \varepsilon'(k) + (q_s + q_{fs} + 1) \cdot (q_s + q_{fs} + q_h) \cdot 2^{-k_0} + 8 \cdot q_g \cdot q_{fs} \cdot 2^{-\min(k_1, k/2)} \\
 &\quad + (q_h + q_s + q_{fs}) \cdot (q_h + q_g + q_s + q_{fs} + 1) \cdot 2^{-k_1} \\
 &\quad + q_h \cdot q_{fs} \cdot 2^{-k_0} + 4 \cdot q_{fs} \cdot 2^{-k/2}
 \end{aligned}$$

Here the attacker can make at most q_h, q_g, q_s, q_{fs} number of h queries, g queries, signature queries and fault signature queries respectively.

Proof. The proof of this theorem is very similar to that of Theorem 1 and hence is omitted.

5 Conclusion

We obtain from the previous theorems that unless the attacker is making more fault oracle queries than hash oracle queries, one gets the same security bound as in the original PSS proof without fault oracle. We note that in practice fault queries are usually more expensive than hash queries, since those hash queries can be made offline when a concrete hash function is used.

In [6] a better security bound was given for PSS (without fault oracle). It was shown that the random size k_0 could be taken as small as $\log_2 q_s$, where q_s is

the maximum number of signature queries; with $q_s = 2^{30}$ this gives $k_0 = 30$ bits. However with a fault oracle one cannot take such a small k_0 , since in this case the random r could be recovered by exhaustive search and the Bellcore attack would still apply.

In summary, any parameters chosen according to the bounds in the original PSS paper [2] give the same level of security against fault attacks. One can take $k = 1024$, $k_0 = k_1 = 128$ as in [2].

References

1. Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols. In: Proceedings of the First Annual Conference on Computer and Communications Security. ACM, New York (1993)
2. 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)
3. Boneh, D., DeMillo, R.A., Lipton, R.J.: On the importance of checking cryptographic protocols for faults. *Journal of Cryptology*, Springer-Verlag 14(2), 101–119 (2001)
4. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited. In: STOC 1998. ACM, New York (1998)
5. Coppersmith, D.: Small solutions to polynomial equations, and low exponent vulnerabilities. *Journal of Cryptology* 10(4), 233–260 (1997)
6. Coron, J.S.: Optimal security proofs for PSS and other signature schemes. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 272–287. Springer, Heidelberg (2002)
7. Coron, J.-S., Joux, A., Kizhvatov, I., Naccache, D., Paillier, P.: Fault Attacks on Randomized RSA Signatures with Partially Unknown Messages. In: CHES 2009, pp. 444–456 (2009), <http://www.jscoron.fr/publications.html>
8. EMVIntegrated circuit card specifications for payment systems, Book 2. Security and Key Management. Version 4.2 (June 2008), <http://www.emvco.com>
9. Goldwasser, S., Micali, S., Rivest, R.: A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal of computing* 17(2), 281–308 (1988)
10. IEEE P1363a, Standard Specifications For Public Key Cryptography: Additional Techniques, <http://www.manta.ieee.org/groups/1363>
11. ISO/IEC 9796-2:2002 Information technology – Security techniques – Digital signature schemes giving message recovery – Part 2: Integer factorization based mechanisms (2002)
12. Lenstra, A., Lenstra Jr., H., Lovász, L.: Factoring polynomials with rational coefficients. *Mathematische Annalen* 261, 513–534 (1982)
13. Nguyễn, P.Q., Stern, J.: Cryptanalysis of a fast public key cryptosystem presented at SAC '97. In: Tavares, S., Meijer, H. (eds.) SAC 1998. LNCS, vol. 1556, p. 213. Springer, Heidelberg (1999)
14. Rivest, R., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public key cryptosystems. *CACM* 21 (1978)

Cache-Timing Template Attacks

Billy Bob Brumley* and Risto M. Hakala

Department of Information and Computer Science,
Helsinki University of Technology,
P.O.Box 5400, FI-02015 TKK, Finland
{billy.brumley,risto.m.hakala}@tkk.fi

Abstract. Cache-timing attacks are a serious threat to security-critical software. We show that the combination of vector quantization and hidden Markov model cryptanalysis is a powerful tool for automated analysis of cache-timing data; it can be used to recover critical algorithm state such as key material. We demonstrate its effectiveness by running an attack on the elliptic curve portion of OpenSSL (0.9.8k and under). This involves automated lattice attacks leading to key recovery within hours. We carry out the attack on live cache-timing data without simulating the side channel, showing these attacks are practical and realistic.

Keywords: cache-timing attacks, side channel attacks, elliptic curve cryptography.

1 Introduction

Traditional cryptanalysis views cryptographic systems as mathematical abstractions, which can be attacked using only the input and output data of the system. As opposed to attacks on the formal description of the system, side channel attacks [1,2] are based on information that is gained from the physical implementation of the system. Side channel leakages might reveal information about the internal state of the system and can be used in conjunction with other cryptanalytic techniques to break the system. Side channel attacks can be based on information obtained from, for example, power consumption, timings, electromagnetic radiation or even sound. Active attacks in which the attacker manipulates the operation of the system by physical means are also considered side channel attacks.

Our focus is on cache-timing attacks in which side channel information is gained by measuring cache access times; these are trace-driven attacks [3]. We place importance on automated analysis for processing large volumes of cache-timing data over many executions of a given algorithm. Hidden Markov models (HMMs) provide a framework, where the relationship between side channel observations and the internal states of the system can be naturally modeled. HMMs for side channel analysis was previously studied by Oswald [4], and models for

* Supported in part by the European Commission's Seventh Framework Programme (FP7) under contract number ICT-2007-216499 (CACE).

key inference given by Karlof and Wagner [5] and Green et al. [6]. While their proposed models make use of an abstract side channel, we are concerned with concrete cache-timing data here.

The analysis additionally makes use of Vector Quantization (VQ) for classification. Cache-timing data is viewed as vectors that are matched to predefined templates, obtained by inducing the algorithm to perform in an unnatural manner. This can often easily be accomplished in software.

Abstractly, it is reasonable to consider the analysis shown here as a form of template attack [7] used in power analysis of symmetric cryptographic primitive implementations, and more recently for asymmetric primitives [8]. Chari et al. [7] formalize exactly what a template is: A precise model for the noise and expected signal for all possible values for part of the key. Their attack is then carried out iteratively to recover successive parts of the key.

It is difficult and not particularly prudent to model cache-timing attacks accordingly. In lieu of such explicit formalization, we borrow from them in name and in spirit: The attacker has some device or code in their possession that they can give input to, program, or modify in some way that forces it to perform in a certain manner, while at the same time obtaining measurements from the side channel.

Using the described analysis method, we carry out an attack on the elliptic curve portion of OpenSSL (0.9.8k). Within hours, we are able to recover the long-term private key in ECDSA by observing cache-timing data, signatures, and messages. Our attack exploits a weakness that stems from the use of a low-weight signed representation for scalars during point multiplication. The algorithm uses a precomputation table of points that are accessed during point addition steps. The lookups are reflected in the cache-timings, leaking critical algorithm state. A significant fraction of ECDSA nonce portions can be determined this way. Given enough such information, we are able to recover the private key using a lattice attack.

The paper is structured as follows. In Sect. 2, we give background on cache architectures and various published cache attacks. In Sect. 3, we review elliptic curve cryptography and the implementation in OpenSSL. Section 4 covers VQ and how to apply it effectively to cache-timing data analysis. In Sect. 5, we discuss HMMs and describe how they are used in our attack, but also how they can be used to facilitate side channel attacks in general. We present our results in Sect. 6, and countermeasures briefly in Sect. 7. We conclude in Sect. 8.

2 Cache Attacks

We begin with a brief review of modern CPU cache architectures. This is followed by a selective literature review of cache attacks on cryptosystem implementations.

2.1 Data Caches

A CPU has a limited number of working registers to store data. Modern processors are equipped with a data cache to offset the high latency of loading data

from main memory into these registers. When the CPU needs to access data, it first looks in the data cache, which is faster but with smaller capacity than main memory. If it finds the data in the cache, it is loaded with minimal latency and this is known as a cache hit; otherwise, a cache miss occurs and the latency is higher as the data is fetched from successive layers of caches or even main memory. Thus access to frequently used data has lower latency. Cache layers L1, L2, and L3 are commonplace, increasing with capacity and latency. We focus on data caches here, but processors often have an instruction cache as well.

The cache replacement policy determines where data from main memory is stored in the cache. At opposite ends of the spectrum are a fully-associative cache and a direct mapped cache. Respectively, these allow data from a given memory location to be stored in any location or one location in the cache. The trade-off is between complexity and latency. A compromise is an N -way associative cache, where each location in memory can be stored in one of N different locations in the cache. The cache locations, or lines, then form a number of associative sets or congruency classes.

We give the L1 data cache details for the two example processors under consideration here.

Intel Atom. The L1 data cache consists of 384 lines of 64B each for a total of 24KB. It is 6-way associative, thus the lines are divided into 64 associative sets.

Intel Pentium 4. The L1 data cache consists of 128 lines of 64B each for a total of 8KB. It is 4-way associative, thus the lines are divided into 32 associative sets.

We focus on these because they implement Intel's HyperThreading, a form of Simultaneous Multithreading (SMT) that allows active execution of multiple threads concurrently. In a cache-timing attack scenario, this relaxes the need to force context switches since the threads naturally compete for shared resources during execution, such as the data caches. The newly-released (Nov. 2008) Intel i7 also features HyperThreading; it has the same number of associative sets as the Intel Atom.

2.2 Published Attacks

Percival [9] demonstrated a cache-timing attack on OpenSSL 0.9.7c (30 Sep. 2003) where a classical sliding window was used twice for exponentiation for two 512-bit exponents in combination with the CRT to carry out a 1024-bit RSA encryption operation. Sliding window exponentiation computes β^e by sliding a width- w window across e with placement such that the value falling in the window is odd. It then uses a precomputation table β^i for all odd $1 \leq i < 2^w$, accessed during multiplication steps; this lookup is reflected in the cache-timings, demonstrated on a Pentium 4 with HyperThreading. The sequence of squarings and multiplications yields significant key data: recovery of 200 bits out of each 512-bit exponent, and [9] claimed an additional 110 bits from each exponent due to fixed memory access patterns revealing information about the index to the

precomputation table and thus key data. Assuming the absence of errors, [9] reasoned how this allows the RSA modulus to be factored in reasonable time. OpenSSL responded to the vulnerability in 0.9.7h (11 Oct. 2005) by modifying the exponentiation routine.

Hlaváč and Rosa [10] used a similar approach to demonstrate a lattice attack on DSA signatures with known nonce portions. They estimated that after observing 6 authentications to an OpenSSH server, which uses OpenSSL (< 0.9.7h) for DSA signatures, an attacker will have a high success probability when running a lattice attack to recover the private key. They state that the side channel was emulated for the experiments.

The numerous published attacks against secret key implementations are noteworthy. Among others, these include attacks on AES by Bernstein [11] and Osvik et al. [12]. Both papers present key recovery attacks on various implementations.

3 Elliptic Curve Cryptography

To demonstrate the effectiveness of the analysis method, we will look at one particular implementation of ECC. We stress that the scope of the analysis is much larger; this is merely one example of how it can be used.

Given a point P on an elliptic curve and scalar k , scalar multiplication computes kP . This operation is the performance benchmark for an elliptic curve cryptosystem. It is normally carried out using a double-and-add approach, of which there are many varieties. We outline a common one later in this section.

Our attack is demonstrated on an implementation of scalar multiplication used by ECDSA signature generation. A signature (r, s) on a message m is produced using

$$r = x(kG) \bmod n \tag{1}$$

$$s = k^{-1}(h(m) + rd) \bmod n \tag{2}$$

with point G of order n , nonce k chosen uniformly from $[1, n)$, $x(P)$ the projection of P to its x -coordinate, h a collision-resistant hash function, and d the long-term private key corresponding to the public key $D = dG$.

3.1 ECC in OpenSSL

OpenSSL treats two cases of elliptic curves over binary and prime fields separately and implements scalar multiplication in two ways accordingly. We consider only the latter case, where a general multi-exponentiation algorithm is used [13][14]. The algorithm works left-to-right and uses interleaving, where one accumulator is used for the result and point doublings are shared; low-weight signed representations are used for individual scalars.

When only one scalar is passed, as in [1] or when creating a signature using the OpenSSL command line tool, it reduces to a rather textbook version of scalar multiplication, in this case using the modified Non-Adjacent Form $mNAF_w$ (see, for example, [15]). This is reflected in the pseudocode below. OpenSSL has the

ability to store the precomputed points in memory, so with a fixed P such as a generator they need not necessarily be recomputed for each invocation.

The representation mNAF_w is very similar to the regular windowed NAF_w . Each non-zero coefficient is followed by at least $w - 1$ zero coefficients, except for the most significant digit which is allowed to violate this condition in some cases to reduce the length of the representation by one while still retaining the same weight. Considering the MSBs of NAF_w , one applies $10^{w-1}\delta \mapsto 010^{w-2}\epsilon$ where $\delta < 0$ and $\epsilon = 2^{w-1} + \delta$ when possible to obtain mNAF_w .

Algorithm: Scalar Multiplication

Input: $k \in \mathbb{Z}$, $P \in E(\mathbb{F}_p)$, width w

Output: kP

$(k_{\ell-1} \dots k_0) \leftarrow \text{mNAF}_w(k)$

Precompute iP for all odd $0 < i < 2^{w-1}$

$Q \leftarrow k_{\ell-1}P$

for $i \leftarrow \ell - 2$ **to** 0 **do**

$Q \leftarrow 2Q$

if $k_i \neq 0$ **then** $Q \leftarrow Q + k_iP$

end

return Q

Algorithm: Modified NAF_w

Input: window width w , $k \in \mathbb{Z}$

Output: $\text{mNAF}_w(k)$

$i \leftarrow 0$

while $k \geq 1$ **do**

if k is odd **then** $k_i \leftarrow k \bmod 2^w$,

$k \leftarrow k - k_i$

else $k_i \leftarrow 0$

$k \leftarrow k/2$, $i \leftarrow i + 1$

end

if $k_{i-1} = 1$ **and** $k_{i-1-w} < 0$ **then**

$k_{i-1-w} \leftarrow k_{i-1-w} + 2^{w-1}$

$k_{i-1} \leftarrow 0$, $k_{i-2} \leftarrow 1$, $i \leftarrow i - 1$

end

return (k_{i-1}, \dots, k_0)

3.2 Cache Attack Vulnerability

Following the description of the mNAF_w representation, knowledge of the curve operation sequence corresponds directly to the algorithm state, yielding quite a lot of key data. Point additions take place when a coefficient $k_i \neq 0$ and these are necessarily followed by w point doublings due to the scalar representation. From the side channel perspective, consecutive doublings allow inference of zero coefficients, and more than w point doublings reveals non-trivial zero coefficients.

Without any countermeasures, the above scalar multiplication routine is vulnerable to cache-timing attacks. The points in the precomputation phase are stored in memory; when a point addition takes place, the point to be added is loaded into the cache. An attacker can detect this by concurrently running a spy process [9] that does nothing more than continually load its own data into the cache and measure the time require to read from all cache lines in a cache set, iterating the process for all cache sets. Fast cache access times indicate cache hits and the scalar multiplication routine has not aggressively accessed those cache locations since the last iteration, which would evict the spy process data from those cache locations, cause a cache miss, and thus slower cache access times for the spy process.

In Fig. 11, we illustrate typical cache timing data obtained from a spy process running on a Pentium 4 (Top) and Atom (Bottom) with OpenSSL 0.9.8k

performing an ECDSA signature operation concurrently. The top eight rows of each graph are metadata; the lower half represents the VQ label and the upper half the algorithm state. We show how we obtained the metadata in Sect. 4 and Sect. 5, respectively. The remaining cells are the actual cache-timing data. Each cell in these figures indicates a cache set access time. Technically, time moves within each individual cell, then from bottom to top through all cache sets, then from left to right repeating the measurements. To visualize the data, it is beneficial to consider the data as vectors with length equal to the number of cache sets, and time simply moves left to right.

To manually analyze such traces and determine what operations are being performed we look for (dis)similarities between neighboring vectors. These graphs show seven (Top) and eight (Bottom) point additions, with repeated point doublings occurring between each addition. As an attacker, we hope to find correlation between these point additions and the cache access times—which we easily find here. Additions in the top graph are visible at rows 13 and 24, among others; the bottom graph, rows 6, 7, 55, 56. The reader is encouraged to use the vector quantization label to help locate the point additions (black label).

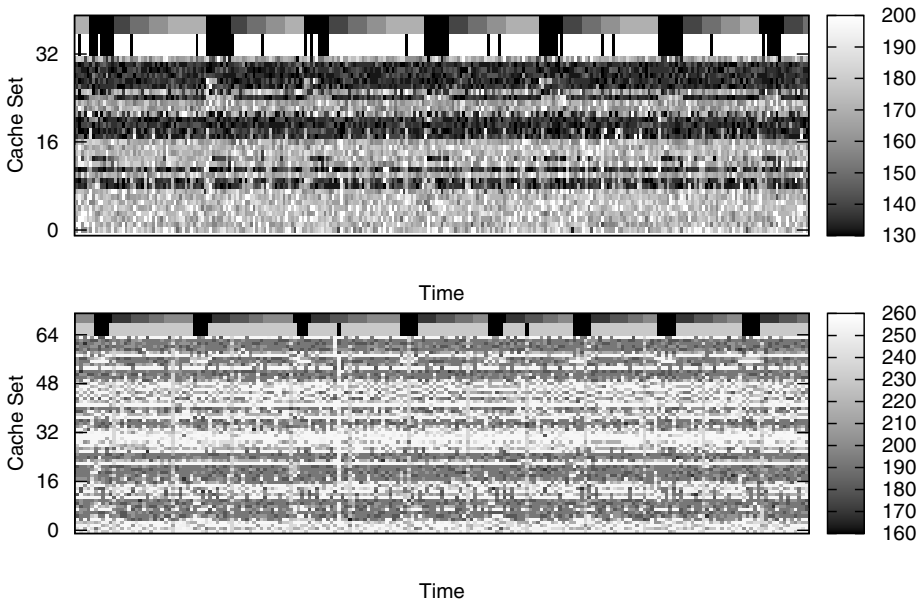


Fig. 1. Cache-timing data from a spy process running concurrently with an OpenSSL 0.9.8k ECDSA signature operation; 160-bit curve, mNAF₄. Top: Pentium 4 timing data, seven point additions. Bottom: Atom timing data, eight point additions. Repeated point doublings occur between the additions. The top eight rows are metadata; the bottom half the VQ label (Sect. 4) and top half the HMM state (Sect. 5). All other cells are the raw timing data, viewed as column vectors from left to right with time.

4 Vector Quantization

Automated analysis of cache-timing data like that shown in Fig. 1 is not a trivial task. When given just one trace, for simplistic algorithms it is sometimes possible to interpret the data manually. For many traces or complex algorithms this is not feasible. We aim to automate the process; the analysis begins with VQ.

A *vector quantizer* is a map $V : \mathcal{R}^n \rightarrow \mathcal{C}$ with $\mathcal{C} \subset \mathcal{R}^n$ where the set $\mathcal{C} = \{c_1, \dots, c_\alpha\}$ is called the *codebook*. A typical definition is $V : v \mapsto \arg \min_{c \in \mathcal{C}} D(v, c)$ where D measures the n -dimensional Euclidean distance between v and c . One also associates a *labelling* $L : \mathcal{C} \rightarrow \mathcal{L}$ with the codebook vectors; this can be as trivial as $\mathcal{L} = \{1, \dots, \alpha\}$ depending on the application.

Here, we are particularly interested in VQ classification; input vectors are mapped to the closest vector in the codebook, then applied the corresponding label for that codebook entry. In this manner, input vectors with the same labelling share some user-defined quality and are grouped accordingly. The classification quality depends on how well the codebook vectors approximate input data for their label. We elaborate on building the codebook \mathcal{C} below.

4.1 Learning Vector Quantization

To learn the codebook vectors, we employ LVQ [16]. This process begins with a set $\mathcal{T} = \{(t_1, l_1), \dots, (t_j, l_j)\}$ of training vectors and predetermined corresponding labels, as well as an approximation to \mathcal{C} . This is commonly derived by taking the k centroids resulting from k -means clustering [17] on all t_i sharing the same label. LVQ in its simplest form then proceeds as follows. For each $t_i, l_i \in \mathcal{T}$ if $L(V(t_i)) = l_i$ the classification is correct and the matching codebook vector is pulled closer to t_i ; otherwise, incorrect and it is pushed away. This process is iterated until an acceptable error rate is achieved.

4.2 Cache-Timing Data Templates

We apply the above techniques to analyze cache-timing data. Taking the working example in Fig. 1, for the Pentium 4 we have $n = 32$ and Atom $n = 64$ the dimension of the cache-timing data vectors; this is the number of cache sets. For simplicity we define $\mathcal{L} = \{D, A, E\}$ to label vectors belonging to respective operations double, addition, or beginning/end.

Next, we build the training data \mathcal{T} . This is somewhat simplified for an attacker as they can create their own private key and generate signatures to produce training data. Nevertheless, extracting individual vectors by hand proves quite tedious and error-prone. Also, if the spy process executes multiple times, there is no guarantee where the memory buffer for the timing data will be allocated. From execution to execution, the vectors will likely look quite different.

Inspired by template attacks [7], we instead modify the software in such a way that it performs only a single task we would like to distinguish. For the scalar multiplication routine shown in Sect. 3, we force the algorithm to perform only point doubling (addition) and collect templates to be used as training vectors

by running the modified algorithm concurrently with the cache spy process, obtaining the needed cache-timing data. This provides large amounts of training vectors and corresponding labels to define \mathcal{T} with minimal effort.

One might be tempted to use these vectors in their entirety for \mathcal{C} . There are a few disadvantages in doing so:

- This would cause VQ to run slower because $\#\mathcal{C}$ would be sizable and contain many vectors such that $L(c_i) = L(c_j)$ where $D(c_i, c_j)$ is needlessly small; codebook redundancy in a sense. In practice we may need to analyze copious volumes of trace data.
- We cannot assume the obtained cache-timing data templates are completely error-free; we strive to curtail the effect of such erroneous vectors.

To circumvent these issues, we partition $\mathcal{T} = \bigcup_{l \in \mathcal{L}} \{(t_i, l_i) : l_i = l\}$ as subsets of all training vectors corresponding to a single label and subsequently perform k -means clustering on the vectors in each subset. The resulting centroids are then added to \mathcal{C} . Finally, with \mathcal{C} and \mathcal{T} realized we employ LVQ to refine \mathcal{C} . This allows experimentation with different values for k in k -means to arrive at a suitably compact \mathcal{C} with small vector classification error rate.

While we expect quality results from VQ classification, errors will nevertheless occur. Furthermore, we are still left with the task of inferring algorithm state. To solve this problem, we turn to hidden Markov models.

5 Hidden Markov Models

HMMs (see, e.g., [18]) are a common method for modeling discrete-time stochastic processes. An HMM is a statistical model in which the system being modeled is assumed to behave like a probabilistic finite state machine with directly unobservable state. The only way of gaining information about the process is through the observations that are emitted from each state.

HMMs have been successfully used in many real life applications; for example, many modern speech recognition methods are based on HMMs [18]. Their usability is based on the ability to model physical systems and gain information about the hidden causes of emitted observations. Thus, it is not very surprising that HMMs can be employed in side channel cryptanalysis as well: the target system can be viewed as the hidden part of the HMM and the emitted observations as information leaked through the side channel. In the following sections, we give a formal definition of an HMM, discuss the three basic problems for HMMs and describe how HMMs are used in our attack. The methodology should give an idea of how to use HMMs in side channel attacks in general.

5.1 Elements of an HMM

An HMM models a discrete-time stochastic process with a finite number of possible states. The state of the process is assumed to be directly unobservable, but information about it can be gained from symbols that are emitted from each

state. The process changes its state based on a set of transition probabilities that indicate the probability of moving from one state to another. An observable symbol is emitted from each process state according to a set of emission probabilities. An example of an HMM is illustrated in Fig. 2. This HMM models a system with three internal states, which are denoted by circles in the figure. Denoted by squares are the two symbols, which can be emitted from the internal states. The state transition probabilities and the emission probabilities are denoted by labeled arrows. For example, the probability of moving from state s_2 to s_3 is a_{23} ; the probability of emitting symbol v_2 from state s_3 is $b_3(2)$. In this HMM, the process always starts from s_1 . Generally, however, there may be several possible first states. The initial state distribution defines the probability distribution for the first state over the states of the HMM.

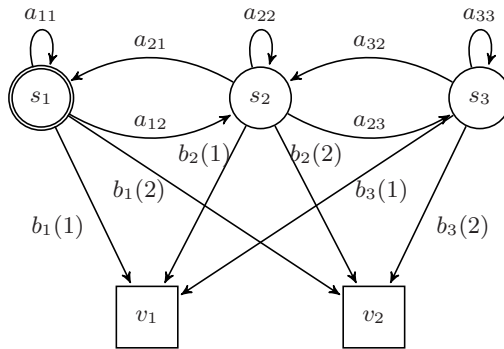


Fig. 2. An example of an HMM

Formally, an HMM is defined by the set of internal states, the set of observation symbols, the transition probabilities between internal states, the emission probabilities for each observable, and the initial state distribution. We denote the set of internal states by $S = \{s_1, s_2, \dots, s_N\}$ and the state at time t by w_t . Correspondingly, the set of observables is denoted by $V = \{v_1, v_2, \dots, v_M\}$ and the observation emitted at time t by o_t . The set of transition probabilities is denoted by $A = \{a_{ij}\}$, where

$$a_{ij} = \Pr(w_{t+1} = s_j | w_t = s_i), \quad 1 \leq i, j \leq N,$$

such that $\sum_{j=1}^N a_{ij} = 1$ for all $1 \leq i \leq N$. Whenever $a_{ij} > 0$, there is a direct transition from state s_i to state s_j ; otherwise, it is not possible to reach s_j from s_i in a single step. An arrow in Fig. 2 denotes a positive transition probability. Thus, s_3 cannot be reached from s_1 in a single step. The set of emission probabilities is denoted by $B = \{b_j(k)\}$, where

$$b_j(k) = \Pr(o_t = v_k | w_t = s_j), \quad 1 \leq j \leq N, \quad 1 \leq k \leq M.$$

The initial state distribution indicates the probability distribution for the first state w_1 . It is denoted by $\pi = \{\pi_i\}$, where

$$\pi_i = \Pr(w_1 = s_i), \quad 1 \leq i \leq N.$$

The first state of the HMM in Fig. 2 is always s_1 , so the initial state distribution for this HMM is defined as $\pi_1 = 1$ and $\pi_i = 0$ for all $i \neq 1$. The three probability measures A , B and π are called the model parameters. For convenience, we will simply write $\lambda = (A, B, \pi)$ to indicate the complete parameter set of an HMM.

5.2 The Three Basic Problems for HMMs

The usefulness of HMMs is based on the ability to model relationships between internal states and observations. Related to this are the following three problems, which are commonly called the three basic problems for HMMs in literature (e.g., [18]):

Problem 1. Given an observation sequence $O = o_1 o_2 \cdots o_T$ and a model $\lambda = (A, B, \pi)$, how do we efficiently compute $\Pr(O|\lambda)$, the probability of the observation sequence given the model?

Problem 2. Given an observation sequence $O = o_1 o_2 \cdots o_T$ and a model λ , what is the most likely state sequence $W = w_1 w_2 \cdots w_T$ that produced the observations?

Problem 3. Given an observation sequence $O = o_1 o_2 \cdots o_T$ and a model λ , how do we adjust the model parameters $\lambda = (A, B, \pi)$ to maximize $\Pr(O|\lambda)$?

We briefly review the methods used to solve these problems; the reader can refer to [18] for a detailed overview. Problem 1 is sometimes called the evaluation problem since it is concerned with finding the probability of a given sequence O . This problem is solved by the forward-backward algorithm (see, e.g., [18]), which is able to efficiently compute the probability $\Pr(O|\lambda)$. Problem 2 poses a problem that is very relevant to our work. It is the problem of finding the most likely explanation for the given observation sequence. The aim is to infer the most likely state sequence W that has produced the given observation sequence O . There are other possible optimality criteria [18], but we are interested in finding W that maximizes $\Pr(W|O, \lambda)$. The problem is known as the decoding problem and it is efficiently solved by the Viterbi algorithm [19]. Another relevant question is posed by Problem 3, which asks how to adjust the model parameters $\lambda = (A, B, \pi)$ to maximize the probability of the observation sequence O . Although there is no known analytical method to adjust λ such that $\Pr(O|\lambda)$ is maximized, the Baum-Welch algorithm [20] provides one method to locally maximize $\Pr(O|\lambda)$. The process is often called training the HMM and it typically involves collecting a set of observation sequences from a real physical phenomenon, which are used in training. This problem is known as the learning problem.

5.3 Use of HMMs in Side-Channel Attacks

HMMs are also useful tools for side channel analysis [4]. Karlof and Wagner [5] and Green et al. [6] use HMMs for modeling side channel attacks. Their research

is concerned with slightly different problems than ours. We outline the differences below.

- They only consider Problem 1 and simulate the side channel. As a result, Problem 3 is not relevant to their work since the artificial side channel actually defines the model that produces the observations. Thus their model parameters are known a priori. This is not the case for our work; Problem 3 is essential.
- They assume one state transition per key digit, in which case the key can be inferred directly from the operation of the algorithm. In our case, the operation sequence does not reveal the entire key, but a significant fraction of the key nevertheless. We use an HMM in which the states correspond only to possible algorithm states.
- They are additionally interested in derivation of the (secret) scalar k in scalar multiplication when the same scalar is used during several runs using a process called belief propagation. This is not helpful in our case, since (EC)DSA uses nonces.

A practical drawback of the HMM presented by Karlof and Wagner was that a single observable needs to correspond to a single key digit (and internal state). Green et al. presented a model, where this is not required: multiple observables can be emitted from each state. This is a more realistic model as one system state may emit variable length data through the side channel. Our model allows this also, but it is based on a different approach.

In the following sections, we describe the HMM used for modeling the OpenSSL scalar multiplication algorithm. We use this model in conjunction with VQ to describe the relationship between the states of the algorithm and the side channel observations. We also describe how to perform side channel data analysis using VQ and the HMM. The aim is to find the most likely state sequence for each trace that is obtained from the side channel. The analysis process can be divided into two steps:

1. The VQ codebook is created and the HMM parameters are adjusted according to obtained training sequences.
2. The actual data analysis is performed. When a sequence of observations is obtained from the side channel, we infer the most likely (hidden) state sequence that has emitted these observations using VQ and the HMM.

Since these states correspond to the internal states of the system, we are able to determine a good estimate of what operations have been done. This information allows us to recover the key.

The following sections give a framework for performing side channel attacks on any system. The main requirements are that we know the specification of the system and have access to do experiments with it or are able to accurately model it.

The HMM for Scalar Multiplication. We construct an HMM where the hidden part models the operation of the algorithm—in this case, scalar multiplication using the modified NAF_w representation, which leaks information about

the algorithm state through the side channel. An illustration of this part (without the transition probabilities) is presented in Fig. 3. The state set is defined as $S = \{s_1, \dots, s_8\}$. Each label denotes the operation that is performed in the corresponding state. In addition, there are separate states to denote the system state preceding and following the execution of the algorithm. These states are denoted by s_1 and s_8 , respectively. OpenSSL uses mNAF₄ for scalars in the case of the 160-bit curve order we are experimenting with, so each point addition is followed by at least 4 point doublings, except in the beginning or end of the process. The states s_3, \dots, s_6 represent these doublings. The most significant digit is handled by the first addition state s_2 .

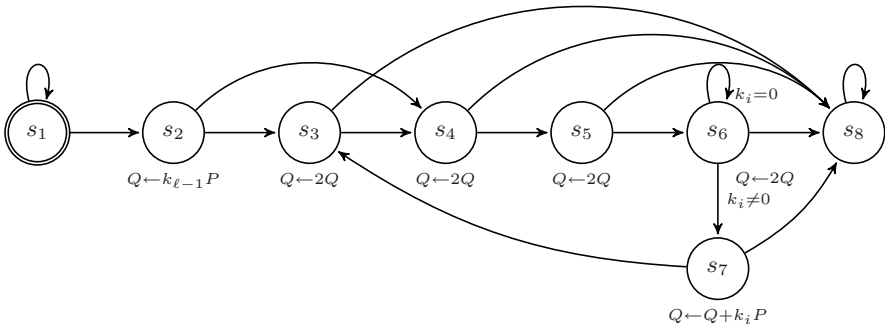


Fig. 3. An HMM transition model for modified NAF₄ scalar multiplication

As can be seen from Fig. 1, the execution of one point doubling or point addition spans several column vectors in the trace. Hence, we should let the internal states emit multiple observations instead of just one. Green et al. [6] solved this problem by introducing an additional variable that counts the cumulative number of emitted observables. This has the drawback of considerably expanding the state space. To avoid this, we solve the problem by introducing substates in each HMM state. One main state consists of a sequence of substates, which are just ordinary HMM states that always emit one observation. Thus, all previously introduced techniques can be used for our HMM.

The set of observables for this HMM is $V = \{D, A, E\}$, which is the same set used for labeling cache-timing data vectors in Sect. 4. We assume that the additions emit mainly *As* and the doublings mainly *Ds*. The s_1 and s_8 states are assumed to emit mainly *Es*. These symbols are connected with side channel observations using VQ as described in Sect. 4. Each vector observation is labeled according to which state—*A*, *D* or *E*—they correspond to. When a new side channel observation is obtained, it can be classified as *A*, *D* or *E* by taking the label of the closest codebook vector. An example of this is shown in Fig. 1, where the rows directly above the observations represent the quantized values. Symbols *A* and *D* are indicated using darker and lighter shades, respectively.

Training of the HMM. Training starts by setting the initial model parameters. These parameters can be rough estimates, since they will be improved during training. To train the model, we obtain a set of sequences in the HMM observation domain. These sequences can be created from the side channel observations as we know how the algorithm operates. The obtained sequences are used for model parameter re-estimation, which is performed using the Baum-Welch algorithm [20]. Next, we create the codebook for VQ as shown in Sect. 4.

Inference of the State Sequence. Given a set of side channel observation sequences from the real target system, we can infer the most likely hidden state sequence for each of them. The first step is to perform VQ, this is, to tag the observations with the label of the closest codebook vector. Thus, we get a set of sequences in the HMM observation domain. By applying the Viterbi algorithm [19], we finally obtain the most likely state sequence for each observation sequence. These state sequences are actually sequences of substates; the actual operation sequence can be recovered based on the transitions that are taken in each state sequence. An example of this is shown in Fig. 1, where the upper rows represent the main states of the algorithm. Additions are indicated using black; doublings are indicated using lighter shades. For example, the first addition on the top trace in Fig. 1 is followed by five doublings.

The state sequences obtained in this step can be used in conjunction with some other method to mount a key recovery attack. In the simplest case, the state sequence reveals the secret key directly and no other methods are needed. However, with $mNAF_4$ this is not the case; we discuss a few practical applications in the next section, as well as give our empirical results.

6 Results

Depending on the attack scenario and the number of traces available, there are at least two interesting ways to apply the analysis to the case of $mNAF_4$ and OpenSSL. The first assumes access to only a single or similarly small number of traces, while the second assumes access to a signature oracle and corresponding side channel information.

Solving Discrete Logs. We consider special versions of the baby-step giant-step algorithm for searching restricted exponent spaces; see [21, Sect. 3.6] for a good overview.

The length- ℓ $mNAF_w$ representation has maximum weight ℓ/w and average weight $\ell/(w+1)$; we denote this weight as t . We assume that the analysis provides us with the position of non-zero coefficients, but not their explicit value or sign; thus each coefficient gives $w - 1$ bits of uncertainty. One can then construct a baby-step giant-step algorithm to solve the ECDLP in this restricted key space. The time and space complexity is $O(2^{(w-1)t/2})$; note that this does not directly depend on ℓ (or further, the group order n). For the curve under consideration, this gives a worst case of $O(2^{60})$ and on average $O(2^{48})$, whereas the complexity without any such side channel information is $O(2^{80})$.

Lattice Attacks. Despite this reduced complexity, an attacker cannot trivially carry out the attack outlined above on a normal desktop PC. Known results on attacking signature schemes with partial knowledge of nonces include [22,23]; the approach is a lattice attack. Formally, the attacker obtains tuples $(r_i, s_i, m_i, \hat{k}_i)$ consisting of a signature (2), message, and partial knowledge of the nonce k obtained through the timing data analysis. For our experiments, not all such tuples are useful in the lattice attack. Using the formalization of [22], we assume \hat{k}_i tells us

$$k_i = z'_i + 2^{\alpha_i} z_i + 2^{\beta_i} z''_i$$

with z_i the only unknown on the right. Our empirical timing data analysis results show that the majority of errors occur when too many or few doubles are placed between an addition; a synchronization error in a sense. So the farther we move towards the MSB, the more likely it is that we have erroneous indexing α_i, β_i and the lattice attack will likely fail.

To mitigate this issue, we instead focus only on the LSBs. We disregard the upper term by setting $z''_i = 0$ and consider only tuples where \hat{k}_i indicates that $z'_i = 0$ and $\alpha_i \geq 6$; that is, the LSBs of k_i are 000000. For k chosen uniformly, this should happen with the reasonable probability of 2^{-6} . Our empirical results are in line with those of [22]: For a 160-bit group order, 41 such tuples is usually enough for the lattice attack to succeed in this case.

Lattice attacks have no recourse to compensate for errors. If our analysis determines $z'_i = 0$ but indeed $z'_i \neq 0$ for some i , that instance of the lattice attack will fail. We thus adopt the naïve strategy of taking random samples of size 41 from the set of tuples until the attack succeeds; an attacker can always check the correctness of a guess by calculating the corresponding public key and comparing it to the actual public key. This strategy is only feasible if the ratio of error-free tuples to erroneous tuples is high.

Finally, we present the automated lattice attack results; 8K signatures with messages and traces were obtained in both cases.

Pentium 4 results. The analysis yielded 122 tuples indicating $z'_i = 0$. The long-term private key d (2) was recovered after 1007 lattice attack iterations (107 correct, 15 incorrect). The analysis ran in less than an hour on a Core 2 Duo.

Atom results. The analysis yielded 147 tuples indicating $z'_i = 0$. We recovered d after a total of 37196 lattice attack iterations (115 correct, 32 incorrect). Our analysis is less accurate in this case, but still accurate enough to recover the key in only a few hours on a Core 2 Duo.

Summary. We omit strategies for finding correlation between the traces and specific key digits. This can be tremendously helpful in further reducing the search space when trying to solve the ECDLP. As such, given only one or a few traces, this analysis method should be used as a tool in conjunction with other heuristics to trim the search space. The lattice attack given here is proof-of-concept. The results suggest that significantly fewer signatures are needed. In practice one can perform a much more intelligent lattice attack, perhaps even considering lattice attacks that account for key digit reuse [24].

7 Countermeasures

An implementation should not rely on any one countermeasure for side channel security, but rather a combination. We briefly discuss countermeasures, with an emphasis on preventing the specific weakness we exploited in OpenSSL.

Scalar Blinding. One often-proposed strategy [1,25,26,27] is to blind the scalar k from the point multiplication routine using randomization. One form is $(k + mn + \tilde{m})P - \tilde{m}P$ with m, \tilde{m} small (e.g. 32-bit) and random. The calculation is then carried out using multi-exponentiation with interleaving. With such a strategy, it suffices that \tilde{m} is low weight—not necessarily short.

Randomized Algorithms. Use random addition-subtraction chains instead of highly regular double-and-add routines. Oswald [28] gave an example and a subsequent attack [4]. Published algorithms tend to be geared towards hardware or resource restricted devices; see [29] for a good review. In a software package like OpenSSL that normally runs on systems with abundant memory, one does not have to rely on simple randomized recoding and can build more flexible addition-subtraction chains.

Shared Context. In OpenSSL’s ECC implementation, the results and illustration in Fig. 1 suggest what is most visible in the traces is not the lookup from the precomputation table, but the dynamic memory for variables in the point addition and doubling functions. OpenSSL is equipped with a shared context [30, pp. 106–107] responsible for allocating memory for curve and finite field arithmetic. Memory from this context should be served up randomly to prevent a clear fixed memory access pattern.

Operation Balancing. In addition to the above shared context, coordinate systems and point addition formulae that are balanced in the number and order of operations are also useful; [31] gives an example.

The above countermeasures restrict to the software engineering view. Clearly operating system-level and hardware-level countermeasures are additionally possible. We leave general countermeasures to this type of attack as an open question.

8 Conclusion

We summarize our contributions as follows:

- We introduced a method for automated cache-timing data analysis, facilitating discovery of critical algorithm state. This is the first work we are aware of that provides this at a framework level, e.g. not specific to one cryptosystem. Consequentially, it bridges the gap between cache attack vulnerabilities [9] and attacks requiring partial key material [22,23].
- We showed how to apply HMM cryptanalysis to cache-timing data; to the best of our knowledge, its first published application to real traces. This builds on existing work in the area of abstract side channel analysis using HMMs [4,5,6], yet departs by tackling practical issues inherent to concrete side channels.

- We demonstrated the method is indeed practical by carrying out an attack on the elliptic curve portion of OpenSSL using live cache-timing data. The attack resulted in complete key recovery, with the analysis running in a matter of hours on a normal desktop PC.

The method works by:

1. Creating cache-timing data vector templates that reflect the algorithm’s cache access behavior.
2. Using VQ to match incoming cache-timing data to these existing templates.
3. Using the output as observation input to an HMM that accurately models the control flow of the algorithm.

The setup phase, including acquiring the templates used to build the VQ code-book vectors and learning the HMM parameters, is the only part by definition requiring any manual work, and the majority of that can in fact be automated by simple modifications to the software under attack. This attack scenario is described for hardware power analysis in [7], but is perhaps even a greater practical threat in this case due to the inherent malleability of software. After the setup phase, cache-timing data analysis is fully automated and requires negligible time.

The analysis given here is not strictly meant for attacking implementations, but for defending them as well. We encourage software developers to analyze their implementations using these methods to discover memory access patterns and apply appropriate countermeasures.

Future Work

One might think to forego the VQ step and use the cache-timing data directly as the sole input to the HMM. In our experience, this only complicates the model and hampers quality results.

The example we gave was tailored to data caches, in particular the L1 data cache. Other data caches could prove equally fruitful. We also plan to apply the analysis method to instruction caches.

While the attack results we gave were for one particular cryptosystem implementation, the analysis method has a much wider range of applications. We in fact found a similar vulnerability in the NSS library’s implementation of elliptic curves. Departing from elliptic curves and public key cryptography, we plan to apply the analysis to an assortment of implementations, asymmetric and symmetric primitives alike.

One of the more interesting planned applications is to algorithms with good side channel resistance properties, such as “Montgomery’s ladder”. While this might be an overwhelming challenge for traditional power analysis, the work here emphasizes the fact that cache-timing attacks are about memory access patterns; a fixed sequence of binary operations cannot be assumed sufficient to thwart cache-timing attacks.

Acknowledgments. We thank the following people for comments and discussions: Dan Bernstein, Kimmo Järvinen, Kaisa Nyberg, and Dan Page.

References

1. 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)
2. 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)
3. Page, D.: Defending against cache based side-channel attacks. Information Security Technical Report 8(1), 30–44 (2003)
4. Oswald, E.: Enhancing simple power-analysis attacks on elliptic curve cryptosystems. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 82–97. Springer, Heidelberg (2003)
5. Karlof, C., Wagner, D.: Hidden Markov model cryptanalysis. In: Walter, C.D., Koç, Ç.K., Paar, C. (eds.) CHES 2003. LNCS, vol. 2779, pp. 17–34. Springer, Heidelberg (2003)
6. Green, P.J., Noad, R., Smart, N.P.: Further hidden Markov model cryptanalysis. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 61–74. Springer, Heidelberg (2005)
7. Chari, S., Rao, J.R., Rohatgi, P.: Template attacks. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 13–28. Springer, Heidelberg (2003)
8. Medwed, M., Oswald, E.: Template attacks on ECDSA. In: Chung, K.-I., Sohn, K., Yung, M. (eds.) WISA 2008. LNCS, vol. 5379, pp. 14–27. Springer, Heidelberg (2009)
9. Percival, C.: Cache missing for fun and profit (2005), <http://www.daemonology.net/papers/cachemissing.pdf>
10. Hlaváč, M., Rosa, T.: Extended hidden number problem and its cryptanalytic applications. In: Biham, E., Youssef, A.M. (eds.) SAC 2006. LNCS, vol. 4356, pp. 114–133. Springer, Heidelberg (2007)
11. Bernstein, D.J.: Cache-timing attacks on AES (2004), <http://cr.yp.to/papers.html#cachetiming>
12. Osvik, D.A., Shamir, A., Tromer, E.: Cache attacks and countermeasures: The case of AES. In: Pointcheval, D. (ed.) CT-RSA 2006. LNCS, vol. 3860, pp. 1–20. Springer, Heidelberg (2006)
13. Möller, B.: Algorithms for multi-exponentiation. In: Vaudenay, S., Youssef, A.M. (eds.) SAC 2001. LNCS, vol. 2259, pp. 165–180. Springer, Heidelberg (2001)
14. Möller, B.: Improved techniques for fast exponentiation. In: Lee, P.J., Lim, C.H. (eds.) ICISC 2002. LNCS, vol. 2587, pp. 298–312. Springer, Heidelberg (2003)
15. Bosma, W.: Signed bits and fast exponentiation. Journal de Théorie des Nombres de Bordeaux 13(1), 27–41 (2001)
16. Kohonen, T.: Self-Organizing Maps. Springer, Heidelberg (1995)
17. Lloyd, S.: Least squares quantization in PCM. IEEE Transactions on Information Theory 28(2), 129–137 (1982)
18. Rabiner, L.R.: A tutorial on hidden Markov models and selected applications in speech recognition. Proceedings of the IEEE 77(2), 257–286 (1989)
19. Viterbi, A.J.: Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. IEEE Transactions on Information Theory 13(2), 260–269 (1967)
20. Baum, L.E., Petrie, T., Soules, G., Weiss, N.: A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. The Annals of Mathematical Statistics 41(1), 164–171 (1970)

21. Menezes, A., van Oorschot, P., Vanstone, S.: Handbook of Applied Cryptography, 5th edn. CRC Press, Boca Raton (2001)
22. Howgrave-Graham, N., Smart, N.P.: Lattice attacks on digital signature schemes. *Designs, Codes and Cryptography* 23(3), 283–290 (2001)
23. Nguyen, P.Q., Shparlinski, I.: The insecurity of the elliptic curve digital signature algorithm with partially known nonces. *Designs, Codes and Cryptography* 30(2), 201–217 (2003)
24. Leadbitter, P.J., Page, D., Smart, N.P.: Attacking DSA under a repeated bits assumption. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 428–440. Springer, Heidelberg (2004)
25. Coron, J.S.: Resistance against differential power analysis for elliptic curve cryptosystems. In: Koç, Ç.K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, pp. 292–302. Springer, Heidelberg (1999)
26. Clavier, C., Joye, M.: Universal exponentiation algorithm: a first step towards provable SPA-resistance. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 300–308. Springer, Heidelberg (2001)
27. Möller, B.: Parallelizable elliptic curve point multiplication method with resistance against side-channel attacks. In: Chan, A.H., Gligor, V.D. (eds.) ISC 2002. LNCS, vol. 2433, pp. 402–413. Springer, Heidelberg (2002)
28. Oswald, E., Aigner, M.: Randomized addition-subtraction chains as a countermeasure against power attacks. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 39–50. Springer, Heidelberg (2001)
29. Walter, C.D.: Randomized exponentiation algorithms. In: Koç, Ç.K. (ed.) *Cryptographic Engineering*. Springer, Heidelberg (2009)
30. Viega, J., Messier, M., Chandra, P.: *Network Security with OpenSSL*. O'Reilly Media, Inc., Sebastopol (2002)
31. Chevallier-Mames, B., Ciet, M., Joye, M.: Low-cost solutions for preventing simple side-channel analysis: Side-channel atomicity. *IEEE Transactions on Computers* 53(6), 760–768 (2004)

Memory Leakage-Resilient Encryption Based on Physically Unclonable Functions

Frederik Armknecht¹, Roel Maes², Ahmad-Reza Sadeghi¹, Berk Sunar³,
and Pim Tuyls^{2,4}

¹ Horst Görtz Institute for IT Security, Ruhr-University Bochum, Germany

² ESAT/COSIC and IBBT, Catholic University of Leuven, Belgium

³ Cryptography & Information Security, WPI, MA USA

⁴ Intrinsic ID, Eindhoven, the Netherlands

Abstract. Physical attacks on cryptographic implementations and devices have become crucial. In this context a recent line of research on a new class of side-channel attacks, called *memory attacks*, has received increasingly more attention. These attacks allow an adversary to measure a significant fraction of secret key bits directly from memory, independent of any computational side-channels.

Physically Unclonable Functions (PUFs) represent a promising new technology that allows to store secrets in a tamper-evident and unclonable manner. PUFs enjoy their security from physical structures at sub-micron level and are very useful primitives to protect against memory attacks.

In this paper we aim at making the first step towards combining and binding algorithmic properties of cryptographic schemes with physical structure of the underlying hardware by means of PUFs. We introduce a new cryptographic primitive based on PUFs, which we call PUF-PRFs. These primitives can be used as a source of randomness like pseudorandom functions (PRFs). We construct a block cipher based on PUF-PRFs that allows simultaneous protection against algorithmic and physical attackers, in particular against memory attacks. While PUF-PRFs in general differ in some aspects from traditional PRFs, we show a concrete instantiation based on established SRAM technology that closes these gaps.

1 Introduction

Modern cryptography provides a variety of tools and methodologies to analyze and to prove the security of cryptographic schemes such as in [7, 8, 6, 9]. These proofs always start from a particular setting with a well-defined adversary model and security notion. The vast majority of these proofs assume a *black box* model: the attacker knows all details of the used algorithms and protocols but has no knowledge of or access to the secrets of the participants, nor can he observe any internal computations. The idealized model allows one to derive security guarantees and gain valuable insights.

However, as soon as this basic assumption fails most security guarantees are off and a new open field of study arises. In cryptographic implementations long-term secret keys are typically stored by configuring a non-volatile memory such as ROM, EEPROM, flash, anti-fuses, poly or e-fuses into a particular state. Computations on these secrets are performed by driving electrical signals from one register to the next and transforming them using combinatorial circuits consisting of digital gates. Side-channel attacks pick up physically leaked key-dependent information from internal computations, e.g. by observing consumed power [27] or emitted radiation [1], making many straightforward algorithms and implementations insecure. It is clear that from an electronic hardware point of view, security is viewed differently, see e.g. [30,49,48,44].

Even when no computation is performed, stored secret bits may leak. For instance, in [43] it was shown that data can be recovered from flash memory even after a number of erasures. By decapsulating the chip and using scanning electron microscopes or transmission electron microscopes the states of anti-fuses and flash can be made visible. Similarly, a typical computer memory is not erased when its power is turned off giving rise to so-called cold-boot attacks [22]. More radical approaches such as opening up an integrated circuit and probing metal wires or scanning non-volatile memories with advanced microscopes or lasers generally lead to a security breach of an algorithm, often immediately revealing an internally stored secret [43].

Given this observation, it becomes natural to investigate security models with the basic assumption: *memory leaks information on the secret key*. Consequently, a recently started line of work has investigated the use of new cryptographic primitives that are less vulnerable to leakage of key bits [2,36]. These works establish security by adapting public-key algorithms to remain secure even after leaking a limited number of key bits. However, no security guarantees can be given when the leakage exceeds a certain threshold, e.g. when the whole non-volatile memory is compromised. Furthermore, they do not provide a solution for the traditional settings, e.g. for securing symmetric encryption schemes.

Here we explore an alternative approach: Instead of making another attempt to solve the problem in an algorithmic manner, we base our solution on a new physical primitive. So-called Physically Unclonable Functions (PUFs) provide a new cryptographic primitive able to store secrets in a non-volatile but highly secure manner. When embedded into an integrated circuit, PUFs are able to use the deep submicron physical uniqueness of the device as a source of randomness [15,14,20,47]. Since this randomness stems from the uncontrollable subtleties of the manufacturing process, rather than from hard-wired bits, it is practically infeasible to externally measure these values during a physical attack. Moreover, any attempt to open up the PUF in order to observe its internals will with overwhelming probability alter these variables and change or even destroy the PUF [47].

In this paper, we take advantage of the useful properties of PUFs to build an encryption scheme resilient against memory leakage adversaries as defined in [2]. We construct a block cipher that explicitly makes use of the algorithmic and

physical properties of PUFs to protect against physical *and* algorithmic attacks at the same time. Other protection mechanisms against physical attacks require either additional algorithmic effort, e.g. [24,34,45,39], on the schemes or separate (possibly expensive) hardware measures.

Our encryption scheme can particularly be used for applications such as secure storage of data on untrusted storage (e.g., harddisk) where (i) no storage of secrets for encryption/decryption is needed and keys are only re-generated when needed, (ii) copying the token is infeasible (unclonability), (iii) temporary unauthorized access to the token will reveal data to the adversary but not the key, or (iv) no non-volatile memory is available.

Contribution. Our contributions are as follows:

A new cryptographic primitive: PUF-PRF. We place the PUFs at the core of a pseudorandom function (PRF) construction that meets well-defined properties. We provide a formal model for this new primitive that we refer to as PUF-PRFs. PRFs [19] are fundamental primitives in cryptography and have many applications, e.g. see [18,32,33].

A PUF-PRF-based provably secure block cipher. One problem with our PUF-PRF construction is that it requires some additional helper data that inevitably leaks some internal information. Hence, PUF-PRFs cannot serve as a direct replacement for PRFs. However, we present a provably secure block cipher based on PUF-PRFs that remains secure despite the information leakage. Furthermore, no secret key needs to be stored, protecting the scheme against memory leakage attacks. The tight integration of PUF-PRFs into the cryptographic construction improves the tamper-resilience of the overall design. Any attempt at accessing the internals of the device will result in a change of the PUF-PRF. Hence, no costly error detection networks or alternative anti-tampering technologies are needed. The unclonability and tamper-resilience properties of the underlying PUFs allow for elegant and cost-effective solutions to specific applications such as software protection or device encryption.

An improved and practical PUF-PRF construction. Although the information leakage through helper data is unavoidable in the general case, the concrete case might allow for more efficient and secure constructions. We introduce SRAM-PRFs, based on so-called SRAM PUFs, which are similar to the general PUF-PRFs but where it can be shown that no information is leaked through the helper data if run in an appropriate mode of operation. Hence, SRAM-PRFs are in all practical views a physical realization of expanding PRFs.

Organization. This paper is organized as follows. First, we give an overview of related work in Section 2. In Section 3, we define and justify the considered attacker model. In Section 4, we introduce a formal model for PUFs. Based on this, we define in Section 5 a new cryptographic primitive, termed PUF-PRFs. Furthermore, we present a provably secure block cipher based on PUF-PRFs that is secure despite the information leakage through helper data. In Section 6,

we explain for the concrete case of SRAM PUFs an improved construction that shares the same benefits like general PUF-PRFs but where it can be argued that the helper data does not leak any information. Finally, in Section 7 we present the conclusions.

2 Related Work

In recent years numerous results in the field of physical attacks emerged showing that the classical black box model is overly optimistic, see e.g. [42,43,3,28,27]. Due to a number of physical leakage channels, the adversary often learns (part of) a stored secret or is able to observe some intermediate results of the private computations. These observations give him a powerful advantage that often breaks the security of the entire scheme. To cope with this reality, a number of new theoretic adversary models were proposed, incorporating possible physical leakage of this kind. Ishai et al. [24] model an adversary which is able to probe, i.e. eavesdrop, a number of lines carrying intermediate results in a private circuit, and show how to create a secure primitive within this computational leakage model. Later, generalizations such as Physically Observable Cryptography proposed by Micali et al. [34] investigate the situation where only computation leaks information while assuming leak-proof secret storages. Recently, Pietrzak [13,39] and Standaert et al. [45] put forward some new models and constructions taking physical side-channel leakage into account.

Complementary to the computation leakage attacks, another line of work explored memory leakage attacks: an adversary learns a fraction of a stored secret [2,36]. In [2] Akavia et al introduced a more realistic model that considers the security against a wide class of side-channel attacks when a function of the secret key bits is leaked. Akavia et al further showed that Regev's lattice-based scheme [41] is resilient to key leakage. More recently Naor et al [36] proposed a generic construction for a public-key encryption scheme that is resilient to key leakage. Although all these papers present strong results from a theoretical security point of view, they are often much too expensive to implement on an integrated circuit (IC), e.g. the size of private circuits in [24] blows up with $O(n^2)$ where n denotes the number of probings by the adversary. Moreover, almost all of these proposals make use of public-key crypto primitives, which introduce a significant overhead in systems where symmetric encryption is desired for improved efficiency.

Besides the information leakage attacks mentioned above, another important field of studies are tampering attacks. Numerous countermeasures have been discussed, e.g., use of a protective coating layer [40] or the application of error detection codes (EDCs) [25,16]. Observe that limitations and benefits of tamper-proof hardware have likewise been theoretically investigated in a series of works [17,26,35,10].

3 Memory Attacks

In this work we consider an extension of memory attacks as introduced by Akavia et. al. [2] where the attacker can extract a bounded number of bits of a stored

secret. The model allows for covering a large variety of different memory attacks, e.g., cold boot attacks described in [22]. However, this general model might not adequately capture certain concrete scenarios. For example, feature sizes on ICs have shrunk to nanometer levels and probing such fine metal wires is even for high-end IC manufacturers a difficult task. During a cryptographic computation a secret state is (temporarily) stored in volatile memory (e.g. in registers and flip-flops). In a typical IC, these structures are relatively small compared to the rest of the circuit, making them very hard to locate and scan properly. Thus, applying these attacks is usually significantly physically more involved for the case of embedded ICs than for the non-embedded PC setting where additional measures to access the memory exist, e.g., through software and networks.

On the other hand, storing long-term secrets, such as private keys, requires non-volatile memory, i.e. memory that sustains its state while the embedding device is powered off. Implementation details of such memories like ROM, EEPROM, flash, anti-fuses, poly or e-fuses and recent results on physical attacks such as [43] indicate that physically attacking non-volatile memory is *much* easier than attacking register files or probing internal busses on recent ICs, making non-volatile memory effectively the weak link in many security implementations.

Motivated by these observations, we consider the following attacker model in this work:

Definition 1 (Non-volatile Memory Attacker). *Let $\alpha : \mathbb{N} \rightarrow \mathbb{N}$ be a function with $\alpha(n) \leq n$ for all $n \in \mathbb{N}$, and let S be a secret stored in non-volatile memory. A α -non-volatile memory attacker can access an oracle \mathcal{O} that takes as input adaptively chosen a polynomial-size circuits h_i and outputs $h_i(S)$ under the condition that the total number of bits that A gets as a result of oracle queries is at most $\alpha(|S|)$.*

The attacker is called a full non-volatile memory attacker if $\alpha = \text{id}$, that is the attacker can extract the whole content of the non-volatile memory.

Obviously, protection against full non-volatile memory attackers is only possible if no long-term secrets are stored within non-volatile memory. One obvious approach is to require a user password before each invocation. However, this reduces usability and is probably subject to password attacks. In this paper, we use another approach and make use of a physical primitive called Physically Unclonable Function (PUF). PUFs allow to *intrinsically* store permanent secrets which are, according to current state of knowledge, not accessible to a non-volatile attacker.

4 Physically Unclonable Functions

In this section, we introduce a formal model for Physically Unclonable Functions (PUFs). We start with some basic definitions. For a probability distribution \mathbb{D} , the expression $x \leftarrow \mathbb{D}$ denotes the event that x has been sampled according to \mathbb{D} . For a set S , $x \stackrel{*}{\leftarrow} S$ means that x has been sampled uniformly random from S . For $m \geq 1$, we denote by \mathbb{U}_m the uniform distribution on $\{0, 1\}^m$. The *min-entropy*

$H_\infty(\mathbb{D})$ of a distribution \mathbb{D} is defined by $H_\infty(\mathbb{D}) \stackrel{\text{def}}{=} -\log_2(\max_x \Pr[x \leftarrow \mathbb{D}])$. Min-entropy can be viewed as the “worst-case” entropy in a random variable sampled according to \mathbb{D} [11] and specifies how many nearly uniform random bits can be extracted from it.

A *distinguisher* \mathcal{D} is a (possibly probabilistic) algorithm that aims for distinguishing between two different distributions \mathbb{D} and \mathbb{D}' . More precisely, \mathcal{D} receives some values (which may depend on adaptively chosen inputs by \mathcal{D}) and outputs a value from $\{0, 1\}$. The *advantage* of \mathcal{D} is defined by $\text{Adv}(\mathcal{D}) \stackrel{\text{def}}{=} |\Pr[1 \leftarrow \mathcal{D}|\mathbb{D}] - \Pr[1 \leftarrow \mathcal{D}|\mathbb{D}']|$. Furthermore, we define the advantage of distinguishing between \mathbb{D} and \mathbb{D}' as $\max_{\mathcal{D}} \text{Adv}(\mathcal{D})$.

In a nutshell, PUFs are physical mechanisms that accept challenges and return responses, that is behaving like functions. The main properties of PUFs that are important in the context of cryptographic applications are *noise* (same challenge can lead to different (but close) responses), *non-uniform distribution* (the distribution of the responses is usually non-uniform), *independence* (two different PUFs show completely independent behavior), *unclonability* (no efficient process is known that allows for physically cloning PUFs), and *tamper evidence* (physically tampering with a PUF will most likely destroy its physical structure, making it unusable, or turn it into a new PUF). We want to emphasize that the properties above are of a physical nature and hence are very hard to prove in the rigorous mathematical sense. However, they are based on experiments conducted worldwide and reflect the current *assumptions* and observations regarding PUFs, e.g., see [47]. We first provide a formal definition for noisy functions before we give a definition for PUFs.

Definition 2 (Noisy functions). For three positive integers $\ell, m, \delta \in \mathbb{N}$ with $0 \leq \delta \leq m$, a (ℓ, m, δ) -noisy function f^* is a probabilistic algorithm which accepts inputs (challenges) $x \in \{0, 1\}^\ell$ and generates outputs (responses) $y \in \{0, 1\}^m$ such that the Hamming distance between two outputs to the same input is at most δ . In a similar manner, we define a (ℓ, m, δ) -noisy family of functions to be a set of (ℓ, m, δ) -noisy functions.

Definition 3 (Physically Unclonable Functions). A $(\ell, m, \delta; q_{\text{puf}}, \epsilon_{\text{puf}})$ -family of PUFs \mathcal{P} is a set of physical realizations of a family of probabilistic algorithms that fulfills the following algorithmic and physical properties.

Algorithmic properties

- **Noise:** \mathcal{P} is a (ℓ, m, δ) -noisy family of functions with $\delta < \frac{m}{2}$
- **Non-uniform output and independence:** There exists a distribution \mathbb{D} on $\{0, 1\}^m$ such that for any input $x \in \{0, 1\}^\ell$, the following two distributions on $(\{0, 1\}^m)^{q_{\text{puf}}}$ can be distinguished with advantage at most ϵ_{puf} .
 1. $(\Pi_1(x), \dots, \Pi_{q_{\text{puf}}}(x))$ for adaptively chosen $\Pi_i \in \mathcal{P}$.
 2. $(y_1, \dots, y_{q_{\text{puf}}})$ with $y_i \leftarrow \mathbb{D}$.
 In order to have a practically useful PUF, it should be that $q_{\text{puf}} \approx |\mathcal{P}|$, ϵ_{puf} is negligible and $H_\infty(\mathbb{D}) > 0$.

Physical properties

- **Unclonability:** No efficient technique is known to physically clone any member $\Pi \in \mathcal{P}$.
- **Tamper evidence:** For any PUF $\Pi \in \mathcal{P}$, any attempt to externally obtain its responses or parameters, e.g. by means of a physical attack, will significantly alter its functionality or destroy it.

A number of constructions for PUFs have been implemented and most of them have been experimentally verified to meet the properties of this theoretical definition. For more details we refer to the literature, e.g. [47,20,29,31,46]. One important observation we make is that a number of PUF implementations can be efficiently implemented on an integrated circuit, e.g. SRAM PUFs [20]. Their challenge-response behavior can hence be easily integrated with a chip’s digital functionality.

Remark 1. Due to their physical properties, PUFs became an interesting building block for protecting against full non-volatile memory attackers. The basic idea is to use a PUF for *implicitly* storing a secret: instead of putting a secret directly into non-volatile memory, it is derived from the PUF responses during run time [20,21].

5 Encrypting with PUFs: A Theoretical Construction

In the previous section, we explained how to use PUFs for protecting any cryptographic scheme against full non-volatile memory attackers (see Remark 1). In the remainder of the paper, we go one step further and explore how to use PUFs for protecting against algorithmic attackers *in addition*. For this purpose, we discuss how to use PUFs as a source of reproducible pseudorandomness. This approach is motivated by the observation that certain PUFs behave to some extent like unpredictable functions. This will allow for constructing (somewhat weaker) physical instantiations of (weak) pseudorandom functions.

5.1 PUF-(w)PRFs

Pseudorandom functions (PRFs) [19] are important cryptographic primitives with various applications (see, e.g., [18,32,33]). We recall their definition.

Definition 4 ((Weak) Pseudorandom Functions). Consider a family of functions \mathcal{F} with input domain $\{0, 1\}^\ell$ and output domain $\{0, 1\}^m$. We say that \mathcal{F} is $(q_{prf}, \epsilon_{prf})$ -pseudorandom in respect to a distribution \mathbb{D} on $\{0, 1\}^m$, if the advantage to distinguish between the following two distributions for adaptively chosen pairwise distinct inputs $x_1, \dots, x_{q_{prf}}$ is at most ϵ_{prf} :

- $y_i = f(x_i)$ where $f \xleftarrow{*} \mathcal{F}$
- $y_i \leftarrow \mathbb{D}$

\mathcal{F} is called weakly pseudorandom if the inputs are not chosen by the distinguisher, but uniformly random sampled from $\{0, 1\}^\ell$ (still under the condition of being pairwise distinct).

\mathcal{F} is called $(q_{prf}, \epsilon_{prf})$ -(weakly)-pseudorandom if it is $(q_{prf}, \epsilon_{prf})$ -(weakly)-pseudorandom with respect to the uniform distribution $\tilde{\mathbb{D}} = \mathbb{U}_m$.

Remark 2. This definition differs in several aspects slightly from the original definition of pseudorandom functions, e.g., [54]. First, specifying the output distribution \mathbb{D} allows for covering families of functions which have a non-uniform output distribution, e.g., PUFs. The original case, as stated in the definition, is $\mathbb{D} = \mathbb{U}_m$.

Second, the requirement of pairwise distinct inputs x_i has been introduced to deal with noisy functions where the same input can lead to different outputs. By disallowing multiple queries on the same input, we do not need to model the noise distribution, which is sometimes hard to characterize in practice. Furthermore, in the case of non-noisy (weak) pseudorandom functions, an attacker gains no advantage by querying the same input more than once. Hence, the requirement does not limit the attacker in the non-noisy case.

Observe that the “non-uniform output and independence” assumption on PUFs (as defined in Definition 3) does not automatically imply (weak) pseudorandomness. The first considers the unpredictability of the response to a specific challenge after making queries to *several different* PUFs while the latter considers the unpredictability of the response to a challenge after making queries to the same PUF.

Obviously, the main obstacle is to convert noisy non-uniform inputs into reliably reproducible, uniformly distributed random strings. For this purpose, we make use of an established tool in cryptography, i.e. *fuzzy extractors* (FE) [12]:

Definition 5 (Fuzzy Extractor). A $(m, n, \delta; \mu_{FE}, \epsilon_{FE})$ -fuzzy extractor E is a pair of randomized procedures, “generate” $\text{Gen} : \{0, 1\}^m \rightarrow \{0, 1\}^n \times \{0, 1\}^*$ and “reproduce” $\text{Rep} : \{0, 1\}^m \times \{0, 1\}^* \rightarrow \{0, 1\}^n$.

The correctness property guarantees that for $(z, \omega) \leftarrow \text{Gen}(y)$ and $y' \in \{0, 1\}^m$ with $\text{dist}(y, y') \leq \delta$, then $\text{Rep}(y', \omega) = z$. If $\text{dist}(y, y') > \delta$, then no guarantee is provided about the output of Rep .

The security property guarantees that for any distribution \mathbb{D} on $\{0, 1\}^m$ of min-entropy μ_{FE} , the string z is nearly uniform even for those who observe ω : if $(z, \omega) \leftarrow \text{Gen}(\mathbb{D})$, then it holds that $\text{SD}((z, \omega), (\mathbb{U}_n, \omega)) \leq \epsilon_{FE}$.

PUFs are most commonly used in combination with fuzzy extractor constructions based on error-correcting codes and universal hash functions. In this case, the helper data consists of a code-offset, which is of the same length as the PUF output, and the seed for the hash function, which is in the order of 100 bits and can often be reused for all outputs.

Theorem 1 (Pseudorandomness of PUF-FE-composition). Let \mathcal{P} be a $(\ell, m, \delta; q_{puf}, \epsilon_{puf})$ -family of PUFs which are $(q_{prf}, \epsilon_{prf})$ -pseudorandom with respect to some distribution \mathbb{D} . Let $E = (\text{Gen}, \text{Rep})$ be an $(m, n, \delta; H_\infty(\mathbb{D}), \epsilon_{FE})$

fuzzy extractor. The advantage of any distinguisher that adaptively chooses pairwise distinct inputs $x_1, \dots, x_{q_{prf}}$ and receives outputs $(z_1, \omega_1), \dots, (z_{q_{prf}}, \omega_{q_{prf}})$ to distinguish the following two distributions is at most $\epsilon_{prf} + q_{prf} \cdot \epsilon_{FE}$:

- $(z_i, \omega_i) = \text{Gen}(\Pi(x_i))$ where $\Pi \xleftarrow{*} \mathcal{P}$
- (z_i, ω_i) where $z_i \leftarrow \mathbb{U}_n$, $(z'_i, \omega_i) = \text{Gen}(\Pi(x_i))$ and $\Pi \xleftarrow{*} \mathcal{P}$

The analogous result holds if \mathcal{P} is $(q_{prf}, \epsilon_{prf})$ -weak-pseudorandom and if the challenges x_i are sampled uniformly random (instead of being adaptively selected), still under the condition of being pairwise distinct.

Proof. We introduce an intermediate case, named case 1', where $(z_i, \omega_i) = \text{Gen}(y_i)$ with $y_i \leftarrow \mathbb{D}$ and $\Pi \xleftarrow{*} \mathcal{P}$. Any distinguisher between case 1 and case 1' can be turned into a distinguisher that distinguishes between PUF outputs and random samples according to \mathbb{D} . Hence, the advantage is at most ϵ_{prf} by assumption. Furthermore, by the usual hybrid argument and the security property of fuzzy extractors, case 1' and case 2 can be distinguished with advantage of at most $q_{prf} \cdot \epsilon_{FE}$. \square

Definition 6 (PUF-(w)PRFs). Consider a family of (weakly)-pseudorandom PUFs \mathcal{P} and a fuzzy extractor $E = (\text{Gen}, \text{Rep})$ (where the parameters are as described in Theorem 4). A family of PUF-(w)PRFs is a set of pairs of randomized procedures, called generation and reproduction. The generation function $\text{Gen} \circ \Pi$ for some PUF $\Pi \in \mathcal{P}$ takes as input $x \in \{0, 1\}^\ell$ outputs $(z, \omega_x) \stackrel{\text{def}}{=} \text{Gen}(\Pi(x)) \in \{0, 1\}^{n \times \{0, 1\}^*}$, while the reproduction function $\text{Rep} \circ \Pi$ takes $(x, \omega_x) \in \{0, 1\}^\ell \times \{0, 1\}^*$ as input and reproduces the value $z = \text{Rep}(\Pi(x), \omega_x)$.

Theorem 4 actually shows that PUF-(w)PRFs and “traditional” (w)PRFs have in common that (part of) the output cannot be distinguished from uniformly random values. One might be tempted to plug in PUF-(w)PRFs wherever PRFs are required. Unfortunately, things are not that simple since the information saved in the helper data is also needed for correct execution. It is a known fact that the helper data of a fuzzy extractor *always* leaks some information about the input, e.g., see [23]. Hence, extra attention must be paid when deploying PUF-PRFs in cryptographic schemes. In the following section, we describe an encryption scheme that achieves real-or-random security *although* the helper data is made public.

5.2 A Luby-Rackoff Cipher Based on PUF-wPRFs

A straightforward approach for using PUF-wPRFs against full non-volatile memory attackers would be to use them for key derivation where the key is afterwards used in some encryption scheme. However, in this construction PUF-wPRFs would ensure security against non-volatile memory attackers only while the security of the encryption scheme would need to be shown separately. In the following, we present a construction that simultaneously protects against algorithmic and physical attacks while the security in both cases can be deduced to PUF-wPRF properties.

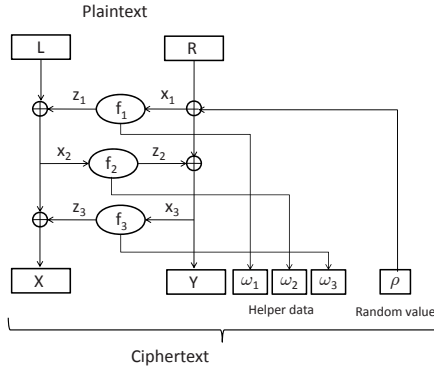


Fig. 1. A randomized 3-round Luby-Rackoff-cipher based on PUF-PRFs

One of the most important results with respect to PRFs was developed by Luby and Rackoff in [33]. They showed how to construct pseudorandom permutations from PRFs. Briefly summarized, a pseudorandom permutation (PRP) is a PRF that is a permutation as well. PRPs can be seen as an idealization of block ciphers. Consequently, the Luby-Rackoff construction is often termed as Luby-Rackoff cipher.

Unfortunately, the Luby-Rackoff result does not automatically apply to the case of PUF-PRFs. As explained in the previous section, PUF-(w)PRFs differ from (w)PRFs as they additionally need some helper data for correct execution. First, it is unclear if and how the existence and necessity of helper data would fit into the established concept of PRPs. Second, an attacker might adaptively choose plaintexts to force internal collisions and use the information leakage of the helper data for checking for these events.

Nonetheless, we can show that a Luby-Rackoff cipher based on PUF-wPRFs also yields a secure block cipher. For this purpose, we consider the set of concrete security notions for symmetric encryption schemes that has been presented and discussed in [4]. More precisely, we prove that a randomized version of a 3-round Luby-Rackoff cipher based on PUF-PRFs fulfills real-or-random indistinguishability against a chosen-plaintext attacker.

In a nutshell, a real-or-random attacker adaptively chooses plaintexts and hands them to an encryption oracle. This oracle either encrypts the received plaintexts (real case) or some random plaintexts (random case). The encryptions are given back to the attacker. Her task is to distinguish between both cases. The scheme is real-or-random indistinguishable if the advantage of winning the game is negligible (in some security parameter). Next, we first define the considered block cipher and prove its security afterwards.

Definition 7 (3-round PUF-wPRF-based Luby-Rackoff cipher). Let \mathcal{F} denote a family of PUF-wPRFs with input and output length n . The 3-round PUF-PRF-based Luby-Rackoff cipher $\mathcal{E}^{\mathcal{F}}$ uses three different PUF-wPRFs $f_i \in$

\mathcal{F} , $i = 1, 2, 3$, as round functions. The working principle is very similar to the original Luby-Rackoff cipher and is displayed in figure 7. The main differences are twofold. First, at the beginning some uniformly random value $\rho \in \{0, 1\}^\ell$ is chosen to randomize the right part R of the plaintext. Second, the round functions are PUF-wPRFs that generate two outputs: z_i and ω_i .

The ciphertext is $(X, Y, \omega_1, \omega_2, \omega_3, \rho)$. Decryption works similar to the case of the "traditional" Luby-Rackoff cipher where the helper data ω_i is used together with the Rep procedure for reconstructing the output z_i of the PUF-PRF f_i and the value ρ to "derandomize" the input to the first round function f_1 .

Since there is no digital secret stored in non-volatile memory, even a full non-volatile memory attacker has no advantage in breaking this scheme. Although this makes encrypting digital communication between two different parties impossible, various applications are imaginable, e.g., for encrypting data stored in untrusted or public storage.

Theorem 2. Let $\mathcal{E}^{\mathcal{F}}$ be the encryption scheme defined in Definition 7 using a family \mathcal{F} of PUF-wPRFs (with parameters as specified in Theorem 1). Then, the advantage of a real-or-random attacker making up to q_{prf} queries is at most $4\epsilon_{prf} + 2q_{prf} \cdot \epsilon_{FE} + 2 \cdot \frac{q_{prf}^2}{2^n}$.

Proof. Let $\{(L^{(i)}, R^{(i)})\}_{i=1, \dots, q_{prf}}$ denote the sequence of the adaptively chosen plaintexts and $x_j^{(i)}, z_j^{(i)}$ be the respective inputs and outputs to round function f_j , and $\rho^{(i)}$ the randomly chosen values. We show the claim by defining a sequence of games and estimating the advantages of distinguishing between them. Let the real game be the scenario that the distinguisher receives the encryptions of the plaintext she did choose.

In game 1, the outputs $z_1^{(i)}$ of the first round function f_1 are replaced by some uniformly random values $\tilde{z}_1^{(i)} \leftarrow^* \{0, 1\}^n$. Under the assumption that the values $x_1^{(i)}$ are pairwise distinct, the advantage to distinguish between both cases is at most $\epsilon_{prf} + q_{prf} \cdot \epsilon_{FE}$ according to Theorem 1. Furthermore, as the values $\rho^{(i)}$ are uniformly random, the probability of a collision in the values $x_1^{(i)}$ is at most $\frac{q_{prf}^2}{2^n}$. As a consequence, the advantage to distinguish between the real game and game 1 is upper bounded by $\epsilon_{prf} + q_{prf} \cdot \epsilon_{FE} + \frac{q_{prf}^2}{2^n}$.

Game 2 is defined like game 1 where now the inputs $x_1^{(i)}$ to the first round function f_1 are randomized to $\tilde{x}_1^{(i)} \leftarrow^* \{0, 1\}^n$. Observe that the values $x_1^{(i)}$ are used in two different contexts: i) for computing the right part of the ciphertext (by XORing with the output of the second round function) and ii) as input to the first round function. Regarding i), observe that the outputs of the second round function are independent of the values $x_1^{(i)}$ as the values $\tilde{z}_1^{(i)}$ (and hence the inputs to f_2) are uniformly random by definition and that the values $x_1^{(i)}$ are independent of the plaintext (because of $\rho^{(i)}$). Hence, i) and ii) represent two independent features, possibly allowing for distinguishing between game 1 and game 2, and hence can be examined separately.

The advantage of distinguishing between games 1 and 2 based on i) is equivalent to deciding whether the values $R^{(i)} \oplus \rho^{(i)} \oplus Y^{(i)}$ are uniformly random or belong to the outputs of the second round function. With the same arguments as above, the advantage is upper bounded by $\epsilon_{prf} + q_{prf} \cdot \epsilon_{FE} + \frac{q_{prf}^2}{2^n}$.

The advantage of distinguishing between game 1 and game 2 based on ii) is at most the advantage of distinguishing $(\Pi_1(x_1^{(1)}), \dots, \Pi_1(x_{q_{prf}}^{(1)}))$ from $(\Pi_1(\tilde{x}_1^{(1)}), \dots, \Pi_1(\tilde{x}_{q_{prf}}^{(1)}))$ where Π_1 denotes the PUF used in f_1 . By the definition of wPRFs (Definition 4), the advantage of distinguishing $(\Pi_1(x_1^{(1)}), \dots, \Pi_1(x_{q_{prf}}^{(1)}))$ from $(y_1, \dots, y_{q_{prf}})$ where $y_i \leftarrow \tilde{\mathbb{D}}$ and $\tilde{\mathbb{D}}$ being an appropriate distribution is at most ϵ_{prf} . Actually, the same holds for $(\Pi_1(\tilde{x}_1^{(1)}), \dots, \Pi_1(\tilde{x}_{q_{prf}}^{(1)}))$ (the fact that the values $\tilde{x}_i^{(1)}$ are unknown cannot increase the advantage). Hence, by the triangular inequality, it follows that the advantage regarding ii) is at most $2\epsilon_{prf}$. In total, the advantage to distinguish between game 1 and game 2 is less than or equal to $3\epsilon_{prf} + q_{prf} \cdot \epsilon_{FE} + \frac{q_{prf}^2}{2^n}$.

Finally, observe that it is indistinguishable whether $x_1^{(i)}$ or $R^{(i)}$ is randomized and likewise whether $z_1^{(i)}$ or $L^{(i)}$. Hence, game 2 is indistinguishable from the random game where the plaintexts are randomized. Summing up, the advantage of a real-or-random attacker is at most $4\epsilon_{prf} + 2q_{prf} \cdot \epsilon_{FE} + 2 \cdot \frac{q_{prf}^2}{2^n}$. \square

6 SRAM PRFs

In the previous section, we showed that secure cryptographic schemes are possible even if helper data is used that leaks information. In this section, we show that in the concrete case, information leakage through helper data can be avoided completely. We illustrate this approach on SRAM PUFs that were originally introduced and experimentally verified in [20]. In respect to our modeling, an SRAM PUF is a realization of a $(\ell, m, \delta; q_{puf}, \epsilon_{puf})$ -PUF that is $(2^\ell, 0)$ -pseudorandom.

We introduce a new mode of operation that, similarly to the fuzzy extractor approach in the previous section, allows for extracting uniform values from SRAM PUFs in a reproducible way. This approach likewise stores some additional helper data but, as opposed to the case of fuzzy extractors, the helper data does not leak any information on the input. Hence, this construction might be of independent interest for SRAM PUF based applications. The proposed construction is based on two techniques: Temporal Majority Voting and Excluding Dark Bits.

We denote the individual bits of a PUF response as $y = (y_0, \dots, y_{m-1})$, with $y_i \in \{0, 1\}$. When performing a response measurement on a PUF Π , every bit y_i of the response is determined by a Bernoulli trial. Every y_i has a most likely value $y_i^{(ML)} \in \{0, 1\}$, and a certain probability $p_i < 1/2$ of differing from this value which we define as its *bit error probability*. We denote $y_i^{(k)}$ as the k -th measurement or sample of bit y_i in a number of repeated measurements.

Definition 8 (Temporal Majority Voting (TMV)). Consider a Bernoulli distributed random bit y_i over $\{0, 1\}$. We define temporal majority voting of y_i

over N votes, with N an odd positive integer, as a function $TMV_N : \{0, 1\}^N \rightarrow \{0, 1\}$, that takes as input N different samples of y_i and outputs the most often occurring value in these samples.

We can calculate the error probability $p_{N,i}$ of bit y_i after TMV with N votes as:

$$p_{N,i} \stackrel{\text{def}}{=} Pr \left[TMV_N \left(y_i^{(0)}, \dots, y_i^{(N-1)} \right) \neq y_i^{(ML)} \right] = 1 - \text{Bin}_{N,p_i} \left(\frac{N-1}{2} \right) \leq p_i, \tag{1}$$

with Bin_{N,p_i} the cumulative distribution function of the binomial distribution. From Eq. (1) it follows that applying TMV to a bit of a PUF response effectively reduces the error probability from p_i to $p_{N,i}$, with $p_{N,i}$ becoming smaller as N increases. We can determine the number of votes N we need to reach a certain threshold p_T such that $p_{N,i} \leq p_T$, given an initial error probability p_i . It turns out that N rises exponentially as p_i gets close to $1/2$. In practice, we also have to put a limit N_T on the number of votes we can perform, since each vote involves a PUF response measurement. We call the pair (N_T, p_T) a *TMV-threshold*.

Definition 9 (Dark Bit (DB)). Let (N_T, p_T) be a TMV-threshold. We define a bit y_i to be dark with respect to this threshold if $p_{N_T,i} > p_T$.

TMV alone cannot decrease the bit error probability to acceptable levels (*e.g.* $\leq 10^{-9}$) because of the non-negligible occurrence of dark bits. We use a *bit mask* γ to identify these dark bits in the generation phase, and exclude them during reproduction. Similar to fuzzy extractors, (N_T, p_T) -TMV and DB can be used for generating and reproducing uniform values from SRAM PUFs.

The **Gen**-procedure takes sufficient measurements of every response bit y_i to make an accurate estimate of its most likely value $y_i^{(ML)}$ and of its error probability p_i . If y_i is dark with respect to (N_T, p_T) , then the corresponding bit γ_i in the bit mask $\gamma \in \{0, 1\}^m$ is set to 0 and y_i is discarded, otherwise γ_i is set to 1 and y_i is appended to the bit string s . The procedure **Gen** outputs a helper string $\omega = (\gamma, \sigma)$ and an extracted string $z = \text{Extract}_\sigma(s)$, with Extract_σ a classical strong extractor [37] with seed σ .

The **Rep**-procedure takes N_T measurements of a response y' and the corresponding helper string $\omega = (\gamma, \sigma)$, with $\gamma \in \{0, 1\}^m$ as input. If γ_i contains a 1, then the result of $TMV_{N_T} \left(y_i'^{(0)}, \dots, y_i'^{(N_T-1)} \right)$ is appended to a bit string s' , otherwise, y_i' is discarded. **Rep** outputs an extracted string $z' = \text{Extract}_\sigma(s')$.

A strong extractor [37] is a function that is able to generate nearly-uniform outputs from inputs coming from a distribution with limited min-entropy. It ensures that the statistical distance of the extracted output to the uniform distribution is negligible. The required compression rate of Extract_σ depends on the remaining min-entropy μ of the PUF response y after the helper data is observed. We call the above construction a **TMV-DB-SRAM-PUF**.

¹ See *e.g.* [37][12] for a definition of a strong extractor. Typical seed lengths of strong extractors are in the order of 100 bits, and in most cases the same seed can be reused for all outputs.

Using analogous arguments as in Theorem 11, one can show that the output of a TMV-DB-SRAM-PUF is indistinguishable from random except with negligible advantage. Additionally, in an SRAM PUF, the most likely value of a bit is independent of whether or not the bit is a dark bit, hence no min-entropy on the PUF output is leaked by the bit mask 12. However, by searching for matching helper strings, an adversary might still be able to find colliding TMV-DB-SRAM-PUF inputs (especially as the input size is small), which can impose a possible security leak. In order to overcome this issue, we present the following way of using a TMV-DB-SRAM-PUF:

Definition 10 (All-at-once mode). Consider a TMV-DB-SRAM-PUF as described above. We define the all-at-once mode of operation to be the pair of procedures (Enroll, Eval).

The enrollment procedure *Enroll* outputs a helper table $\Omega \in \{0, 1\}^{2^\ell \times *}$ when executed. The helper table is constructed by running $\forall x \in \{0, 1\}^\ell$ the generation function $(\text{Gen} \circ \Pi)(x)$, and storing the obtained helper data ω_x as the x -th element in Ω , i.e. $\Omega[x] := \omega_x$.

The evaluation function *Eval*: $\{0, 1\}^\ell \times \{0, 1\}^{2^\ell \times *} \rightarrow \{0, 1\}^n$ takes an element $x \in \{0, 1\}^\ell$ and a helper table $\Omega \in \{0, 1\}^{2^\ell \times *}$ as inputs and (after internal computation) outputs a value $\text{Eval}(x, \Omega) = z \in \{0, 1\}^n$, with $z = (\text{Rep} \circ \Pi)(x, \Omega[x])$.

The *Enroll*-procedure has to be executed before the *Eval*-procedure, but it has to be run only once for every PUF. Every invocation of *Eval* can take the same (public) helper table Ω as one of its inputs. However, in order to conceal exactly which helper string is used, it is important that the *Eval*-procedure takes Ω as a whole as input, and does not just do a look-up of $\Omega[x]$ in a public table Ω . The all-at-once mode prevents an adversary from learning which particular helper string is used during the *internal* computation.

Definition 11 (SRAM-PRF). An SRAM-PRF is a TMV-DB-SRAM-PUF that runs in the all-at-once mode.

Using the arguments given above we argue that SRAM-PRFs are in all practical views a physical realization of PRFs. Observe that one major drawback of SRAM-PRFs is that the hardware size grows exponentially with the input length. Thus, SRAM-PRFs cannot be used as a concrete instantiation of PUF-PRFs for our construction from Section 5.2. This section rather shows up an alternative approach for constructing cryptographic mechanisms based on PUFs despite of the noise problem. As a possible application of SRAM-PRFs, we discuss an expanding Luby-Rackoff cipher where the round functions are replaced by SRAM-PRFs that take 8-bit challenges as input and produce 120-bit extracted outputs. According to [38], at least 48 rounds are necessary for security reasons.

As an instantiation for the PUF, we take an SRAM PUF with an assumed average bit error probability of 15% and an estimated min-entropy content of 0.95 bit/cell. We use TMV-threshold of $(N_T = 99, p_T = 10^{-9})$. Simulations and

² By consequence, also no min-entropy on the PUF *input* is leaked.

experiments on the SRAM PUF show that about 30% of the SRAM cells produce a dark bit with respect to this TMV-threshold. The strong extractor only has to compress by a factor of $\frac{1}{0.95}$, accounting for the limited min-entropy in the PUF response. Hence, $\frac{1}{0.95} \cdot \frac{2^8 \cdot 120}{70\%} \text{ bits} = 5.6 \text{ kbyte}$ of SRAM cells is needed to build one SRAM-PRF. Thus, the entire block cipher uses $48 \cdot 5.6 \text{ kbyte} \approx 271 \text{ kbyte}$ of SRAM cells. The helper tables also require 5.6 *kbyte* each.

Implementing 48 SRAM PUFs using a total of 271 *kbyte* of SRAM cells is feasible on recent ICs, and 48 rounds can be evaluated relatively fast. Storing and loading 48 helper tables of 5.6 *kbyte* each is also achievable in practice. Observe that the size depends linearly on the number of rounds. The according parameters for more rounds can be easily derived. Reducing the input size of the SRAM-PRF will yield an even smaller amount of needed SRAM cells and smaller helper tables, but the number of rounds will increase. A time-area trade-off is hence possible.

7 Conclusions

In this paper we propose a leakage-resilient encryption scheme that makes use of Physically Unclonable Functions (PUFs). The core component is a new PUF-based cryptographic primitive, termed PUF-PRF, that is similar to a pseudo-random function (PRF). We showed that PUF-PRFs possess cryptographically useful algorithmic and physical properties that come from the random character of their physical structures.

Of course, any physical model can only approximately describe real life. Although experiments support our model for the considered PUF implementations, more analysis is necessary. In this context it would be interesting to consider other types of PUFs which fit into our model or might be used for other cryptographic applications. Furthermore, a natural continuation of this work would be to explore other cryptographic schemes based on PUF-PRFs, e.g., hash functions or public key encryption.

Acknowledgements

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. The work of Berk Sunar was supported by the National Science Foundation Cybertrust grant No. CNS-0831416. The work of Roel Maes is funded by IWT-Flanders grant No. 71369 and is in part supported by the IAP Program P6/26 BCRYPT of the Belgian State and K.U.Leuven BOF funding (OT/06/04).

References

1. Agrawal, D., Archambeault, B., Rao, J.R., Rohatgi, P.: The em side-channel(s). In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 29–45. Springer, Heidelberg (2003)

2. Akavia, A., Goldwasser, S., Vaikuntanathan, V.: Simultaneous hardcore bits and cryptography against memory attacks. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 474–495. Springer, Heidelberg (2009)
3. Anderson, R.J., Kuhn, M.G.: Low cost attacks on tamper resistant devices. In: Proceedings of the 5th International Workshop on Security Protocols, London, UK, pp. 125–136. Springer, Heidelberg (1998)
4. Bellare, M., Desai, A., Jokipii, E., Rogaway, P.: A concrete security treatment of symmetric encryption. In: FOCS 1997: Proceedings of the 38th Annual Symposium on Foundations of Computer Science (FOCS 1997), Washington, DC, USA, p. 394. IEEE Computer Society, Los Alamitos (1997)
5. Bellare, M., Kilian, J., Rogaway, P.: The security of cipher block chaining. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 341–358. Springer, Heidelberg (1994)
6. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated key exchange secure against dictionary attacks. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 139–155. Springer, Heidelberg (2000)
7. 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)
8. Bellare, M., Rogaway, P.: Provably secure session key distribution: the three party case. In: STOC 1995: Proceedings of the twenty-seventh annual ACM symposium on Theory of computing, pp. 57–66. ACM, New York (1995)
9. 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)
10. Chandran, N., Goyal, V., Sahai, A.: New constructions for UC secure computation using tamper-proof hardware. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 545–562. Springer, Heidelberg (2008)
11. Chor, B., Goldreich, O.: Unbiased bits from sources of weak randomness and probabilistic communication complexity. *SIAM J. Comput.* 17(2), 230–261 (1988)
12. Dodis, Y., Ostrovsky, R., Reyzin, L., Smith, A.: Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.* 38(1), 97–139 (2008)
13. Dziembowski, S., Pietrzak, K.: Leakage-resilient cryptography. In: FOCS '08: Proceedings of the 2008 49th Annual IEEE Symposium on Foundations of Computer Science, Washington, DC, USA, pp. 293–302. IEEE Computer Society, Los Alamitos (2008)
14. Gassend, B., Clarke, D., van Dijk, M., Devadas, S.: Controlled Physical Random Functions. In: Annual Computer Security Applications Conference — ACSAC 2002, Washington, DC, USA, p. 149. IEEE Computer Society, Los Alamitos (2002)
15. Gassend, B., Clarke, D.E., van Dijk, M., Devadas, S.: Silicon physical unknown functions. In: Atluri, V. (ed.) ACM Conference on Computer and Communications Security — CCS 2002, pp. 148–160. ACM, New York (2002)
16. Gaubatz, G., Sunar, B., Karpovsky, M.G.: Non-linear residue codes for robust public-key arithmetic. In: Breveglieri, L., Koren, I., Naccache, D., Seifert, J.-P. (eds.) FDTC 2006. LNCS, vol. 4236, pp. 173–184. Springer, Heidelberg (2006)
17. Gennaro, R., Lysyanskaya, A., Malkin, T., Micali, S., Rabin, T.: Algorithmic tamper-proof (atp) security: Theoretical foundations for security against hardware tampering. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 258–277. Springer, Heidelberg (2004)

18. Goldreich, O., Goldwasser, S., Micali, S.: On the cryptographic applications of random functions. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 276–288. Springer, Heidelberg (1985)
19. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. *J. ACM* 33(4), 792–807 (1986)
20. Guajardo, J., Kumar, S.S., Schrijen, G.-J., Tuyls, P.: FPGA Intrinsic PUFs and Their Use for IP Protection. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 63–80. Springer, Heidelberg (2007)
21. Guajardo, J., Kumar, S.S., Schrijen, G.-J., Tuyls, P.: Physical Unclonable Functions and Public Key Crypto for FPGA IP Protection. In: International Conference on Field Programmable Logic and Applications — FPL 2007, August 27–30, pp. 189–195. IEEE, Los Alamitos (2007)
22. 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. In: van Oorschot, P.C. (ed.) USENIX Security Symposium, pp. 45–60. USENIX Association (2008)
23. Ignatenko, T., Willems, F.: On the security of the XOR-method in biometric authentication systems. In: Twenty-seventh symposium on Information Theory in the Benelux, pp. 197–204 (2006)
24. 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)
25. Karpovsky, M., Kulikowski, K., Taubin, A.: Robust protection against fault-injection attacks on smart cards implementing the advanced encryption standard. In: Proc. Int. Conference on Dependable Systems and Networks (DNS 2004) (July 2004)
26. Katz, J.: Universally composable multi-party computation using tamper-proof hardware. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 115–128. Springer, Heidelberg (2007)
27. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
28. 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)
29. Kumar, S.S., Guajardo, J., Maes, R., Schrijen, G.-J., Tuyls, P.: The Butterfly PUF: Protecting IP on every FPGA. In: IEEE International Workshop on Hardware-Oriented Security and Trust – HOST 2008, June 9. IEEE, Los Alamitos (2008)
30. Lemke, K.: Embedded security: Physical protection against tampering attacks. In: Lemke, C.P.K., Wolf, M. (eds.) Embedded Security in Cars, ch. 2, pp. 207–217. Springer, Heidelberg (2006)
31. Lim, D., Lee, J.W., Gassend, B., Suh, G.E., van Dijk, M., Devadas, S.: Extracting secret keys from integrated circuits. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 13(10), 1200–1205 (2005)
32. Luby, M.: Pseudo-randomness and applications. Princeton University Press, Princeton (1996)
33. Luby, M., Rackoff, C.: How to construct pseudorandom permutations from pseudorandom functions. *SIAM J. Comput.* 17(2), 373–386 (1988)
34. Micali, S., Reyzin, L.: Physically observable cryptography (extended abstract). In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 278–296. Springer, Heidelberg (2004)

35. Moran, T., Segev, G.: David and goliath commitments: UC computation for asymmetric parties using tamper-proof hardware. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 527–544. Springer, Heidelberg (2008)
36. 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)
37. Nisan, N., Zuckerman, D.: More deterministic simulation in logspace. In: STOC 1993: Proceedings of the twenty-fifth annual ACM symposium on Theory of computing, pp. 235–244. ACM, New York (1993)
38. Patarin, J., Nachev, 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)
39. Pietrzak, K.: A leakage-resilient mode of operation. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 462–482. Springer, Heidelberg (2009)
40. Posch, R.: Protecting devices by active coating. *Journal of Universal Computer Science* 4, 652–668 (1998)
41. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Proceedings of the 37th Annual ACM Symposium on Theory of Computing, pp. 84–93 (2005)
42. Samyde, D., Skorobogatov, S., Anderson, R., Quisquater, J.-J.: On a new way to read data from memory. In: SISW 2002: Proceedings of the First International IEEE Security in Storage Workshop, Washington, DC, USA, p. 65. IEEE Computer Society, Los Alamitos (2002)
43. Skorobogatov, S.P.: Data remanence in flash memory devices. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 339–353. Springer, Heidelberg (2005)
44. Smith, S.W.: Fairy dust, secrets, and the real world [computer security]. *IEEE Security and Privacy* 1(1), 89–93 (2003)
45. Standaert, F.-X., Malkin, T., Yung, M.: A unified framework for the analysis of side-channel key recovery attacks. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 134–152. Springer, Heidelberg (2009)
46. Edward Suh, G., Devadas, S.: Physical Unclonable Functions for Device Authentication and Secret Key Generation. In: Proceedings of the 44th Design Automation Conference, DAC 2007, San Diego, CA, USA, June 4–8, pp. 9–14. ACM, New York (2007)
47. Tuyls, P., Schrijen, G.-J., Škorić, B., van Geloven, J., Verhaegh, N., Wolters, R.: Read-proof hardware from protective coatings. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 369–383. Springer, Heidelberg (2006)
48. Verbauwheide, I., Schaumont, P.: Design methods for security and trust. In: Proc. of Design Automation and Test in Europe (DATE 2008), NICE, FR, p. 6 (2007)
49. Weingart, S.H.: Physical security devices for computer subsystems: A survey of attacks and defences. In: Paar, C., Koç, Ç.K. (eds.) CHES 2000. LNCS, vol. 1965, pp. 302–317. Springer, Heidelberg (2000)

Signature Schemes with Bounded Leakage Resilience

Jonathan Katz^{1,*} and Vinod Vaikuntanathan²

¹ University of Maryland

`jkatz@cs.umd.edu`

² IBM Research

`vinodv@alum.mit.edu`

Abstract. A *leakage-resilient* cryptosystem remains secure even if arbitrary, but bounded, information about the secret key (and possibly other internal state information) is leaked to an adversary. Denote the length of the secret key by n . We show:

- A full-fledged signature scheme tolerating leakage of $n - n^\epsilon$ bits of information about the secret key (for any constant $\epsilon > 0$), based on general assumptions.
- A one-time signature scheme, based on the minimal assumption of one-way functions, tolerating leakage of $(\frac{1}{4} - \epsilon) \cdot n$ bits of information about the signer’s entire state.
- A more efficient one-time signature scheme, that can be based on several specific assumptions, tolerating leakage of $(\frac{1}{2} - \epsilon) \cdot n$ bits of information about the signer’s entire state.

The latter two constructions extend to give leakage-resilient t -time signature schemes. All the above constructions are in the standard model.

1 Introduction

Proofs of security for cryptographic primitives traditionally treat the primitive as a “black box” that an adversary is able to access in a relatively limited fashion. For example, in the usual model for proving security of signature schemes, an adversary is given the public key and allowed to request signatures on any messages of its choice, but is unable to get any *other* information about the secret key or any internal randomness or state information used during signature generation.

In real-world implementations of cryptographic primitives, on the other hand, an adversary may be able to recover a significant amount of additional information not captured by standard security models. Examples include information leaked by side-channel cryptanalysis [20,21], fault attacks [5,3], or timing attacks [4], or even bits of the secret key itself in case this key is improperly stored

* Work done while visiting IBM, and supported in part by NSF grants #0627306 and #0716651.

or erased [17]. Potentially, schemes can also be attacked when they are implemented using poor random number generation [28] (which can be viewed as giving the adversary additional information on the internal state, beyond what would be available if the output were truly random), or when the same key is used in multiple contexts (e.g., for decryption and signing).

In the past few years, cryptographers have made tremendous progress toward modeling security in the face of such information leakage [25,35], and in constructing *leakage-resilient* cryptosystems secure even in case such leakage occurs. (There has also been corresponding work on reducing unwanted leakage by, e.g., building tamper-proof hardware; this is not the focus of our work.) Most relevant to the current work is a recent series of results [11,13,19,10,26,2] showing cryptosystems that guarantee security even when *arbitrary* information about the secret key is leaked (under suitable restrictions); we discuss this work, along with other related results, in further detail below. This prior work gives constructions of stream ciphers [11,31] (and hence stateful symmetric-key encryption and MACs), symmetric-key encryption schemes [9], public-key encryption schemes [11,10,26], and signature schemes [2] achieving various notions of leakage resilience.

Most prior work has focused on primitives for ensuring *secrecy*. The only work of which we are aware that deals with *authenticity* is that of Alwen et al. [2] which shows, among other results, leakage-resilient signature schemes based on number-theoretic assumptions in the random oracle model.¹ Here we give constructions of leakage-resilient signature schemes based on *general assumptions* in the *standard model*; our main construction also tolerates more leakage than the schemes of [2]. (In the full version we also show some technical improvements to the results of [2].) We postpone a more thorough discussion of our results until after we define leakage resilience in more detail.

1.1 Modeling Leakage Resilience

At a high level, definitions of leakage resilience take the following form: Begin with a “standard” security notion (e.g., existential unforgeability under adaptive chosen message attacks [15]) and modify this definition by allowing the adversary to (adaptively) specify a series of *leakage functions* f_1, \dots . The adversary, in addition to getting whatever other information is specified by the original security definition, is given the result of applying f_i to the secret key and possibly other internal state of the honest party (e.g., the signer). We then require that the adversary’s success probability — for signature schemes, the probability with which it can output a forged signature — remain negligible. It should be clear that this is a general methodology that can be applied to many different primitives. The exact model is then determined by the restrictions placed on the leakage function(s) f_i :

Limited vs. arbitrary information. A first consideration regards whether the $\{f_i\}$ can be arbitrary (polynomial-time computable) functions, or whether

¹ The results of [2] were obtained independently of our own work.

they are restricted to be in some more limited class. Early work considered the latter case, for example where the adversary is restricted to learning *specific bits* of the secret key [6], or the values on *specific wires* of the circuit implementing the primitive [19]. More recent work [11,13,9,10,26,2] allows arbitrary $\{f_i\}$.

Bounded vs. unbounded information leakage. Let n denote the length of the secret key. If the secret key does not change over time, and the $\{f_i\}$ are allowed to be arbitrary, then security in the traditional sense cannot be achieved once the total length of the *leakage* — that is, the outputs of all the $\{f_i\}$ — is n bits or more. For the case of signatures, the length of the leakage must also be less than the signature length. This inherent restriction is used in [10,26]. (Alwen et al. [2] do not impose this restriction, but as a consequence can only achieve a weaker notion of security.)

One can avoid this restriction, and potentially tolerate an unbounded amount of leakage overall, if the secret key is updated over time; even in this case, one must somehow limit the amount of leakage between successive key updates. This approach to leakage resilience was considered in [11,31] in the context of stateful symmetric-key primitives, and [12] in the context of stateful signature schemes.

One can also avoid imposing a bound on the leakage by restricting the $\{f_i\}$, as discussed next.

Computational min-entropy of the secret key. If the leakage is much shorter than the secret key (as discussed above), then the secret key will have high min-entropy conditioned on the leakage. This setting is considered in [1,26,10,2], and is also enforced on a per-period basis in the work of [11,31] (i.e., the leakage per time period is required to be shorter than the secret key). More recent work [9,10] shows schemes that remain secure for leakage of arbitrary length, as long as the secret key remains exponentially hard to compute given the leakage (but even if the secret key is fully determined by the leakage in an information-theoretic sense). A drawback of this guarantee is that given some collection of functions $\{f_i\}$ (say, as determined experimentally for some particular set of side-channel attacks) there is no way to tell, in general, whether they satisfy the stated requirement or not. Furthermore, existing results in this direction currently require super-polynomial hardness assumptions.

Inputs to the leakage functions. A final issue is the allowed inputs to the leakage functions. Work of [11,31] assumes, following [25], that *only computation leaks information*; this is modeled by letting each f_i take as input only those portions of the secret key that are accessed during the i th phase of the scheme. Halderman et al. [17], however, show that memory contents can be leaked even when they are not being accessed. Motivated (in part) by this result, the schemes of [1,9,10,26,2] allow the $\{f_i\}$ to take the entire secret key as input at all times.

For the specific primitives considered in [11,13,9,10,26], the secret key sk is the only internal state maintained by the party holding the secret key, and

so allowing the $\{f_i\}$ to depend on sk is (almost) the most general choice.² For signature schemes, however, any randomness used during signing might also be leaked to an adversary. The strongest definition of leakage resilience is thus obtained by allowing the $\{f_i\}$ to depend on *all* the state information used by the honest signer during the course of the experiment.

All these variants may be meaningful depending on the particular attacks one is trying to model. Memory attacks [17,1], which probe long-term secret information during a time when computation is *not* taking place, can be faithfully modeled by allowing the leakage functions to take only sk as input. On the other hand, side-channel attacks that collect information while computation is occurring might be more accurately captured by allowing the leakage functions to take as input only those portions of the internal state that are being accessed.

1.2 Our Results

With the preceding discussion in mind, we can now describe our results in further detail. In all cases, we allow the leakage function(s) to be *arbitrary* as long as the total leakage is *bounded* as some function of the secret key length n ; recall that such a restriction on the leakage is essential if the secret key is unchanging, as it is in all our schemes. Our results can be summarized as follows:

1. We show a construction of a leakage-resilient signature scheme that is existentially unforgeable against chosen-message attacks in the standard model, based on general (as opposed to number-theoretic) assumptions. This scheme tolerates leakage of $n - n^\epsilon$ bits of information about the secret key for any $\epsilon > 0$ based on polynomial hardness assumptions, and can tolerate (optimal) $n - \omega(\log n)$ bits of leakage based on sub-exponential hardness assumptions.
2. We also construct two leakage-resilient *one-time* (resp., *t-time*) signature schemes in the standard model. These schemes are more efficient than the scheme above; they also tolerate leakage that may depend on the entire state of the signer (rather than just the secret key).
 - Our first scheme is based on the minimal assumption that one-way functions exist, and tolerates leakage of $(\frac{1}{4} - \epsilon) \cdot n$ bits for any $\epsilon > 0$. The construction extends to give a *t-time* signature scheme tolerating leakage of $\Theta(n/t)$ bits.
 - Our second scheme, which can be based on various concrete assumptions, is more efficient and tolerates leakage of up to $(\frac{1}{2} - \epsilon) \cdot n$ bits for any $\epsilon > 0$. This construction also extends to give a *t-time* signature scheme tolerating leakage of $\Theta(n/t)$ bits.

In the full version of this work, we also discuss efficient constructions of full-fledged signature schemes based on number-theoretic assumptions (in the random oracle model) that are secure as long as the leakage is bounded by $(\frac{1}{2} - \epsilon) \cdot n$

² More generally, one could also allow the $\{f_i\}$ to depend on the *randomness* used to generate the (public and) secret key(s); this possibility is mentioned in [26, Section 8.2]. (For the specific schemes considered in [11,10,31,9,10,26], however, this makes no substantive difference.)

bits for any $\epsilon > 0$. Similar schemes were discovered independently by Alwen et al. [2], but our analysis offers some advantages as compared to theirs. Specifically, we make explicit the fact that the leakage can depend on the entire state of the signer, and we allow leakage queries to depend on the random oracle.

Independent of our work, Faust et al. [12] describe a transformation from any 3-time signature scheme tolerating $\alpha(n)$ bits of leakage to a full-fledged (but stateful) signature scheme where the secret key is updated over time; the resulting scheme tolerates $\alpha(n)$ bits of leakage *between key updates*, and unbounded leakage overall. (In the transformed signature scheme, security is ensured as long as the leakage depends only on the *active portion* of the secret-key.) Applying this transformation to our constructions, we get *full-fledged* signature schemes that tolerate *unbounded leakage* (subject to the restrictions mentioned above).

1.3 Overview of Our Techniques

Our constructions all rely on the same basic idea. Roughly, we consider signature schemes with the following properties:

- A given public key pk corresponds to a set S_{pk} of *exponentially many* secret keys. Furthermore, given (sk, pk) with $sk \in S_{pk}$ it remains hard to compute any other $sk' \in S_{pk}$.
- The secret key sk used by the signer has high min-entropy (at least in a computational sense) even for an adversary who observes signatures on messages of its choice. (For our one-time scheme, this is only required to hold for an adversary who observes a single signature.)
- A signature forgery can be used to compute a secret key in S_{pk} .

To prove that any such signature scheme is leakage resilient, we show how to use an adversary \mathcal{A} attacking the scheme to find distinct $sk, sk' \in S_{pk}$ given (sk, pk) (in violation of the assumed hardness of doing so). Given (sk, pk) , we simply run \mathcal{A} on input pk and respond to its signing queries using the given key sk . Leakage queries can also be answered using sk . If the adversary forges a signature, we extract some $sk' \in S_{pk}$; it remains only to show that $sk' \neq sk$ with high probability. Let $n = \log |S_{pk}|$ be the (computational) min-entropy of sk conditioned on pk and the signatures seen by the adversary. (We assume that all secret keys in S_{pk} are equally likely, which will be the case in our constructions.) A standard argument (cf. Lemma 1) shows that if the leakage is bounded by ℓ bits, then the conditional min-entropy of the secret key is still at least $n - \ell - t$ bits except with probability 2^{-t} . So as long as the leakage is bounded away from n , with high probability the min-entropy of sk conditioned on \mathcal{A} 's entire view is still at least 1. But then $sk' \neq sk$ with probability at least $1/2$. This concludes the outline of the proof. We remark, however, that various subtleties arise in the formal proofs of security.

Some existing signature schemes in the random oracle model already satisfy the requirements stated above. In particular, these include schemes constructed using the Fiat-Shamir transform [13] applied to a witness-indistinguishable Σ -protocol where there are an *exponential* number of witnesses for to a given

statement. Concrete examples include the signature schemes of Okamoto [29] (extending the Schnorr [34] and Guillou-Quisquater [16] schemes) based on the discrete logarithm or RSA assumptions, as well as the signature scheme of Fischlin and Fischlin [14] (extending the Ong-Schnorr [30] scheme) based on the hardness of factoring. This class of schemes was also considered by Alwen et al. [2]. See the full version of our paper for further discussion.

We are not aware of any existing signature scheme in the standard model that meets our requirements. We construct one as follows. Let H be a universal one-way hash function (UOWHF) [27] mapping n -bit inputs to n^ϵ -bit outputs. The secret key of the signature scheme is $x \in \{0,1\}^n$, and the public key is $(y = H(x), pk, r)$ where pk is a public key for a CPA-secure public-key encryption scheme, and r is a common reference string for an unbounded simulation-sound NIZK proof system [33,8]. A signature on a message m consists of an encryption $C \leftarrow \text{Enc}_{pk}(m||x)$ of both m and x , along with a proof π that C is an encryption of $m||x'$ with $H(x') = y$. Observe that, with high probability over choice of x , there are exponentially many pre-images of $y = H(x)$ and hence exponentially many valid secret keys; furthermore, finding another such secret key $sk' \neq sk$ requires finding a collision in H . Details are given in Section 3.

Our leakage-resilient one-time signature schemes are constructed using a similar idea. The first construction is inspired by the Lamport signature scheme [23]. The secret key is $\{(x_{i,0}, x_{i,1})\}_{i=1}^k$ and the public key is $\{(y_{i,0}, y_{i,1})\}_{i=1}^k$ where $y_{i,b} = H(x_{i,b})$ for H a UOWHF. Once again, there are exponentially many secret keys associated with any public key and finding any two such keys yields a collision in H . Adapting the Lamport scheme, so that the signature on a message $m = m_1 \cdots m_k$ is $\{x_{i,m_i}\}_{i=1}^k$, yields a signature scheme secure against leakage of $n^{1-\epsilon}$ bits. By first encoding the message using an error-correcting code with high minimum distance, it is possible to “boost” the leakage resilience to $(\frac{1}{4} - \epsilon) \cdot n$ bits. Using cover-free families this approach extends also to give a leakage-resilient t -time signature scheme. These constructions are all described in Section 4.

Our second construction builds on ideas that can be traced back to [7,24]. Roughly, let (G, \oplus) and (G', \otimes) be groups with $\log |G'| \leq \epsilon \cdot \log |G|$, and let $\mathcal{H} = \{H_s : G \rightarrow G'\}$ be a family of collision-resistant hash functions that are also *homomorphic* (i.e., for which $H_s(a) \otimes H_s(b) = H_s(a \oplus b)$); such hash functions can be constructed based on a variety of concrete assumptions (see Section 4.3). The secret key is a pair of elements $a, b \in G$, and the public-key is $(s, H_s(a), H_s(b))$ for a random key s . Note, there are exponentially many secret keys associated with any public key and finding any two such secret keys yields a collision in H_s . The signature on a message $m \in \{1, \dots, \text{ord}(G)\}$ is simply $\sigma = a \oplus mb$, which can be verified by checking that $H_s(\sigma) \stackrel{?}{=} H_s(a) \otimes m H_s(b)$. The important property for our purposes is that given a single signature $a \oplus mb$, the secret key (a, b) still has high min-entropy. So if the adversary forges another signature σ' for a message $m' \neq m$, with high probability it holds that $\sigma' \neq a \oplus m'b$ and we obtain a collision in H_s .

2 Definitions and Preliminaries

We provide a formal definition of leakage resilience for signature schemes, and state a technical lemma that will be used in our analysis. We denote the security parameter by k , and let PPT stand for “probabilistic polynomial time”.

Definition 1. A signature scheme is a tuple of PPT algorithms $(\text{Gen}, \text{Sign}, \text{Vrfy})$ such that:

- Gen is a randomized algorithm that takes as input 1^k and outputs (pk, sk) , where pk is the public key and sk is the secret key.
- Sign is a (possibly) randomized algorithm that takes as input the secret key sk , the public key pk , and a message m , and outputs a signature σ . We denote this by $\sigma \leftarrow \text{Sign}_{sk}(m)$, leaving the public key implicit³.
- Vrfy is a deterministic algorithm that takes as input a public key pk , a message m , and a purported signature σ . It outputs a bit b indicating acceptance or rejection, and we write this as $b := \text{Vrfy}_{pk}(m, \sigma)$.

It is required that for all k , all (pk, sk) output by $\text{Gen}(1^k)$, and all messages m in the message space, we have $\text{Vrfy}_{pk}(m, \text{Sign}_{sk}(m)) = 1$.

Our definition of leakage resilience is the standard notion of existential unforgeability under adaptive chosen-message attacks [15], except that we additionally allow the adversary to specify arbitrary leakage functions $\{f_i\}$ and obtain the value of these functions applied to the secret key (and possibly other state information).

Definition 2. Let $\Pi = (\text{Gen}, \text{Sign}, \text{Vrfy})$ be a signature scheme, and let λ be a function. Given an adversary \mathcal{A} , define the following experiment parameterized by k :

1. Choose $r \leftarrow \{0, 1\}^*$ and compute $(pk, sk) := \text{Gen}(1^k; r)$. Set $\text{state} := \{r\}$.
2. Run $\mathcal{A}(1^k, pk)$. The adversary may then adaptively access a signing oracle $\text{Sign}_{sk}(\cdot)$ and a leakage oracle $\text{Leak}(\cdot)$ that have the following functionality:
 - In response to the i th query $\text{Sign}_{sk}(m_i)$, this oracle chooses random $r_i \leftarrow \{0, 1\}^*$, computes $\sigma_i := \text{Sign}_{sk}(m_i; r_i)$, and returns σ_i to \mathcal{A} . It also sets $\text{state} := \text{state} \cup \{r_i\}$.
 - In response to the i th query $\text{Leak}(f_i)$ (where f_i is specified as a circuit), this oracle gives $f_i(\text{state})$ to \mathcal{A} . (To make the definition meaningful in the random oracle model, the $\{f_i\}$ are allowed to be oracle circuits that depend on the random oracle H .)
The $\{f_i\}$ can be arbitrary, subject to the restriction that the total output length of all the f_i is at most $\lambda(|sk|)$.
3. At some point, \mathcal{A} outputs (m, σ) .

³ Usually one assumes without loss of generality that the public key is included as part of the secret key. Since we measure leakage as a function of the secret-key length, however, we seek to minimize the size of the secret key.

\mathcal{A} succeeds if (1) $\text{Vrfy}_{pk}(m, \sigma) = 1$ and (2) m was not previously queried to the $\text{Sign}_{sk}(\cdot)$ oracle. We denote the probability of this event by $\Pr[\text{Succ}_{\mathcal{A}, \Pi}^{\lambda\text{-leakage}^*}(k)]$. We say Π is fully λ -leakage resilient if $\Pr[\text{Succ}_{\mathcal{A}, \Pi}^{\lambda\text{-leakage}^*}(k)]$ is negligible for every PPT adversary \mathcal{A} .

If state is not updated after each signing query (and therefore, always contains only the randomness r used to generate the secret key), we denote the probability of success by $\Pr[\text{Succ}_{\mathcal{A}, \Pi}^{\lambda\text{-leakage}}(k)]$ and say Π is λ -leakage resilient if $\Pr[\text{Succ}_{\mathcal{A}, \Pi}^{\lambda\text{-leakage}}(k)]$ is negligible for every PPT adversary \mathcal{A} .

Leakage resilience in the definition above corresponds to the *memory attacks* of [1] (except that we allow the leakage to depend also on the random coins used to generate the secret key). Other variations of the definition are, of course, also possible: state could include only sk (and not the random coins r used to generate it), or could include only the most recently used random coins r_i .

2.1 A Technical Lemma

Let X be a random variable taking values in $\{0, 1\}^n$. The *min-entropy* of X is

$$H_\infty(X) \stackrel{\text{def}}{=} \min_{x \in \{0, 1\}^n} \{-\log_2 \Pr[X = x]\}.$$

The conditional min-entropy of X given an event E is defined as:

$$H_\infty(X | E) \stackrel{\text{def}}{=} \min_{x \in \{0, 1\}^n} \{-\log_2 \Pr[X = x | E]\}.$$

Lemma 1. *Let X be a random variable with $H \stackrel{\text{def}}{=} H_\infty(X)$, and fix $\delta \in [0, H]$. Let f be a function whose range has size 2^λ , and set*

$$Y \stackrel{\text{def}}{=} \{y \in \{0, 1\}^\lambda \mid H_\infty(X | y = f(X)) \leq H - \delta\}.$$

Then

$$\Pr[f(X) \in Y] \leq 2^{\lambda - \delta}.$$

In words: the probability that knowledge of $f(X)$ decreases the min-entropy of X by δ or more is at most $2^{\lambda - \delta}$. Put differently, the min-entropy of X after observing $f(X)$ is greater than H' except with probability at most $2^{\lambda - H + H'}$.

Proof. Fix y in the range of f and $x \in \{0, 1\}^n$ with $f(x) = y$. Since

$$\Pr[X = x \mid y = f(X)] = \frac{\Pr[X = x]}{\Pr[y = f(X)]},$$

we have that $y \in Y$ only if $\Pr[y = f(X)] \leq 2^{-\delta}$. The assumption regarding the range of f implies $|Y| \leq 2^\lambda$, and so $\Pr[f(X) \in Y] \leq 2^{\lambda - \delta}$ as claimed.

3 A Leakage-Resilient Signature Scheme

We construct a leakage-resilient signature scheme in the standard model, following the intuition described in Section 1.2. Let (Gen_H, H) be a public-coin⁴ UOWHF [27] mapping n -bit inputs to $\frac{1}{2} \cdot n^\epsilon$ -bit outputs for $n = \text{poly}(k)$ and $\epsilon \in (0, 1)$. Let $(\text{Gen}_E, \text{Enc}, \text{Dec})$ be a CPA-secure, dense⁵ public-key encryption scheme, and let $(\ell, \mathcal{P}, \mathcal{V}, \mathcal{S}_1, \mathcal{S}_2)$ be an unbounded simulation-sound NIZK proof system [8] for the following language L :

$$L = \{(s, y, pk, m, C) : \exists x, \omega \text{ s.t. } C = \text{Enc}_{pk}(x; \omega) \text{ and } H_s(x) = y\}.$$

The signature scheme is defined as follows:

- Key generation:** Choose random $x \leftarrow \{0, 1\}^n$ and compute $s \leftarrow \text{Gen}_H(1^k)$. Obviously sample a public key pk for the encryption scheme, and choose a random string $r \leftarrow \{0, 1\}^{\ell(k)}$. The public key is $(s, y := H_s(x), pk, r)$ and the secret key is x .
- Signing:** To sign message m using secret key x and public key (s, y, pk, r) , first choose random ω and compute $C := \text{Enc}_{pk}(x; \omega)$. Then compute $\pi \leftarrow \mathcal{P}_r((s, y, pk, m, C), (x, \omega))$; i.e., π is a proof that $(s, y, pk, m, C) \in L$ using witness (x, ω) . The signature is (C, π) .
- Verification:** Given a signature (C, π) on the message m with respect to the public key (s, y, pk, r) , output 1 iff $\mathcal{V}_r((s, y, pk, m, C), \pi) = 1$.

Theorem 1. *Under the stated assumptions, the scheme above is $(n - n^\epsilon)$ -leakage resilient.*

Proof (Sketch). Let Π denote the scheme given above, and let \mathcal{A} be a PPT adversary with $\delta = \delta(k) \stackrel{\text{def}}{=} \Pr[\text{Succ}_{\mathcal{A}, \Pi}^{\lambda\text{-leakage}}(k)]$. We consider a sequence of experiments, and let $\Pr_i[\cdot]$ denote the probability of an event in experiment i . We abbreviate $\text{Succ}_{\mathcal{A}, \Pi}^{\lambda\text{-leakage}}(k)$ by Succ .

Experiment 0: This is the experiment of Definition 2. Given the public key (s, y, pk, r) defined by the experiment, Succ denotes the event that \mathcal{A} outputs $(m, (C, \pi))$ where $\mathcal{V}_r((s, y, pk, m, C), \pi) = 1$ and m was never queried to the signing oracle. By assumption, we have $\Pr_0[\text{Succ}] = \delta$.

Experiment 1: We introduce the following differences with respect to the preceding experiment: when setting up the public key, we now generate the common random string r of the simulation-sound NIZK by computing $(r, \tau) \leftarrow \mathcal{S}_1(1^k)$. Furthermore, signing queries are now answered as follows: to sign m , generate $C \leftarrow \text{Enc}_{pk}(x)$ as before but compute π as $\pi \leftarrow \mathcal{S}_2((s, y, pk, m, C), \tau)$.

⁴ For a public-coin UOWHF (cf. [18]), it is hard to find a second pre-image even given the randomness used to generate the hash key. Standard constructions of UOWHFs have this property.

⁵ This means it is possible to sample a public key “obliviously,” without knowing the corresponding secret key.

It follows from the (adaptive) zero-knowledge property of $(\ell, \mathcal{P}, \mathcal{V}, \mathcal{S}_1, \mathcal{S}_2)$, that the difference $|\Pr_1[\text{Succ}] - \Pr_0[\text{Succ}]|$ must be negligible.

Experiment 2: We modify the preceding experiment in the following way: to answer a signing query for a message m , compute $C \leftarrow \text{Enc}_{pk}(0^n)$ (and then compute π as in Experiment 1). CPA-security of the encryption scheme implies that $|\Pr_2[\text{Succ}] - \Pr_1[\text{Succ}]|$ is negligible.

Experiment 3: We now change the way the public key is generated. Namely, instead of obviously sampling the encryption public key pk we compute it as $(pk, sk) \leftarrow \text{Gen}_E(1^k)$. Note that this is only a syntactic change and so $\Pr_3[\text{Succ}] = \Pr_2[\text{Succ}]$. (This assumes perfect oblivious sampling; if an obviously generated public key and a legitimately generated public key are only computationally indistinguishable, then the probability of Succ is affected by a negligible amount.)

Given the public key (s, y, pk, r) defined by the experiment, let Ext be the event that \mathcal{A} outputs $(m, (C, \pi))$ such that the event Succ occurs and furthermore, $H_s(\text{Dec}_{sk}(C)) = y$. Unbounded simulation soundness of the NIZK proof system implies that $|\Pr_3[\text{Ext}] - \Pr_3[\text{Succ}]|$ is negligible. (Note that by definition of L the message m is included as part of the statement being proved, and so if \mathcal{A} did not request a signature on m then it was never given a simulated proof of the statement (s, y, pk, m, C) .)

To complete the proof, we show that $\Pr_3[\text{Ext}]$ is negligible. Consider the following adversary \mathcal{B} finding a second preimage in the UOWHF: \mathcal{B} chooses random $x \leftarrow \{0, 1\}^n$ and is given key s (along with the randomness used to generate s). \mathcal{B} then runs Experiment 3 with \mathcal{A} . In this experiment all signatures given to \mathcal{A} are simulated (as described in Experiment 3 above); furthermore \mathcal{B} can easily answer any leakage queries made by \mathcal{A} since \mathcal{B} knows a legitimate secret key. (Recall that here we allow the leakage functions to be applied only to [the randomness used to generate] the secret key, but not to any auxiliary state used during signing.) If event Ext occurs when \mathcal{A} terminates, then \mathcal{B} recovers a value $x' \stackrel{\text{def}}{=} \text{Dec}_{sk}(C)$ for which $H_s(x') = y = H_s(x)$; i.e., \mathcal{B} recovers such an x' with probability exactly $\Pr_3[\text{Ext}]$. We now argue that $x' \neq x$ with high probability.

The only information about x revealed to \mathcal{A} in Experiment 3 comes from the value y included in the public key and the leakage queries asked by \mathcal{A} ; these total at most $\frac{1}{2} \cdot n^\epsilon + (n - n^\epsilon) = n - \frac{1}{2} \cdot n^\epsilon$ bits. Using Lemma 1 with $\Delta = H_\infty(x) = n$, the probability that $H_\infty(x \mid \mathcal{A}'\text{ view}) = 0$ (i.e., the probability that x is uniquely determined by the view of \mathcal{A}) is at most $2^{-n^\epsilon/2}$, which is negligible. When the conditional min-entropy of x is greater than 0 there are at least two (equally probable) possibilities for x and so $x' \neq x$ with probability at least $\frac{1}{2}$. Taken together, the probability that \mathcal{B} recovers $x' \neq x$ with $H_s(x') = H_s(x)$ is at least

$$\frac{1}{2} \cdot \left(\Pr_3[\text{Ext}] - 2^{-n^\epsilon/2} \right).$$

We thus see that if $\Pr_3[\text{Ext}]$ is not negligible then \mathcal{B} violates the security of the UOWHF with non-negligible probability, a contradiction. \square

If we are willing to rely on sub-exponential hardness assumptions, we can construct a UOWHF with $\omega(\log n)$ -bit outputs. In that case, the same signature scheme tolerates (optimal) leakage of $n - \omega(\log n)$ bits.

4 Fully Leakage-Resilient Bounded-Use Signature Schemes

In this section we describe constructions of fully leakage-resilient one-time and t -time signature schemes. These results are incomparable to the result of the previous section: on the positive side, here we achieve *full* leakage resilience (that is, where the leakage depends not only on the secret-key, but also on the randomness used by the signer) as well as better efficiency (and, in one case, rely on weaker assumptions); on the downside, the schemes given here are only secure when the adversary obtains a bounded number of signatures, and the leakage that can be tolerated is lower.

4.1 A Construction Based on One-Way Functions

We describe a basic one-time signature scheme, and then present an extension that tolerates leakage of up to a constant fraction of the secret key length. Let (Gen_H, H) be a UOWHF mapping k^c -bit inputs to k -bit outputs for some $c > 1$. (As before, we assume that H is a public-coin UOWHF, i.e., it is secure even given the randomness used to generate the hash key.) Our basic scheme is a variant on Lamport's signature scheme [23], using H as the one-way function:

Key generation: Choose random $x_{i,0}, x_{i,1} \leftarrow \{0,1\}^{k^c}$ for $i = 1, \dots, k$, and generate $s \leftarrow \text{Gen}_H(1^k)$. Compute $y_{i,b} := H_s(x_{i,b})$ for $i \in \{1, \dots, k\}$ and $b \in \{0,1\}$. The public key is $(s, \{y_{i,b}\})$ and the secret key is $\{x_{i,b}\}$.

Signing: The signature on a k -bit message $m = m_1 \cdots m_k$ consists of the k values $x_{1,m_1}, \dots, x_{k,m_k}$.

Verification: Given a signature x_1, \dots, x_k on the k -bit message $m = m_1 \cdots m_k$ with respect to the public key $(s, \{y_{i,b}\})$, output 1 iff $y_{i,m_i} \stackrel{?}{=} H_s(x_i)$ for all i .

It can be shown that the above scheme is fully $n^{(c-1)/(c+1)}$ -leakage resilient (as a one-time signature scheme), where $n = 2k^{c+1}$ denotes the length of the secret key. Setting c appropriately, the above approach thus tolerates leakage $n^{1-\epsilon}$ for any desired $\epsilon > 0$. (We omit the proof, since we will prove security for an improved scheme below.) The bound on the leakage is essentially tight, since an adversary who obtains the signature on the message 0^k and then leaks the value $x_{1,1}$ (which is only $k^c = (n/2)^{c/(c+1)}$ bits) can forge a signature on the message 10^{k-1} .

Tolerating leakage linear in the secret key length. An extension of the above scheme allows us to tolerate greater leakage. Specifically, we apply Lamport's scheme to a high-distance *encoding* of the message. Details follow.

If A is a $k \times \ell$ matrix over $\{0,1\}$ (viewed as the field \mathbb{F}_2), then A defines a (linear) error-correcting code $\mathcal{C} \subset \{0,1\}^\ell$ where the message $m \in \{0,1\}^k$ (viewed

as a row vector) is mapped to the codeword $m \cdot A$. It is well known that for every $\epsilon > 0$ there exists a constant R such that choosing $A \in \{0, 1\}^{k \times Rk}$ uniformly at random defines a code with relative minimum distance $\frac{1}{2} - \epsilon$, except with probability negligible in k . (We will not need efficient decodability.)

Fix a constant $\epsilon \in (0, 1)$ and let R be as above; set $\ell = Rk$. Let (Gen_H, H) be a UOWHF mapping ℓ_{in} -bit inputs to k -bit outputs where $\ell_{in} = 2k/\epsilon$. The signature scheme is defined as:

Key generation: Choose random $A \in \{0, 1\}^{k \times \ell}$ and $x_{i,0}, x_{i,1} \leftarrow \{0, 1\}^{\ell_{in}}$ for $i = 1, \dots, \ell$. Generate $s \leftarrow \text{Gen}_H(1^k)$. Compute $y_{i,b} := H_s(x_{i,b})$ for $i \in \{1, \dots, \ell\}$ and $b \in \{0, 1\}$. The public key is $(A, s, \{y_{i,b}\})$ and the secret key is $\{x_{i,b}\}$.

Signing: To sign a message $m \in \{0, 1\}^k$, first compute $\bar{m} = m \cdot A \in \{0, 1\}^\ell$. The signature then consists of the ℓ values $x_{1,\bar{m}_1}, \dots, x_{\ell,\bar{m}_\ell}$.

Verification: Given a signature x_1, \dots, x_ℓ on the message m with respect to the public key $(A, s, \{y_{i,b}\})$, first compute $\bar{m} = m \cdot A$ and then output 1 iff $y_{i,\bar{m}_i} \stackrel{?}{=} H_s(x_i)$ for all i .

Theorem 2. *If H is a UOWHF then the scheme above is a one-time signature scheme that is fully $(\frac{1}{4} - \epsilon) \cdot n$ -leakage resilient, where $n = 2\ell \cdot \ell_{in}$ denotes the length of the secret key.*

Proof. Let Π denote the scheme given above, and let \mathcal{A} be a PPT adversary with $\delta = \delta(k) \stackrel{\text{def}}{=} \Pr[\text{Succ}_{\mathcal{A}, \Pi}^{\lambda\text{-leakage}^*}(k)]$. We construct an adversary \mathcal{B} breaking the security of H with probability at least $(\delta - \text{negl}(k))/4\ell$, implying that δ must be negligible.

\mathcal{B} chooses random $A \in \{0, 1\}^{k \times \ell}$ and $x_{i,0}, x_{i,1} \leftarrow \{0, 1\}^{\ell_{in}}$ for $i = 1, \dots, \ell$; we let $\mathcal{X} = \{x_{i,b}\}$ denote the set of secret key values \mathcal{B} chooses and observe that $H_\infty(\mathcal{X}) = 2\ell \cdot \ell_{in}$. Next, \mathcal{B} selects a random $b^* \in \{0, 1\}$ and a random index $i^* \in \{1, \dots, \ell\}$, and outputs x_{i^*, b^*} ; it is given in return a hash key s . Then \mathcal{B} computes $y_{i,b} := H_s(x_{i,b})$ for all i, b and gives the public key $(A, s, \{y_{i,b}\})$ to \mathcal{A} .

\mathcal{B} answers the signing and leakage queries of \mathcal{A} using the secret key $\{x_{i,b}\}$ that it knows. Since this secret key is distributed identically to the secret key of an honest signer, the simulation for \mathcal{A} is perfect and \mathcal{A} outputs a forgery with probability δ .

Let \bar{m} denote the encoding of the message m whose signature was requested by \mathcal{A} . The information \mathcal{A} has about the secret-key \mathcal{X} consists of: (1) the signature $(x_{1,\bar{m}_1}, \dots, x_{\ell,\bar{m}_\ell})$ it obtained; (2) the values $\{y_{i,1-\bar{m}_i}\}_{i=1}^\ell$ from the public key and (3) the answers to the leakage queries asked by \mathcal{A} . Together, these total $\ell \cdot \ell_{in} + \ell k + (\frac{1}{4} - \epsilon) \cdot 2\ell \cdot \ell_{in}$ bits. By Lemma [11](#), it follows that $H_\infty(\mathcal{X} \mid \mathcal{A}'\text{s view}) > (\frac{1}{2} + \epsilon) \cdot \ell \cdot \ell_{in}$ except with probability at most

$$2^{(\ell \cdot \ell_{in} + \ell k + (\frac{1}{2} - 2\epsilon)\ell \cdot \ell_{in}) - 2\ell \cdot \ell_{in} + (\frac{1}{2} + \epsilon) \cdot \ell \cdot \ell_{in}} = 2^{\ell k - \epsilon \ell \cdot \ell_{in}},$$

which is negligible.

Assuming $H_\infty(\mathcal{X} \mid \mathcal{A}'\text{s view}) > (\frac{1}{2} + \epsilon) \cdot \ell \cdot \ell_{in}$, there is no set $I \subseteq [\ell]$ with $|I| \geq (\frac{1}{2} - \epsilon) \cdot \ell$ for which the values $\{x_{i,1-\bar{m}_i}\}_{i \in I}$ are all fixed given $\mathcal{A}'\text{s view}$. To see this, assume the contrary. Then

$$H_\infty(\mathcal{X} \mid \mathcal{A}'\text{'s view}) \leq \sum_{i \notin I} H_\infty(x_{i,1-\bar{m}_i} \mid \mathcal{A}'\text{'s view}) \leq \left(\frac{1}{2} + \epsilon\right) \ell \cdot \ell_{in},$$

in contradiction to the assumed bound on the conditional min-entropy of \mathcal{X} .

Let $(m^*, (x_1^*, \dots, x_\ell^*))$ denote the forgery output by \mathcal{A} , and let $\bar{m}^* = m^* \cdot A$ denote the encoding of m^* . Let I be the set of indices where \bar{m} and \bar{m}^* differ; with all but negligible probability over choice of the matrix A it holds that $|I| \geq (\frac{1}{2} - \epsilon) \cdot \ell$ and so we assume this to be the case. By the argument of the previous paragraph, it cannot be the case that the $\{x_{i,1-\bar{m}_i}\}_{i \in I}$ are all fixed given $\mathcal{A}'\text{'s view}$. But then with probability at least half we have $x_i^* \neq x_{i,\bar{m}_i^*}$ for at least one index $i \in I$. Assuming this to be the case, with probability at least $1/2\ell$ this difference occurs at the index (i^*, b^*) guessed at the outset by \mathcal{B} ; when this happens \mathcal{B} has found a collision in H for the given hash key s . Putting everything together, we see that \mathcal{B} finds a collision in H with probability at least $(\delta - \text{negl}(k)) \cdot \frac{1}{2} \cdot \frac{1}{2\ell}$, as claimed.

A t -time signature scheme. The idea above can be further extended to give a fully leakage resilient t -time signature scheme using *cover-free families*. We follow the definition of [22].

Definition 3. A family of non-empty sets $\mathcal{S} = \{S_1, \dots, S_N\}$, where $S_i \subset U$, is $(t, \frac{1}{2})$ -cover-free if for all distinct $S, S_1, \dots, S_t \in \mathcal{S}$ we have $|S \setminus \bigcup_{i=1}^t S_i| \geq |S|/2$.

Porat and Rothschild [32] show an explicit construction that, for any t and k , yields a $(t, \frac{1}{2})$ -cover free family $\mathcal{S} = \{S_1, \dots, S_N\}$ where the number of sets is $N = \Omega(2^k)$, the size of each set is $|S_i| = \mathcal{O}(kt)$, and the universe size is $|U| = \mathcal{O}(kt^2)$. If we let $f : \{0, 1\}^k \rightarrow \mathcal{S}$ denote an injective map, we obtain the following scheme:

Key generation: Set $\ell = \mathcal{O}(kt^2)$ and $\ell_{in} = 8tk$. Choose $x_i \leftarrow \{0, 1\}^{\ell_{in}}$ for $i = 1, \dots, \ell$. Generate $s \leftarrow \text{Gen}_H(1^k)$, and compute $y_i := H_s(x_i)$ for $i \in \{1, \dots, \ell\}$.

The public key is $(s, \{y_i\}_{i=1}^\ell)$ and the secret key is $\{x_i\}_{i=1}^\ell$.

Signing: To sign a message $m \in \{0, 1\}^k$, first compute $f(m) = S_m \in \mathcal{S}$. The signature then consists of $\{x_i\}_{i \in S_m}$.

Verification: Given a signature $\{x_i\}$ on the message m with respect to the public key $(s, \{y_i\})$, first compute $S_m = f(m)$ and then output 1 iff $y_i \stackrel{?}{=} H_s(x_i)$ for all $i \in S_m$.

A proof of the following proceeds along exactly the same lines as the proof of Theorem 2:

Theorem 3. If H is a UOWHF then the scheme above is a t -time signature scheme that is fully $\Theta(n/t)$ -leakage resilient, where $n = \ell \cdot \ell_{in}$ denotes the length of the secret key.

4.2 A Construction from Homomorphic Collision-Resistant Hashing

Our second construction of fully leakage-resilient bounded-use signature schemes relies on homomorphic collision-resistant hash functions, defined below. In Section 4.3, we describe efficient instantiations of the hash functions we need based on several concrete assumptions.

We concentrate on the case of one-time signatures, and defer a treatment of t -time signatures to the full version.

Definition 4. Fix $\epsilon \in (0, 1)$. A pair of PPT algorithms (Gen_H, H) is an ϵ -homomorphic collision-resistant hash function family (ϵ -hCRHF) if:

1. $\text{Gen}_H(1^k)$ outputs a key s that specifies groups $(G, \oplus), (G', \otimes)$ (written additively), and two sets $S, T \subseteq G$ such that
 - $\log |S| = \omega(\log k)$ and $\log |G'| \leq \epsilon \cdot \log |S|$ and $\log |T| \leq (1 + \epsilon) \log |S|$.
 - S is efficiently sampleable, and elements of S can be represented using $\log |S| + \mathcal{O}(1)$ bits.
 - T is efficiently recognizable, and $\{x + my \mid x, y \in S, 0 \leq m < 2^k\} \subseteq T$.
2. The key s defines a function $H_s : G \rightarrow G'$ with $H_s(x \oplus y) = H_s(x) \otimes H_s(y)$ for all $x, y \in G$.
3. There exists a constant c (independent of k) for which the following holds. For any s , any m, m' with $0 \leq m < m' < 2^k$, and any σ, σ' :

$$\left| \{x, y \in S \mid H_s(x + my) = \sigma \wedge H_s(x + m'y) = \sigma'\} \right| \leq 2^c.$$

4. No PPT algorithm \mathcal{A} can find two elements $x, y \in T$ such that $H_s(x) = H_s(y)$. Namely, the following is negligible for all PPT \mathcal{A} :

$$\Pr[s \leftarrow \text{Gen}_H(1^k); (x, y) \leftarrow \mathcal{A}(s) : x, y \in T_k \wedge x \neq y \wedge H_s(x) = H_s(y)].$$

If the above holds even when \mathcal{A} is given the randomness used to generate s , then (Gen_H, H) is a strong ϵ -hCRHF.

Define a signature scheme as follows.

Key generation: Compute $s \leftarrow \text{Gen}_H(1^k)$; this specifies groups $(G, \oplus), (G', \otimes)$ and sets S, T . Choose x, y uniformly at random from S . Output $sk := (x, y)$ and $pk := (s, H_s(x), H_s(y))$.

Signing: The scheme is defined for messages m satisfying $0 \leq m < 2^k$. Given m , output the signature $\sigma := x \oplus my$.

Verification: Given a signature σ on the message m with respect to the public key $pk = (s, a, b)$, output 1 iff $\sigma \in T$ and $H_s(\sigma) \stackrel{?}{=} a \otimes mb$.

Theorem 4. If (Gen_H, H) is a (strong) ϵ -hCRHF, then the above is a one-time signature scheme that is (fully) $(\frac{1}{2} - 2\epsilon) \cdot n$ -leakage resilient.

Proof. Correctness is easily verified. Let Π denote the scheme given above, and let \mathcal{A} be a PPT adversary with $\delta = \delta(k) \stackrel{\text{def}}{=} \Pr[\text{Succ}_{\mathcal{A}, \Pi}^{\lambda\text{-leakage}^*}(k)]$. We construct an adversary \mathcal{B} breaking the security of (Gen_H, H) with probability at least $\delta/2 - \text{negl}(k)$, implying that δ must be negligible.

\mathcal{B} is given as input a key s (along with the randomness used to generate it). \mathcal{B} chooses $x, y \in S$, sets $sk := (x, y)$, and gives the public key $pk := (s, H_s(x), H_s(y))$ to \mathcal{A} . Algorithm \mathcal{B} then answers the signing and leakage queries of \mathcal{A} using the secret key (x, y) that it knows. Since this secret key is distributed

identically to the secret key of an honest signer, the simulation for \mathcal{A} is perfect and \mathcal{A} outputs a valid forgery (m', σ') with probability δ . If this occurs, then \mathcal{B} outputs $(\sigma', x \oplus m'y)$ as a candidate collision for H_s .

Note that $x \oplus m'y \in T$. If σ' is a valid signature on m' , we have $\sigma' \in T$ and

$$H_s(\sigma') = H_s(x) \otimes m'H_s(y) = H_s(x \oplus m'y).$$

It remains to show that $\sigma' \neq x \oplus m'y$ with significant probability.

Let c be the constant guaranteed to exist by condition **3** of Definition **4**. The length of the secret key is $n \stackrel{\text{def}}{=} 2 \log |S|$ bits.⁶ The information \mathcal{A} has about $sk = (x, y)$ consists of: (1) the signature $x \oplus my$ it obtained; (2) the values $H_s(x), H_s(y)$ from the public key; and (3) the answers to the leakage queries asked by \mathcal{A} . These total at most

$$\begin{aligned} \log |T| + 2 \log |G'| + \left(\frac{1}{2} - 2\epsilon\right) 2 \log |S| &\leq (1 + \epsilon) \log |S| + 2\epsilon \log |S| \\ &\quad + \log |S| - 4\epsilon \log |S| \\ &= 2 \log |S| - \epsilon \log |S| \end{aligned}$$

bits of information about sk . The min-entropy of sk is $2 \log |S|$ bits, so by Lemma **1** it follows that $H_\infty(sk \mid \mathcal{A}'\text{'s view}) \geq c + 1$ except with probability at most $2^{-\epsilon \log |S| + c + 1}$, which is negligible.

Assuming $H_\infty(sk \mid \mathcal{A}'\text{'s view}) \geq c + 1$, we claim that for any $m' \neq m$ (with $0 \leq m' < 2^k$) the value $x \oplus m'y$ has min-entropy at least 1; this follows from the fact that, for any fixed $\hat{\sigma}'$, the two equations $\sigma = x \oplus my$ and $\hat{\sigma}' = x \oplus m'y$ constrain (x, y) to a set of size at most 2^c (by condition **3** of Definition **4**). Thus, $\sigma' = x \oplus m'y$ with probability at most $1/2$. Putting everything together, we see that \mathcal{B} finds a collision in H_s with probability at least $(\delta - \text{negl}(k)) \cdot \frac{1}{2}$ as claimed.

4.3 Constructing (Strong) Homomorphic CRHFs

Homomorphic CRHFs can be constructed from a variety of standard assumptions. Here, we describe constructions based on the discrete logarithm and the RSA assumptions; in the full version, we show a construction based on lattices. All except the RSA-based construction are *strong* ϵ -hCRHFs.

An instantiation based on the discrete logarithm assumption. Let G' be a group of prime order $p > 2^k$ where the discrete logarithm problem is hard. Let $\ell = \lceil \frac{1}{\epsilon} \rceil$, and set $S = T = G = \mathbb{Z}_p^\ell$.

The key-generation algorithm Gen_H outputs random $g_1, \dots, g_\ell \in G$ as the key. Given $s = (g_1, \dots, g_\ell)$, define $H_s(x_1, \dots, x_\ell) = \prod_{i=1}^\ell g_i^{x_i}$. This function is clearly homomorphic, and collision resistance follows by standard arguments.

An instantiation based on the RSA assumption. Fix $\ell = \lceil \frac{2}{\epsilon} \rceil$. On security parameter k , algorithm $\text{Gen}_H(1^k)$ chooses safe primes $p = 2p' + 1$ and $q = 2q' + 1$

⁶ We assume for simplicity that elements of S can be described using exactly $\log |S|$ bits; the proof can be modified suitably if this is not the case.

with $p', q' > 2^k$, and sets $N = pq$. (The primes p and q are not used after key generation, but because they are in memory during key generation this construction is not strong.) Gen_H then chooses a random element $u \in \mathbb{Z}_N^*$, as well as a prime $e > 2^{(\ell+1) \cdot k}$. The key is $s = (N, e, u)$.

Let $G = \mathbb{Z}_N^* \times \mathbb{Z}$ and $G' = \mathbb{Z}_N^*$. Define

$$H_s(r, x) = r^e \cdot u^x \pmod{N}.$$

Take $S = \mathcal{QR}_N \times \{0, \dots, 2^{\ell k}\} \subset G$ (where \mathcal{QR}_N denotes the set of quadratic residues modulo N) and $T = \mathbb{Z}_N^* \times \{0, \dots, 2^{(\ell+1) \cdot k}\}$.

The homomorphic property of H_s is easy to see. One can also verify that:

1. $\log |S| = \omega(\log k)$ and $\log |G'| \leq \epsilon \cdot \log |S|$ and $\log |T| \leq (1 + \epsilon) \log |S|$.
2. T is efficiently recognizable, and $\{x + my \mid x, y \in S, 0 \leq m < 2^k\} \subseteq T$.
3. For any s , any m, m' with $0 \leq m < m' < 2^k$, and any σ, σ' :

$$\left| \{x, y \in S \mid H_s(x + my) = \sigma \wedge H_s(x + m'y) = \sigma'\} \right| \leq 1.$$

(This uses the fact that $\mathcal{QR}_N \simeq \mathbb{Z}_{p'} \times \mathbb{Z}_{q'}$ has no elements other than the identity whose order is less than 2^k .)

Collision resistance follows via standard arguments (e.g., [29]).

Acknowledgments

We are grateful to Yael Tauman Kalai for stimulating discussions, and collaboration during the early stages of this work. We also thank Krzysztof Pietrzak for suggesting the extension to the one-time signature scheme (based on one-way functions) that tolerates leakage linear in the secret key length.

References

1. 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)
2. Alwen, J., Dodis, Y., Wichs, D.: Public key cryptography in the bounded retrieval model and security against side-channel attacks. In: Halevi, S. (ed.) Crypto 2009. LNCS, vol. 5677, pp. 1–17. Springer, Heidelberg (2009)
3. Biham, E., Carmeli, Y., Shamir, A.: Bug attacks. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 221–240. Springer, Heidelberg (2008)
4. Boneh, D., Brumley, D.: Remote timing attacks are practical. *Computer Networks* 48(5), 701–716 (2005)
5. Boneh, D., DeMillo, R.A., Lipton, R.J.: On the importance of checking cryptographic protocols for faults. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 37–51. Springer, Heidelberg (1997)
6. 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)

7. Cramer, R., Damgård, I.: Secure signature schemes based on interactive protocols. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 297–310. Springer, Heidelberg (1995)
8. 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)
9. Dodis, Y., Kalai, Y., Lovett, S.: On cryptography with auxiliary input. In: 41st Annual ACM Symposium on Theory of Computing (STOC), pp. 621–630. ACM, New York (2009)
10. Dodis, Y., Kalai, Y., Vaikuntanathan, V.: Public-key encryption schemes with auxiliary inputs (manuscript, 2009)
11. Dziembowski, S., Pietrzak, K.: Leakage-resilient cryptography. In: 49th Annual Symposium on Foundations of Computer Science (FOCS), pp. 293–302. IEEE, Los Alamitos (2008), Full version: <http://eprint.iacr.org/2008/240>
12. Faust, S., Kiltz, E., Pietrzak, K., Rothblum, G.: Leakage-resilient signatures, <http://eprint.iacr.org/2009/282>
13. 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)
14. Fischlin, M., Fischlin, R.: The representation problem based on factoring. In: Preneel, B. (ed.) CT-RSA 2002. LNCS, vol. 2271, pp. 96–113. Springer, Heidelberg (2002)
15. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing* 17(2), 281–308 (1988)
16. Guillou, L.C., Quisquater, J.-J.: A “paradoxical” indentity-based signature scheme resulting from zero-knowledge. In: Goldwasser, S. (ed.) CRYPTO 1988. LNCS, vol. 403, pp. 216–231. Springer, Heidelberg (1990)
17. Halderman, A., Schoen, S., Heninger, N., Clarkson, W., Paul, W., Calandrino, J., Feldman, A., Applebaum, J., Felten, E.: Lest we remember: Cold boot attacks on encryption keys. In: Proc. 17th USENIX Security Symposium, pp. 45–60. USENIX Association (2008)
18. 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)
19. 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)
20. 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)
21. 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)
22. Kumar, R., Rajagopalan, S., Sahai, A.: Coding constructions for blacklisting problems without computational assumptions. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 609–623. Springer, Heidelberg (1999)
23. Lamport, L.: Constructing digital signatures from a one-way function. Technical Report SRI-CSL-98, SRI International Computer Science Laboratory (October 1979)

24. Lyubashevsky, V., Micciancio, D.: Asymptotically efficient lattice-based digital signatures. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 37–54. Springer, Heidelberg (2008)
25. Micali, S., Reyzin, L.: Physically observable cryptography. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 278–296. Springer, Heidelberg (2004)
26. 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), <http://eprint.iacr.org/2009/105>
27. Naor, M., Yung, M.: Universal one-way hash functions and their cryptographic applications. In: 21st Annual ACM Symposium on Theory of Computing (STOC), pp. 33–43. ACM Press, New York (1989)
28. Nguyen, P.Q., Shparlinski, I.: The insecurity of the digital signature algorithm with partially known nonces. *Journal of Cryptology* 15(3), 151–176 (2002)
29. 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)
30. Ong, H., Schnorr, C.-P.: Fast signature generation with a Fiat-Shamir-like scheme. In: Damgård, I.B. (ed.) EUROCRYPT 1990. LNCS, vol. 473, pp. 432–440. Springer, Heidelberg (1991)
31. Pietrzak, K.: A leakage-resilient mode of operation. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 462–482. Springer, Heidelberg (2009)
32. Porat, E., Rothschild, A.: Explicit non-adaptive combinatorial group testing schemes. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfssdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part I. LNCS, vol. 5125, pp. 748–759. Springer, Heidelberg (2008)
33. Sahai, A.: Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In: 40th Annual Symposium on Foundations of Computer Science (FOCS), pp. 543–553. IEEE, Los Alamitos (1999)
34. 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)
35. Standaert, F.-X., Malkin, T., 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)

Author Index

- Abe, Masayuki 435
Aoki, Kazumaro 578
Armknecht, Frederik 685
Ateniese, Giuseppe 319
Aumasson, Jean-Philippe 542
- Bellare, Mihir 232
Benadjila, Ryad 162
Billet, Olivier 162, 451
Biryukov, Alex 1
Boldyreva, Alexandra 524
Brakerski, Zvika 232
Brier, Eric 560
Brumley, Billy Bob 667
- Çalık, Çağdaş 542
Cao, Zhenfu 303
Cash, David 524
Castagnos, Guilhem 469
Cathalo, Julien 179
Chen, Liqun 505
Choi, Seung Geol 268, 287
Coron, Jean-Sébastien 653
- Dachman-Soled, Dana 287
Damgård, Ivan 52
Ding, Ning 303
- Elbaz, Ariel 268
- Finiasz, Matthieu 88
Fischlin, Marc 524
- Gaži, Peter 37
Gueron, Shay 162
Guo, Jian 578
- Hakala, Risto M. 667
Herrmann, Mathias 487
- Jager, Tibor 399
Joux, Antoine 347, 469
- Kamara, Seny 319
Katz, Jonathan 197, 319, 636, 703
- Khazaei, Shahram 560
Khovratovich, Dmitry 1
Kurosawa, Kaoru 334
- Laguillaumie, Fabien 469
Lai, Xuejia 19
Lamberger, Mario 126
Lehmann, Anja 364
Libert, Benoît 179
Lucks, Stefan 347
Lunemann, Carolin 52
Lyubashevsky, Vadim 598
- Ma, Rong 303
Macario-Rat, Gilles 451
Maes, Roel 685
Malkin, Tal 268, 287
Mandal, Avradip 653
Matusiewicz, Krystian 106, 578
Maurer, Ueli 37
May, Alexander 487
Meier, Willi 542, 560
Mendel, Florian 126, 144
Morrissey, Paul 505
- Naito, Yusuke 382
Naor, Moni 232
Naya-Plasencia, María 106
Nguyen, Phong Q. 469
Nikolić, Ivica 106
Nojima, Ryo 334
- Ohkubo, Miyako 435
Ohta, Kazuo 382
Okamoto, Tatsuaki 214
Özen, Onur 542
- Peyrin, Thomas 560
Phan, Raphael C.-W. 542
Pinkas, Benny 250
- Rechberger, Christian 126, 144
Rijmen, Vincent 126
Ristenpart, Thomas 232
Robshaw, Matt J.B. 162

- Sadeghi, Ahmad-Reza 685
Salvail, Louis 70
Sasaki, Yu 106, 578
Schaffner, Christian 70
Schläffer, Martin 106, 126, 144
Schneider, Thomas 250
Schwenk, Jörg 399
Segev, Gil 232
Sendrier, Nicolas 88
Shacham, Hovav 232
Smart, Nigel P. 250, 505
Sotáková, Miroslava 70
Stehlé, Damien 617
Steinfeld, Ron 617
Sun, Xiaorui 19
Sunar, Berk 685
Takashima, Katsuyuki 214
Tanaka, Keisuke 617
Tessaro, Stefano 364
Tuyls, Pim 685
Vaikuntanathan, Vinod 636, 703
Varıcı, Kerem 542
Wang, Lei 382, 578
Warinschi, Bogdan 505, 524
Wee, Hoeteck 287, 417
Williams, Stephen C. 250
Xagawa, Keita 617
Yerukhimovich, Arkady 197
Yilek, Scott 232
Yoneyama, Kazuki 382
Yung, Moti 179, 268
Zhang, Zongyang 303