

Rosario Gennaro  
Matthew Robshaw (Eds.)

LNCS 9215

# Advances in Cryptology – CRYPTO 2015

35th Annual Cryptology Conference  
Santa Barbara, CA, USA, August 16–20, 2015  
Proceedings, Part I

1  
Part I



 Springer

*Commenced Publication in 1973*

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

## Editorial Board

David Hutchison

*Lancaster University, Lancaster, UK*

Takeo Kanade

*Carnegie Mellon University, Pittsburgh, PA, USA*

Josef Kittler

*University of Surrey, Guildford, UK*

Jon M. Kleinberg

*Cornell University, Ithaca, NY, USA*

Friedemann Mattern

*ETH Zurich, Zürich, Switzerland*

John C. Mitchell

*Stanford University, Stanford, CA, USA*

Moni Naor

*Weizmann Institute of Science, Rehovot, Israel*

C. Pandu Rangan

*Indian Institute of Technology, Madras, India*

Bernhard Steffen

*TU Dortmund University, Dortmund, Germany*

Demetri Terzopoulos

*University of California, Los Angeles, CA, USA*

Doug Tygar

*University of California, Berkeley, CA, USA*

Gerhard Weikum

*Max Planck Institute for Informatics, Saarbrücken, Germany*

More information about this series at <http://www.springer.com/series/7410>

Rosario Gennaro · Matthew Robshaw (Eds.)

# Advances in Cryptology – CRYPTO 2015

35th Annual Cryptology Conference  
Santa Barbara, CA, USA, August 16–20, 2015  
Proceedings, Part I

*Editors*

Rosario Gennaro  
City College of New York  
New York, NY  
USA

Matthew Robshaw  
Impinj, Inc.  
Seattle, WA  
USA

ISSN 0302-9743                      ISSN 1611-3349 (electronic)  
Lecture Notes in Computer Science  
ISBN 978-3-662-47988-9              ISBN 978-3-662-47989-6 (eBook)  
DOI 10.1007/978-3-662-47989-6

Library of Congress Control Number: 2015944435

LNCS Sublibrary: SL4 – Security and Cryptology

Springer Heidelberg New York Dordrecht London  
© International Association for Cryptologic Research 2015

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made.

Printed on acid-free paper

Springer-Verlag GmbH Berlin Heidelberg is part of Springer Science+Business Media  
([www.springer.com](http://www.springer.com))

## Preface

CRYPTO 2015, the 35th Annual International Cryptology Conference, was held August 16–20, 2015, on the campus of the University of California, Santa Barbara. The event was sponsored by the International Association for Cryptologic Research (IACR) in cooperation with the UCSB Computer Science Department.

The program of CRYPTO 2015 reflects significant advances and trends in all areas of cryptology. Seventy-four papers were included in the program; this two-volume proceedings contains the revised versions of these papers. The program also included two invited talks: Shai Halevi on ‘The state of cryptographic multilinear maps’ and Ed Felten on ‘Cryptography, Security, and Public Safety: A Policy Perspective’. The paper ‘Integral Cryptanalysis on Full MISTY1’ by Yosuke Todo was selected for both the best paper award and the award for the best paper authored by a young researcher.

This year we received a record number of submissions (266), and in an effort to accommodate as many high-quality submissions as possible, the conference ran in two parallel sessions.

The papers were reviewed by a Program Committee (PC) consisting of 40 leading researchers in the field, in addition to the two co-chairs. Each PC member was allowed to submit two papers. Papers were reviewed in a double-blind fashion, with each paper assigned to three reviewers (four for PC-authored papers). During the discussion phase, when necessary, extra reviews were solicited.

We would like to sincerely thank the authors of all submissions—those whose papers made it into the program and those whose papers did not. Our deep appreciation also goes out to the PC members, who invested an extraordinary amount of time in reviewing papers, and to the many external reviewers who significantly contributed to the comprehensive evaluation of the submissions. A list of PC members and external reviewers follows. Despite all our efforts, the list of external reviewers may contain errors or omissions; we apologize for that in advance.

We would like to thank Tom Ristenpart, the general chair, for working closely with us throughout the whole process and providing the much-needed support at every step, including artfully creating and maintaining the website and taking care of all aspects of the conference’s logistics—particularly the novel double-track arrangements.

As always, special thanks are due to Shai Halevi for providing his tireless support of the *websubrev* software, which we used for the whole conference planning and operation, including paper submission and evaluation, interaction among PC members, and communication with the authors. Alfred Hofmann and his colleagues at Springer provided a meticulous service for the timely production of this volume.

Finally, we would like to thank Qualcomm, NSF, and Microsoft for sponsoring the conference, and Cryptography Research for their continuous support.

# CRYPTO 2015

## The 35th IACR International Cryptology Conference

University of California, Santa Barbara, CA, USA  
August 16–20, 2015

Sponsored by the *International Association for Cryptologic Research*

### General Chair

Thomas Ristenpart      Cornell Tech, New York, USA

### Program Chairs

Rosario Gennaro      The City College of New York, USA  
Matthew Robshaw      Impinj, USA

### Program Committee

Michel Abdalla	École Normale Supérieure and CNRS, France
Masayuki Abe	NTT Labs, Japan
Paulo Barreto	University of Sao Paulo, Brazil
Colin Boyd	University of Science and Technology, Norway
Zvika Brakerski	Weizmann Institute of Science, Israel
Emmanuel Bresson	Airbus Cybersecurity, France
Anne Canteaut	Inria, France
Dario Catalano	Università di Catania, Italy
Nishanth Chandran	Microsoft Research, India
Melissa Chase	Microsoft Research, USA
Joan Daemen	ST Microelectronics, Belgium
Orr Dunkelman	University of Haifa, Israel
Karim ElDefrawy	HRL Laboratories, USA
Dario Fiore	IMDEA Software Institute, Spain
Steven Galbraith	Auckland University, New Zealand
Sanjam Garg	University of California, Berkeley, USA
Carmit Hazay	Bar-Ilan University, Israel
Tetsu Iwata	Nagoya University, Japan
Stas Jarecki	University of California, Irvine, USA
Thomas Johansson	Lund University, Sweden
Lars R. Knudsen	Technical University of Denmark

Gregor Leander	Ruhr-Universität Bochum, Germany
Allison B. Lewko	Columbia University, USA
Huijia (Rachel) Lin	University of California, Santa Barbara, USA
Mitsuru Matsui	Mitsubishi Electric, Japan
Sarah Meiklejohn	University College London, UK
Daniele Micciancio	University of California, San Diego, USA
Steve Myers	Indiana University, USA
Bryan Parno	Microsoft Research, USA
Giuseppe Persiano	Università di Salerno, Italy
Thomas Peyrin	Nanyang Technological University, Singapore
Josef Pieprzyk	Queensland University of Technology, Australia
Axel Poschmann	NXP Semiconductors, Germany
Bart Preneel	KU Leuven, Belgium
Mariana Raykova	SRI International, USA
Carla Ràfols	Ruhr-Universität Bochum, Germany
Palash Sarkar	Indian Statistical Institute, India
Nigel Smart	University of Bristol, UK
François-Xavier Standaert	Université catholique de Louvain, Belgium
John Steinberger	Tsinghua University, China

## Additional Reviewers

Divesh Aggarwal	Harry Bartlett	Ran Canetti
Shashank Agrawal	Georg Becker	Angelo De Caro
Shweta Agrawal	Christof Beierle	David Cash
Martin Albrecht	Sonia Belaid	Debrup Chakraborty
Mehrdad Aliasgari	Mihir Bellare	Eshan Chattopadhyay
Prabhanjan Ananth	Fabrice Benhamouda	Binyi Chen
Elena Andreeva	Guido Bertoni	Jie Chen
Kazumaro Aoki	Nir Bitansky	Mahdi Cheraghchi
Daniel Apon	Olivier Blazy	Céline Chevalier
Benny Applebaum	Celine Blondeau	Chongwon Cho
Frederik Armknecht	Florian Boehl	Joo Yeon Cho
Hassan Asghar	Sonia Bogos	Ashish Choudhury
Gilad Asharov	Jonathan Bootle	Michele Ciampi
Gilles Van Assche	Joppe Bos	Ran Cohen
Nuttapong Attrapadung	Christina Boura	Dana Dachman-Soled
Jean-Philippe Aumasson	Elette Boyle	Hani T. Dawoud
Shi Bai	Cerys Bradley	Ed Dawson
Josep Balasch	Anne Broadbent	Yi Deng
Foteini Baldimtsi	Andre Chailloux	Claus Diem
Achiya Bar-On	Christian Cachin	Itai Dinur
Joshua Baron	Seyit Camtepe	Yevgeniy Dodis



Alexandre Duc	Marcel Keller	Rafael Misoczki
Leo Ducas	Nathan Keller	Payman Mohassel
Stefan Dziembowski	Carmen Kempka	Amir Moradi
Oriol Farràs	Sotirios Kentros	Pawel Morawiecki
Sebastian Faust	Dmitry Khovratovich	Paz Morillo
Serge Fehr	Dakshita Khurana	Nicky Mouha
Joan Feigenbaum	Aggelos Kiayias	Pratyay Mukherjee
Ben Fisch	Hyun-Jin (Tiffany) Kim	Sean Murphy
Marc Fischlin	Susumu Kiyoshima	Michael Naehrig
Christopher Fletcher	Miroslav Knezevic	Preetum Nakkiran
Georg Fuchsbauer	Markulf Kohlweiss	Chanathip Namprempre
Thomas Fuhr	Ilan Komargodski	Mara Naya-Plasencia
Eiichiro Fujisaki	Venkata Koppula	Phong Nguyen
Marc Fyrbiak	Luke Kowalczyk	Jesper Buus Nielsen
Romain Gay	Thorsten Kranz	Ivica Nikolic
Ran Gelles	Ranjit Kumaresan	Ventzi Nikov
Craig Gentry	Junichiro Kume	Svetla Nikova
Hossein Ghodosi	Eyal Kushilevitz	Ryo Nishimaki
Kristian Gjøsteen	Tatsuya Kyogoku	Luca Nizzardo
Florian Gopfert	Thijs Laarhoven	Adam O'Neill
Vincent Grosso	Mario Lamberger	Miyako Ohkubo
Jian Guo	Joshua Lampkins	Olya Ohrimenko
Divya Gupta	Martin Mehl Lauridsen	Tatsuaki Okamoto
Shai Halevi	Tancrede Lepoint	Claudio Orlandi
Brett Hemenway	Gaëtan Leurent	Rafail Ostrovsky
Nadia Heninger	Anthony Leverrier	Carles Padro
Javier Herranz	Benoit Libert	Jiaxin Pan
Ryo Hiromasa	Fuchun Lin	Omer Paneth
Shoichi Hirose	Zhen Liu	Saurabh Panjwani
Viet Tung Hoang	Steve Lu	Alain Passelègue
Justin Holmgren	Atul Luykx	Valerio Pastro
Naofumi Homma	Anna Lysyanskaya	Arpita Patra
Yan Huang	Vadim Lyubashevsky	Michaël Peeters
Vincenzo Iovino	Mohammad Mahmoody	Roel Peeters
Yuval Ishai	Antonio Marcedone	Chris Peikert
Zahra Jafargholi	Daniel Masny	Christopher Peikert
Tibor Jager	Alexander May	Olivier Pereira
Abhishek Jain	Willi Meier	Thomas Peters
Jrmy Jean	Carlos Aguilar Melchor	Duong Hieu Phan
Anthony Journault	Florian Mendel	Krzysztof Pietrzak
Saqib A. Kakvi	Bart Mennink	Benny Pinkas
Pierre Karpman	Peihan Miao	Oxana Poburinnaya
Elham Kashefi	Eric Miles	David Pointcheval
Aniket Kate	Brice Minaud	Joop van de Pol
Jonathan Katz	Kazuhiko Minematsu	Antigoni Polychriadiou
Stefan Katzenbeisser	Ilya Mironov	Christopher Portmann

Romain Poussier	Karn Seth	Damien Vergnaud
Manoj Prabhakaran	Yannick Seurin	Thomas Vidick
Emmanuel Prouff	Barak Shani	Jorge L. Villar
Orazio Puglisi	Kyoji Shibusaki	D. Vinayagamurthy
Elizabeth Quaglia	Adam Shull	Ivan Visconti
Kenneth Radke	Marcos A. Simplicio Jr.	Shabsi Walfish
Mario Di Raimondo	Luisa Siniscalchi	Michael Walter
Somindu C. Ramanna	Boris Skoric	Lei Wang
Vanishree Rao	Adam Smith	Meiqin Wang
Micha Ren	Douglas Stebila	Xiao Wang
Renato Renner	Igor Stepanovs	Hoeteck Wee
Oscar Reparaz	Marc Stoettinger	Carolyn Whinnall
Vincent Rijmen	Takeshi Sugawara	Daniel Wichs
Thomas Ristenpart	David Sutter	Cyrille Wiedling
Florentin Rochet	Daisuke Suzuki	David Wu
Phil Rogaway	Bjorn Tackmann	Keita Xagawa
Mike Rosulek	Katsuyuki Takashima	Sophia Yakoubov
Ron Rothblum	Sid Telang	Shota Yamada
Arnab Roy	Sidharth Telang	Takashi Yamakawa
Kikuchi Ryo	Stefano Tessaro	Jun Yan
Rei Safavi-Naini	Susan Thomson	Yanqing Yao
Amit Sahai	Mehdi Tibouch	Kazuki Yoneyama
Louis Salvail	Elmar Tischhauser	Yu Yu
Palash Sarkar	Toyohiro Tsurumaru	Samee Zahur
Yu Sasaki	Dominique Unruh	Mahdi Zamani
Alessandra Scafuro	Berkant Ustaoglu	Mark Zhandry
Benedikt Schmidt	Vinod Vaikuntanathan	Bingsheng Zhang
Tobias Schneider	Kerem Varici	Hong-Sheng Zhou
Peter Scholl	Vesselin Velichkov	Vassilis Zikas
Dominique Schroeder	M. Venkatasubramanian	
Gil Segev	Daniele Venturi	

# Contents – Part I

## Lattice-Based Cryptography

Sieving for Shortest Vectors in Lattices Using Angular Locality-Sensitive Hashing . . . . .	3
<i>Thijs Laarhoven</i>	
Coded-BKW: Solving LWE Using Lattice Codes . . . . .	23
<i>Qian Guo, Thomas Johansson, and Paul Stankovski</i>	
An Improved BKW Algorithm for LWE with Applications to Cryptography and Lattices . . . . .	43
<i>Paul Kirchner and Pierre-Alain Fouque</i>	
Provably Weak Instances of Ring-LWE . . . . .	63
<i>Yara Elias, Kristin E. Lauter, Ekin Ozman, and Katherine E. Stange</i>	

## Cryptanalytic Insights

Links Among Impossible Differential, Integral and Zero Correlation Linear Cryptanalysis . . . . .	95
<i>Bing Sun, Zhiqiang Liu, Vincent Rijmen, Ruilin Li, Lei Cheng, Qingju Wang, Hoda Alkhzaimi, and Chao Li</i>	
On Reverse-Engineering S-Boxes with Hidden Design Criteria or Structure. . . . .	116
<i>Alex Biryukov and Léo Perrin</i>	
Capacity and Data Complexity in Multidimensional Linear Attack . . . . .	141
<i>Jialin Huang, Serge Vaudenay, Xuejia Lai, and Kaisa Nyberg</i>	
Observations on the SIMON Block Cipher Family . . . . .	161
<i>Stefan Kölbl, Gregor Leander, and Tyge Tiessen</i>	

## Modes and Constructions

Tweaking Even-Mansour Ciphers . . . . .	189
<i>Benoît Cogliati, Rodolphe Lampe, and Yannick Seurin</i>	
Multi-key Security: The Even-Mansour Construction Revisited. . . . .	209
<i>Nicky Mouha and Atul Luykx</i>	

Reproducible Circularly-Secure Bit Encryption: Applications  
and Realizations . . . . . 224  
*Mohammad Hajiabadi and Bruce M. Kapron*

**Multilinear Maps and IO**

Zeroizing Without Low-Level Zeroes: New MMAP Attacks  
and Their Limitations . . . . . 247  
*Jean-Sébastien Coron, Craig Gentry, Shai Halevi, Tancrede Lepoint,  
Hemanta K. Maji, Eric Miles, Mariana Raykova, Amit Sahai,  
and Mehdi Tibouchi*

New Multilinear Maps Over the Integers . . . . . 267  
*Jean-Sébastien Coron, Tancrede Lepoint, and Mehdi Tibouchi*

Constant-Round Concurrent Zero-Knowledge from Indistinguishability  
Obfuscation . . . . . 287  
*Kai-Min Chung, Huijia Lin, and Rafael Pass*

Indistinguishability Obfuscation from Compact Functional Encryption . . . . . 308  
*Prabhanjan Ananth and Abhishek Jain*

**Pseudorandomness**

Efficient Pseudorandom Functions via On-the-Fly Adaptation . . . . . 329  
*Nico Döttling and Dominique Schröder*

The Iterated Random Permutation Problem with Applications  
to Cascade Encryption . . . . . 351  
*Brice Minaud and Yannick Seurin*

The Exact PRF Security of Truncation: Tight Bounds for Keyed Sponges  
and Truncated CBC. . . . . 368  
*Peter Gaži, Krzysztof Pietrzak, and Stefano Tessaro*

An Algebraic Framework for Pseudorandom Functions and Applications  
to Related-Key Security . . . . . 388  
*Michel Abdalla, Fabrice Benhamouda, and Alain Passelègue*

**Block Cipher Cryptanalysis**

Integral Cryptanalysis on Full MISTY1 . . . . . 413  
*Yosuke Todo*

New Attacks on Feistel Structures with Improved Memory Complexities . . . . . 433  
*Itai Dinur, Orr Dunkelman, Nathan Keller, and Adi Shamir*

Known-Key Distinguisher on Full PRESENT . . . . . 455  
*Céline Blondeau, Thomas Peyrin, and Lei Wang*

Key-Recovery Attack on the ASASA Cryptosystem with Expanding  
S-Boxes. . . . . 475  
*Henri Gilbert, Jérôme Plût, and Joana Treger*

**Integrity**

Online Authenticated-Encryption and its Nonce-Reuse Misuse-Resistance . . . 493  
*Viet Tung Hoang, Reza Reyhanitabar, Phillip Rogaway,  
and Damian Vizár*

Relational Hash: Probabilistic Hash for Verifying Relations, Secure  
Against Forgery and More . . . . . 518  
*Avradip Mandal and Arnab Roy*

Explicit Non-malleable Codes Against Bit-Wise Tampering  
and Permutations. . . . . 538  
*Shashank Agrawal, Divya Gupta, Hemanta K. Maji, Omkant Pandey,  
and Manoj Prabhakaran*

**Assumptions**

Cryptanalysis of the Co-ACD Assumption . . . . . 561  
*Pierre-Alain Fouque, Moon Sung Lee, Tancrede Lepoint,  
and Mehdi Tibouchi*

Last Fall Degree, HFE, and Weil Descent Attacks on ECDLP . . . . . 581  
*Ming-Deh A. Huang, Michiel Koster, and Sze Ling Ye*

A Quasipolynomial Reduction for Generalized Selective Decryption  
on Trees . . . . . 601  
*Georg Fuchsbauer, Zahra Jafarholi, and Krzysztof Pietrzak*

**Hash Functions and Stream Cipher Cryptanalysis**

Practical Free-Start Collision Attacks on 76-step SHA-1 . . . . . 623  
*Pierre Karpman, Thomas Peyrin, and Marc Stevens*

Fast Correlation Attacks over Extension Fields, Large-Unit Linear  
Approximation and Cryptanalysis of SNOW 2.0 . . . . . 643  
*Bin Zhang, Chao Xu, and Willi Meier*

Cryptanalysis of Full Sprout . . . . . 663  
*Virginie Lallemand and Maria Naya-Plasencia*

Higher-Order Differential Meet-in-the-middle Preimage Attacks on SHA-1 and BLAKE . . . . . 683  
*Thomas Espitau, Pierre-Alain Fouque, and Pierre Karpman*

**Implementations**

Decaf: Eliminating Cofactors Through Point Compression . . . . . 705  
*Mike Hamburg*

Actively Secure OT Extension with Optimal Overhead . . . . . 724  
*Marcel Keller, Emmanuela Orsini, and Peter Scholl*

Algebraic Decomposition for Probing Security . . . . . 742  
*Claude Carlet, Emmanuel Prouff, Matthieu Rivain, and Thomas Roche*

Consolidating Masking Schemes . . . . . 764  
*Oscar Reparaz, Begül Bilgin, Svetla Nikova, Benedikt Gierlichs, and Ingrid Verbauwhede*

**Author Index** . . . . . 785

## Contents – Part II

### Multiparty Computation I

A Simpler Variant of Universally Composable Security for Standard Multiparty Computation . . . . .	3
<i>Ran Canetti, Asaf Cohen, and Yehuda Lindell</i>	
Concurrent Secure Computation via Non-Black Box Simulation . . . . .	23
<i>Vipul Goyal, Divya Gupta, and Amit Sahai</i>	
Concurrent Secure Computation with Optimal Query Complexity . . . . .	43
<i>Ran Canetti, Vipul Goyal, and Abhishek Jain</i>	
Constant-Round MPC with Fairness and Guarantee of Output Delivery . . . . .	63
<i>S. Dov Gordon, Feng-Hao Liu, and Elaine Shi</i>	

### Zero-Knowledge

Statistical Concurrent Non-malleable Zero-Knowledge from One-Way Functions . . . . .	85
<i>Susumu Kiyoshima</i>	
Implicit Zero-Knowledge Arguments and Applications to the Malicious Setting . . . . .	107
<i>Fabrice Benhamouda, Geoffroy Couteau, David Pointcheval, and Hoeteck Wee</i>	
Impossibility of Black-Box Simulation Against Leakage Attacks . . . . .	130
<i>Rafail Ostrovsky, Giuseppe Persiano, and Ivan Visconti</i>	
Efficient Zero-Knowledge Proofs of Non-algebraic Statements with Sublinear Amortized Cost . . . . .	150
<i>Zhangxiang Hu, Payman Mohassel, and Mike Rosulek</i>	

### Theory

Parallel Hashing via List Recoverability . . . . .	173
<i>Iftach Haitner, Yuval Ishai, Eran Omri, and Ronen Shaltiel</i>	
Cryptography with One-Way Communication . . . . .	191
<i>Sanjam Garg, Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai</i>	

(Almost) Optimal Constructions of UOWHFs from 1-to-1, Regular One-Way Functions and Beyond. . . . . 209  
*Yu Yu, Dawu Gu, Xiangxue Li, and Jian Weng*

**Signatures**

Practical Round-Optimal Blind Signatures in the Standard Model . . . . . 233  
*Georg Fuchsbauer, Christian Hanser, and Daniel Slamanig*

Programmable Hash Functions Go Private: Constructions and Applications to (Homomorphic) Signatures with Shorter Public Keys. . . . . 254  
*Dario Catalano, Dario Fiore, and Luca Nizzardo*

Structure-Preserving Signatures from Standard Assumptions, Revisited . . . . . 275  
*Eike Kiltz, Jiaxin Pan, and Hoeteck Wee*

Short Group Signatures via Structure-Preserving Signatures: Standard Model Security from Simple Assumptions . . . . . 296  
*Benoît Libert, Thomas Peters, and Moti Yung*

**Multiparty Computation II**

Efficient Constant Round Multi-party Computation Combining BMR and SPDZ . . . . . 319  
*Yehuda Lindell, Benny Pinkas, Nigel P. Smart, and Avishay Yanai*

Round-Optimal Black-Box Two-Party Computation. . . . . 339  
*Rafail Ostrovsky, Silas Richelson, and Alessandra Scafuro*

Secure Computation with Minimal Interaction, Revisited . . . . . 359  
*Yuval Ishai, Ranjit Kumaresan, Eyal Kushilevitz, and Anat Paskin-Cherniavsky*

PoW-Based Distributed Cryptography with No Trusted Setup. . . . . 379  
*Marcin Andrychowicz and Stefan Dziembowski*

**Non-signaling and Information-Theoretic Crypto**

Multi-prover Commitments Against Non-signaling Attacks. . . . . 403  
*Serge Fehr and Max Fillinger*

Arguments of Proximity [Extended Abstract] . . . . . 422  
*Yael Tauman Kalai and Ron D. Rothblum*

Distributions Attaining Secret Key at a Rate of the Conditional Mutual Information . . . . . 443  
*Eric Chitambar, Benjamin Fortescue, and Min-Hsiu Hsieh*



Privacy with Imperfect Randomness . . . . . 463  
*Yevgeniy Dodis and Yanqing Yao*

**Attribute-Based Encryption**

Communication Complexity of Conditional Disclosure of Secrets  
 and Attribute-Based Encryption . . . . . 485  
*Romain Gay, Iordanis Kerenidis, and Hoeteck Wee*

Predicate Encryption for Circuits from LWE . . . . . 503  
*Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee*

Bilinear Entropy Expansion from the Decisional Linear Assumption . . . . . 524  
*Lucas Kowalczyk and Allison Bishop Lewko*

**New Primitives**

Data Is a Stream: Security of Stream-Based Channels . . . . . 545  
*Marc Fischlin, Felix Günther, Giorgia Azzurra Marson,  
 and Kenneth G. Paterson*

Bloom Filters in Adversarial Environments . . . . . 565  
*Moni Naor and Eylon Yosev*

Proofs of Space . . . . . 585  
*Stefan Dziembowski, Sebastian Faust, Vladimir Kolmogorov,  
 and Krzysztof Pietrzak*

**Fully Homomorphic/Functional Encryption**

Quantum Homomorphic Encryption for Circuits of Low T-gate Complexity . . . . . 609  
*Anne Broadbent and Stacey Jeffery*

Multi-identity and Multi-key Leveled FHE from Learning with Errors . . . . . 630  
*Michael Clear and Ciarán McGoldrick*

From Selective to Adaptive Security in Functional Encryption . . . . . 657  
*Prabhanjan Ananth, Zvika Brakerski, Gil Segev,  
 and Vinod Vaikuntanathan*

A Punctured Programming Approach to Adaptively Secure Functional  
 Encryption . . . . . 678  
*Brent Waters*

**Multiparty Computation III**

Secure Computation from Leaky Correlated Randomness . . . . . 701  
*Divya Gupta, Yuval Ishai, Hemanta K. Maji, and Amit Sahai*

Efficient Multi-party Computation: From Passive to Active Security via Secure SIMD Circuits . . . . .	721
<i>Daniel Genkin, Yuval Ishai, and Antigoni Polychroniadou</i>	
Large-Scale Secure Computation: Multi-party Computation for (Parallel) RAM Programs. . . . .	742
<i>Elette Boyle, Kai-Min Chung, and Rafael Pass</i>	
Incoercible Multi-party Computation and Universally Composable Receipt-Free Voting . . . . .	763
<i>Joël Alwen, Rafail Ostrovsky, Hong-Sheng Zhou, and Vassilis Zikas</i>	
<b>Author Index</b> . . . . .	781

# **Lattice-Based Cryptography**

# Sieving for Shortest Vectors in Lattices Using Angular Locality-Sensitive Hashing

Thijs Laarhoven<sup>(✉)</sup>

Department of Mathematics and Computer Science,  
Eindhoven University of Technology, Eindhoven, The Netherlands  
mail@thijs.com

**Abstract.** By replacing the brute-force list search in sieving algorithms with Charikar’s angular locality-sensitive hashing (LSH) method, we get both theoretical and practical speedups for solving the shortest vector problem (SVP) on lattices. Combining angular LSH with a variant of Nguyen and Vidick’s heuristic sieve algorithm, we obtain heuristic time and space complexities for solving SVP of  $2^{0.3366n+o(n)}$  and  $2^{0.2075n+o(n)}$  respectively, while combining the same hash family with Micciancio and Voulgaris’ GaussSieve algorithm leads to an algorithm with (conjectured) heuristic time and space complexities of  $2^{0.3366n+o(n)}$ . Experiments with the GaussSieve-variant show that in moderate dimensions the proposed HashSieve algorithm already outperforms the GaussSieve, and the practical increase in the space complexity is much smaller than the asymptotic bounds suggest, and can be further reduced with probing. Extrapolating to higher dimensions, we estimate that a fully optimized and parallelized implementation of the GaussSieve-based HashSieve algorithm might need a few core years to solve SVP in dimension 130 or even 140.

**Keywords:** Lattices · Shortest vector problem (SVP) · Sieving algorithms · Approximate nearest neighbor problem · Locality-sensitive hashing (LSH)

## 1 Introduction

*Lattice Cryptography.* Over the past few decades, lattice-based cryptography has attracted wide attention from the cryptographic community, due to e.g. its presumed resistance against quantum attacks [10], average-case hardness guarantees [3], the existence of lattice-based fully homomorphic encryption schemes [16], and efficient cryptographic primitives like NTRU [17]. An important problem related to lattice cryptography is to estimate the hardness of the underlying hard lattice problems, such as finding short vectors; a good understanding is critical for accurately choosing parameters in lattice cryptography [28, 39].

*Finding Short Vectors.* Given a basis  $\{\mathbf{b}_1, \dots, \mathbf{b}_n\} \subset \mathbb{R}^n$  of an  $n$ -dimensional lattice  $\mathcal{L} = \sum_{i=1}^n \mathbb{Z}\mathbf{b}_i$ , finding a shortest non-zero lattice vector (with respect to the Euclidean norm) or approximating it up to a constant factor is well-known

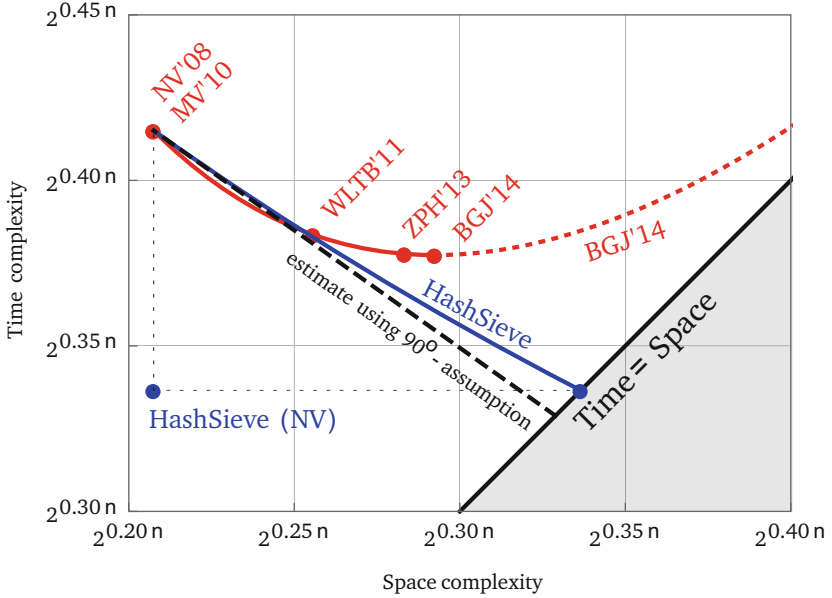
to be NP-hard under randomized reductions [4, 21]. For large approximation factors, various fast algorithms for finding short vectors are known, such as the lattice basis reduction algorithms LLL [26] and BKZ [43, 44]. The latter has a block-size parameter  $\beta$  which can be tuned to obtain a trade-off between the time complexity and the quality of the output; the higher  $\beta$ , the longer the algorithm takes and the shorter the vectors in the output basis. BKZ uses an algorithm for solving the exact shortest vector problem (SVP) in lattices of dimension  $\beta$  as a subroutine, and the runtime of BKZ largely depends on the runtime of this subroutine. Estimating the complexity of solving exact SVP therefore has direct consequences for the estimated hardness of solving approximate SVP with BKZ.

*Finding Shortest Vectors.* In the original description of BKZ, enumeration was used as the SVP subroutine [14, 20, 38, 44]. This method has a low (polynomial) space complexity, but its runtime is superexponential ( $2^{\Omega(n \log n)}$ ), which is known to be suboptimal: sieving [5], the Voronoi cell algorithm [32], and the recent discrete Gaussian sampling approach [2] all run in single exponential time ( $2^{O(n)}$ ). The main drawbacks of the latter methods are that their space complexities are exponential in  $n$  as well, and due to larger hidden constants in the exponents enumeration is commonly still considered more practical than these other methods in moderate dimensions  $n$  [34].

*Sieving in Arbitrary Lattices.* On the other hand, these other SVP algorithms are relatively new, and recent improvements have shown that at least sieving may be able to compete with enumeration in the future. While the original work of Ajtai et al. [5] showed only that sieving solves SVP in time and space  $2^{O(n)}$ , later work showed that one can provably solve SVP in arbitrary lattices in time  $2^{2.47n+o(n)}$  and space  $2^{1.24n+o(n)}$  [35, 40]. Heuristic analyses of sieving algorithms further suggest that one may be able to solve SVP in time  $2^{0.42n+o(n)}$  and space  $2^{0.21n+o(n)}$  [7, 33, 35], or optimizing for time, in time  $2^{0.38n+o(n)}$  and space  $2^{0.29n+o(n)}$  [7, 45, 46]. Other works have shown how to speed up sieving in practice [11, 15, 19, 29, 30, 41], and sieving recently made its way to the top 25 of the SVP challenge hall of fame [42], using the GaussSieve algorithm [23, 33].

*Sieving in Ideal Lattices.* The potential of sieving is further illustrated by recent results in ideal lattices [11, 19]; while it is not known how to use the additional structure in ideal lattices (commonly used in lattice cryptography) for enumeration or other SVP algorithms, sieving does admit significant polynomial speedups for ideal lattices, and the GaussSieve was recently used to solve SVP on an ideal lattice in dimension 128 [11, 19, 37]. This is higher than the highest dimension for which enumeration was used to find a record in either lattice challenge [37, 42], which further illustrates the potential of sieving and the possible impact of further improvements to sieving and, in particular, the GaussSieve algorithm.

**Contributions.** In this work we show how to obtain exponential trade-offs and speedups for sieving using (angular) locality-sensitive hashing [12, 18], a technique from the field of nearest neighbor searching. In short, for each list vector  $\mathbf{w}$  we store low-dimensional, lossy *sketches* (hashes), such that vectors that



**Fig. 1.** The heuristic space-time trade-off of various heuristic sieves from the literature (red), and the heuristic trade-off between the space and time complexities obtained with the HashSieve (blue curve). For the NV-sieve, we can further process the hash tables sequentially to obtain a speedup rather than a trade-off (blue point). The dashed, gray line shows the estimate for the space-time trade-off of the HashSieve obtained by assuming that all reduced vectors are orthogonal (cf. Proposition 1). The referenced works are: NV’08 [35]; MV’10 [33]; WLTB’11 [45]; ZPH’13 [46]; BGJ’14 [7] (Color figure online).

are nearby have a higher probability of having the same sketch (hash value) than vectors which are far apart. To search the list for nearby vectors we then do not go through the entire list of lattice vectors, but only consider those vectors that have at least one matching sketch (hash value) in one of the hash tables. Storing all list vectors in exponentially many hash tables requires exponentially more space, but searching for nearby vectors can then be done exponentially faster as well, as many distant vectors are not considered for reductions. Optimizing for time, the resulting HashSieve algorithm has heuristic time and space complexities both bounded by  $2^{0.3366n+o(n)}$ , while tuning the parameters differently, we get a continuous heuristic trade-off between the space and time complexities as illustrated by the solid blue curve in Fig. 1.

*From a Tradeoff to a Speedup.* Applying angular LSH to a variant of the Nguyen-Vidick sieve [35], we further obtain an algorithm with heuristic time and space complexities of  $2^{0.3366n+o(n)}$  and  $2^{0.2075n+o(n)}$  respectively, as illustrated by the blue point in Fig. 1. The key observation is that the hash tables of the HashSieve can be processed sequentially (similar to [8]), storing one hash table at a time. The resulting algorithm achieves the same heuristic speed-up, but the asymptotic

space complexity remains the same as in the original NV-sieve algorithm. This improvement is explained in detail in the full version. Note that this speedup does not appear to be compatible with the GaussSieve and only works with the NV-sieve, which may make the resulting algorithm slower in moderate dimensions, even though the memory used is much smaller.

*Experimental Results.* Practical experiments with the (GaussSieve-based) HashSieve algorithm validate our heuristic analysis, and show that (i) already in low dimensions, the HashSieve outperforms the GaussSieve; and (ii) the increase in the space complexity is significantly smaller than one might guess from only looking at the leading exponent of the space complexity. We also show how to further reduce the space complexity at almost no cost by a technique called probing, which reduces the required number of hash tables by a factor  $\text{poly}(n)$ . In the end, these results will be an important guide for estimating the hardness of exact SVP in moderate dimensions, and for the hardness of approximate SVP in high dimensions using BKZ with sieving as the SVP subroutine.

*Main Ideas.* While the use of LSH was briefly considered in the context of sieving by Nguyen and Vidick [35, Sect. 4.2.2], there are two main differences:

- Nguyen and Vidick considered LSH families based on *Euclidean distances* [6], while we will argue that it seems more natural to consider hash families based on *angular distances* or *cosine similarities* [12].
- Nguyen and Vidick focused on the *worst-case* difference between nearby and faraway vectors, while we will focus on the *average-case* difference.

To illustrate the second point: the *smallest* angle between pairwise reduced vectors in the GaussSieve may be only slightly bigger than  $60^\circ$  (i.e. hardly any bigger than angles of non-reduced vectors), while in high dimensions the *average* angle between two pairwise reduced vectors is actually close to  $90^\circ$ .

*Outlook.* Although this work focuses on applying angular LSH to sieving, more generally this work could be considered the first to succeed in applying LSH to lattice algorithms. Various recent follow-up works have already further investigated the use of different LSH methods and other nearest neighbor search methods in the context of lattice sieving [8, 9, 25, 31], and an open problem is whether other lattice algorithms (e.g. provable sieving algorithms, the Voronoi cell algorithm) may benefit from related techniques as well.

**Roadmap.** In Sect. 2 we describe the technique of (angular) LSH for finding near(est) neighbors, and Sect. 3 describes how to apply these techniques to the GaussSieve. Section 4 states the main result regarding the time and space complexities of sieving using angular LSH, and describes the technique of probing. In Sect. 5 we finally describe experiments performed using the GaussSieve-based HashSieve, and possible consequences for the estimated complexity of SVP in high dimensions. The full version [24] contains details on how angular LSH may be combined with the NV-sieve, and how the memory can be reduced to obtain a memory-wise asymptotically superior NV-sieve-based HashSieve.

## 2 Locality-Sensitive Hashing

### 2.1 Introduction

The near(est) neighbor problem is the following [18]: Given a list of  $n$ -dimensional vectors  $L = \{\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_N\} \subset \mathbb{R}^n$ , preprocess  $L$  in such a way that, when later given a target vector  $\mathbf{v} \notin L$ , one can efficiently find an element  $\mathbf{w} \in L$  which is close(st) to  $\mathbf{v}$ . While in low (fixed) dimensions  $n$  there may be ways to answer these queries in time sub-linear or even logarithmic in the list size  $N$ , in high dimensions it seems hard to do better than with a naive brute-force list search of time  $O(N)$ . This inability to efficiently store and query lists of high-dimensional objects is sometimes referred to as the “curse of dimensionality” [18].

Fortunately, if we know that the list of objects  $L$  has a certain structure, or if we know that there is a significant gap between what is meant by “nearby” and “far away,” then there are ways to preprocess  $L$  such that queries can be answered in time sub-linear in  $N$ . For instance, for the Euclidean norm, if it is known that the closest point  $\mathbf{w}^* \in L$  lies at distance  $\|\mathbf{v} - \mathbf{w}^*\| = r_1$ , and all other points  $\mathbf{w} \in L$  are at distance at least  $\|\mathbf{v} - \mathbf{w}\| \geq r_2 = (1 + \varepsilon)r_1$  from  $\mathbf{v}$ , then it is possible to preprocess  $L$  using time and space  $O(N^{1+\rho})$ , and answer queries in time  $O(N^\rho)$ , where  $\rho = (1 + \varepsilon)^{-2} < 1$  [6]. For  $\varepsilon > 0$ , this corresponds to a sub-linear time and sub-quadratic (super-linear) space complexity in  $N$ .

### 2.2 Hash Families

The method of [6] described above, as well as the method we will use later, relies on using *locality-sensitive hash functions* [18]. These are functions  $h$  which map an  $n$ -dimensional vector  $\mathbf{v}$  to a low-dimensional *sketch* of  $\mathbf{v}$ , such that vectors which are nearby in  $\mathbb{R}^n$  have a high probability of having the same sketch, while vectors which are far away have a low probability of having the same image under  $h$ . Formalizing this property leads to the following definition of a *locality-sensitive hash family*  $\mathcal{H}$ . Here, we assume  $D$  is a certain similarity measure<sup>1</sup>, and the set  $U$  below may be thought of as (a subset of) the natural numbers  $\mathbb{N}$ .

**Definition 1.** [18] A family  $\mathcal{H} = \{h : \mathbb{R}^n \rightarrow U\}$  is called  $(r_1, r_2, p_1, p_2)$ -sensitive for a similarity measure  $D$  if for any  $\mathbf{v}, \mathbf{w} \in \mathbb{R}^n$  we have

- If  $D(\mathbf{v}, \mathbf{w}) \leq r_1$  then  $\mathbb{P}_{h \in \mathcal{H}}[h(\mathbf{v}) = h(\mathbf{w})] \geq p_1$ .
- If  $D(\mathbf{v}, \mathbf{w}) \geq r_2$  then  $\mathbb{P}_{h \in \mathcal{H}}[h(\mathbf{v}) = h(\mathbf{w})] \leq p_2$ .

Note that if we are given a hash family  $\mathcal{H}$  which is  $(r_1, r_2, p_1, p_2)$ -sensitive with  $p_1 \gg p_2$ , then we can use  $\mathcal{H}$  to distinguish between vectors which are at most  $r_1$  away from  $\mathbf{v}$ , and vectors which are at least  $r_2$  away from  $\mathbf{v}$  with non-negligible probability, by only looking at their hash values (and that of  $\mathbf{v}$ ).

---

<sup>1</sup> A similarity measure  $D$  may informally be thought of as a “slightly relaxed” distance metric, which may not satisfy all properties associated to distance metrics.



### 2.3 Amplification

Before turning to how such hash families may actually be constructed or used to find nearest neighbors, note that in general it is unknown whether efficiently computable  $(r_1, r_2, p_1, p_2)$ -sensitive hash families even exist for the ideal setting of  $r_1 \approx r_2$  and  $p_1 \approx 1$  and  $p_2 \approx 0$ . Instead, one commonly first constructs an  $(r_1, r_2, p_1, p_2)$ -sensitive hash family  $\mathcal{H}$  with  $p_1 \approx p_2$ , and then uses several AND- and OR-compositions to turn it into an  $(r_1, r_2, p'_1, p'_2)$ -sensitive hash family  $\mathcal{H}'$  with  $p'_1 > p_1$  and  $p'_2 < p_2$ , thereby amplifying the gap between  $p_1$  and  $p_2$ .

**AND-composition.** Given an  $(r_1, r_2, p_1, p_2)$ -sensitive hash family  $\mathcal{H}$ , we can construct an  $(r_1, r_2, p_1^k, p_2^k)$ -sensitive hash family  $\mathcal{H}'$  by taking  $k$  different, pairwise independent functions  $h_1, \dots, h_k \in \mathcal{H}$  and a one-to-one mapping  $f : U^k \rightarrow U$ , and defining  $h \in \mathcal{H}'$  as  $h(\mathbf{v}) = f(h_1(\mathbf{v}), \dots, h_k(\mathbf{v}))$ . Clearly  $h(\mathbf{v}) = h(\mathbf{w})$  iff  $h_i(\mathbf{v}) = h_i(\mathbf{w})$  for all  $i \in [k]$ , so if  $\mathbb{P}[h_i(\mathbf{v}) = h_i(\mathbf{w})] = p_j$  for all  $i$ , then  $\mathbb{P}[h(\mathbf{v}) = h(\mathbf{w})] = p_j^k$  for  $j = 1, 2$ .

**OR-composition.** Given an  $(r_1, r_2, p_1, p_2)$ -sensitive hash family  $\mathcal{H}$ , we can construct an  $(r_1, r_2, 1 - (1 - p_1)^t, 1 - (1 - p_2)^t)$ -sensitive hash family  $\mathcal{H}'$  by taking  $t$  different, pairwise independent functions  $h_1, \dots, h_t \in \mathcal{H}$ , and defining  $h \in \mathcal{H}'$  by the relation  $h(\mathbf{v}) = h(\mathbf{w})$  iff  $h_i(\mathbf{v}) = h_i(\mathbf{w})$  for *at least one*  $i \in [t]$ . Clearly  $h(\mathbf{v}) \neq h(\mathbf{w})$  iff  $h_i(\mathbf{v}) \neq h_i(\mathbf{w})$  for all  $i \in [t]$ , so if  $\mathbb{P}[h_i(\mathbf{v}) \neq h_i(\mathbf{w})] = 1 - p_j$  for all  $i$ , then  $\mathbb{P}[h(\mathbf{v}) \neq h(\mathbf{w})] = (1 - p_j)^t$  for  $j = 1, 2$ .<sup>2</sup>

Combining a  $k$ -wise AND-composition with a  $t$ -wise OR-composition, we can turn an  $(r_1, r_2, p_1, p_2)$ -sensitive hash family  $\mathcal{H}$  into an  $(r_1, r_2, 1 - (1 - p_1^k)^t, 1 - (1 - p_2^k)^t)$ -sensitive hash family  $\mathcal{H}'$  as follows:

$$(r_1, r_2, p_1, p_2) \xrightarrow{k\text{-AND}} (r_1, r_2, p_1^k, p_2^k) \xrightarrow{t\text{-OR}} (r_1, r_2, (1 - p_1^k)^t, (1 - p_2^k)^t).$$

As long as  $p_1 > p_2$ , we can always find values  $k$  and  $t$  such that  $p_1^* = 1 - (1 - p_1^k)^t \approx 1$  is close to 1 and  $p_2^* = 1 - (1 - p_2^k)^t \approx 0$  is very small.

### 2.4 Finding Nearest Neighbors

To use these hash families to find nearest neighbors, we may use the following method first described in [18]. First, we choose  $t \cdot k$  random hash functions  $h_{i,j} \in \mathcal{H}$ , and we use the AND-composition to combine  $k$  of them at a time to build  $t$  different hash functions  $h_1, \dots, h_t$ . Then, given the list  $L$ , we build  $t$  different hash tables  $T_1, \dots, T_t$ , where for each hash table  $T_i$  we insert  $\mathbf{w}$  into the bucket labeled  $h_i(\mathbf{w})$ . Finally, given the vector  $\mathbf{v}$ , we compute its  $t$  images  $h_i(\mathbf{v})$ , gather all the candidate vectors that collide with  $\mathbf{v}$  in at least one of these hash tables (an OR-composition) in a list of candidates, and search this set of candidates for a nearest neighbor.

Clearly, the quality of this algorithm for finding nearest neighbors depends on the quality of the underlying hash family  $\mathcal{H}$  and on the parameters  $k$  and  $t$ .

<sup>2</sup> Note that  $h$  is strictly not a function and only defines a relation.

Larger values of  $k$  and  $t$  amplify the gap between the probabilities of finding ‘good’ (nearby) and ‘bad’ (faraway) vectors, which makes the list of candidates shorter, but larger parameters come at the cost of having to compute many hashes (both during the preprocessing and querying phases) and having to store many hash tables in memory. The following lemma shows how to balance  $k$  and  $t$  so that the overall time complexity is minimized.

**Lemma 1.** [18] *Suppose there exists a  $(r_1, r_2, p_1, p_2)$ -sensitive hash family  $\mathcal{H}$ . Then, for a list  $L$  of size  $N$ , taking*

$$\rho = \frac{\log(1/p_1)}{\log(1/p_2)}, \quad k = \frac{\log(N)}{\log(1/p_2)}, \quad t = O(N^\rho), \quad (1)$$

*with high probability we can either (a) find an element  $\mathbf{w}^* \in L$  that satisfies  $D(\mathbf{v}, \mathbf{w}^*) \leq r_2$ , or (b) conclude that with high probability, no elements  $\mathbf{w} \in L$  with  $D(\mathbf{v}, \mathbf{w}) > r_1$  exist, with the following costs:*

- (1) *Time for preprocessing the list:  $\tilde{O}(kN^{1+\rho})$ .*
- (2) *Space complexity of the preprocessed data:  $\tilde{O}(N^{1+\rho})$ .*
- (3) *Time for answering a query  $\mathbf{v}$ :  $\tilde{O}(N^\rho)$ .*
  - (3a) *Hash evaluations of the query vector  $\mathbf{v}$ :  $O(N^\rho)$ .*
  - (3b) *List vectors to compare to the query vector  $\mathbf{v}$ :  $O(N^\rho)$ .*

Although Lemma 1 only shows how to choose  $k$  and  $t$  to minimize the time complexity, we can also tune  $k$  and  $t$  so that we use more time and less space. In a way this algorithm can be seen as a generalization of the naive brute-force search solution for finding nearest neighbors, as  $k = 0$  and  $t = 1$  corresponds to checking the whole list for nearby vectors in linear time and linear space.

## 2.5 Angular Hashing

Let us now consider actual hash families for the similarity measure  $D$  that we are interested in. As argued in the next section, what seems a more natural choice for  $D$  than the Euclidean distance is the *angular distance*, defined on  $\mathbb{R}^n$  as

$$D(\mathbf{v}, \mathbf{w}) = \theta(\mathbf{v}, \mathbf{w}) = \arccos\left(\frac{\mathbf{v}^T \mathbf{w}}{\|\mathbf{v}\| \cdot \|\mathbf{w}\|}\right). \quad (2)$$

With this similarity measure, two vectors are ‘nearby’ if their common angle is small, and ‘far apart’ if their angle is large. In a sense, this is similar to the Euclidean norm: if two vectors have similar Euclidean norms, their distance is large iff their angular distance is large. For this similarity measure  $D$ , the following hash family  $\mathcal{H}$  was first described in [12]:

$$\mathcal{H} = \{h_{\mathbf{a}} : \mathbf{a} \in \mathbb{R}^n, \|\mathbf{a}\| = 1\}, \quad h_{\mathbf{a}}(\mathbf{v}) \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } \mathbf{a}^T \mathbf{v} \geq 0; \\ 0 & \text{if } \mathbf{a}^T \mathbf{v} < 0. \end{cases} \quad (3)$$

Intuitively, the vector  $\mathbf{a}$  defines a hyperplane (for which  $\mathbf{a}$  is a normal vector), and  $h_{\mathbf{a}}$  maps the two regions separated by this hyperplane to different bits.

To see why this is a non-trivial locality-sensitive hash family for the angular distance, consider two vectors  $\mathbf{v}, \mathbf{w} \in \mathbb{R}^n$ . These two vectors lie on a two-dimensional plane passing through the origin, and with probability 1 a hash vector  $\mathbf{a}$  does not lie on this plane (for  $n > 2$ ). This means that the hyperplane defined by  $\mathbf{a}$  intersects this plane in some line  $\ell$ . Since  $\mathbf{a}$  is taken uniformly at random from the unit sphere, the line  $\ell$  has a uniformly random ‘direction’ in the plane, and maps  $\mathbf{v}$  and  $\mathbf{w}$  to different hash values iff  $\ell$  separates  $\mathbf{v}$  and  $\mathbf{w}$  in the plane. Therefore the probability that  $h(\mathbf{v}) \neq h(\mathbf{w})$  is directly proportional to their common angle  $\theta(\mathbf{v}, \mathbf{w})$  as follows [12]:

$$\mathbb{P}_{h_{\mathbf{a}} \in \mathcal{H}}[h_{\mathbf{a}}(\mathbf{v}) \neq h_{\mathbf{a}}(\mathbf{w})] = 1 - \frac{\theta(\mathbf{v}, \mathbf{w})}{\pi}. \quad (4)$$

So for any two angles  $\theta_1 < \theta_2$ , the family  $\mathcal{H}$  is  $(\theta_1, \theta_2, 1 - \frac{\theta_1}{\pi}, 1 - \frac{\theta_2}{\pi})$ -sensitive. In particular, Charikar’s hyperplane hash family is  $(\frac{\pi}{3}, \frac{\pi}{2}, \frac{2}{3}, \frac{1}{2})$ -sensitive.

### 3 From the GaussSieve to the HashSieve

Let us now describe how locality-sensitive hashing can be used to speed up sieving algorithms, and in particular how we can speed up the GaussSieve of Micciancio and Voulgaris [33]. We have chosen this algorithm as our main focus since it seems to be the most practical sieving algorithm to date, which is further motivated by the extensive attention it has received in recent years [15, 19, 23, 29, 30, 41] and by the fact that the highest sieving record in the SVP challenge database was obtained using (a modification of) the GaussSieve [23, 42]. Note that the same ideas can also be applied to the Nguyen-Vidick sieve [35], which has proven complexity bounds. Details on this combination are in the full version.

#### 3.1 The GaussSieve Algorithm

A simplified version of the GaussSieve algorithm of Micciancio and Voulgaris is described in Algorithm 1. The algorithm iteratively builds a longer and longer list  $L$  of lattice vectors, occasionally reducing the lengths of list vectors in the process, until at some point this list  $L$  contains a shortest vector. Vectors are sampled from a discrete Gaussian over the lattice, using e.g. the sampling algorithm of Klein [22, 33], or popped from the stack. If list vectors are modified or newly sampled vectors are reduced, they are pushed to the stack.

In the GaussSieve, the reductions in Lines 5 and 6 follow the rule:

$$\text{Reduce } \mathbf{u}_1 \text{ with } \mathbf{u}_2 : \quad \text{if } \|\mathbf{u}_1 \pm \mathbf{u}_2\| < \|\mathbf{u}_1\| \text{ then } \mathbf{u}_1 \leftarrow \mathbf{u}_1 \pm \mathbf{u}_2. \quad (5)$$

Throughout the execution of the algorithm, the list  $L$  is always pairwise reduced w.r.t. (5), i.e.,  $\|\mathbf{w}_1 \pm \mathbf{w}_2\| \geq \max\{\|\mathbf{w}_1\|, \|\mathbf{w}_2\|\}$  for all  $\mathbf{w}_1, \mathbf{w}_2 \in L$ . This implies that two list vectors  $\mathbf{w}_1, \mathbf{w}_2 \in L$  always have an angle of at least  $60^\circ$ ; otherwise

**Algorithm 1.** The GaussSieve algorithm (simplified)

---

```

1: Initialize an empty list  $L$  and an empty stack  $S$ 
2: repeat
3:   Get a vector  $v$  from the stack (or sample a new one if  $S = \emptyset$ )
4:   for each  $w \in L$  do
5:     Reduce  $v$  with  $w$ 
6:     Reduce  $w$  with  $v$ 
7:     if  $w$  has changed then
8:       Remove  $w$  from the list  $L$ 
9:       Add  $w$  to the stack  $S$  (unless  $w = \mathbf{0}$ )
10:  if  $v$  has changed then
11:    Add  $v$  to the stack  $S$  (unless  $v = \mathbf{0}$ )
12:  else
13:    Add  $v$  to the list  $L$ 
14: until  $v$  is a shortest vector

```

---

one of them would have been used to reduce the other before being added to the list. Since all angles between list vectors are always at least  $60^\circ$ , the size of  $L$  is bounded by the *kissing constant* in dimension  $n$ : the maximum number of vectors in  $\mathbb{R}^n$  one can find such that any two vectors have an angle of at least  $60^\circ$ . Bounds and conjectures on the kissing constant in high dimensions lead us to believe that the size of the list  $L$  will therefore not exceed  $2^{0.2075n+o(n)}$  [13].

While the space complexity of the GaussSieve is reasonably well understood, there are no proven bounds on the time complexity of this algorithm. One might estimate that the time complexity is determined by the double loop over  $L$ : at any time each pair of vectors  $w_1, w_2 \in L$  was compared at least once to see if one could reduce the other, so the time complexity is at least quadratic in  $|L|$ . The algorithm further seems to show a similar asymptotic behavior as the NV-sieve [35], for which the asymptotic time complexity is heuristically known to be quadratic in  $|L|$ , i.e., of the order  $2^{0.415n+o(n)}$ . One might therefore conjecture that the GaussSieve also has a time complexity of  $2^{0.415n+o(n)}$ , which closely matches previous experiments with the GaussSieve in high dimensions [23].

### 3.2 The GaussSieve with Angular Reductions

Since the heuristic bounds on the space and time complexities are only based on the fact that each pair of vectors  $w_1, w_2 \in L$  has an angle of at least  $60^\circ$ , the same heuristics apply to any reduction method that guarantees that angles between vectors in  $L$  are at least  $60^\circ$ . In particular, if we reduce vectors only if their angle is at most  $60^\circ$  using the following rule:

Reduce  $u_1$  with  $u_2$  :

$$\text{if } \theta(u_1, \pm u_2) < 60^\circ \text{ and } \|u_1\| \geq \|u_2\| \text{ then } u_1 \leftarrow u_1 \pm u_2, \quad (6)$$

then we expect the same heuristic bounds on the time and space complexities to apply. More precisely, the list size would again be bounded by  $2^{0.208n+o(n)}$ ,

---

**Algorithm 2.** The GaussSieve-based HashSieve algorithm

---

```

1: Initialize an empty list  $L$  and an empty stack  $S$ 
2: Initialize  $t$  empty hash tables  $T_i$ 
3: Sample  $k \cdot t$  random hash vectors  $\mathbf{a}_{i,j}$ 
4: repeat
5:   Get a vector  $\mathbf{v}$  from the stack (or sample a new one if  $S = \emptyset$ )
6:   Obtain the set of candidates  $C = \bigcup_{i=1}^t T_i[h_i(\mathbf{v})]$ 
7:   for each  $\mathbf{w} \in C$  do
8:     Reduce  $\mathbf{v}$  with  $\mathbf{w}$ 
9:     Reduce  $\mathbf{w}$  with  $\mathbf{v}$ 
10:    if  $\mathbf{w}$  has changed then
11:      Remove  $\mathbf{w}$  from the list  $L$ 
12:      Remove  $\mathbf{w}$  from all  $t$  hash tables  $T_i$ 
13:      Add  $\mathbf{w}$  to the stack  $S$  (unless  $\mathbf{w} = \mathbf{0}$ )
14:    if  $\mathbf{v}$  has changed then
15:      Add  $\mathbf{v}$  to the stack  $S$  (unless  $\mathbf{v} = \mathbf{0}$ )
16:    else
17:      Add  $\mathbf{v}$  to the list  $L$ 
18:      Add  $\mathbf{v}$  to all  $t$  hash tables  $T_i$ 
19: until  $\mathbf{v}$  is a shortest vector

```

---

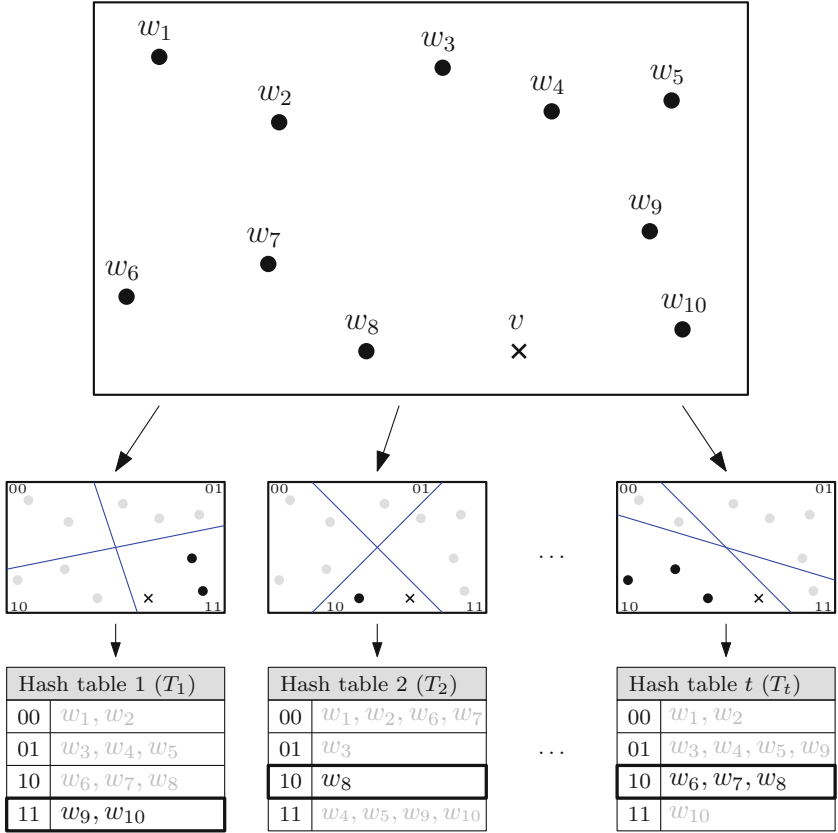
and the time complexity may again be estimated to be of the order  $2^{0.415n+o(n)}$ . Basic experiments show that, although with this notion of reduction the list size increases, this factor indeed appears to be sub-exponential in  $n$ .

### 3.3 The HashSieve with Angular Reductions

Replacing the stronger notion of reduction of (5) by the weaker one of (6), we can clearly see the connection with angular hashing. Considering the GaussSieve with angular reductions, we are repeatedly sampling new target vectors  $\mathbf{v}$  (with each time almost the same list  $L$ ), and each time we are looking for vectors  $\mathbf{w} \in L$  whose angle with  $\mathbf{v}$  is at most  $60^\circ$ . Replacing the brute-force list search in the original algorithm with the technique of angular locality-sensitive hashing, we obtain Algorithm 2. Blue lines in Algorithm 2 indicate modifications to the GaussSieve. Note that the setup costs of locality-sensitive hashing are spread out over the various iterations; at each iteration we only update the parts of the hash tables that were affected by updating  $L$ . This means that we only pay the setup costs of locality-sensitive hashing once, rather than once for each search.

### 3.4 The (GaussSieve-Based) HashSieve Algorithm

Finally, note that there seems to be no point in skipping potential reductions in Lines 8 and 9. So while for our intuition and for the theoretical motivation we may consider the case where the reductions are based on (6), in practice we will again reduce vectors based on (5). The algorithm is illustrated in Fig. 2.



**Fig. 2.** An example of the HashSieve, using  $k = 2$  hyperplanes and  $2^k = 4$  buckets in each hash table. Given 10 list vectors  $L = \{w_1, \dots, w_{10}\}$  and a target vector  $v$ , for each of the  $t$  hash tables we first compute  $v$ 's hash value (i.e. compute the region in which it lies), look up vectors with the same hash value, and compare  $v$  with those vectors. Here we will try to reduce  $v$  with  $C = \{w_6, w_7, w_8, w_9, w_{10}\}$  and vice versa.

### 3.5 Relation with Leveled Sieving

Overall, the crucial modification going from the GaussSieve to the HashSieve is that by using hash tables and looking up vectors to reduce the target vector with in these hash tables, we make the search space smaller; instead of comparing a new vector to *all* vectors in  $L$ , we only compare the vector to a much smaller subset of candidates  $C \subset L$ , which mostly contains good candidates for reduction, and does not contain many of the ‘bad’ vectors in  $L$  which are not useful for reductions anyway.

In a way, the idea of the HashSieve is similar to the technique previously used in two- and three-level sieving [45, 46]. There, the search space of candidate nearby vectors was reduced by partitioning the space into regions, and for each vector storing in which region it lies. In those algorithms, two nearby vectors in

adjacent regions are not considered for reductions, which means one needs more vectors to saturate the space (a higher space complexity) but less time to search the list of candidates for nearby vectors (a lower time complexity). The key difference between leveled sieving and our method is in the way the partitions of  $\mathbb{R}^n$  are chosen: using giant balls in leveled sieving (similar to the Euclidean LSH method of [6]), and using intersections of half-spaces in the HashSieve.

## 4 Theoretical Results

For analyzing the time complexity of sieving with angular LSH, for clarity of exposition we will analyze the GaussSieve-based HashSieve and assume that the GaussSieve has a time complexity which is quadratic in the list size, i.e. a time complexity of  $2^{0.415n+o(n)}$ . We will then show that using angular LSH, we can reduce the time complexity to  $2^{0.337n+o(n)}$ . Note that although practical experiments in high dimensions seem to verify this assumption [23], in reality it is not known whether the time complexity of the GaussSieve is quadratic in  $|L|$ . At first sight this therefore may not guarantee a heuristic time complexity of the order  $2^{0.337n+o(n)}$ . In the full version we illustrate how the same techniques can be applied to the sieve of Nguyen and Vidick [35], for which the heuristic time complexity is in fact known to be at most  $2^{0.415n+o(n)}$ , and for which we get the same speedup. This implies that indeed, with sieving we can provably solve SVP in time and space  $2^{0.337n+o(n)}$  under the same heuristic assumptions of Nguyen and Vidick [35]. For clarity of exposition, in the main text we will continue focusing on the GaussSieve due to its better practical performance, even though theoretically one might rather apply this analysis to the algorithm of Nguyen and Vidick due to their heuristic bounds on the time and space complexities.

### 4.1 High-Dimensional Intuition

So for now, suppose that the GaussSieve has a time complexity quadratic in  $|L|$  and that  $|L| \leq 2^{0.208n+o(n)}$ . To estimate the complexities of the HashSieve, we will use the following assumption previously described in [35]:

**Heuristic 1.** *The angle  $\Theta(\mathbf{v}, \mathbf{w})$  between random sampled/list vectors  $\mathbf{v}$  and  $\mathbf{w}$  follows the same distribution as the distribution of angles  $\Theta(\mathbf{v}, \mathbf{w})$  obtained by drawing  $\mathbf{v}, \mathbf{w} \in \mathbb{R}^n$  at random from the unit sphere.*

Note that under this assumption, in high dimensions angles close to  $90^\circ$  are much more likely to occur between list vectors than smaller angles. So one might guess that for two vectors  $\mathbf{w}_1, \mathbf{w}_2 \in L$  (which necessarily have an angle larger than  $60^\circ$ ), with high probability their angle is close to  $90^\circ$ . On the other hand, vectors that can reduce one another always have an angle less than  $60^\circ$ , and by similar arguments we expect this angle to always be close to  $60^\circ$ . Under the extreme assumption that all ‘reduced angles’ between vectors that are unable to reduce each other are *exactly*  $90^\circ$  (and non-reduced angles are at most  $60^\circ$ ), we obtain the following estimate for the costs of the HashSieve algorithm.

**Proposition 1.** *Assuming that reduced vectors are always pairwise orthogonal, the HashSieve with parameters  $k = 0.2075n + o(n)$  and  $t = 2^{0.1214n + o(n)}$  heuristically solves SVP in time and space  $2^{0.3289n + o(n)}$ . We further obtain the trade-off between the space and time complexities indicated by the dashed line in Fig. 1.*

*Proof.* If all reduced angles are  $90^\circ$ , then we can simply let  $\theta_1 = \frac{\pi}{3}$  and  $\theta_2 = \frac{\pi}{2}$  and use the hash family described in Sect. 2.5 with  $p_1 = \frac{2}{3}$  and  $p_2 = \frac{1}{2}$ . Applying Lemma 1, we can perform a single search in time  $N^\rho$  with  $\rho = \frac{\log(1/p_1)}{\log(1/p_2)} = \log_2(\frac{3}{2}) \approx 0.585$ . Since we need to perform these searches  $\tilde{O}(|L|) = \tilde{O}(N)$  times, the time complexity is of the order  $\tilde{O}(N^{1+\rho}) = 2^{0.3289n + o(n)}$ .  $\square$

## 4.2 Heuristically Solving SVP in Time and Space $2^{0.3366n + o(n)}$

Of course, in practice not all reduced angles are actually  $90^\circ$ , and one should carefully analyze what is the real probability that a vector  $w$  whose angle with  $v$  is more than  $60^\circ$ , is found as a candidate due to a collision in one of the hash tables. The following central theorem follows from this analysis and shows how to choose the parameters to optimize the asymptotic time complexity. A rigorous proof of Theorem 1 based on the NV-sieve can be found in the full version.

**Theorem 1.** *Sieving with angular locality-sensitive hashing with parameters*

$$k = 0.2206n + o(n), \quad t = 2^{0.1290n + o(n)}, \quad (7)$$

*heuristically solves SVP in time and space  $2^{0.3366n + o(n)}$ . Tuning  $k$  and  $t$  differently, we further obtain the trade-off indicated by the solid blue line in Fig. 1.*

Note that the optimized values in Theorem 1 and Proposition 1, and the associated curves in Fig. 1 are very similar. So the simple estimate based on the intuition that in high dimensions “everything is orthogonal” is not far off.

## 4.3 Heuristically Solving SVP in Time $2^{0.3366n}$ and Space $2^{0.2075n}$

For completeness let us briefly explain how for the NV-sieve [35], we can in fact process the hash tables sequentially and eliminate the need of storing exponentially many hash tables in memory, for which full details are given in the full version. To illustrate the idea, recall that in the Nguyen-Vidick sieve we are given a list  $L$  of size  $2^{0.21n + o(n)}$  of vectors of norm at most  $R$ , and we want to build a new list  $L'$  of similar size  $2^{0.21n + o(n)}$  of vectors of norm at most  $\gamma R$  with  $\gamma < 1$ . To do this, we look at (almost) all pairs of vectors in  $L$ , and see if their difference (sum) is short; if so, we add it to  $L'$ . As the probability of finding a short vector is roughly  $2^{-0.21n + o(n)}$  and we have  $2^{0.42n + o(n)}$  pairs of vectors, this will result in enough vectors to continue in the next iterations.

The natural way to apply angular LSH to this algorithm would be to add all vectors in  $L$  to  $t$  independent hash tables, and to find short vectors to add to



$L'$  we then compute a new vector  $\mathbf{v}$ 's hash value for each of these  $t$  hash tables, look for potential short vectors  $\mathbf{v} \pm \mathbf{w}$  by comparing  $\mathbf{v}$  with the colliding vectors  $\mathbf{w} \in \bigcup_{i=1}^t T_i[h_i(\mathbf{v})]$ , and process all vectors one by one. This results in similar asymptotic time and space complexities as illustrated above.

The crucial modification that we can make to this algorithm (similar to [8]) is that we process the tables one by one; we first construct the first hash table, add all vectors in  $L$  to this hash table, and look for short difference vectors inside each of the buckets of  $L$  to add to  $L'$ . The cost of building and processing one hash table is of the order  $2^{0.21n+o(n)}$ , and the number of vectors found that can be added to  $L'$  is of the order  $2^{0.08n+o(n)}$ . By then deleting the hash table from memory and building new hash tables over and over ( $t = 2^{0.13n+o(n)}$  times) we keep building a longer list  $L'$  until finally we will again have found  $2^{0.21n+o(n)}$  short vectors for the next iteration. In this case however we never stored all hash tables in memory at the same time, and the memory increase compared to the NV-sieve is asymptotically negligible. This leads to the following result.

**Theorem 2.** *Sieving with angular locality-sensitive hashing with parameters*

$$k = 0.2206n + o(n), \quad t = 2^{0.1290n+o(n)}, \quad (8)$$

*heuristically solves SVP in time  $2^{0.3366n+o(n)}$  and space  $2^{0.2075n+o(n)}$ . These complexities are indicated by the left-most blue point in Fig. 1.*

Note that this choice of parameters balances the costs of computing hashes and comparing vectors; the fact that the blue point in Fig. 1 does not lie on the ‘‘Time = Space’’-line does not mean we can further reduce the time complexity.

#### 4.4 Reducing the Space Complexity with Probing

Finally, as the above modification only seems to work with the less practical NV-sieve (and not with the GaussSieve), and since for the GaussSieve-based HashSieve the memory requirement increases exponentially, let us briefly sketch how we can reduce the required amount of memory in practice for the (GaussSieve-based) HashSieve using *probing* [36]. The key observation here is that, as illustrated in Fig. 2, we only check one bucket in each hash table for nearby vectors, leading to  $t$  hash buckets in total that are checked for candidate reductions. This seems wasteful, as the hash tables contain more information: we also know for instance which hash buckets are next-most likely to contain nearby vectors, which are buckets with very similar hash values. By also probing these buckets in a clever way and checking multiple hash buckets per hash table, we can significantly reduce the number of hash tables  $t$  in practice such that in the end we still find as many good vectors. Using  $\ell$  levels of probing (checking all buckets with hash value at Hamming distance at most  $\ell$  to  $h(\mathbf{v})$ ) we can reduce  $t$  by a factor  $O(n^\ell)$  at the cost of increasing the time complexity by a factor at most  $2^\ell$ . This does not constitute an exponential improvement, but the polynomial reduction in memory may be worthwhile in practice. More details on probing can be found in the full version.

## 5 Practical Results

### 5.1 Experimental Results in Moderate Dimensions

To verify our theoretical analysis, we implemented both the GaussSieve and the GaussSieve-based HashSieve to try to compare the asymptotic trends of these algorithms. For implementing the HashSieve, we note that we can use various simple tweaks to further improve the algorithm’s performance. These include:

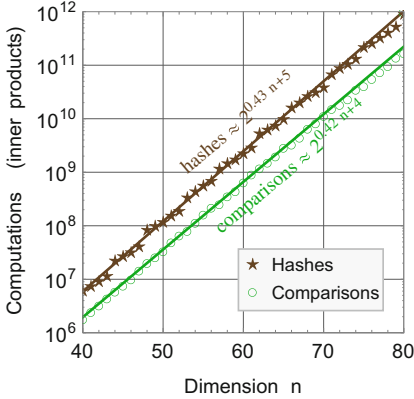
- (a) With the HashSieve, maintaining a list  $L$  is no longer needed.
- (b) Instead of making a list of candidates, we go through the hash tables one by one, checking if collisions in this table lead to reductions. If a reducing vector is found early on, this may save up to  $t \cdot k$  hash computations.
- (c) As  $h_i(-\mathbf{v}) = -h_i(\mathbf{v})$  the hash of  $-\mathbf{v}$  can be computed for free from  $h_i(\mathbf{v})$ .
- (d) Instead of comparing  $\pm\mathbf{v}$  to all candidate vectors  $\mathbf{w}$ , we only compare  $+\mathbf{v}$  to the vectors in the bucket  $h_i(\mathbf{v})$  and  $-\mathbf{v}$  to the vectors in the bucket labeled  $-h_i(\mathbf{v})$ . This further reduces the number of comparisons by a factor 2 compared to the GaussSieve, where both comparisons are done for each potential reduction.
- (e) For choosing vectors  $\mathbf{a}_{i,j}$  to use for the hash functions  $h_i$ , there is no reason to assume that drawing  $\mathbf{a}$  from a specific, sufficiently large random subset of the unit sphere would lead to substantially different results. In particular, using sparse vectors  $\mathbf{a}_{i,j}$  makes hash computations significantly cheaper, while retaining the same performance [1, 27]. Our experiments indicated that even if all vectors  $\mathbf{a}_{i,j}$  have only two equal non-zero entries, the algorithm still finds the shortest vector in (roughly) the same number of iterations.
- (f) We should not store the actual vectors, but only pointers to vectors in each hash table  $T_i$ . This means that compared to the GaussSieve, the space complexity roughly increases from  $O(N \cdot n)$  to  $O(N \cdot n + N \cdot t)$  instead of  $O(N \cdot n \cdot t)$ , i.e., an asymptotic increase of a factor  $t/n$  rather than  $t$ .

With these tweaks, we performed several experiments of finding shortest vectors using the lattices of the SVP challenge [42]. We generated lattice bases for different seeds and different dimensions using the SVP challenge generator, used NTL (Number Theory Library) to preprocess the bases (LLL reduction), and then used our implementations of the GaussSieve and the HashSieve to obtain these results. For the HashSieve we chose the parameters  $k$  and  $t$  by rounding the theoretical estimates of Theorem 1 to the nearest integers, i.e.,  $k = \lfloor 0.2206n \rfloor$  and  $t = \lfloor 2^{0.1290n} \rfloor$  (see Fig. 3a). Note that clearly there are ways to further speed up both the GaussSieve and the HashSieve, using e.g. better preprocessing, vectorized code, parallel implementations, optimized samplers, etc. The purpose of our experiments is only to obtain a fair comparison of the two algorithms and to try to estimate and compare the asymptotic behaviors of these algorithms. Details on a more optimized implementation of the HashSieve are given in [31].

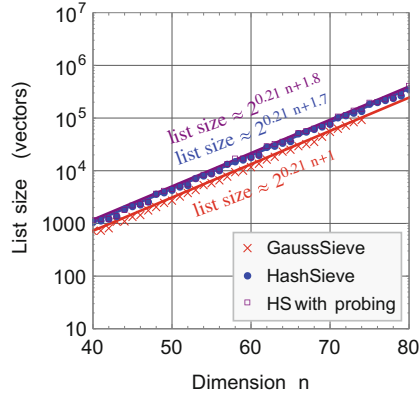
**Computations.** Figure 3b shows the number of inner products computed by the HashSieve for comparing vectors and for computing hashes. We have chosen

Dimension ( $n$ )	40	45	50	55	60	65	70	75	80
Hash length ( $k$ )	9	10	11	12	13	14	15	17	18
Hash tables ( $t$ )	36	56	87	137	214	334	523	817	1278
... with probing ( $t_1$ )	7	9	13	19	28	41	60	88	130

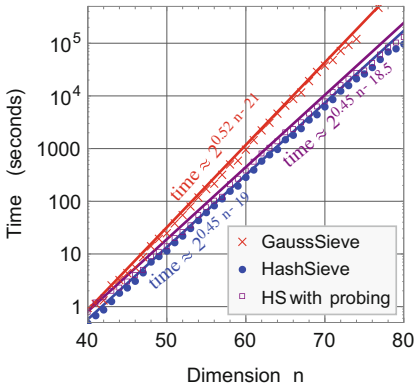
(a) Parameters used in HashSieve experiments, without ( $t$ ) and with ( $t_1$ ) probing



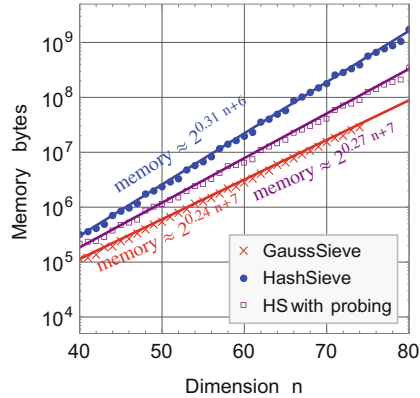
(b) HashSieve computations (no prob.)



(c) List sizes



(d) Time complexities



(e) Space complexities

**Fig. 3.** Experimental data for the GaussSieve and the HashSieve (with/without probing). Markers indicate experiments, lines and labels represent least-squares fits. Figure 3b shows the time spent on hashing and comparing vectors in the HashSieve. Figure 3c confirms our intuition that if we miss a small fraction of the reducing vectors, the list size increases by a small factor. Figure 3d compares the time complexities of the algorithms, confirming our theoretical analysis of a speedup of roughly  $2^{0.07n}$  over the GaussSieve. Figure 3e illustrates the space requirements of each algorithm. Note that probing decreases the required memory at the cost of a small increase in the time. Also note that the step-wise behavior of some curves in Fig. 3 is explained by the fact that  $k$  is small but integral, and increases by 1 only once every four/five dimensions.

$k$  and  $t$  so that the total time for each of these operations is roughly balanced, and indeed this seems to be the case. The total number of inner products for hashing seems to be a constant factor higher than the total number of inner products computed for comparing vectors, which may also be desirable, as hashing is significantly cheaper than comparing vectors using sparse hash vectors. Tuning the parameters differently may slightly change this ratio.

**List Sizes.** In the analysis, we assumed that if reductions are missed with a constant probability, then the list size also increases by a constant factor. Figure 3c seems to support this intuition, as indeed the list sizes in the HashSieve seem to be a (small) constant factor larger than in the GaussSieve.

**Time Complexities.** Figure 3d compares the timings of the GaussSieve and HashSieve on a single core of a Dell Optiplex 780, which has a processor speed of 2.66 GHz. Theoretically, we expect to achieve a speedup of roughly  $2^{0.078n}$  for each list search, and in practice we see that the asymptotic speedup of the HashSieve over the GaussSieve is close to  $2^{0.07n}$  using a least-squares fit.

Note that the coefficients in the least-squares fits for the time complexities of the GaussSieve and HashSieve are higher than theory suggests, which is in fact consistent with previous experiments in low dimensions [15, 19, 29, 30, 33]. This phenomenon seems to be caused purely by the low dimensionality of our experiments. Figure 3d shows that in higher dimensions, the points start to deviate from the straight line, with a better scaling of the time complexity in higher dimensions. High-dimensional experiments of the GaussSieve ( $80 \leq n \leq 100$ ) and the HashSieve ( $86 \leq n \leq 96$ ) demonstrated that these algorithms start following the expected trends of  $2^{0.42n+o(n)}$  (GaussSieve) and  $2^{0.34n+o(n)}$  (HashSieve) as  $n$  gets larger [23, 31]. In high dimensions we therefore expect the coefficient 0.3366 to be accurate. For more details, see [31].

**Space Complexities.** Figure 3e illustrates the experimental space complexities of the tested algorithms for various dimensions. For the GaussSieve, the total space complexity is dominated by the memory required to store the list  $L$ . In our experiments we stored each vector coordinate in a register of 4 bytes, and since each vector has  $n$  entries, this leads to a total space complexity for the GaussSieve of roughly  $4nN$  bytes. For the HashSieve the asymptotic space complexity is significantly higher, but recall that in our hash tables we only store pointers to vectors, which may also be only 4 bytes each. For the HashSieve, we estimate the total space complexity as  $4nN + 4tN \sim 4tN$  bytes, i.e., roughly a factor  $\frac{t}{n} \approx 2^{0.1290n}/n$  higher than the space complexity of the GaussSieve. Using probing, the memory requirement is further reduced by a significant amount, at the cost of a small increase in the time complexity (Fig. 3d).

## 5.2 High-Dimensional Extrapolations

As explained at the start of this section, the experiments in Sect. 5.1 are aimed at verifying the heuristic analysis and at establishing trends which hold regardless of the amount of optimization of the code, the quality of preprocessing of the

input basis, the amount of parallelization etc. However, the linear estimates in Fig. 3 may not be accurate. For instance, the time complexities of the GaussSieve and HashSieve seem to scale better in higher dimensions; the time complexities may well be  $2^{0.415n+o(n)}$  and  $2^{0.337n+o(n)}$  respectively, but the contribution of the  $o(n)$  only starts to fade away for large  $n$ . To get a better feeling of the actual time complexities in high dimensions, one would have to run these algorithms in higher dimensions. In recent work, Mariano et al. [31] showed that the HashSieve can be parallelized in a similar fashion as the GaussSieve [29]. With better preprocessing and optimized code (but without probing), Mariano et al. were able to solve SVP in dimensions up to 96 in less than one day on one machine using the HashSieve<sup>3</sup>. Based on experiments in dimensions 86 up to 96, they further estimated the time complexity to lie between  $2^{0.32n-15}$  and  $2^{0.33n-16}$ , which is close to the theoretical estimate  $2^{0.3366n+o(n)}$ . So although the points in Fig. 3d almost seem to lie on a line with a different leading constant, these leading constants should not be taken for granted for high-dimensional extrapolations; the theoretical estimate  $2^{0.3366n+o(n)}$  seems more accurate.

Finally, let us try to estimate the highest practical dimension  $n$  in which the HashSieve may be able to solve SVP right now. The current highest dimension that was attacked using the GaussSieve is  $n = 116$ , for which 32 GB RAM and about 2 core years were needed [23]. Assuming the theoretical estimates for the GaussSieve ( $2^{0.4150n+o(n)}$ ) and HashSieve ( $2^{0.3366n+o(n)}$ ) are accurate, and assuming there is a constant overhead of approximately  $2^2$  of the HashSieve compared to the GaussSieve (based on the exponents in Fig. 3d), we might estimate the time complexities of the GaussSieve and HashSieve to be  $G(n) = 2^{0.4150n+C}$  and  $H(n) = 2^{0.3366n+C+2}$  respectively. To solve SVP in the same dimension  $n = 116$ , we therefore expect to use a factor  $G(116)/H(116) \approx 137$  less time using the HashSieve, or five core days on the same machine. With approximately two core years, we may further be able to solve SVP in dimension 138 using the HashSieve, which would place sieving near the very top of the SVP hall of fame [42]. This does not take into account the space complexity though, which at this point may have increased to several TBs. Several levels of probing may significantly reduce the required amount of RAM, but further experiments have to be conducted to see how practical the HashSieve is in high dimensions. As in high dimensions the space requirement also becomes an issue, studying the memory-efficient NV-sieve-based HashSieve (with space complexity  $2^{0.2075n+o(n)}$ ) may be an interesting topic for future work.

**Acknowledgments.** The author is grateful to Meilof Veenigen and Niels de Vreede for their help and advice with implementations. The author thanks the anonymous reviewers, Daniel J. Bernstein, Marleen Kooiman, Tanja Lange, Artur Mariano, Joop van de Pol, and Benne de Weger for their valuable suggestions and comments. The author further thanks Michele Mosca for funding a research visit to Waterloo to collaborate on lattices and quantum algorithms, and the author thanks Stacey Jeffery, Michele Mosca, Joop van de Pol, and John M. Schanck for valuable discussions there. The author also thanks Memphis Depay for his inspiration.

<sup>3</sup> At the time of writing, Mariano et al.’s highest SVP challenge records obtained using the HashSieve are in dimension 107, using five days on one multi-core machine.

## References

1. Achlioptas, D.: Database-friendly random projections. In: PODS (2001)
2. Aggarwal, D., Dadush, D., Regev, O., Stephens-Davidowitz, N.: Solving the shortest vector problem in  $2^n$  time via discrete Gaussian sampling. In: STOC (2015)
3. Ajtai, M.: Generating hard instances of lattice problems (extended abstract). In: STOC, pp. 99–108 (1996)
4. Ajtai, M.: The shortest vector problem in  $L_2$  is NP-hard for randomized reductions (extended abstract). In: STOC, pp. 10–19 (1998)
5. Ajtai, M., Kumar, R., Sivakumar, D.: A sieve algorithm for the shortest lattice vector problem. In: STOC, pp. 601–610 (2001)
6. Andoni, A., Indyk, P.: Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In: FOCS, pp. 459–468 (2006)
7. Becker, A., Gama, N., Joux, A.: A sieve algorithm based on overlattices. In: ANTS, pp. 49–70 (2014)
8. Becker, A., Gama, N., Joux, A.: Speeding-up lattice sieving without increasing the memory, using sub-quadratic nearest neighbor search. Preprint (2015)
9. Becker, A., Laarhoven, T.: Efficient sieving in (ideal) lattices using cross-polytopic LSH. Preprint (2015)
10. Bernstein, D.J., Buchmann, J., Dahmen, E.: Post-Quantum Cryptography. Springer, Heidelberg (2009)
11. Bos, J.W., Naehrig, M., van de Pol, J.: Sieving for shortest vectors in ideal lattices: a practical perspective. Cryptology ePrint Archive, Report 2014/880 (2014)
12. Charikar, M.S.: Similarity estimation techniques from rounding algorithms. In: STOC, pp. 380–388 (2002)
13. Conway, J.H., Sloane, N.J.A.: Sphere Packings, Lattices and Groups. Springer, New York (1999)
14. Fincke, U., Pohst, M.: Improved methods for calculating vectors of short length in a lattice. *Math. Comput.* **44**(170), 463–471 (1985)
15. Fitzpatrick, R., Bischof, C., Buchmann, J., Dagdelen, Ö., Göpfert, F., Mariano, A., Yang, B.-Y.: Tuning GaussSieve for speed. In: Aranha, D.F., Menezes, A. (eds.) LATINCRYPT 2014. LNCS, vol. 8895, pp. 288–305. Springer, Heidelberg (2015)
16. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: STOC (2009)
17. 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)
18. Indyk, P., Motwani, R.: Approximate nearest neighbors: towards removing the curse of dimensionality. In: STOC, pp. 604–613 (1998)
19. Ishiguro, T., Kiyomoto, S., Miyake, Y., Takagi, T.: Parallel gauss sieve algorithm: solving the svp challenge over a 128-dimensional ideal lattice. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 411–428. Springer, Heidelberg (2014)
20. Kannan, R.: Improved algorithms for integer programming and related lattice problems. In: STOC, pp. 193–206 (1983)
21. Khot, S.: Hardness of approximating the shortest vector problem in lattices. In: FOCS, pp. 126–135 (2004)
22. Klein, P.: Finding the closest lattice vector when it’s unusually close. In: SODA, pp. 937–941 (2000)
23. Kleinjung, T.: Private Communication (2014)
24. Laarhoven, T.: Sieving for shortest vectors in lattices using angular locality-sensitive hashing (2015). Full version at <http://eprint.iacr.org/2014/744>

25. Laarhoven, T., de Weger, B.: Faster sieving for shortest lattice vectors using spherical locality-sensitive hashing. In: LATINCRYPT (2015)
26. Lenstra, A.K., Lenstra, H.W., Lovász, L.: Factoring polynomials with rational coefficients. *Math. Ann.* **261**(4), 515–534 (1982)
27. Li, P., Hastie, T.J., Church, K.W.: Very sparse random projections. In: KDD, pp. 287–296 (2006)
28. Lindner, R., Peikert, C.: Better key sizes (and attacks) for LWE-based encryption. In: Kiayias, A. (ed.) CT-RSA 2011. LNCS, vol. 6558, pp. 319–339. Springer, Heidelberg (2011)
29. Mariano, A., Timnat, S., Bischof, C.: Lock-free GaussSieve for linear speedups in parallel high performance SVP calculation. In: SBAC-PAD (2014)
30. Mariano, A., Dagdelen, Ö., Bischof, C.: A comprehensive empirical comparison of parallel ListSieve and GaussSieve. In: Lopes, L., et al. (eds.) Euro-Par 2014: Parallel Processing Workshops, Part I. LNCS, vol. 8805, pp. 48–59. Springer, Switzerland (2014)
31. Mariano, A., Laarhoven, T., Bischof, C.: Parallel (probable) lock-free HashSieve: a practical sieving algorithm for the SVP. In: ICPP (2015)
32. Micciancio, D., Voulgaris, P.: A deterministic single exponential time algorithm for most lattice problems based on Voronoi cell computations. In: STOC (2010)
33. Micciancio, D., Voulgaris, P.: Faster exponential time algorithms for the shortest vector problem. In: SODA, pp. 1468–1480 (2010)
34. Micciancio, D., Walter, M.: Fast lattice point enumeration with minimal overhead. In: SODA, pp. 276–294 (2015)
35. Nguyen, P.Q., Vidick, T.: Sieve algorithms for the shortest vector problem are practical. *J. Math. Crypt.* **2**(2), 181–207 (2008)
36. Panigraphy, R.: Entropy based nearest neighbor search in high dimensions. In: SODA, pp. 1186–1195 (2006)
37. Plantard, T., Schneider, M.: Ideal lattice challenge. <http://latticechallenge.org/ideallattice-challenge/> (2014)
38. Pohst, M.E.: On the computation of lattice vectors of minimal length, successive minima and reduced bases with applications. *ACM Bull.* **15**(1), 37–44 (1981)
39. van de Pol, J., Smart, N.P.: Estimating key sizes for high dimensional lattice-based systems. In: Stam, M. (ed.) IMACC 2013. LNCS, vol. 8308, pp. 290–303. Springer, Heidelberg (2013)
40. Pujol, X., Stehlé, D.: Solving the shortest lattice vector problem in time  $2^{2.465n}$ . *Cryptology ePrint Archive*, Report 2009/605 (2009)
41. Schneider, M.: Sieving for shortest vectors in ideal lattices. In: Youssef, A., Nitaj, A., Hassanien, A.E. (eds.) AFRICACRYPT 2013. LNCS, vol. 7918, pp. 375–391. Springer, Heidelberg (2013)
42. Schneider, M., Gama, N., Baumann, P., Nobach, L.: SVP challenge (2014). <http://latticechallenge.org/svp-challenge>
43. Schnorr, C.-P.: A hierarchy of polynomial time lattice basis reduction algorithms. *Theoret. Comput. Sci.* **53**(2), 201–224 (1987)
44. Schnorr, C.-P., Euchner, M.: Lattice basis reduction: improved practical algorithms and solving subset sum problems. *Math. Programming* **66**(2), 181–199 (1994)
45. Wang, X., Liu, M., Tian, C., Bi, J.: Improved Nguyen-Vidick heuristic sieve algorithm for shortest vector problem. In: ASIACCS, pp. 1–9 (2011)
46. Zhang, F., Pan, Y., Hu, G.: A three-level sieve algorithm for the shortest vector problem. In: Lange, T., Lauter, K., Lisoněk, P. (eds.) SAC 2013. LNCS, vol. 8282, pp. 29–47. Springer, Heidelberg (2014)

# Coded-BKW: Solving LWE Using Lattice Codes

Qian Guo<sup>1,2</sup>, Thomas Johansson<sup>1</sup>(✉), and Paul Stankovski<sup>1</sup>

<sup>1</sup> Department of Electrical and Information Technology, Lund University,  
Lund, Sweden

{qian.guo, thomas.johansson, paul.stankovski}@eit.lth.se

<sup>2</sup> Shanghai Key Laboratory of Intelligent Information Processing,  
School of Computer Science, Fudan University, Shanghai, China

**Abstract.** In this paper we propose a new algorithm for solving the Learning With Errors (LWE) problem based on the steps of the famous Blum-Kalai-Wasserman (BKW) algorithm. The new idea is to introduce an additional procedure of mapping subvectors into codewords of a lattice code, thereby increasing the amount of positions that can be cancelled in each BKW step. The procedure introduces an additional noise term, but it is shown that by using a sequence of lattice codes with different rates the noise can be kept small. Developed theory shows that the new approach compares favorably to previous methods. It performs particularly well for the BINARY-LWE case, i.e., when the secret vector is sampled from  $\{0, 1\}^*$ .

**Keywords:** LWE · BINARY-LWE · BKW · Coded-BKW · Lattice codes

## 1 Introduction

Learning with Errors (LWE) is a problem that has received a lot of attention recently and can be considered as a generalization of the Learning Parity with Noise (LPN) problem. Regev introduced LWE in [31], and it has proved to be a very useful tool for constructing cryptographic primitives. Although a great number of different constructions of cryptographic primitives have been given since the introduction of the LWE problem, one of the most interesting ones is the work on constructing fully homomorphic encryption schemes [8, 10, 19, 20].

There are several motivating reasons for the interest in LWE-based cryptography. One is the simplicity of the constructions, sometimes giving rise to very efficient implementations which run much faster than competing alternative solutions. Another reason is the well-developed theory on lattice problems, which gives insights into the hardness of the LWE problem. There are theoretical reductions from worst-case lattice problems to average-case LWE [31].

---

Q. Guo—Supported in part by Shanghai Key Program of Basic Research (No. 12JC1401400), the National Defense Basic Research Project (No. JCYJ-1408).

T. Johansson and P. Stankovski—Supported by the Swedish Research Council (Grants No. 621-2012-4259).



A third motivating reason is the fact that LWE-based cryptography is one of the areas where a quantum computer is not known to be able to break the primitives (contrary to factoring-based and discrete log-based primitives). This is sometimes referred to as being a tool in *post-quantum cryptography*.

Let us state the LWE problem.

**Definition 1.** Let  $n$  be a positive integer,  $q$  an odd prime, and let  $\mathcal{X}$  be an error distribution selected as the discrete Gaussian distribution on  $\mathbb{Z}_q$ . Fix  $\mathbf{s}$  to be a secret vector in  $\mathbb{Z}_q^n$ , chosen according to a uniform distribution. Denote by  $L_{\mathbf{s},\mathcal{X}}$  the probability distribution on  $\mathbb{Z}_q^n \times \mathbb{Z}_q$  obtained by choosing  $\mathbf{a} \in \mathbb{Z}_q^n$  uniformly at random, choosing an error  $e \in \mathbb{Z}_q$  according to  $\mathcal{X}$  and returning

$$(\mathbf{a}, z) = (\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e)$$

in  $\mathbb{Z}_q^n \times \mathbb{Z}_q$ . The (search) LWE problem is to find the secret vector  $\mathbf{s}$  given a fixed number of samples from  $L_{\mathbf{s},\mathcal{X}}$ .

The definition above gives the *search* LWE problem, as the problem description asks for the recovery of the secret vector  $\mathbf{s}$ . Another variant is the so-called *decision* LWE problem. In this case the problem is to distinguish samples drawn from  $L_{\mathbf{s},\mathcal{X}}$  and samples drawn from a uniform distribution on  $\mathbb{Z}_q^n \times \mathbb{Z}_q$ . Typically, we are then interested in distinguishers with non-negligible advantage.

The parameters of an LWE instance are typically chosen with some internal relations. The prime  $q$  is chosen as a polynomial in  $n$ , and the discrete Gaussian distribution  $\mathcal{X}$  has mean zero and standard deviation  $\sigma = \alpha \cdot q$  for some small  $\alpha$ . For example, in [31], Regev proposed to use parameters  $q \approx n^2$  and  $\alpha = 1/(\sqrt{2\pi n} \cdot \log_2^2 n)$ .

## 1.1 Previous Work

A number of algorithms for solving the LWE problem have been given, using different approaches. As there is a strong connection to lattice problems, a direction for a subset of the algorithms has been to either rewrite the LWE problem as the problem of finding a short vector in a dual lattice, the Short Integer Solution (SIS) problem, or to solve the Bounded Distance Decoding (BDD) problem. Lattice reduction algorithms may be applied to solve these problems. Even though there has been a lot of research devoted to the study of lattice reduction algorithms, there still seems to be quite some uncertainty about the complexity and performance of such algorithms for higher dimensions.

Another very interesting approach was given by Arora and Ge in [5], where they proposed a novel algebraic approach to solve the LWE problem. The asymptotic complexity of this algorithm is subexponential when  $\sigma \leq \sqrt{n}$ , but fully exponential otherwise. The algorithm is mainly of asymptotic interest as applying it on specific instances gives higher complexity than other solvers.

Finally, much work has been done on combinatorial algorithms for solving LWE, all taking the famous Blum-Kalai-Wasserman (BKW) algorithm [7] as a basis. The BKW algorithm resembles the generalized birthday approach by

Wagner [34] and was originally given as an algorithm for solving the LPN problem. These combinatorial algorithms have the advantage that their complexity can be analyzed in a standard way and we can get explicit values on the complexity for different instantiations of the LWE problem. Even though we use approximations in the analysis, the deviation between theoretical analysis and actual performance seems to be small [3, 17]. This approach tends to give algorithms with the best performance for some important parameter choices. A possible drawback with BKW-based algorithms is that they usually require a huge amount of memory, often of the same order as the time complexity. Some recent work in this direction is [1, 3, 17].

## 1.2 Motivation and Contributions

We know that the theoretical hardness of the LWE problem is well-established, through reductions to hard lattice problems [9, 30, 31]. This can be transferred to asymptotic statements on the security. In fact, most proposals of LWE-based cryptographic primitives rely only on asymptotics when arguing about security.

But there is also a huge interest in studying the actual hardness of specific instances of the LWE problem. How does the choice of parameters  $(n, q, \sigma)$  influence the complexity of solving LWE? What are the smallest parameters we can have and still achieve, say, 80-bit security?

In this paper we introduce a new algorithm for the LWE problem which again uses the BKW algorithm as a basis. The novel idea is to introduce a modified BKW step, we call it a coded-BKW step, that allows us to cancel out more positions in the  $\mathbf{a}$  vectors than a traditional BKW step. The coded-BKW step involves mapping the considered part of an  $\mathbf{a}$  vector into a nearest codeword in a lattice code (a linear code over  $\mathbb{Z}_q$ , where the distance is the Euclidean norm). The mapping to the nearest codeword introduces some noise, but with

**Table 1.** Time complexity comparison for solving various LWE instances.

n	q	$\sigma$	Complexity ( $\log_2 \#\mathbb{Z}_q$ )			
			This paper (Sect. 5)	Duc et al. [17]	NTL-BKZ Lindner-Peikert model [1, 25]	BKZ 2.0 Simulator Model [1, 11, 26]
Regev [31]						
128	16,411	11.81	84.5	95.0	61.6	61.9
256	65,537	25.53	145.1	178.7	175.5	174.5
512	262,147	57.06	287.6	357.5	386.8	518.6
Lindner & Peikert [25]						
128	2,053	2.70	69.7	83.7	54.5	57.1
256	4,099	3.34	123.8	154.2	156.2	151.2
512	4,099	2.90	209.2	271.8	341.9	424.5

**Table 2.** Time complexity comparison for solving various BINARY-LWE instances.

n	q	$\sigma$	Complexity ( $\log_2 \#\mathbb{Z}_2$ )				
			This paper (Sect. 7)	Albrecht et al. [3]		NTL-BKZ L-P model [1, 25]	BKZ 2.0 Sim. model [1, 11, 26]
				w/o Unnatural Selection	Improved version		
Regev [31]							
128	16,411	11.81	58.8	78.2	74.2	65.4	65.7
256	65,537	25.53	97.9	142.7	132.5	179.5	178.5
512	262,147	57.06	163.7	251.2	241.8	390.9	522.8

a proper selection of parameters this can be kept bounded and small enough not to influence the total noise in the BKW procedure too much. Whenever any pair of  $\mathbf{a}$  vectors map to the same codeword, they are added together creating a new sample with a part of the  $\mathbf{a}$  vector cancelled, as is the usual result of a BKW step. These samples are the input to the next step in the BKW procedure. The algorithm also contains some additional new steps using the discrete Fast Fourier Transform (FFT) to provide some additional improvement. The new ideas have some connection to the recent paper on LPN [22], but in that paper a binary covering code (see [12] for definition) was used after the BKW steps. Also the recent work [3] has some connections as it introduces additional noise in the BKW steps by the so-called lazy modulus switching. Still, the new algorithm outperforms previous BKW-type algorithms for solving LWE — even when compared with the most recent work [17], we improve significantly (as detailed in Table 1).

We also apply the algorithm in a slightly modified form on the BINARY-LWE problem. The BINARY-LWE problem is the LWE problem when the secret vector  $\mathbf{s}$  is chosen uniformly from  $\{0, 1\}^n$ . In this case we have a huge improvement (see Table 2) in performance compared with other algorithms.

Tables 1 and 2 show comparisons of different algorithms for solving various LWE and BINARY-LWE instances, respectively. We compare the performance of the new algorithm with the previous best BKW variant (i.e., Duc et al. [17] for LWE or Albrecht et al. [3] for BINARY-LWE) and the estimates (under certain models [11, 25, 26, 29]) for distinguishing LWE (or BINARY-LWE) samples from uniform using lattice reduction algorithms, when LWE is reduced to SIS. The results consolidate the understanding that BKW is asymptotically efficient. For the toy LWE instances with  $n = 128$ , the SIS approach still beats all the BKW variants, including ours; but the recent variant has greatly narrowed the gap. The situation alters when the parameter  $n$  increases.

We also obtain a significant improvement (i.e., with a factor of more than  $2^{11}$  in time) on solving an LWE (136, 2003, 5.19)-instance, which first appeared

in [29] and was then adopted as an example in [25], compared with the estimates in [1] that use the BDD approach.

Thus, we are close to a conclusion that, when choosing LWE instances for today’s cryptosystems (e.g., achieving an 80-bit or higher security level), thwarting of BKW-type attacks must be taken into consideration.

The remainder of the paper is organized as follows. In Sect. 2 we describe the basic theory around the LWE problem. We give a short description of the BKW algorithm in Sect. 3, and then present the novel modification in the next section. We detail the algorithm in Sect. 5, analyze its complexity in Sect. 6, and then propose a variant for BINARY-LWE in Sect. 7. This is followed by the sections of implementation and results. We finally concludes this paper in Sect. 10.

## 2 Background

On an  $n$ -dimensional Euclidean space  $\mathbb{R}^n$ , the intuitive notion of length of a vector  $\mathbf{x} = (x_1, x_2, \dots, x_n)$  is captured by the  $L_2$ -norm;  $\|\mathbf{x}\| = \sqrt{x_1^2 + \dots + x_n^2}$ . The Euclidean distance between two vectors  $\mathbf{x}$  and  $\mathbf{y}$  in  $\mathbb{R}^n$  is defined as  $\|\mathbf{x} - \mathbf{y}\|$ . For a given set of vectors  $\mathcal{L}$ , the minimum mean square error (MMSE) estimator assigns each vector in  $\mathbb{R}^n$  to the vector  $\mathbf{l} \in \mathcal{L}$  such that  $\|\mathbf{x} - \mathbf{l}\|$  is minimized. Let us shortly introduce the discrete Gaussian distribution.

### 2.1 Discrete Gaussian Distribution

Let  $x \in \mathbb{Z}$ . The discrete Gaussian distribution on  $\mathbb{Z}$  with mean 0 and variance  $\sigma^2$ , denoted  $D_{\mathbb{Z},\sigma}$ , is the probability distribution obtained by assigning a probability proportional to  $\exp(-x^2/2\sigma^2)$  to each  $x \in \mathbb{Z}$ . The  $\mathcal{X}$  distribution<sup>1</sup> with variance  $\sigma^2$  is the distribution on  $\mathbb{Z}_q$  obtained by folding  $D_{\mathbb{Z},\sigma} \bmod q$ , i.e., accumulating the value of the probability mass function over all integers in each residue class *mod*  $q$ . Similarly, we define the discrete Gaussian over  $\mathbb{Z}^n$  with variance  $\sigma^2$ , denoted  $D_{\mathbb{Z}^n,\sigma}$ , as the product distribution of  $n$  independent copies of  $D_{\mathbb{Z},\sigma}$ .

In general, the discrete Gaussian distribution does not exactly inherit the usual properties from the continuous case, but in our considered cases it will be close enough and we will use properties from the continuous case, as they are approximately correct. For example, if  $X$  is drawn from  $\mathcal{X}_{\sigma_1}$  and  $Y$  is drawn from  $\mathcal{X}_{\sigma_2}$ , then we consider  $X + Y$  to be drawn from  $\mathcal{X}_{\sqrt{\sigma_1^2 + \sigma_2^2}}$ . This follows the path of previous work [1].

A central point in cryptanalysis is to estimate the number of samples required to distinguish between two distributions, in our case the uniform distribution on  $\mathbb{Z}_q$  and  $\mathcal{X}_\sigma$ . The solution to this distinguishing problem leads to an efficient key recovery: we assume that for a right guess, the observed symbol is  $\mathcal{X}_\sigma$  distributed; otherwise, it is uniformly random. Thus, we need to distinguish the secret from  $Q$  candidates. We follow the theory from linear cryptanalysis [6] (also similar to that in correlation attacks [15]), that the number  $M$  of required samples to test

<sup>1</sup> It is also denoted  $\mathcal{X}_\sigma$ , and we omit  $\sigma$  if there is no ambiguity.

is about  $\mathcal{O}\left(\frac{\ln(Q)}{\Delta(\mathcal{X}_\sigma\|U)}\right)$ , where  $\Delta(\mathcal{X}_\sigma\|U)$  is the divergence<sup>2</sup> between  $\mathcal{X}_\sigma$  and the uniform distribution  $U$  in  $\mathbb{Z}_q$ .

## 2.2 LWE Problem Description

We already gave the definition of the search LWE problem in Definition 1. As we will focus on this problem, we skip giving a more formal definition of the decision version of LWE. Instead we reformulate the search LWE problem a bit. Assume that we ask for  $m$  samples from the LWE distribution  $L_{\mathbf{s},\mathcal{X}}$  and the response is denoted as

$$(\mathbf{a}_1, z_1), (\mathbf{a}_2, z_2), \dots, (\mathbf{a}_m, z_m),$$

where  $\mathbf{a}_i \in \mathbb{Z}_q^n, z_i \in \mathbb{Z}_q$ . We introduce  $\mathbf{z} = (z_1, z_2, \dots, z_m)$  and  $\mathbf{y} = (y_1, y_2, \dots, y_m) = \mathbf{s}\mathbf{A}$ . We can then write  $\mathbf{A} = [\mathbf{a}_1^T \ \mathbf{a}_2^T \ \dots \ \mathbf{a}_m^T]$  and  $\mathbf{z} = \mathbf{s}\mathbf{A} + \mathbf{e}$ , where  $z_i = y_i + e_i = \langle \mathbf{s}, \mathbf{a}_i \rangle + e_i$  and  $e_i \stackrel{\S}{\leftarrow} \mathcal{X}$  is the noise. We see that the problem has been reformulated as a decoding problem. The matrix  $\mathbf{A}$  serves as the generator matrix for a linear code over  $\mathbb{Z}_q$  and  $\mathbf{z}$  is the received word. Finding the codeword  $\mathbf{y} = \mathbf{s}\mathbf{A}$  such that the distance  $\|\mathbf{y} - \mathbf{z}\|$  is minimum will give the secret vector  $\mathbf{s}$ .

If the secret vector  $\mathbf{s}$  is drawn from the uniform distribution, there is a simple transformation [4, 23] that can be applied, namely, we may through Gaussian elimination transform  $\mathbf{A}$  into systematic form. Assume that the first  $n$  columns are linearly independent and form the matrix  $\mathbf{A}_0$ . Define  $\mathbf{D} = \mathbf{A}_0^{-1}$ . With a change of variables  $\hat{\mathbf{s}} = \mathbf{s}\mathbf{D}^{-1} = (z_1, z_2, \dots, z_n)$  we get an equivalent problem described by  $\hat{\mathbf{A}} = (\mathbf{I}, \hat{\mathbf{a}}_{n+1}^T, \hat{\mathbf{a}}_{n+2}^T, \dots, \hat{\mathbf{a}}_m^T)$ , where  $\hat{\mathbf{A}} = \mathbf{D}\mathbf{A}$ . We compute

$$\hat{\mathbf{z}} = \mathbf{z} - (z_1, z_2, \dots, z_n)\hat{\mathbf{A}} = (\mathbf{0}, \hat{z}_{n+1}, \hat{z}_{n+2}, \dots, \hat{z}_m).$$

After this initial step, each entry in the secret vector  $\hat{\mathbf{s}}$  is now distributed according to  $\mathcal{X}$ .

## 2.3 Lattice Codes and Construction A

A lattice  $\Lambda$  is a discrete additive subgroup of  $\mathbb{R}^n$ . Reformulated,  $\Lambda$  is a lattice iff there are linearly independent vectors  $\mathbf{v}_1, \dots, \mathbf{v}_m \in \mathbb{R}^n$ , such that any  $\mathbf{y} \in \Lambda$  can be written as  $\mathbf{y} = \sum_{i=1}^m \alpha_i \mathbf{v}_i$ , where  $\alpha_i \in \mathbb{Z}$ . The set  $\mathbf{v}_1, \dots, \mathbf{v}_m$  is called a basis for  $\Lambda$ . A matrix whose columns are these vectors is said to be a generator matrix for  $\Lambda$ .

Furthermore, let  $\text{Vol}(\cdot)$  denote the volume of a closed set in  $\mathbb{R}^n$  and let  $\mathcal{V}$  be the fundamental Voronoi region of  $\Lambda$ , i.e.,

$$\mathcal{V} = \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\| \leq \|\mathbf{x} - \mathbf{w}\|, \forall \mathbf{w} \in \Lambda\}.$$

<sup>2</sup> Divergence has a couple of aliases in literature: relative entropy, information divergence, Kullback-Leibler divergence, etc. We refer the interested reader to [15] for the rigorous definition. In this paper, the divergence  $\Delta(\mathcal{X}_\sigma\|U)$  will be computed numerically.

We will be interested in a more narrow class of lattices based on  $q$ -ary linear codes. If  $\mathcal{C}$  is a linear  $[N, k]$  code over the alphabet of size  $q$ , where  $q$  is a prime, then a lattice over this code is

$$\Lambda(\mathcal{C}) = \{\lambda \in \mathbb{R}^n : \lambda = (\mathbf{c} \bmod q), \mathbf{c} \in \mathcal{C}\}.$$

The mapping from the code to the lattice is often referred to as Construction A [14]. The lattice  $\Lambda(\mathcal{C})$  is called the  $q$ -ary lattice associated with  $\mathcal{C}$ .

A typical application is to use the lattice  $\Lambda(\mathcal{C})$  as a codebook for quantization of sequences  $\mathbf{x} \in \mathbb{R}^n$ . Let  $Q(\mathbf{x})$  be the lattice point closest to  $\mathbf{x}$  if the squared error is used as a fidelity criterion. We call  $Q(\mathbf{x})$  an MSE quantizer. The second moment of  $\Lambda$ , denoted by  $\sigma^2 = \sigma^2(\Lambda)$ , is defined as the second moment per dimension of a uniform distribution over its fundamental region  $\mathcal{V}$ , i.e.,

$$\sigma^2 = \frac{1}{n} \cdot \int_{\mathcal{V}} \|\mathbf{x}\|^2 \frac{1}{\text{Vol}(\mathcal{V})} d\mathbf{x}. \quad (1)$$

From the literature [13,14] of lattice codes, we know that the value  $\sigma^2$  can be represented as

$$\sigma^2 = G(\Lambda) \cdot \text{Vol}(\mathcal{V})^{\frac{2}{n}}, \quad (2)$$

where  $G(\Lambda)$  is called the normalized second moment, which represents a figure of merit of a lattice quantizer with respect to the MSE distortion measure. We denote the minimum possible value of  $G(\Lambda)$  over all lattices in  $\mathbb{R}^n$  by  $G(\Lambda_n)$  and it is known that

$$\frac{1}{2\pi e} < G(\Lambda_n) \leq \frac{1}{12}, \quad (3)$$

where the upper bound is achieved when the lattice is generated by  $\mathbb{Z}^n$ , and the lower one is achieved asymptotically by lattices generated by Construction A from  $q$ -ary random linear codes [18,27,35].

### 3 The BKW Algorithm

The BKW algorithm was proposed by Blum et al. [7] and was originally targeting the LPN problem. However, it is trivially adopted also to the LWE problem.

As with Wagner's generalized birthday algorithm, the BKW approach uses an iterative collision procedure on the columns in the generator matrix  $\mathbf{A}$ , which step by step reduces the dimension of  $\mathbf{A}$ . Summing together columns that collide in some subset of positions and keeping them as columns in a new matrix reduces the dimension but increases the size of the noise.

A brief description inspired by the notation in [22] follows. Initially, one searches for all combinations of two columns in  $\mathbf{A}$  with the same last  $b$  entries. Assume that one finds two columns  $\mathbf{a}_{i_1}^T, \mathbf{a}_{i_2}^T$  such that

$$\mathbf{a}_{i_1} - \mathbf{a}_{i_2} = (* * \cdots * \underbrace{00 \cdots 0}_{b \text{ symbols}}),$$

where  $*$  means any value. Then a new vector  $\mathbf{a}_1^{(2)} = \mathbf{a}_{i_1} - \mathbf{a}_{i_2}$  is formed. An “observed symbol” is also formed, corresponding to this new column by forming  $z_1^{(2)} = z_{i_1} - z_{i_2}$ . If  $y_1^{(2)} = \langle \mathbf{s}, \mathbf{a}_1^{(2)} \rangle$ , then  $z_1^{(2)} = y_1^{(2)} + e_1^{(2)}$ , where now  $e_1^{(2)} = e_{i_1} - e_{i_2}$ . Recall that noise like  $e_{i_1}$  follows the Gaussian distribution with variance  $\sigma^2$ , so  $e_1^{(2)} = e_{i_1} - e_{i_2}$  is considered to be Gaussian distributed with variance  $2\sigma^2$ . There is also a second obvious way of getting collisions, namely, combining any two vectors where the sum of the collision sets is zero. The procedure is analog to the above, just replacing subtraction with addition.

There are different approaches to realizing the above merging procedure. We consider the approach called LF1 in [24], which computes the difference between one fixed column and any other column with the same last  $b$  entries (in absolute value), and forwards this to the next BKW step.

Put all such new columns in a matrix  $\mathbf{A}_2$ ,

$$\mathbf{A}_2 = (\mathbf{a}_1^{(2)\top} \mathbf{a}_2^{(2)\top} \dots \mathbf{a}_{m_2}^{(2)\top}).$$

If  $m$  is the number of columns in  $\mathbf{A}$ , then we have the number of columns in  $\mathbf{A}_2$  to be  $m_2 = m - \frac{q^{b-1}}{2}$ . Hence, using the LF1 approach, the number of samples (columns) forwarded to the next step of BKW is slowly decreasing (by  $\frac{q^{b-1}}{2}$  for each step). It is known from simulation, that the LF2 approach [24] which gives more surviving samples, performs well and could be chosen in an actual implementation.

Now the last  $b$  entries of columns in  $\mathbf{A}_2$  are all zero. In connection to this matrix, the vector of observed symbols is

$$\mathbf{z}_2 = (z_1^{(2)} z_2^{(2)} \dots z_{m - \frac{q^{b-1}}{2}}^{(2)}),$$

where  $z_i^{(2)} - y_i^{(2)}$  are assumed Gaussian with variance  $2\sigma^2$ , for  $1 \leq i \leq m - \frac{q^{b-1}}{2}$ . This completes one step of the BKW algorithm.

We then iterate the same for  $i = 2, 3, \dots, t$ , picking a new collision set of size  $\frac{q^{b-1}}{2}$  and finding colliding columns in  $\mathbf{A}_i$ , giving new vectors with an additional  $b$  entries being zero, forming the columns of  $\mathbf{A}_{i+1}$ . Repeating the same procedure an additional  $t - 2$  times will reduce the number of unknowns in the secret vector  $\mathbf{s}$  to  $n - bt$  in the remaining problem.

For each iteration the noise is increased. After  $t$  BKW steps the noise connected to each column is of the form

$$e = \sum_{j=1}^{2^t} e_{i_j},$$

and the total noise is approximately Gaussian with variance  $2^t \cdot \sigma^2$ .

Altogether we have reduced the LWE instance to a smaller instance, where now the length of the secret vector is  $n' = n - tb$ , but the noise has variance  $2^t \cdot \sigma^2$ . The remaining unknown part of the secret vector  $\mathbf{s}$  is guessed (a total

of  $q^{n-tb}$ ) and for each guess we check through a hypothesis test whether the remaining samples follow the Gaussian distribution. The number of remaining samples is at least  $m - t \cdot \frac{q^b - 1}{2}$ .

Note that there is an improved version of BKW using lazy modulus reduction [3] and the very recent improvement in [17].

## 4 A Modified BKW Algorithm for the LWE Problem

The new algorithm we propose uses the same structure as the BKW algorithm. The new idea involves changing the BKW step to a more advanced step that can remove more positions in the treated vectors at the expense of leaving an additional noise term.

We introduce some additional notation. For the index set  $I$ , we make use of  $\mathbf{v}_I$  to denote the vector with entries indexed by  $I$ . Alternatively, we utilize the symbol  $\mathbf{v}_{[1, \dots, n]}$  to denote the vector containing the first  $n$  entries of  $\mathbf{v}$ , etc.

### 4.1 A New BKW Step

Recall the BKW step, taking a large number of vectors  $\mathbf{a}_i$  and trying to collide them in a set of positions determined by an index set  $I$ . This part of the vector  $\mathbf{a}$  is written as  $\mathbf{a}_I$ . The size of the collision set ( $\frac{q^b - 1}{2}$ ) and the number of vectors have to be of the same order, which essentially determines the complexity of the BKW algorithm, as the number of steps we can perform is determined by the variance of the noise.

We propose to do the BKW step in a different manner. Assuming that we are considering step  $i$  in the BKW process, we fix a  $q$ -ary linear code with parameters  $(N_i, b)$ , called  $\mathcal{C}_i$ . The code gives rise to a lattice code. Now, for any given vector  $\mathbf{a}_I$  as input to this BKW step, we approximate the vector by one of the codewords in the code  $\mathcal{C}_i$ .

We rewrite  $\mathbf{a}_I$  into two parts, the codeword part  $\mathbf{c}_I \in \mathcal{C}_i$  and an error part  $\mathbf{e}_I \in \mathbb{Z}_q^{N_i}$ , i.e.,

$$\mathbf{a}_I = \mathbf{c}_I + \mathbf{e}_I. \tag{4}$$

Clearly, we desire the error part to be as small as possible, so we adopt a decoding procedure to find the nearest codeword in the chosen code  $\mathcal{C}_i$  using the Euclidean metric. Here, we utilize syndrome decoding by maintaining a large syndrome table, and details will be discussed thoroughly later.

Each vector  $\mathbf{a}_I$  is then sorted according to which codeword it was mapped to. Altogether, there are  $q^b$  possible codewords. Finally, generate new vectors for the next BKW step by subtracting vectors mapped to the same codeword (or adding to the zero codeword).

The inner product  $\langle \mathbf{s}_I, \mathbf{a}_I \rangle$  is equal to

$$\langle \mathbf{s}_I, \mathbf{a}_I \rangle = \langle \mathbf{s}_I, \mathbf{c}_I \rangle + \langle \mathbf{s}_I, \mathbf{e}_I \rangle.$$

By subtracting two vectors mapped to the same codeword we cancel out the first part of the right hand side and we are left with the noise. The latter term is referred to as the error term introduced by coding.



Let us examine the samples we have received after  $t$  BKW steps of this kind. In step  $i$  we have removed  $N_i$  positions, so in total we have now removed  $\sum_{i=1}^t N_i$  positions ( $N_i \geq b$ ). The received samples are created from summing  $2^t$  original samples, so after guessing the remaining symbols in the secret vector and adjusting for its contribution, a received symbol  $z$  can be written as a sum of noise variables,

$$z = \sum_{j=1}^{2^t} e_{i_j} + \sum_{i=1}^n s_i (E_i^{(1)} + E_i^{(2)} + \cdots + E_i^{(t)}), \quad (5)$$

where  $E_i^{(h)} = \sum_{j=1}^{2^{t-h+1}} \hat{e}_{i_j}^{(h)}$  and  $\hat{e}_{i_j}^{(h)}$  is the coding noise introduced in step  $h$  of the modified BKW algorithm. Note that on one position  $i$ , at most one error term  $E_i^{(h)}$  is non-zero.

We observe that noise introduced in early steps is increased exponentially in the remaining steps, so the procedure will use a sequence of codes with decreasing rate. In this way the error introduced in early steps will be small and then it will eventually increase.

## 4.2 Analyzing the Error Distribution

There are many approaches to estimating the error distribution introduced by coding. The simplest way is just assuming that the value is a summation of several independent discrete Gaussian random variables. This estimation is easily performed and fairly accurate. A second approach is to compute the error distribution accurately (to sufficient precision) by computer. We should note that the error distribution is determined from the linear code employed. We now rely on some known result on lattice codes to provide a good estimate on the size of the noise introduced by coding.

We assume that the error vector  $\mathbf{e}$  introduced by the coding technique remains discrete Gaussian, and their summation is discrete Gaussian as well, just as in previous research. As the error is distributed symmetrically we should estimate the value  $\mathbb{E}[||\mathbf{e}||^2]$  to bound the effect of the error, where  $\mathbf{e}$  is the error vector distributed uniformly on the integer points inside the fundamental region  $\mathcal{V}$  of the lattice generated by Construction A.

Thus, the problem of decoding transforms to an MMSE quantizing problem over the corresponding lattice. For simplicity of analysis, we change the hypothesis and assume that the error vector  $\mathbf{e}$  is distributed uniformly and continuously on  $\mathcal{V}$ . Thus we can utilize the theory on lattice codes to give a fairly accurate estimation of the value  $\frac{1}{N} \mathbb{E}[||\mathbf{e}||^2]$ , which exactly corresponds to the second moment of the lattice  $\sigma^2$ . As given in Eq. (2), we can write it as,

$$\sigma^2 = G(A) \cdot Vol(\mathcal{V})^{\frac{2}{N}}.$$

In our scheme, although we employ several different linear codes with different rates, we also try to make the contribution of every dimension equal. We generate a lattice  $A$  by Construction A, given a linear code. We denote the minimum

possible value of  $G(\Lambda)$  over all lattices  $\Lambda$  in  $\mathbb{Z}^n$  generated by Construction A from an  $[N, k]$  linear code as  $G(\Lambda_{N,k})$ .

Definitely  $G(\Lambda_{N,k})$  is no less than  $G(\Lambda_N)$ ; thus it is lower bounded by the value  $\frac{1}{2\pi e}$  and this bound can be achieved asymptotically. For the lattice generated by  $\mathbb{Z}^N$ , i.e., employing a trivial linear code without redundancy, its normalized second moment is  $\frac{1}{12}$ . Therefore, the value  $G(\Lambda_{N,k})$  satisfies

$$\frac{1}{2\pi e} < G(\Lambda_{N,k}) \leq \frac{1}{12}.$$

We set  $G(\Lambda_{N,k})$  to be  $\frac{1}{12}$  and surely this is a pessimistic estimation. Since the lattice is built from a linear code by Construction A, the volume of  $\mathcal{V}$  is  $q^{N-k}$ . Thus, we can approximate  $\sigma$  by

$$\sigma \approx q^{1-k/N} \cdot \sqrt{G(\Lambda_{N,k})} = \frac{q^{1-k/N}}{\sqrt{12}}. \tag{6}$$

We have numerically tested the smallest possible variance of errors introduced by coding, given several small sizes of  $N, k$  and  $q$ , (e.g.,  $[N, k]$  is  $[3, 1]$  or  $[2, 1]$ ,  $q$  is 631, 2053 or 16411) and verified that the above estimation works (see Table 3, where  $1/G$  is bounding  $1/G(\Lambda_{N,k})$ ). We choose  $[N, 1]$  codes since for the covering or MMSE property, lower rate means worse performance.

It is folklore that the value  $G$  will decrease when the dimension and length becomes larger, and all the cases listed in Table 3 fully obey the rule. Thus we believe that we may have even better performance when employing a more complicated code for a larger problem. Actually, the values without a † sign in Table 3 is computed using randomly chosen linear codes, and they still outperform our estimation greatly. This observation fits the theory well that when the dimension  $n$  is large, a random linear code may act nearly optimally.

From Eq. (6) we know the variance of the error term from the coding part. Combining this with Eq. (5), we get an estimation of the variance of the total noise for the samples that we create after  $t$  modified BKW steps.

### 4.3 Decoding Method and Constraint

Here we discuss details of syndrome decoding and show that the additional cost is under control. Generally, we characterize the employed  $[N, k]$  linear code by a systematic generator matrix  $\mathbf{M} = [\mathbf{I} \mathbf{F}']_{k \times N}$ . Thus, a corresponding parity-check matrix  $\mathbf{H} = [\mathbf{F}'^T \mathbf{I}]_{(N-k) \times N}$  is directly obtained.

**Table 3.** Numerical evaluations on  $1/G$

$q$	631			2053			16411	
	[2,1]	[3,1]	[4,1]	[2,1]	[3,1]	[4,1]	[2,1]	[3,1]
$E[\ \mathbf{e}\ ^2]$	101.26 <sup>†</sup>	1277.31	4951.53	329.24 <sup>†</sup>	6185.67	29107.73	2631.99 <sup>†</sup>	99166.25
$1/G$	12.46	12.71	12.80	12.47	12.65	12.78	12.47	12.62

The value with a † sign means that it is optimal.

The syndrome decoding procedure is described as follows. (1) We construct a constant-time query table containing  $q^{N-k}$  items, in each of which we store the syndrome and its corresponding error vector with minimum Euclidean distance. (2) When the syndrome is computed, by checking the table, we locate its corresponding error vector and add them together, thereby yielding the desired nearest codeword.

We generalize the method in [22] to the non-binary case  $\mathbb{Z}_q$  for computing the syndrome efficiently. Starting by sorting the vectors  $\mathbf{a}_I$  by the first  $k$  entries, we then partition them accordingly; thus there are  $q^k$  partitions denoted  $\mathcal{P}_j$ , for  $1 \leq j \leq q^k$ . We can read the syndrome from its last  $N - k$  entries directly if the vector  $\mathbf{a}_I$  belongs to the partition with the first  $k$  entries all zero. Then we operate inductively. If we know one syndrome, we can compute another one in the same partition within  $2(N - k)$   $\mathbb{Z}_q$  operations, or compute one in a different partition whose first  $k$  entries with distance 1 from that in the known partition within  $3(N - k)$   $\mathbb{Z}_q$  operations. Suppose we have  $m_{dec}$  vectors to decode here (generally, the value  $m_{dec}$  is larger than  $q^k$ ), then the complexity of this part is bounded by  $(N - k)(2m_{dec} + q^k) < 3m_{dec}(N - k)$ . Since the cost of adding error vectors for the codewords is  $m_{dec}N$ , we can give an upper bound for the decoding cost, which is roughly  $4m_{dec}N$ .

**Concatenated Constructions.** The drawback of the previous decoding strategy is that a large table is required to be stored with size exponential in  $N - k$ . On the other hand, there is an inherent memory constraint, i.e.,  $\mathcal{O}(q^b)$ , when the size  $b$  is fixed, which dominates the complexity of the BKW-type algorithm.

We make use of a narrow sense concatenated code defined by direct summing several smaller linear codes to simplify the decoding procedure, when the decoding table is too large. This technique is not favored in coding theory since it diminishes the decoding capability, but it works well for our purpose.

## 5 Algorithm Description

We present a detailed description of the new algorithm in this section, containing five steps. This is illustrated in Algorithm 1 below.

### 5.1 Gaussian Elimination

The goal of this step is to transform the distribution of secret vector  $\mathbf{s}$  to be that of the error (c.f. [4, 23] for similar ideas). We refer to the full version for details on deriving the complexity of this step.

The complexity of this step is as follows,

$$C_0 = (m - n') \cdot (n + 1) \cdot \lceil \frac{n'}{b - 1} \rceil < m(n + 1) \cdot \lceil \frac{n'}{b - 1} \rceil, \quad (7)$$

where  $n' = n - t_1 b$ .

---

**Algorithm 1.** New LWE solving algorithm (main steps)
 

---

**Input:** Matrix  $\mathbf{A}$  with  $n$  rows and  $m$  columns, received vector  $\mathbf{z}$  of length  $m$  and algorithm parameters  $t_1, t_2, b, n_{test}, l, d$

**repeat**

- Step 1: Gaussian elimination to change the distribution of the secret vector;
- Step 2: Use  $t_1$  BKW steps to remove the bottom  $t_1 b$  entries;
- Step 3: Use  $t_2$  coded-BKW steps to remove the bottom  $n_{cod}$  entries;
- Step 4: **for every guess of the top  $n_{top}$  entries do**
  - Subspace Hypothesis Testing using an  $[n_{test}, l]$  linear code and FFT;

**until** *acceptable hypothesis is found*

---

## 5.2 Standard BKW Reductions

The previously described coded-BKW in Sect. 4 introduces noise that grows with each iteration, so it makes sense to start with a number of pure BKW reductions. We start by performing  $t_1$  standard BKW steps to balance the two noise parts, i.e., the noise increased by merging and the noise introduced by coding. This step zeros out the bottom  $t_1 \cdot b$  bits. We now explain the details.

Given the output of the Gaussian elimination, i.e.,  $\hat{\mathbf{z}}$  and  $\hat{\mathbf{A}} = (\mathbf{I}\mathbf{L}_0)$ , we process only on the non-systematic part of  $\hat{\mathbf{A}}$ , denoted by  $\mathbf{L}_0$ . Similar as the other BKW procedures [7], in each step we sort the vector by the last  $b$  unprocessed entries and thus divide the total samples into at most  $\frac{q^b-1}{2}$  classes. Then, we merge (adding or subtracting) those in the same class to zero the considered  $b$  entries, forming new samples as the input to the next BKW step,  $\mathbf{L}_1, \mathbf{L}_2$ , etc.

The output of this step is a matrix  $(\mathbf{I}\mathbf{L}_t)$ , where all  $t_1 b$  last entries in each column of  $\mathbf{L}_t$  are zero. Collecting the first  $n - t_1 b$  rows of the matrix  $\mathbf{L}_t$  and appending the identity matrix in front, we have a series of new LWE samples with dimension  $n - t_1 b$ . The complexity of this step is

$$C_1 = \sum_{i=1}^{t_1} (n + 1 - ib) \left( m - \frac{i(q^b - 1)}{2} \right). \quad (8)$$

## 5.3 Coded-BKW Reductions

We next continue to perform  $t_2$  coded-BKW steps, in each of which an  $[N_i, b]$   $q$ -ary linear code is utilized. Here various rates are employed to equalize the error contribution per dimension. The code length  $N_i$  in the  $(t_2 - i + 1)$ th coded-BKW step is a function of a preset variance value  $\sigma_{set}^2$  which is determined by the error level introduced by the codes utilized in the last phase — subspace hypothesis testing. We know that in the final error expression there are  $2^{t_2 - i + 1}$  error terms from the  $i$ -th coded BKW step. Thus, we have the following equation,

$$\sigma_{set}^2 = \frac{2^i q^{2(1 - \frac{b}{N_i})}}{12}.$$

Thus, the code length  $N_i$  is chosen as,

$$N_i = \left\lceil \frac{b}{1 - \frac{1}{2} \log_q(12 \cdot \frac{\sigma_{\text{test}}^2}{2^t})} \right\rceil.$$

By  $n_{\text{cod}}$  we denote the total number of positions canceled by the coded-BKW steps, i.e.,  $n_{\text{cod}} = \sum_{i=1}^{t_2} N_i$ . We denote the number of samples after the last coded-BKW step by  $M$ . Following Sect. 4.3, the decoding cost is upper bounded by

$$C_2' = \sum_{i=1}^{t_2} 4(M + \frac{i(q^b - 1)}{2})N_i,$$

where  $(M + \frac{i(q^b - 1)}{2})$  is the number of samples processed in the  $(t_2 - i + 1)$ -th step. Thus, the overall complexity of this step is

$$C_2 = C_2' + \sum_{i=1}^{t_2} (n_{\text{top}} + n_{\text{test}} + \sum_{j=1}^i N_j) (M + \frac{(i-1)(q^b - 1)}{2}). \quad (9)$$

#### 5.4 Partial Guessing

The previous step outputs samples with smaller dimension but higher noise variance. In order to deal with the remaining unknowns in the secret  $\hat{\mathbf{s}}$  vector, we use a combination of testing all values by guessing and performing a hypothesis test using an FFT.

In this step we perform a simple partial guessing technique, which balances the complexity of the previous steps and the later FFT testing phase. We exhaust the top  $n_{\text{top}}$  entries of  $\hat{\mathbf{s}}$  with the absolute value less than  $d$ ; thus there are  $(2d+1)^{n_{\text{top}}}$  candidates. Thus, the complexity of this step is just that of updating the observed symbol, i.e.,

$$C_3 = Mn_{\text{top}}(2d+1)^{n_{\text{top}}}. \quad (10)$$

The upcoming last step is performed for each such guess.

#### 5.5 Subspace Hypothesis Testing

Here we generalize the subspace hypothesis testing technique first proposed in [22] to  $\mathbb{Z}_q$  case, and then combine with Fast Fourier Transform to calculate the occurrences of different symbols in  $\mathbb{Z}_q$  efficiently. This information would yield an optimal distinguisher with a small additional cost.

We use a polynomial in the quotient ring  $\mathbb{Z}[X]/(X^q - 1)$  to record the occurrences. The modulus  $(X^q - 1)$  is determined by the group property of  $\mathbb{Z}_q$ . We employ an  $[n_{\text{test}}, l]$  systematic linear code, group the samples  $(\hat{\mathbf{a}}'_i, \hat{z}'_i)$  from the previous steps in sets  $L(\mathbf{c}_i)$  according to their nearest codewords and define the function  $f_L^{\mathbf{c}_i}(X)$  as

$$f_L^{\mathbf{c}_i}(X) = \sum_{(\mathbf{a}'_i, z'_i) \in L(\mathbf{c}_i)} X^{\hat{z}'_i \pmod q}.$$

Due to the systematic feature of the code utilized, we rewrite  $f_L^{\mathbf{c}_i}(X)$  as a function of the information part  $\mathbf{u}$  of the codeword  $\mathbf{c}_i$ , denoted by  $h_{\mathbf{u}}(X) = f_L^{\mathbf{c}_i}(X)$ , and later we exhaust all the  $q^l$  possible values of the vector  $\mathbf{u}$ . Define

$$H_{\mathbf{y}}(X) = \sum_{\mathbf{u} \in \mathbb{Z}_q^l} h_{\mathbf{u}}(X) \cdot X^{-\langle \mathbf{y}, \mathbf{u} \rangle}.$$

Here we exhaust all candidates of  $\mathbf{y} \in \mathbb{Z}_q^l$ . Then, there exists a unique vector  $\mathbf{y} \in \mathbb{Z}_q^l$ , s.t.,  $\langle \mathbf{y}, \mathbf{u} \rangle = \langle \hat{\mathbf{s}}, \mathbf{c}_i \rangle$ . For the right guess, the polynomial  $H_{\mathbf{y}}(X)$  will record the occurrences of the error symbols which are discrete Gaussian distributed; otherwise, it should be uniformly distributed.

The calculation of the polynomial  $H_{\mathbf{y}}(X)$  can be accelerated by Fast Fourier Transform. Let  $\omega$  be a primitive  $q$ -th root of unity in the complex field  $\mathbb{C}$ . We can interpolate the polynomial  $H_{\mathbf{y}}(X)$  if we know its  $q$  values at the  $q$  different points  $(1, \omega, \omega^2, \dots, \omega^{q-1})$  with complexity about  $\mathcal{O}(q \log_2(q))$ . Thus, the problem is transformed to a polynomial evaluation problem.

We first evaluate  $q^l$  polynomials  $h_{\mathbf{u}}(X)$  on  $q$  different points  $(1, \omega, \omega^2, \dots, \omega^{q-1})$  with the complexity  $\mathcal{O}(q^l \cdot q \log_2 q)$ . Then with these values stored, we can evaluate the polynomial  $H_{\mathbf{y}}(X)$  using  $q$  FFTs, each of which costs  $\mathcal{O}(q^l \log_2(q^l))$ .

If the symbol occurrences are known, then we obtain the belief levels of all the candidates using a Neyman-Pearson test [15]. We choose the one with the highest rank and output it. This testing adds  $\mathcal{O}(q^{l+1})$   $\mathbb{Z}_q$ -operations. Similar to that in the LPN case [22], recovering the remaining information can be done by iteratively employing this procedure to solve smaller LWE instances whose complexity is negligible compared to that of knowing the first part.

The employed  $[n_{test}, l]$  linear code brings in an error with variance per dimension  $\frac{q^{2(1-l/n_{test})}}{12}$ . We denote it by  $\sigma_{set}^2$ , which is manipulated as a preset parameter in the previous coded-BKW phase to control the code sizes. As before, the decoding cost in this step is upper bounded by

$$C'_4 = 4Mn_{test}.$$

We introduce a new notation  $n_{tot} = n_{cod} + n_{test}$  to denote the total length of the subvectors affected by coding. The overall complexity of this step is given as

$$C_4 = C'_4 + (2d + 1)^{n_{top}} (C_{FFT} \cdot q^{l+1} (l + 1) \log_2 q + q^{l+1}). \quad (11)$$

Here  $C_{FFT}$  is the constant before the complexity order of an FFT.

## 6 Analysis of the New Approach for BKW

We denote by  $P(d)$  the probability that the absolute value of one guessed symbol  $\hat{s}_i$  is smaller than  $d$ , where  $\hat{s}_i \stackrel{\$}{\leftarrow} \mathcal{X}_\sigma$ . Here we obtain a lower bound of  $P(d)$  by ignoring the folding feature of the distribution as  $P(d) > \text{erf}(\frac{d}{\sqrt{2}\sigma})$ , where  $\text{erf}$  is the error function  $\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$ .

In the testing step, we preset a noise level  $\gamma^2\sigma^2\sigma_{set}^2n_{tot}$  to be the variance of the noise introduced by coding, and then compute the required number of samples to perform a successful distinguishing. The process may fail if the size of the information subvector to be tested, denoted  $\hat{\mathbf{s}}_{test}$ , is too large to distinguish. Thus we need a new notion,  $P_{test}$ , to denote the probability that the Euclidean length of  $\hat{\mathbf{s}}_{test}$  is less than a preset value  $\gamma\sqrt{n_{tot}}\sigma$ . Using the following lemma from [28], which is a tail bound on discrete Gaussians, we can upper bound the failure probability by  $(\gamma e^{\frac{1-\gamma^2}{2}})^{n_{tot}}$ .

**Lemma 1.** *For any  $\gamma \geq 1$ ,  $\Pr[\|\mathbf{v}\| > \gamma\sigma\sqrt{n}; \mathbf{v} \stackrel{\$}{\leftarrow} D_{\mathbb{Z}^n, \sigma}] < (\gamma e^{\frac{1-\gamma^2}{2}})^n$ .*

Later we set the value  $\gamma$  to be 1.2. Then, the estimated success probability is larger than 97.5% in most of the applications. We summarize our findings in the following theorem.

**Theorem 1 (The Complexity of Algorithm 1).** *Let  $(n, q, \sigma)$  be the parameters of the chosen LWE instance. Let  $d, t_1, t_2, b, l, n_{test}$  be algorithm parameters. The number of  $\mathbb{Z}_q$  operations required for a successful run of the new attack is*

$$C = \frac{C_0 + C_1 + C_2 + C_3 + C_4}{(P(d))^{n_{top}} \cdot P_{test}}, \quad (12)$$

with  $C_0, \dots, C_4$  as in Eqs. (7)–(11).

The required number of samples  $M$  for testing is set to be<sup>3</sup>

$$M = \frac{4 \ln((2d+1)^{n_{top}} q^l)}{\Delta(\mathcal{X}_{\sigma_{final}} \| U)},$$

where  $U$  is the uniform distribution in  $\mathbb{Z}_q$  and  $\sigma_{final}^2 = 2^{t_1+t_2}\sigma^2 + \gamma^2\sigma^2\sigma_{set}^2n_{tot}$ . Thus, the number of calls to the LWE oracle is

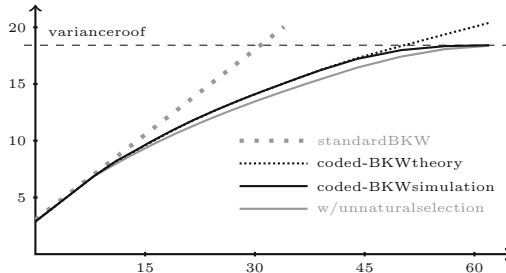
$$m = \frac{(t_1 + t_2)(q^b - 1)}{2} + M.$$

*Proof.* The cost for one iteration is  $C_0 + C_1 + C_2 + C_3 + C_4$ , which should be divided by its expected success probability  $(P(d))^{n_{top}} \cdot P_{test}$ .

## 7 A Variant of Coded-BKW for Binary-LWE

We can derive an efficient algorithm for BINARY-LWE by modifying certain steps accordingly. First, the distribution of the information vector is already of small size in  $\mathbb{Z}_q$ ; therefore we skip the Gaussian elimination step. In addition, since the prime  $q$  is a relatively large symbol, it is beneficial to replace the step of the FFT hypothesis testing by a simple step exhausting all the combinations of the top  $n_{top}$  entries, which are uniformly chosen from the binary set  $\{0, 1\}^{n_{top}}$ . The variant is similar to Algorithm 1, so we omit it here and refer the interested reader to the full version for details.

<sup>3</sup> The constant factor in the formula is chosen as 4. According to some estimates on linear and differential cryptanalysis (e.g., [6, 33]), its failure probability is fairly low.



**Fig. 1.** Number of eliminated rows vs.  $\log_2$  of error variance. Number of eliminated rows vs.  $\log_2$  of error variance.

## 8 Simulation

We have performed simulations to support our theoretical results. A simulation with parameters  $(q, \sigma, \#samples) = (2053, 2.70, 2^{25})$  is shown in Fig. 1, plotting the number of eliminated rows vs.  $\log_2$  of the variance of the samples errors. Four standard 2-row BKW steps were used initially, followed by three iterations each of [3,2]-, [4,2]-, [5,2]- and [6,2]-coding steps. The dashed horizontal line shows the variance of the corresponding uniform distribution (variance roof) of the errors, setting an upper bound for variance in simulations. The four curves show the performances of 2-step BKW (theoretical), theoretical coded-BKW (according to Sect. 6), coded-BKW simulation, and coded-BKW simulation when employing the unnatural selection heuristic (see [3]).

It is clear that coded-BKW significantly outperforms plain BKW. Furthermore, it can be seen that the developed theoretical estimations for coded-BKW very closely match actual simulation performance.

Last but not least, the unnatural selection heuristic can be employed by producing more samples, but retaining only the ones with the smallest coding errors. Instead of producing  $2^{25}$  samples,  $2^{27}$  were produced at each step. There is a clear gain in variance performance, and that gain is even larger when the sample factor is increased. These results will be detailed in the full version.

## 9 Summary of Results

We now present numerical results, as shown in Tables 1 and 2, using the new algorithms to solve the LWE and BINARY-LWE problems for various parameter settings, including instances from Regev’s cryptosystem [31] or from Lindner and Peikert’s paper [25]. As in [17], we consider operations over  $\mathbb{C}$  to have the same complexity as the operation in  $\mathbb{Z}_q$ , and set  $C_{FFT}$  to be 1, which is the best we can obtain for an FFT. We also set  $\gamma = 1.2$  and  $d = 3\sigma$ .

As in [1], we apply the new method to the instances proposed in a somewhat homomorphic encryption scheme [2], which can be considered as LWE instances using linearization. Our method yields substantial improvements in all cases



and especially solves an instance with the number of variables in the linearized system  $n = 153$  (targeting 128-bit security [1]), in about  $2^{119}$  bit operations, thereby breaking the claimed security level.

We present here additional information about the comparisons in Tables 1 and 2. Firstly, only the new algorithms and the algorithm proposed in [17] are key-recovery attacks; all the others belong to the class of distinguishing attacks. Secondly, the counterpart proposed by Albrecht et al. [3] is the version without unnatural selection, since we can also improve our algorithm by this heuristic. Thus, we accelerate the BKW-type BINARY-LWE solver by a factor of almost  $2^{20}$ , for the toy instance  $n = 128$  in Regev’s parameter setting. Last, we adopt the estimating model in [1, 3] using data from the implementations in [11, 25, 26] to evaluate the performance of the lattice reduction distinguisher, when LWE is reduced to SIS. We refer the interested readers to these two papers for details.

When reducing LWE to BDD, also named “Decode” in [25], Lindner and Peikert reported the running time of this attack on two LWE instances. Albrecht et al. [1] multiplied the time by the clock speed of the CPU used, compared with their BKW variant, and finally reached the conclusion that the BDD approach would yield substantially lower complexity. Specifically, their estimation about this “Decode” approach on one (with parameter (136, 2003, 5.19)) of the two instances is about  $2^{91.4} \mathbb{Z}_q$  operations. We obtain a much better time complexity of about  $2^{80.6}$  operations over  $\mathbb{Z}_q$ , when applying Algorithm 1 to this instance.

As RING-LWE is a sub-problem of LWE, the new algorithm can be employed to attack some recent RING-LWE-based cryptosystems [16, 21, 32]. We solve the underlying RING-LWE (256, 7681, 4.51) and RING-LWE (512, 12289, 4.86) instantiations in  $2^{123}$  and  $2^{225}$  bit-operations, respectively, thereby breaking the claimed 128-bit and 256-bit security levels.

## 10 Conclusion

We have proposed a new algorithm to solve the LWE problem by modifying the steps of the BKW algorithm using lattice codes. Our algorithm outperforms the previous BKW variants for all instantiations we considered and also all the lattice reduction approaches from some size of instances and onwards. To the best of our knowledge, it is the best LWE solver when the dimension  $n$  is large enough and it seems to cover the choices of today’s and future security levels. Another application is that it outperforms all the other approaches drastically on the BINARY-LWE problem.

## References

1. Albrecht, M.R., Cid, C., Faugère, J.C., Fitzpatrick, R., Perret, L.: On the Complexity of the BKW Algorithm on LWE. *Desig. Codes Crypt.* **74**, 1–30 (2013)
2. Albrecht, M.R., Farshim, P., Faugère, J.-C., Perret, L.: Polly cracker, revisited. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 179–196. Springer, Heidelberg (2011)

3. Albrecht, M.R., Faugère, J.-C., Fitzpatrick, R., Perret, L.: Lazy modulus switching for the BKW algorithm on LWE. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 429–445. Springer, Heidelberg (2014)
4. Applebaum, B., Cash, D., Peikert, C., Sahai, A.: Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 595–618. Springer, Heidelberg (2009)
5. Arora, S., Ge, R.: New algorithms for learning in presence of errors. In: Aceto, L., Henzinger, M., Sgall, J. (eds.) ICALP 2011, Part I. LNCS, vol. 6755, pp. 403–415. Springer, Heidelberg (2011)
6. Baignères, T., Junod, P., Vaudenay, S.: How far can we go beyond linear cryptanalysis? In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 432–450. Springer, Heidelberg (2004)
7. Blum, A., Kalai, A., Wasserman, H.: Noise-tolerant learning, the parity problem, and the statistical query model. *J. ACM* **50**(4), 506–519 (2003)
8. Brakerski, Z.: Fully homomorphic encryption without modulus switching from classical GapSVP. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 868–886. Springer, Heidelberg (2012)
9. Brakerski, Z., Langlois, A., Peikert, C., Regev, O., Stehlé, D.: Classical hardness of learning with errors. In: Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing, STOC 2013, pp. 575–584. ACM (2013)
10. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) LWE. In: Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science, pp. 97–106. IEEE Computer Society (2011)
11. Chen, Y., Nguyen, P.Q.: BKZ 2.0: better lattice security estimates. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 1–20. Springer, Heidelberg (2011)
12. Cohen, G., Honkala, I., Litsyn, S., Lobstein, A.: *Covering Codes*, vol. 54. Elsevier, Amsterdam (1997)
13. Conway, J., Sloane, N.: Voronoi regions of lattices, second moments of polytopes, and quantization. *IEEE Trans. Inf. Theory* **28**(2), 211–226 (1982)
14. Conway, J.H., Sloane, N.J.A., Bannai, E., Leech, J., Norton, S., Odlyzko, A., Parker, R., Queen, L., Venkov, B.: *Sphere Packings, Lattices and Groups*, vol. 3. Springer, New York (1993)
15. Cover, T.M., Thomas, J.A.: *Elements of Information Theory*. Wiley, New York (2012)
16. De Clercq, R., Roy, S.S., Vercauteren, F., Verbauwhede, I.: Efficient software implementation of Ring-LWE Encryption. In: Design, Automation and Test in Europe (DATE 2015) (2015)
17. Duc, A., Tramèr, F., Vaudenay, S.: Better algorithms for LWE and LWR. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 173–202. Springer, Heidelberg (2015)
18. Erez, U., Litsyn, S., Zamir, R.: Lattices which are good for (almost) everything. *IEEE Trans. Inf. Theory* **51**(10), 3401–3416 (2005)
19. Gentry, C.: A fully homomorphic encryption scheme. Ph.D. thesis, Stanford University (2009)
20. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: conceptually-simpler, asymptotically-faster, attribute-based. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 75–92. Springer, Heidelberg (2013)

21. Göttert, N., Feller, T., Schneider, M., Buchmann, J., Huss, S.: On the design of hardware building blocks for modern lattice-based encryption schemes. In: Prouff, E., Schaumont, P. (eds.) CHES 2012. LNCS, vol. 7428, pp. 512–529. Springer, Heidelberg (2012)
22. Guo, Q., Johansson, T., Löndahl, C.: Solving LPN using covering codes. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8873, pp. 1–20. Springer, Heidelberg (2014)
23. Kirchner, P.: Improved generalized birthday attack. Cryptology ePrint Archive, Report 2011/377 (2011)
24. Leveil, É., Fouque, P.-A.: An improved LPN algorithm. In: De Prisco, R., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 348–359. Springer, Heidelberg (2006)
25. Lindner, R., Peikert, C.: Better key sizes (and Attacks) for LWE-based encryption. In: Kiayias, A. (ed.) CT-RSA 2011. LNCS, vol. 6558, pp. 319–339. Springer, Heidelberg (2011)
26. Liu, M., Nguyen, P.Q.: Solving BDD by enumeration: an update. In: Dawson, E. (ed.) CT-RSA 2013. LNCS, vol. 7779, pp. 293–309. Springer, Heidelberg (2013)
27. Loeliger, H.A.: Averaging bounds for lattices and linear codes. *IEEE Trans. Inf. Theory* **43**(6), 1767–1773 (1997)
28. Lyubashevsky, V.: Lattice signatures without trapdoors. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 738–755. Springer, Heidelberg (2012)
29. Micciancio, D., Regev, O.: Lattice-based cryptography. In: Bernstein, D.J., Buchmann, J., Dahmen, E. (eds.) Post-Quantum Cryptography, pp. 147–191. Springer, Berlin Heidelberg (2009)
30. Peikert, C.: Public-key cryptosystems from the worst-case shortest vector problem. In: Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing, pp. 333–342. ACM (2009)
31. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. *J. ACM* **56**(6), 34:1–34:40 (2009)
32. Roy, S.S., Vercauteren, F., Mentens, N., Chen, D.D., Verbauwhe, I.: Compact Ring-LWE cryptoprocessor. In: Batina, L., Robshaw, M. (eds.) CHES 2014. LNCS, vol. 8731, pp. 371–391. Springer, Heidelberg (2014)
33. Selçuk, A.A.: On probability of success in linear and differential cryptanalysis. *J. Crypt.* **21**(1), 131–147 (2008)
34. Wagner, D.: A generalized birthday problem. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, p. 288. Springer, Heidelberg (2002)
35. Zamir, R., Feder, M.: On lattice quantization noise. *IEEE Trans. Inf. Theory* **42**(4), 1152–1159 (1996)

# An Improved BKW Algorithm for LWE with Applications to Cryptography and Lattices

Paul Kirchner<sup>1</sup> and Pierre-Alain Fouque<sup>2</sup>(✉)

<sup>1</sup> École Normale Supérieure, Paris, France  
paul.kirchner@ens.fr

<sup>2</sup> Université de Rennes 1 and Institut Universitaire de France, Rennes, France  
pierre-alain.fouque@ens.fr

**Abstract.** In this paper, we study the Learning With Errors problem and its binary variant, where secrets and errors are binary or taken in a small interval. We introduce a new variant of the Blum, Kalai and Wasserman algorithm, relying on a quantization step that generalizes and fine-tunes modulus switching. In general this new technique yields a significant gain in the constant in front of the exponent in the overall complexity. We illustrate this by solving within half a day a LWE instance with dimension  $n = 128$ , modulus  $q = n^2$ , Gaussian noise  $\alpha = 1/(\sqrt{n/\pi} \log^2 n)$  and binary secret, using  $2^{28}$  samples, while the previous best result based on BKW claims a time complexity of  $2^{74}$  with  $2^{60}$  samples for the same parameters.

We then introduce variants of BDD, GapSVP and UniqueSVP, where the target point is required to lie in the fundamental parallelepiped, and show how the previous algorithm is able to solve these variants in subexponential time. Moreover, we also show how the previous algorithm can be used to solve the BinaryLWE problem with  $n$  samples in subexponential time  $2^{(\ln 2/2 + o(1))n/\log \log n}$ . This analysis does not require any heuristic assumption, contrary to other algebraic approaches; instead, it uses a variant of an idea by Lyubashevsky to generate many samples from a small number of samples. This makes it possible to asymptotically and heuristically break the NTRU cryptosystem in subexponential time (without contradicting its security assumption). We are also able to solve subset sum problems in subexponential time for density  $o(1)$ , which is of independent interest: for such density, the previous best algorithm requires exponential time. As a direct application, we can solve in subexponential time the parameters of a cryptosystem based on this problem proposed at TCC 2010.

## 1 Introduction

The Learning With Errors (LWE) problem has been an important problem in cryptography since its introduction by Regev in [34]. Many cryptosystems have been proven secure assuming the hardness of this problem, including Fully Homomorphic Encryption schemes [11, 16]. The decision version of the problem can be described as follows: given  $m$  samples of the form  $(\mathbf{a}, b) \in (\mathbb{Z}_q)^n \times \mathbb{Z}_q$ , where  $\mathbf{a}$  are

uniformly distributed in  $(\mathbb{Z}_q)^n$ , distinguish whether  $b$  is uniformly chosen in  $\mathbb{Z}_q$  or is equal to  $\langle \mathbf{a}, \mathbf{s} \rangle + e$  for a fixed secret  $\mathbf{s} \in (\mathbb{Z}_q)^n$  and  $e$  a noise value in  $\mathbb{Z}_q$  chosen according to some probability distribution. Typically, the noise is sampled from some distribution concentrated on small numbers, such as a discrete Gaussian distribution with standard deviation  $\alpha q$  for  $\alpha = o(1)$ . In the search version of the problem, the goal is to recover  $\mathbf{s}$  given the promise that the sample instances come from the latter distribution. Initially, Regev showed that if  $\alpha q \geq 2\sqrt{n}$ , solving LWE on average is at least as hard as approximating lattice problems in the worst case to within  $\tilde{O}(n/\alpha)$  factors with a quantum algorithm. Peikert shows a classical reduction when the modulus is large  $q \geq 2^n$  in [32]. Finally, in [10], Brakerski *et al.* prove that solving LWE instances with polynomial-size modulus in polynomial time implies an efficient solution to GapSVP.

There are basically three approaches to solving LWE: the first relies on lattice reduction techniques such as the LLL [23] algorithm and further improvements [12] as exposed in [25, 26]; the second uses combinatorial techniques [9, 35]; and the third uses algebraic techniques [6]. According to Regev in [1], the best known algorithm to solve LWE is the algorithm by Blum, Kalai and Wasserman in [9], originally proposed to solve the Learning Parities with Noise (LPN) problem, which can be viewed as a special case of LWE where  $q = 2$ . The time and memory requirements of this algorithm are both exponential for LWE and subexponential for LPN in  $2^{\mathcal{O}(n/\log n)}$ . During the first stage of the algorithm, the dimension of  $\mathbf{a}$  is reduced, at the cost of a (controlled) decrease of the bias of  $b$ . During the second stage, the algorithm distinguishes between LWE and uniform by evaluating the bias.

Since the introduction of LWE, some variants of the problem have been proposed in order to build more efficient cryptosystems. Some of the most interesting variants are Ring-LWE by Lyubashevsky, Peikert and Regev in [29], which aims to reduce the space of the public key using cyclic samples; and the cryptosystem by Döttling and Müller-Quade [14], which uses short secret and error. In 2013, Micciancio and Peikert [30] as well as Brakerski *et al.* [10] proposed a binary version of the LWE problem and obtained a hardness result.

**Related Work.** Albrecht *et al.* have presented an analysis of the BKW algorithm as applied to LWE in [3, 4]. It has been recently revisited by Duc *et al.*, who use a multi-dimensional FFT in the second stage of the algorithm [15]. However, the main bottleneck is the first BKW step and since the proposed algorithms do not improve this stage, the overall asymptotic complexity is unchanged.

In the case of the BinaryLWE variant, where the error and secret are binary (or sufficiently small), Micciancio and Peikert show that solving this problem using  $m = n(1 + \Omega(1/\log(n)))$  samples is at least as hard as approximating lattice problems in the worst case in dimension  $\Theta(n/\log(n))$  with approximation factor  $\tilde{O}(\sqrt{n}q)$ . We show in the full version that existing lattice reduction techniques require exponential time. Arora and Ge describe a  $2^{\tilde{O}(\alpha q)^2}$ -time algorithm when  $q > n$  to solve the LWE problem [6]. This leads to a subexponential time algorithm when the error magnitude  $\alpha q$  is less than  $\sqrt{n}$ . The idea is to transform this system into a noise-free polynomial system and then use root finding

algorithms for multivariate polynomials to solve it, using either relinearization in [6] or Gröbner basis in [2]. In this last work, Albrecht *et al.* present an algorithm whose time complexity is  $2^{\frac{(\omega+o(1))n \log \log \log n}{8 \log \log n}}$  when the number of samples  $m = (1 + o(1))n \log \log n$  is super-linear, where  $\omega < 2.3728$  is the linear algebra constant, under some assumption on the regularity of the polynomial system of equations; and when  $m = \mathcal{O}(n)$ , the complexity becomes exponential.

**Contribution.** Our first contribution is to present in a unified framework the BKW algorithm and all its previous improvements in the binary case [8, 18, 21, 24] and in the general case [4]. We introduce a new quantization step, which generalizes modulus switching [4]. This yields a significant decrease in the constant of the exponential of the complexity for LWE. Moreover our proof does not require Gaussian noise, and does not rely on unproven independence assumptions. Our algorithm is also able to tackle problems with larger noise.

We then introduce generalizations of the BDD, GapSVP and UniqueSVP problems, and prove a reduction from these variants to LWE. When particular parameters are set, these variants impose that the lattice point of interest (the point of the lattice that the problem essentially asks to locate: for instance, in the case of BDD, the point of the lattice closest to the target point) lie in the fundamental parallelepiped; or more generally, we ask that the coordinates of this point relative to the basis defined by the input matrix  $\mathbf{A}$  has small infinity norm, bounded by some value  $B$ . For small  $B$ , our main algorithm yields a subexponential-time algorithm for these variants of BDD, GapSVP and UniqueSVP.

Through a reduction to our variant of BDD, we are then able to solve the subset-sum problem in subexponential time when the density is  $o(1)$ , and in time  $2^{(\ln 2/2+o(1))n/\log \log n}$  if the density is  $\mathcal{O}(1/\log n)$ . This is of independent interest, as existing techniques for density  $o(1)$ , based on lattice reduction, require exponential time. As a consequence, the cryptosystems of Lyubashevsky, Palacio and Segev at TCC 2010 [28] can be solved in subexponential time.

As another application of our main algorithm, we show that BinaryLWE with reasonable noise can be solved in time  $2^{(\ln 2/2+o(1))n/\log \log n}$  instead of  $2^{\Omega(n)}$ ; and the same complexity holds for secret of size up to  $2^{\log^{o(1)} n}$ . As a consequence, we can heuristically recover the secret polynomials  $\mathbf{f}, \mathbf{g}$  of the NTRU problem in subexponential time  $2^{(\ln 2/2+o(1))n/\log \log n}$  (without contradicting its security assumption). The heuristic assumption comes from the fact that NTRU samples are not random, since they are rotations of each other: the heuristic assumption is that this does not significantly hinder BKW-type algorithms. Note that there is a large value hidden in the  $o(1)$  term, so that our algorithm does not yield practical attacks for recommended NTRU parameters.

Our results are extended to the case where the secret is small with respect to the L2 norm in the full version.

## 2 Preliminaries

We identify any element of  $\mathbb{Z}/q\mathbb{Z}$  to the smallest of its equivalence class, the positive one in case of tie. Any vector  $\mathbf{x} \in (\mathbb{Z}/q\mathbb{Z})^n$  has an Euclidean norm

$\|\mathbf{x}\| = \sqrt{\sum_{i=0}^{n-1} x_i^2}$  and  $\|\mathbf{x}\|_\infty = \max_i |x_i|$ . A matrix  $\mathbf{B}$  can be Gram-Schmidt orthogonalized in  $\tilde{\mathbf{B}}$ , and its norm  $\|\mathbf{B}\|$  is the maximum of the norm of its columns. We denote by  $(\mathbf{x}|\mathbf{y})$  the vector obtained as the concatenation of vectors  $\mathbf{x}, \mathbf{y}$ . Let  $\mathbf{I}$  be the identity matrix and we denote by  $\ln$  the neperian logarithm and  $\log$  the binary logarithm. A lattice is the set of all integer linear combinations  $\Lambda(\mathbf{b}_1, \dots, \mathbf{b}_n) = \sum_i \mathbf{b}_i \cdot x_i$  (where  $x_i \in \mathbb{Z}$ ) of a set of linearly independent vectors  $\mathbf{b}_1, \dots, \mathbf{b}_n$  called the basis of the lattice. If  $\mathbf{B} = [\mathbf{b}_1, \dots, \mathbf{b}_n]$  is the matrix basis, lattice vectors can be written as  $\mathbf{B}\mathbf{x}$  for  $\mathbf{x} \in \mathbb{Z}^n$ . Its dual  $\Lambda^*$  is the set of  $\mathbf{x} \in \mathbb{R}^n$  such that  $(\mathbf{x}, \Lambda) \subset \mathbb{Z}^n$ . We have  $\Lambda^{**} = \Lambda$ . We borrow Bleichenbacher's definition of bias [31].

**Definition 1.** *The bias of a probability distribution  $\phi$  over  $\mathbb{Z}/q\mathbb{Z}$  is*

$$\mathbb{E}_{x \sim \phi}[\exp(2i\pi x/q)].$$

This definition extends the usual definition of the bias of a coin in  $\mathbb{Z}/2\mathbb{Z}$ : it preserves the fact that any distribution with bias  $b$  can be distinguished from uniform with constant probability using  $\Omega(1/b^2)$  samples, as a consequence of Hoeffding's inequality; moreover the bias of the sum of two independent variable is still the product of their biases. We also have the following simple lemma:

**Lemma 1.** *The bias of the Gaussian distribution of mean 0 and standard deviation  $q\alpha$  is  $\exp(-2\pi^2\alpha^2)$ .*

*Proof.* The bias is the value of the Fourier transform at  $-1/q$ . □

We introduce a non standard definition for the LWE problem. However as a consequence of Lemma 1, this new definition naturally extends the usual Gaussian case (as well as its standard extensions such as the bounded noise variant [10, Definition 2.14]), and it will prove easier to work with.

**Definition 2.** *Let  $n \geq 0$  and  $q \geq 2$  be integers. Given parameters  $\alpha$  and  $\epsilon$ , the LWE distribution is, for  $\mathbf{s} \in (\mathbb{Z}/q\mathbb{Z})^n$ , a distribution on pairs  $(\mathbf{a}, b) \in (\mathbb{Z}/q\mathbb{Z})^n \times (\mathbb{R}/q\mathbb{Z})$  such that  $\mathbf{a}$  is sampled uniformly, and for all  $\mathbf{a}$ ,*

$$|\mathbb{E}[\exp(2i\pi(\langle \mathbf{a}, \mathbf{s} \rangle - b)/q)|\mathbf{a}] \exp(\alpha'^2) - 1| \leq \epsilon$$

for some universal  $\alpha' \leq \alpha$ .

For convenience, we define  $\beta = \sqrt{n/2}/\alpha$ . In the remainder,  $\alpha$  is called the noise parameter<sup>1</sup>, and  $\epsilon$  the distortion parameter. Also, we say that a LWE distribution has a noise distribution  $\phi$  if  $b$  is distributed as  $\langle \mathbf{a}, \mathbf{s} \rangle + \phi$ .

**Definition 3.** *The Decision-LWE problem is to distinguish a LWE distribution from the uniform distribution over  $(\mathbf{a}, b)$ . The Search-LWE problem is, given samples from a LWE distribution, to find  $\mathbf{s}$ .*

<sup>1</sup> Remark that it differs by a constant factor from other authors' definition of  $\alpha$ .

**Definition 4.** *The real  $\lambda_i$  is the radius of the smallest ball, centered in  $\mathbf{0}$ , such that it contains  $i$  vectors of the lattice  $\Lambda$  which are linearly independent.*

We define  $\rho_s(\mathbf{x}) = \exp(-\pi\|\mathbf{x}\|^2/s^2)$  and  $\rho_s(S) = \sum_{\mathbf{x} \in S} \rho_s(\mathbf{x})$  (and similarly for other functions). The discrete Gaussian distribution  $D_{E,s}$  over a set  $E$  and of parameter  $s$  is such that the probability of  $D_{E,s}(\mathbf{x})$  of drawing  $\mathbf{x} \in E$  is equal to  $\rho_s(\mathbf{x})/\rho_s(E)$ . To simplify notation, we will denote by  $D_E$  the distribution  $D_{E,1}$ .

**Definition 5.** *The smoothing parameter  $\eta_\epsilon$  of the lattice  $\Lambda$  is the smallest  $s$  such that  $\rho_{1/s}(\Lambda^*) = 1 + \epsilon$ .*

Now, we will generalize the BDD, UniqueSVP and GapSVP problems by using another parameter  $B$  that bounds the target lattice vector. For  $B = 2^n$ , we recover the usual definitions if the input matrix is reduced.

**Definition 6.** *The  $\text{BDD}_{B,\beta}^{\|\cdot\|_\infty}$  (resp.  $\text{BDD}_{B,\beta}^{\|\cdot\|}$ ) problem is, given a basis  $\mathbf{A}$  of the lattice  $\Lambda$ , and a point  $\mathbf{x}$  such that  $\|\mathbf{A}\mathbf{s} - \mathbf{x}\| \leq \lambda_1/\beta < \lambda_1/2$  and  $\|\mathbf{s}\|_\infty \leq B$  (resp.  $\|\mathbf{s}\| \leq B$ ), to find  $\mathbf{s}$ .*

**Definition 7.** *The  $\text{UniqueSVP}_{B,\beta}^{\|\cdot\|_\infty}$  (resp.  $\text{UniqueSVP}_{B,\beta}^{\|\cdot\|}$ ) problem is, given a basis  $\mathbf{A}$  of the lattice  $\Lambda$ , such that  $\lambda_2/\lambda_1 \geq \beta$  and there exists a vector  $\mathbf{s}$  such that  $\|\mathbf{A}\mathbf{s}\| = \lambda_1$  with  $\|\mathbf{s}\|_\infty \leq B$  (resp.  $\|\mathbf{s}\| \leq B$ ), to find  $\mathbf{s}$ .*

**Definition 8.** *The  $\text{GapSVP}_{B,\beta}^{\|\cdot\|_\infty}$  (resp.  $\text{GapSVP}_{B,\beta}^{\|\cdot\|}$ ) problem is, given a basis  $\mathbf{A}$  of the lattice  $\Lambda$  to distinguish between  $\lambda_1(\Lambda) \geq \beta$  and if there exists  $\mathbf{s} \neq \mathbf{0}$  such that  $\|\mathbf{s}\|_\infty \leq B$  (resp.  $\|\mathbf{s}\| \leq B$ ) and  $\|\mathbf{A}\mathbf{s}\| \leq 1$ .*

**Definition 9.** *Given two probability distributions  $P$  and  $Q$  on a finite set  $S$ , the Kullback-Leibler (or KL) divergence between  $P$  and  $Q$  is*

$$D_{\text{KL}}(P||Q) = \sum_{x \in S} \ln \left( \frac{P(x)}{Q(x)} \right) P(x) \quad \text{with } \ln(x/0) = +\infty \text{ if } x > 0.$$

The following two lemmata are proven in [33]:

**Lemma 2.** *Let  $P$  and  $Q$  be two distributions over  $S$ , such that for all  $x$ ,  $|P(x) - Q(x)| \leq \delta(x)P(x)$  with  $\delta(x) \leq 1/4$ . Then:*

$$D_{\text{KL}}(P||Q) \leq 2 \sum_{x \in S} \delta(x)^2 P(x).$$

**Lemma 3.** *Let  $A$  be an algorithm which takes as input  $m$  samples of  $S$  and outputs a bit. Let  $x$  (resp.  $y$ ) be the probability that it returns 1 when the input is sampled from  $P$  (resp.  $Q$ ). Then:*

$$|x - y| \leq \sqrt{m D_{\text{KL}}(P||Q)/2}.$$

Finally, we say that an algorithm has a negligible probability of failure if its probability of failure is  $2^{-\Omega(n)}$ .<sup>2</sup>

<sup>2</sup> Some authors use another definition.



## 2.1 Secret-Error Switching

At a small cost in samples, it is possible to reduce any LWE instance to an instance where the secret follows the same distribution as the error [5, 10].

**Theorem 1.** *Given an oracle that solves LWE with  $m$  samples in time  $t$  with the secret coming from the rounded error distribution, it is possible to solve LWE with  $m + \mathcal{O}(n \log \log q)$  samples with the same error distribution (and any distribution on the secret) in time  $t + \mathcal{O}(mn^2 + (n \log \log q)^3)$ , with negligible probability of failure.*

*Furthermore, if  $q$  is prime, we lose  $n + k$  samples with probability of failure bounded by  $q^{-1-k}$ .*

*Proof.* First, select an invertible matrix  $\mathbf{A}$  from the vectorial part of  $\mathcal{O}(n \log \log q)$  samples in time  $\mathcal{O}((n \log \log q)^3)$  [10, Claim 2.13].

Let  $\mathbf{b}$  be the corresponding rounded noisy dot products. Let  $\mathbf{s}$  be the LWE secret and  $\mathbf{e}$  such that  $\mathbf{A}\mathbf{s} + \mathbf{e} = \mathbf{b}$ . Then the subsequent  $m$  samples are transformed in the following way. For each new sample  $(\mathbf{a}', b')$  with  $b' = \langle \mathbf{a}', \mathbf{s} \rangle + e'$ , we give the sample  $(-{}^t\mathbf{A}^{-1}\mathbf{a}', b' - \langle {}^t\mathbf{A}^{-1}\mathbf{a}', \mathbf{b} \rangle)$  to our LWE oracle.

Clearly, the vectorial part of the new samples remains uniform and since

$$b' - \langle {}^t\mathbf{A}^{-1}\mathbf{a}', \mathbf{b} \rangle = \langle -{}^t\mathbf{A}^{-1}\mathbf{a}', \mathbf{b} - \mathbf{A}\mathbf{s} \rangle + b' - \langle \mathbf{a}', \mathbf{s} \rangle = \langle -{}^t\mathbf{A}^{-1}\mathbf{a}', \mathbf{e} \rangle + e'$$

the new errors follow the same distribution as the original, and the new secret is  $\mathbf{e}$ . Hence the oracle outputs  $\mathbf{e}$  in time  $t$ , and we can recover  $\mathbf{s}$  as  $\mathbf{s} = \mathbf{A}^{-1}(\mathbf{b} - \mathbf{e})$ .

If  $q$  is prime, the probability that the  $n + k$  first samples are in some hyperplane is bounded by  $q^{n-1}q^{-n-k} = q^{-1-k}$ .  $\square$

## 2.2 Low Dimension Algorithms

Our main algorithm will return samples from a LWE distribution, while the bias decreases. We describe two fast algorithms when the dimension is small enough.

**Theorem 2.** *If  $n = 0$  and  $m = k/b^2$ , with  $b$  smaller than the real part of the bias, the Decision-LWE problem can be solved with advantage  $1 - 2^{-\Omega(k)}$  in time  $\mathcal{O}(m)$ .*

*Proof.* The algorithm DISTINGUISH computes  $x = \frac{1}{m} \sum_{i=0}^{m-1} \cos(2i\pi b_i/q)$  and returns the boolean  $x \geq b/2$ . If we have a uniform distribution then the average of  $x$  is 0, else it is larger than  $b/2$ . The Hoeffding inequality shows that the probability of  $|x - \mathbb{E}[x]| \geq b/2$  is  $2^{-k/8}$ , which gives the result.  $\square$

**Lemma 4.** *For all  $\mathbf{s} \neq \mathbf{0}$ , if  $\mathbf{a}$  is sampled uniformly,  $\mathbb{E}[\exp(2i\pi \langle \mathbf{a}, \mathbf{s} \rangle / q)] = 0$ .*

*Proof.* Multiplication by  $s_0$  in  $\mathbb{Z}_q$  is a  $\gcd(s_0, q)$ -to-one map because it is a group morphism, therefore  $a_0 s_0$  is uniform over  $\gcd(s_0, q)\mathbb{Z}_q$ . Thus, by using  $k = \gcd(q, s_0, \dots, s_{n-1}) < q$ ,  $\langle \mathbf{a}, \mathbf{s} \rangle$  is distributed uniformly over  $k\mathbb{Z}_q$  so

$$\mathbb{E}[\exp(2i\pi \langle \mathbf{a}, \mathbf{s} \rangle / q)] = \frac{q}{k} \sum_{j=0}^{q/k-1} \exp(2i\pi jk/q) = 0. \quad \square$$

**Algorithm 1.** FindSecret

---

```

function FINDSECRET( $\mathcal{L}$ )
  for all  $(\mathbf{a}, b) \in \mathcal{L}$  do
     $f[\mathbf{a}] \leftarrow f[\mathbf{a}] + \exp(2i\pi b/q)$ 
  end for
   $t \leftarrow \text{FASTFOURIERTRANSFORM}(f)$ 
  return  $\arg \max_{\mathbf{s} \in (\mathbb{Z}/q\mathbb{Z})^n} \Re(t[\mathbf{s}])$ 
end function

```

---

**Theorem 3.** *The algorithm FINDSECRET, when given  $m > (8n \log q + k)/b^2$  samples from a LWE problem with bias whose real part is superior to  $b$  returns the correct secret in time  $\mathcal{O}(m + n \log^2(q)q^n)$  except with probability  $2^{-\Omega(k)}$ .*

*Proof.* The fast Fourier transform needs  $\mathcal{O}(nq^n)$  operations on numbers of bit size  $\mathcal{O}(\log(q))$ . The Hoeffding inequality shows that the difference between  $t[\mathbf{s}']$  and  $\mathbb{E}[\exp(2i\pi(b - \langle \mathbf{a}, \mathbf{s}' \rangle)/q)]$  is at most  $b/2$  except with probability at most  $2 \exp(-mb^2/2)$ . Consequently, it holds for all  $\mathbf{s}'$  except with probability at most  $2q^n \exp(-mb^2/2) = 2^{-\Omega(k)}$  using the union bound. Then  $t[\mathbf{s}] \geq b - b/2 = b/2$  and for all  $\mathbf{s}' \neq \mathbf{s}$ ,  $t[\mathbf{s}'] < b/2$  so the algorithm returns  $\mathbf{s}$ .  $\square$

### 3 Main Algorithm

In this section, we present our main algorithm, prove its asymptotical complexity, and present practical results in dimension  $n = 128$ .

#### 3.1 Rationale

A natural idea in order to distinguish between an instance of LWE (or LPN) and a uniform distribution is to select some  $k$  samples that add up to zero, yielding a new sample of the form  $(\mathbf{0}, e)$ . It is then enough to distinguish between  $e$  and a uniform variable. However, if  $\delta$  is the bias of the error in the original samples, the new error  $e$  has bias  $\delta^k$ , hence roughly  $\delta^{-2k}$  samples are necessary to distinguish it from uniform. Thus it is crucial that  $k$  be as small as possible.

The idea of the algorithm by Blum, Kalai and Wasserman BKW is to perform “blockwise” Gaussian elimination. The  $n$  coordinates are divided into  $k$  blocks of length  $b = n/k$ . Then, samples that are equal on the first  $b$  coordinates are subtracted together to produce new samples that are zero on the first block. This process is iterated over each consecutive block. Eventually samples of the form  $(\mathbf{0}, e)$  are obtained.

Each of these samples ultimately results from the addition of  $2^k$  starting samples, so  $k$  should be at most  $\mathcal{O}(\log(n))$  for the algorithm to make sense. On the other hand  $\Omega(q^b)$  data are clearly required at each step in order to generate enough collisions on  $b$  consecutive coordinates of a block. This naturally results in a complexity roughly  $2^{(1+o(1))n/\log(n)}$  in the original algorithm for LPN. This algorithm was later adapted to LWE in [3], and then improved in [4].

The idea of the latter improvement is to use so-called “lazy modulus switching”. Instead of finding two vectors that are equal on a given block in order to generate a new vector that is zero on the block, one uses vectors that are merely close to each other. This may be seen as performing addition modulo  $p$  instead of  $q$  for some  $p < q$ , by rounding every value  $x \in \mathbb{Z}_q$  to the value nearest  $xp/q$  in  $\mathbb{Z}_p$ . Thus at each step of the algorithm, instead of generating vectors that are zero on each block, small vectors are produced. This introduces a new “rounding” error term, but essentially reduces the complexity from roughly  $q^b$  to  $p^b$ . Balancing the new error term with this decrease in complexity results in a significant improvement.

However it may be observed that this rounding error is much more costly for the first few blocks than the last ones. Indeed samples produced after, say, one iteration step are bound to be added together  $2^{a-1}$  times to yield the final samples, resulting in a corresponding blowup of the rounding error. By contrast, later terms will undergo less additions. Thus it makes sense to allow for progressively coarser approximations (i.e. decreasing the modulus) at each step. On the other hand, to maintain comparable data requirements to find collisions on each block, the decrease in modulus is compensated by progressively longer blocks.

What we propose here is a more general view of the BKW algorithm that allows for this improvement, while giving a clear view of the different complexity costs incurred by various choice of parameters. Balancing these terms is the key to finding an optimal complexity. We forego the “modulus switching” point of view entirely, while retaining its core ideas. The resulting algorithm generalizes several variants of BKW, and will be later applied in a variety of settings.

### 3.2 Quantization

The goal of quantization is to associate to each point of  $\mathbb{R}^k$  a center from a *small* set, such that the expectancy of the distance between a point and its center is small. We will then be able to produce small vectors by subtracting vectors associated to the same center.

Modulus switching amounts to a simple quantizer which rounds every coordinate to the nearest multiple of some constant. Our proven algorithm uses a similar quantizer, except the constant depends on the index of the coordinate.

It is possible to decrease the average distance from a point to its center by a constant factor for large moduli [17], but doing so would complicate our proof without improving the leading term of the complexity. When the modulus is small, it might be worthwhile to use error-correcting codes as in [18].

### 3.3 Main Algorithm

Let us denote by  $\mathcal{L}_0$  the set of starting samples, and  $\mathcal{L}_i$  the sample list after  $i$  reduction steps. The numbers  $d_0 = 0 \leq d_1 \leq \dots \leq d_k = n$  partition the  $n$  coordinates of sample vectors into  $k$  buckets. Let  $\mathbf{D} = (D_0, \dots, D_{k-1})$  be the vector of quantization coefficients associated to each bucket.

**Algorithm 2.** Main resolution

---

```

1: function REDUCE( $\mathcal{L}_{in}, D_i, d_i, d_{i+1}$ )
2:    $\mathcal{L}_{out} \leftarrow \emptyset$ 
3:    $t[] \leftarrow \emptyset$ 
4:   for all  $(\mathbf{a}, b) \in \mathcal{L}_{in}$  do
5:      $\mathbf{r} = \lfloor \frac{(a_{d_i}, \dots, a_{d_{i+1}-1})}{D} \rfloor$ 
6:     if  $t[\mathbf{r}] = \emptyset$  then
7:        $t[\mathbf{r}] \leftarrow (\mathbf{a}, b)$ 
8:     else
9:        $\mathcal{L}_{out} \leftarrow \mathcal{L}_{out} \cup \{(\mathbf{a}, b) - t[\mathbf{r}]\}$ 
10:       $t[\mathbf{r}] \leftarrow \emptyset$ 
11:    end if
12:  end for
13:  return  $\mathcal{L}_{out}$ 
14: end function
15: function SOLVE( $\mathcal{L}_0, \mathbf{D}, (d_i)$ )
16:   for  $0 \leq i < k$  do
17:      $\mathcal{L}_{i+1} \leftarrow \text{REDUCE}(\mathcal{L}_i, D_i, d_i, d_{i+1})$ 
18:   end for
19:   return  $\text{DISTINGUISH}\{b | (\mathbf{a}, b) \in \mathcal{L}_k\}$ 
20: end function

```

---

In order to allow for a uniform presentation of the BKW algorithm, applicable to different settings, we do not assume a specific distribution on the secret. Instead, we assume there exists some *known*  $\mathbf{B} = (B_0, \dots, B_{n-1})$  such that  $\sum_i (s_i/B_i)^2 \leq n$ . Note that this is in particular true if  $|s_i| \leq B_i$ . We shall see how to adapt this to the standard Gaussian case later on. Without loss of generality,  $\mathbf{B}$  is non increasing.

There are  $a$  phases in our reduction: in the  $i$ -th phase, the coordinates from  $d_i$  to  $d_{i+1}$  are reduced. We define  $m = |\mathcal{L}_0|$ .

**Lemma 5.** SOLVE terminates in time  $\mathcal{O}(mn \log q)$ .

*Proof.* The REDUCE algorithm clearly runs in time  $\mathcal{O}(|\mathcal{L}|n \log q)$ . Moreover,  $|\mathcal{L}_{i+1}| \leq |\mathcal{L}_i|/2$  so that the total running time of SOLVE is  $\mathcal{O}(n \log q \sum_{i=0}^k m/2^i) = \mathcal{O}(mn \log q)$ .  $\square$

**Lemma 6.** Write  $\mathcal{L}'_i$  for the samples of  $\mathcal{L}_i$  where the first  $d_i$  coordinates of each sample vector have been truncated. Assume  $|s_j|D_i < 0.23q$  for all  $d_i \leq j < d_{i+1}$ . If  $\mathcal{L}'_i$  is sampled according to the LWE distribution of secret  $\mathbf{s}$  and noise parameters  $\alpha$  and  $\epsilon \leq 1$ , then  $\mathcal{L}'_{i+1}$  is sampled according to the LWE distribution of the truncated secret with parameters:

$$\alpha'^2 = 2\alpha^2 + 4\pi^2 \sum_{j=d_i}^{d_{i+1}-1} (s_j D_i / q)^2 \quad \text{and} \quad \epsilon' = 3\epsilon.$$

On the other hand, if  $D_i = 1$ , then  $\alpha'^2 = 2\alpha^2$ .

*Proof.* The independence of the outputted samples and the uniformity of their vectorial part are clear. Let  $(\mathbf{a}, b)$  be a sample obtained by subtracting two samples from  $\mathcal{L}_i$ . For  $\mathbf{a}'$  the vectorial part of a sample, define  $\epsilon(\mathbf{a}')$  such that  $\mathbb{E}[\exp(2i\pi(\langle \mathbf{a}', \mathbf{s} \rangle - b)/q) | \mathbf{a}'] = (1 + \epsilon(\mathbf{a}')) \exp(-\alpha^2)$ . By definition of LWE,  $|\epsilon(\mathbf{a}')| \leq \epsilon$ , and by independence:

$$\mathbb{E}[\exp(2i\pi(\langle \mathbf{a}, \mathbf{s} \rangle - b)/q) | \mathbf{a}] = \exp(-2\alpha^2) \mathbb{E}_{\mathbf{a}' - \mathbf{a}'' = \mathbf{a}}[(1 + \epsilon(\mathbf{a}'))(1 + \epsilon(\mathbf{a}''))],$$

with  $|\mathbb{E}_{\mathbf{a}' - \mathbf{a}'' = \mathbf{a}}[(1 + \epsilon(\mathbf{a}'))(1 + \epsilon(\mathbf{a}''))] - 1| \leq 3\epsilon$ .

Thus we computed the noise corresponding to adding two samples of  $\mathcal{L}_i$ . To get the noise for a sample from  $\mathcal{L}_{i+1}$ , it remains to truncate coordinates from  $d_i$  to  $d_{i+1}$ . A straightforward induction on the coordinates shows that this noise is:

$$\exp(-2\alpha^2) \mathbb{E}_{\mathbf{a}' - \mathbf{a}'' = \mathbf{a}}[(1 + \epsilon(\mathbf{a}'))(1 + \epsilon(\mathbf{a}''))] \prod_{j=d_i}^{d_{i+1}-1} \mathbb{E}[\exp(2i\pi \mathbf{a}_j \mathbf{s}_j / q)].$$

Indeed, if we denote by  $\mathbf{a}^{(j)}$  the vector  $\mathbf{a}$  where the first  $j$  coordinates are truncated and  $\alpha_j$  the noise parameter of  $\mathbf{a}^{(j)}$ , we have:

$$\begin{aligned} & |\mathbb{E}[\exp(2i\pi(\langle \mathbf{a}^{(j+1)}, \mathbf{s}^{(j+1)} \rangle - b)/q) | \mathbf{a}^{(j+1)}] - \exp(-\alpha_n^2) \mathbb{E}[\exp(2i\pi \mathbf{a}_j \mathbf{s}_j / q)]| \\ &= |\mathbb{E}[\exp(-2i\pi \mathbf{a}_j \mathbf{s}_j / q) (\exp(2i\pi(\langle \mathbf{a}^{(j)}, \mathbf{s}^{(j)} \rangle - b)/q) - \exp(-\alpha_j^2))]| \\ &\leq \epsilon' \exp(-\alpha_j^2) \mathbb{E}[\exp(2i\pi \mathbf{a}_j \mathbf{s}_j / q)]. \end{aligned}$$

It remains to compute  $\mathbb{E}[\exp(2i\pi \mathbf{a}_j \mathbf{s}_j / q)]$  for  $d_i \leq j < d_{i+1}$ . Let  $D = D_i$ . The distribution of  $\mathbf{a}_j$  is even, so  $\mathbb{E}[\exp(2i\pi \mathbf{a}_j \mathbf{s}_j / q)]$  is real. Furthermore, since  $|\mathbf{a}_j| \leq D$ ,

$$\mathbb{E}[\exp(2i\pi \mathbf{a}_j \mathbf{s}_j / q)] \geq \cos(2\pi \mathbf{s}_j D / q).$$

Assuming  $|\mathbf{s}_j| D < 0.23q$ , simple function analysis shows that

$$\mathbb{E}[\exp(2i\pi \mathbf{a}_j \mathbf{s}_j / q)] \geq \exp(-4\pi^2 \mathbf{s}_j^2 D^2 / q^2).$$

On the other hand, if  $D_i = 1$  then  $\mathbf{a}_j = 0$  and  $\mathbb{E}[\exp(2i\pi \mathbf{a}_j \mathbf{s}_j / q)] = 1$ . □

Finding optimal parameters for BKW amounts to balancing various costs: the baseline number of samples required so that the final list  $\mathcal{L}_k$  is non-empty, and the additional factor due to the need to distinguish the final error bias. This final bias itself comes both from the blowup of the original error bias by the BKW additions, and the ‘‘rounding errors’’ due to quantization. Balancing these costs essentially means solving a system.

For this purpose, it is convenient to set the overall target complexity as  $2^{n(x+o(1))}$  for some  $x$  to be determined. The following auxiliary lemma essentially gives optimal values for the parameters of SOLVE assuming a suitable value of  $x$ . The actual value of  $x$  will be decided later on.

**Lemma 7.** *Pick some value  $x$  (dependent on LWE parameters). Choose:*

$$k \leq \left\lfloor \log \left( \frac{nx}{6\alpha^2} \right) \right\rfloor \quad m = n2^k 2^{nx}$$

$$D_i \leq \frac{q\sqrt{x/6}}{\pi B_{d_i} 2^{(a-i+1)/2}} \quad d_{i+1} = \min \left( d_i + \left\lfloor \frac{nx}{\log(1+q/D_i)} \right\rfloor, n \right).$$

Assume  $d_k = n$  and  $\epsilon \leq 1/(\beta^2 x)^{\log 3}$ , and for all  $i$  and  $d_i \leq j < d_{i+1}$ ,  $|s_j| D_i < 0.23q$ . SOLVE runs in time  $\mathcal{O}(mn)$  with negligible failure probability.

*Proof.* Remark that for all  $i$ ,

$$|\mathcal{L}_{i+1}| \geq (|\mathcal{L}_i| - (1+q/D_i)^{d_{i+1}-d_i})/2 \geq (|\mathcal{L}_i| - 2^{nx})/2.$$

Using induction, we then have  $|\mathcal{L}_i| \geq (|\mathcal{L}_0| + 2^{nx})/2^i - 2^{nx}$  so that  $|\mathcal{L}_k| \geq n2^{nx}$ .

By induction and using the previous lemma, the input of DISTINGUISH is sampled from a LWE distribution with noise parameter:

$$\alpha'^2 = 2^k \alpha^2 + 4\pi^2 \sum_{i=0}^{k-1} 2^{k-i-1} \sum_{j=d_i}^{d_{i+1}-1} (s_j D_i / q)^2.$$

By choice of  $k$  the first term is smaller than  $nx/6$ . As for the second term, since  $B$  is non increasing and by choice of  $D_i$ , it is smaller than:

$$4\pi^2 \sum_{i=0}^{k-1} 2^{k-i-1} \frac{x/6}{\pi^2 2^{k-i+1}} \sum_{j=d_i}^{d_{i+1}-1} \left( \frac{s_j}{B_j} \right)^2 \leq (x/6) \sum_{j=0}^{n-1} \left( \frac{s_j}{B_j} \right)^2 \leq nx/6.$$

Thus the real part of the bias is superior to  $\exp(-nx/3)(1-3^a\epsilon) \geq 2^{-nx/2}$ , and hence by Theorem 2.2, DISTINGUISH fails with negligible probability.  $\square$

**Theorem 4.** *Assume that for all  $i$ ,  $|s_i| \leq B$ ,  $B \geq 2$ ,  $\max(\beta, \log(q)) = 2^{o(n/\log n)}$ ,  $\beta = \omega(1)$ , and  $\epsilon \leq 1/\beta^4$ . Then SOLVE takes time  $2^{(n/2+o(n))/\ln(1+\log \beta/\log B)}$ .*

*Proof.* We apply Lemma 7, choosing

$$k = \lfloor \log(\beta^2/(12 \ln(1 + \log \beta))) \rfloor = (2 - o(1)) \log \beta \in \omega(1)$$

and we set  $D_i = q/(Bk2^{(k-i)/2})$ . It now remains to show that this choice of parameters satisfies the conditions of the lemma.

First, observe that  $BD_i/q \leq 1/k = o(1)$  so the condition  $|s_j| D_i < 0.23q$  is fulfilled. Then,  $d_k \geq n$ , which amounts to:

$$\sum_{i=0}^{k-1} \frac{x}{(k-i)/2 + \log \mathcal{O}(kB)} \geq 2x \ln(1 + k/2/\log \mathcal{O}(kB)) \geq 1 + k/n = 1 + o(1)$$

If we have  $\log k = \omega(\log \log B)$  (so in particular  $k = \omega(\log B)$ ), we get  $\ln(1 + k/2/\log \mathcal{O}(kB)) = (1 + o(1)) \ln(k) = (1 + o(1)) \ln(1 + \log \beta/\log B)$ .

Else,  $\log k = \mathcal{O}(\log \log B) = o(\log B)$  (since necessarily  $B = \omega(1)$  in this case), so we get  $\ln(1 + k/2/\log \mathcal{O}(kB)) = (1 + o(1)) \ln(1 + \log \beta/\log B)$ .

Thus our choice of  $x$  fits both cases and we have  $1/x \leq 2 \ln(1 + \log \beta)$ . Second, we have  $1/k = o(\sqrt{x})$  so  $D_i$ ,  $\epsilon$  and  $k$  are also sufficiently small and the lemma applies. Finally, note that the algorithm has complexity  $2^{\Omega(n/\log n)}$ , so a factor  $n2^k \log(q)$  is negligible.  $\square$

This theorem can be improved when the use of the given parameters yields  $D < 1$ , since  $D = 1$  already gives a lossless quantization.

**Theorem 5.** *Assume that for all  $i$ ,  $|s_i| \leq B = n^{b+o(1)}$ . Let  $\beta = n^c$  and  $q = n^d$  with  $d \geq b$  and  $c + b \geq d$ . Assume  $\epsilon \leq 1/\beta^4$ . Then SOLVE takes time  $2^{n/(2(c-d+b)/d+2\ln(d/b)-o(1))}$ .*

*Proof.* Once again we aim to apply Lemma 7, and choose  $k$  as above:

$$k = \log(\beta^2/(12 \ln(1 + \log \beta))) = (2c - o(1)) \log n$$

If  $i < \lceil 2(c - d + b) \log n \rceil$ , we take  $D_i = 1$ , else we choose  $q/D_i = \Theta(B2^{(a-i)/2})$ . Satisfying  $d_a \geq n - 1$  amounts to:

$$\begin{aligned} & 2x(c - d + b) \log n / \log q + \sum_{i=\lceil 2(c-d+b) \log n \rceil}^{a-1} \frac{x}{(a-i)/2 + \log \mathcal{O}(B)} \\ & \geq 2x(c - d + b)/d + 2x \ln((a - 2(c - d + b) \log n + 2 \log B)/2 / \log \mathcal{O}(B)) \\ & \geq 1 + a/n = 1 + o(1) \end{aligned}$$

So that we can choose  $1/x = 2(c - d + b)/d + 2 \ln(d/b) - o(1)$ .  $\square$

**Corollary 1.** *Given a LWE problem with  $q = n^d$ , Gaussian errors with  $\beta = n^c$ ,  $c > 1/2$  and  $\epsilon \leq n^{-4c}$ , we can find a solution in  $2^{n/(1/d+2\ln(d/(1/2+d-c))-o(1))}$  time.*

*Proof.* Apply Theorem 1: with probability  $2/3$ , the secret is now bounded by  $B = \mathcal{O}(q\sqrt{n}/\beta\sqrt{\log n})$ . The previous theorem gives the complexity of an algorithm discovering the secret, using  $b = 1/2 - c + d$ , and which works with probability  $2/3 - 2^{-\Omega(n)}$ . Repeating  $n$  times with different samples, the correct secret will be outputted at least  $n/2 + 1$  times, except with negligible probability. By returning the most frequent secret, the probability of failure is negligible.  $\square$

In particular, if  $c \leq d$ , it is possible to quantumly approximate lattice problems within factor  $\mathcal{O}(n^{c+1/2})$  [34]. Setting  $c = d$ , the complexity is  $2^{n/(1/c+2\ln(2c)-o(1))}$ , so that the constant slowly converges to 0 when  $c$  goes to infinity.

A simple BKW using the bias would have a complexity of  $2^{d/cn+o(n)}$ , the analysis of [4] or [3] only conjectures  $2^{dn/(c-1/2)+o(n)}$  for  $c > 1/2$ . In [4], the authors incorrectly claim a complexity of  $2^{cn+o(n)}$  when  $c = d$ , because the blowup in the error is not explicitly computed.

Finally, if we want to solve the LWE problem for different secrets but with the same vectorial part of the samples, it is possible to be much faster if we work with a bigger final bias, since the REDUCE part needs to be called only once.

### 3.4 Experimentation

We have implemented our algorithm, in order to test its efficiency in practice, as well as that of the practical improvements in the appendix of the full version. We have chosen dimension  $n = 128$ , modulus  $q = n^2$ , binary secret, and Gaussian errors with noise parameter  $\alpha = 1/(\sqrt{n/\pi} \log^2 n)$ . The previous best result for these parameters, using a BKW algorithm with lazy modulus switching, claims a time complexity of  $2^{74}$  with  $2^{60}$  samples [4].

Using our improved algorithm, we were able to recover the secret using  $m = 2^{28}$  samples within 13 hours on a single PC equipped with a 16-core Intel Xeon. The computation time proved to be devoted mostly to the computation of  $9 \cdot 10^{13}$  norms, computed in fixed point over 16 bits in SIMD.

In appendix of the full version, we compare the different techniques to solve the LWE problem when the number of samples is large or small. We were able to solve the same problem using BKZ with block size 40 followed by an enumeration in two minutes.

## 4 Applications to Lattice Problems

We first show that  $\text{BDD}_{B,\beta}^{\|\cdot\|_\infty}$  is easier than  $\text{LWE}_{B,\beta}$  for some large enough modulus and then that  $\text{UniqueSVP}_{B,\beta}^{\|\cdot\|_\infty}$  and  $\text{GapSVP}_{B,\beta}^{\|\cdot\|_\infty}$  are easier than  $\text{BDD}_{B,\beta}^{\|\cdot\|_\infty}$ . In appendix of the full version, we prove the same result for  $\text{BDD}_{B,\beta}^{\|\cdot\|}$ .

### 4.1 Variant of Bounding Distance Decoding

The main result of this subsection is close to the classic reduction of [34]. However, our definition of LWE allows to simplify the proof, and gain a constant factor in the decoding radius. The use of the KL divergence instead of the statistical distance also allows to gain a constant factor, when we need an exponential number of samples, or when  $\lambda_n^*$  is really small.

The core of the reduction lies in Lemma 8, assuming access to a Gaussian sampling oracle. This hypothesis will be taken care of in Lemma 9.

**Lemma 8.** *Let  $\mathbf{A}$  be a basis of the lattice  $\Lambda$  of full rank  $n$ . Assume we are given access to an oracle outputting a vector sampled under the law  $D_{\Lambda^*,\sigma}$  and  $\sigma \geq q\eta_\epsilon(\Lambda^*)$ , and to an oracle solving the LWE problem in dimension  $n$ , modulus  $q \geq 2$ , noise parameter  $\alpha$ , and distortion parameter  $\xi$  which fails with negligible probability and use  $m$  vectors if the secret  $\mathbf{s}$  verifies  $|s_i| \leq B_i$ .*

*Then, if we are given a point  $\mathbf{x}$  such that there exists  $\mathbf{s}$  with  $\mathbf{v} = \mathbf{A}\mathbf{s} - \mathbf{x}$ ,  $\|\mathbf{v}\| \leq \sqrt{1/\pi}\alpha q/\sigma$ ,  $|s_i| \leq B_i$  and  $\rho_{\sigma/q}(\Lambda \setminus \{\mathbf{0}\} + \mathbf{v}) \leq \xi \exp(-\alpha^2)/2$ , we are able to find  $\mathbf{s}$  in at most  $mn$  calls to the Gaussian sampling oracle,  $n$  calls to the LWE solving oracle, with a probability of failure  $n\sqrt{m}\epsilon + 2^{-\Omega(n)}$  and complexity  $\mathcal{O}(mn^3 + n^c)$  for some  $c$ .*



In the previous lemma, we required access to a  $D_{\Lambda^*, \sigma}$  oracle. However, for large enough  $\sigma$ , this hypothesis comes for free, as shown by the following lemma, which we borrow from [10].

**Lemma 9.** *If we have a basis  $\mathbf{A}$  of the lattice  $\Lambda$ , then for  $\sigma \geq \mathcal{O}(\sqrt{\log n} \|\tilde{\mathbf{A}}\|)$ , it is possible to sample in polynomial time from  $D_{\Lambda, \sigma}$ .*

We will also need the following lemma, due to Banaszczyk [7]. For completeness, a proof is provided in the appendix of the full version.

**Lemma 10.** *For a lattice  $\Lambda$ ,  $\mathbf{c} \in \mathbb{R}^n$ , and  $t \geq 1$ ,*

$$\frac{\rho((\Lambda + \mathbf{c}) \setminus \mathcal{B}(0, t\sqrt{\frac{n}{2\pi}}))}{\rho(\Lambda)} \leq \exp(-n(t^2 - 2\ln t - 1)/2) \leq \exp(-n(t-1)^2/2).$$

**Theorem 6.** *Assume we have a LWE solving oracle of modulus  $q \geq 2^n$ , parameters  $\beta$  and  $\xi$  which needs  $m$  samples.*

*If we have a basis  $\mathbf{A}$  of the lattice  $\Lambda$ , and a point  $\mathbf{x}$  such that  $\mathbf{A}\mathbf{s} - \mathbf{x} = \mathbf{v}$  with  $\|\mathbf{v}\| \leq (1-1/n)\lambda_1/\beta/t < \lambda_1/2$  and  $4\exp(-n(t-1/\beta-1)^2/2) \leq \xi \exp(-n/2/\beta^2)$ , then with  $n^2$  calls to the LWE solving oracle with secret  $\mathbf{s}$ , we can find  $\mathbf{s}$  with probability of failure  $2\sqrt{m}\exp(-n(t^2 - 2\ln t - 1)/2)$  for any  $t \geq 1 + 1/\beta$ .*

*Proof.* Using Lemma 10, we can prove that  $\sigma = t\sqrt{n/2/\pi}/\lambda_1 \leq \eta_\epsilon(\Lambda^*)$  for  $\epsilon = 2\exp(-n(t^2 - 2\ln t - 1)/2)$  and

$$\rho_{1/\sigma}(\Lambda \setminus \{\mathbf{0}\} + \mathbf{v}) \leq 2\exp(-n(t(1-1/\beta/t) - 1)^2/2).$$

Using LLL, we can find a basis  $\mathbf{B}$  of  $\Lambda$  so that  $\|\tilde{\mathbf{B}}^*\| \leq 2^{n/2}/\lambda_1$ , and therefore, it is possible to sample in polynomial time from  $D_{\Lambda, q\sigma}$  since  $q \geq 2^n$  for sufficiently large  $n$ .

The LLL algorithm also gives a non zero lattice vector of norm  $\ell \leq 2^n \lambda_1$ . For  $i$  from 0 to  $n^2$ , we let  $\lambda = \ell(1-1/n)^i$ , we use the algorithm of Lemma 8 with standard deviation  $tq\sqrt{n/2/\pi}/\lambda$ , which uses only one call to the LWE solving oracle, and return the closest lattice vector of  $\mathbf{x}$  in all calls.

Since  $\ell(1-1/n)^{n^2} \leq 2^n \exp(-n)\lambda_1 \leq \lambda_1$ , with  $0 \leq i \leq n^2$  be the smallest integer such that  $\lambda = \ell(1-1/n)^i \leq \lambda_1$ , we have  $\lambda \geq (1-1/n)\lambda_1$ . Then the lemma applies since

$$\|\mathbf{v}\| \leq (1-1/n)\lambda_1/\beta/t \leq \sqrt{1/\pi}\sqrt{n/2}/\beta q/(tq\sqrt{n/2/\pi}/\lambda) = \lambda/t/\beta.$$

Finally, the distance bound makes  $\mathbf{A}\mathbf{s}$  the unique closest lattice point of  $\mathbf{x}$ .  $\square$

Using self-reduction, it is possible to remove the  $1-1/n$  factor [27].

**Corollary 2.** *It is possible to solve  $\text{BDD}_{B, \beta}^{\|\cdot\|_\infty}$  in time  $2^{(n/2+o(n))/\ln(1+\log \beta/\log B)}$  if  $\beta = \omega(1)$ ,  $\beta = 2^{o(n/\log n)}$  and  $\log B = \mathcal{O}(\log \beta)$ .*

*Proof.* Apply the previous theorem and Theorem 4 with some sufficiently large constant for  $t$ , and remark that dividing  $\beta$  by some constant does not change the complexity.  $\square$

Note that since we can solve LWE for many secrets in essentially the same time than for one, we have the same property for BDD.

## 4.2 UniqueSVP and GapSVP

In this section, we show how  $\text{GapSVP}_{B,\beta}^{\|\cdot\|_\infty}$  and  $\text{UniqueSVP}_{B,\beta}^{\|\cdot\|_\infty}$  can be reduced to  $\text{BDD}_{B,\beta}^{\|\cdot\|_\infty}$ , and hence to LWE. Proofs are provided in the appendix of the full version.

**Theorem 7.** *Given a  $\text{BDD}_{B,\beta}^{\|\cdot\|_\infty}$  oracle, it is possible to solve  $\text{UniqueSVP}_{B,\beta}^{\|\cdot\|_\infty}$  in polynomial time of  $n$  and  $\beta$ .*

**Theorem 8.** *We can solve any  $\text{GapSVP}_{\substack{o(B\sqrt{\log \log \log \beta / \log \log \beta}), \beta}}^{\|\cdot\|_\infty}$  instances in time  $2^{(n/2+o(n))/\ln(1+\log \beta / \log B)}$  for  $\beta = 2^{o(n/\log n)}$ ,  $\beta = \omega(1)$ ,  $B \geq 2$ .*

**Corollary 3.** *It is possible to solve any  $\text{GapSVP}_{2^{\sqrt{\log n}}, n^c}^{\|\cdot\|_\infty}$  with  $c > 0$  in time  $2^{(n+o(n))/\ln \ln n}$ .*

*Proof.* Use Theorem 8 with  $B = 2^{\sqrt{\log n}} \log \log n$  and  $\beta = n^c$ . □

**Theorem 9.** *If it is possible to solve  $\text{BDD}_{B,\beta}^{\|\cdot\|_\infty}$  in polynomial time, then it is possible to solve in randomized polynomial time  $\text{GapSVP}_{B/\sqrt{n}, \beta\sqrt{n/\log n}}^{\|\cdot\|_\infty}$ .*

## 5 Other Applications

### 5.1 Low Density Subset-Sum Problem

**Definition 10.** *We are given a vector  $\mathbf{a} \in \mathbb{Z}^n$  whose coordinates are sampled independently and uniformly in  $[0; M)$ , and  $\langle \mathbf{a}, \mathbf{s} \rangle$  where the coordinates of  $\mathbf{s}$  are sampled independently and uniformly in  $\{0, 1\}$ . The goal is to find  $\mathbf{s}$ . The density is defined as  $d = \frac{n}{\log M}$ .*

Note that this problem is trivially equivalent to the *modular* subset-sum problem, where we are given  $\langle \mathbf{a}, \mathbf{s} \rangle \bmod M$  by trying all possible  $\lfloor \langle \mathbf{a}, \mathbf{s} \rangle / M \rfloor$ .

In [13, 22], Lagarias *et al.* reduce the subset sum problem to UniqueSVP, even though this problem was not defined at that time. We will show a reduction to  $\text{BDD}_{1, \Omega(2^{1/d})}^{\|\cdot\|_\infty}$ , which is essentially the same. First, we need two geometric lemmata.

**Lemma 11.** *Let  $\mathcal{B}_n(r)$ , the number of points of  $\mathbb{Z}^n$  of norm smaller than  $r$ , and  $V_n$  the volume of the unit ball. Then,*

$$\mathcal{B}_n(r) \leq V_n \left( r + \frac{\sqrt{n}}{2} \right)^n.$$

*Proof.* For each  $\mathbf{x} \in \mathbb{Z}^n$ , let  $E_{\mathbf{x}}$  be a cube of length 1 centered on  $\mathbf{x}$ . Let  $E$  be the union of all the  $E_{\mathbf{x}}$  which have a non empty intersection with the ball of center  $\mathbf{0}$  and radius  $r$ . Therefore  $\text{vol}(E) \geq \mathcal{B}_n(r)$  and since  $E$  is included in the ball of center  $\mathbf{0}$  and radius  $r + \frac{\sqrt{n}}{2}$ , the claim is proven. □

**Lemma 12.** *For  $n \geq 4$  we have*

$$V_n = \frac{\pi^{n/2}}{(n/2)!} \leq (\sqrt{\pi e/n})^n.$$

**Theorem 10.** *Using one call to a  $\text{BDD}_{1,c2^{1/d}}^{\|\cdot\|_\infty}$  oracle with any  $c < \sqrt{2/\pi/e}$  and  $d = o(1)$ , and polynomial time, it is possible to solve a subset-sum problem of density  $d$ , with negligible probability of failure.*

*Proof.* With the matrix:

$$\mathbf{A} = \begin{pmatrix} \mathbf{I} \\ C\mathbf{a} \end{pmatrix}$$

for some  $C > c2^{1/d}\sqrt{n}/2$  and  $\mathbf{b} = (1/2, \dots, 1/2, C\langle \mathbf{a}, \mathbf{s} \rangle)$ , return  $\text{BDD}(\mathbf{A}, \mathbf{b})$ . It is clear that  $\|\mathbf{A}\mathbf{s} - \mathbf{b}\| = \sqrt{n}/2$ . Now, let  $\mathbf{x}$  such that  $\|\mathbf{A}\mathbf{x}\| = \lambda_1$ . If  $\langle \mathbf{a}, \mathbf{x} \rangle \neq 0$ , then  $\lambda_1 = \|\mathbf{A}\mathbf{x}\| \geq C$  therefore  $\beta \geq c2^{1/d}$ . Else,  $\langle \mathbf{a}, \mathbf{x} \rangle = 0$ . Without loss of generality,  $x_0 \neq 0$ , we let  $y = -(\sum_{i>0} a_i x_i)/x_0$  and the probability over  $\mathbf{a}$  that  $\langle \mathbf{a}, \mathbf{x} \rangle = 0$  is:

$$\Pr[\langle \mathbf{a}, \mathbf{x} \rangle = 0] = \Pr[a_0 = y] = \sum_{z=0}^{M-1} \Pr[y = z] \Pr[a_0 = z] \leq \frac{1}{M}.$$

Therefore, the probability of failure is at most, for sufficiently large  $n$ ,

$$\begin{aligned} \mathcal{B}_n(\beta\sqrt{n}/2)/M &\leq (\sqrt{\pi e/n})^n (c2^{1/d}\sqrt{n}/2 + \sqrt{n}/2)^n / 2^{n/d} \\ &= (\sqrt{\pi e/2}(c + 2^{-1/d}))^n = 2^{-\Omega(n)}. \end{aligned} \quad \square$$

**Corollary 4.** *For any  $d = o(1)$  and  $d = \omega(\log n/n)$ , we can solve the subset-sum problem of density  $d$  with negligible probability of failure in time  $2^{(n/2+o(n))/\ln(1/d)}$ .*

The cryptosystem of Lyubashevsky *et al.* [28] uses  $2^{1/d} > 10n \log^2 n$  and is therefore broken in time  $2^{(\ln 2/2+o(1))n/\log \log n}$ . Current lattice reduction algorithms are slower than this one when  $d = \omega(1/(\log n \log \log n))$ .

## 5.2 Sample Expander and Application to LWE with Binary Errors

**Definition 11.** *Let  $q$  be a prime number. The problem Small-DecisionLWE is to distinguish  $(\mathbf{A}, \mathbf{b})$  with  $\mathbf{A}$  sampled uniformly with  $n$  columns and  $m$  rows,  $\mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e}$  such that  $\|\mathbf{s}\|^2 + \|\mathbf{e}\|^2 \leq nk^2$  and  $\|\mathbf{s}\|_\infty \leq B$  from  $(\mathbf{A}, \mathbf{b})$  sampled uniformly. Also, the distribution  $(\mathbf{s}, \mathbf{e})$  is efficiently samplable.*

*The problem Small-SearchLWE is to find  $\mathbf{s}$  given  $(\mathbf{A}, \mathbf{b})$  with  $\mathbf{A}$  sampled uniformly and  $\mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e}$  with the same conditions on  $\mathbf{s}$  and  $\mathbf{e}$ .*

These problems are generalizations of BinaryLWE where  $\mathbf{s}$  and  $\mathbf{e}$  have coordinates sampled uniformly in  $\{0, 1\}$ . In this case, remark that each sample is a root of a known quadratic polynomial in the coordinates of  $\mathbf{s}$ . Therefore, it is easy

to solve this problem when  $m \geq n^2$ . For  $m = \mathcal{O}(n)$ , a Gröbner basis algorithm applied on this system will (heuristically) have a complexity of  $2^{\Omega(n)}$  [2]. For  $m = \mathcal{O}(n/\log n)$  and  $q = n^{\mathcal{O}(1)}$ , it has been shown to be harder than a lattice problem in dimension  $\Theta(n/\log n)$  [30].

In appendix of the full version, we prove the following theorem<sup>3</sup>, with the coordinates of  $\mathbf{x}$  and  $\mathbf{y}$  distributed according to a samplable  $\mathcal{D}$ :

**Theorem 11.** *Assume there is an efficient distinguisher which uses  $k$  samples for Decision-LWE (respectively a solver for Search-LWE) with error distribution  $\langle \mathbf{s}, \mathbf{y} \rangle + \langle \mathbf{e}, \mathbf{x} \rangle$  of advantage (resp. success probability)  $\epsilon$ .*

*Then, either there is an efficient distinguisher for Decision-LWE with samples and secret taken uniformly, and error distribution  $\mathcal{D}$  in dimension  $m-1$  and with  $n+m$  samples of advantage  $\frac{\xi}{4qk} - q^{-n} - q^{-m}$ ; or there is an efficient distinguisher of advantage  $\epsilon - \xi$  for Small-Decision-LWE (resp. solver of success probability  $\epsilon - \xi$  for Small-Search-LWE).*

**Lemma 13.** *Let  $\mathcal{D} = D_{\mathbb{Z}, \sigma}$  for  $\sigma \geq 1$ . Then, the advantage of a distinguisher for Decision-LWE of dimension  $m$  with  $m+n$  samples of noise distribution  $\mathcal{D}$  is at most  $\sqrt{q^n/\sigma^{n+m}}$ . Furthermore, the bias of  $\langle (\mathbf{s}|\mathbf{e}), (\mathbf{x}|\mathbf{y}) \rangle$ , for fixed  $\mathbf{s}$  and  $\mathbf{e}$ , is at least  $\exp(-\pi(\|\mathbf{s}\|^2 + \|\mathbf{e}\|^2)\sigma^2/q^2)$ .*

*Proof.* We have  $\mathcal{D}^{m+n}(\mathbf{a}) \leq \mathcal{D}(0)^{m+n} = 1/\rho_\sigma(\mathbb{Z})^{m+n}$  and  $\rho_\sigma(\mathbb{Z}) = \sigma\rho_{1/\sigma}(\mathbb{Z}) \geq \sigma$  using a Poisson summation. The first property is then a direct application of the leftover hash lemma, since  $q$  is prime.

The bias of  $\lambda\mathcal{D}$  can be computed using a Poisson summation as:

$$\sum_{a \in \mathbb{Z}} \rho_\sigma(a) \cos(2\pi\lambda a/q) = \rho_{1/\sigma}(\mathbb{Z} + \lambda/q) \geq \exp(-\pi\lambda^2\sigma^2/q^2).$$

Therefore, the second property follows from the independency of the coordinates of  $\mathbf{x}$  and  $\mathbf{y}$ .  $\square$

**Corollary 5.** *Let  $q$ ,  $n$  and  $m$  such that  $m \log q/(n+m) = o(n/\log n)$ , then  $(m-3) \log q/(n+m) - \log k = \omega(\log B)$  and  $m = \omega(1)$ . Then, we can solve the Small-Decision-LWE problem in time*

$$2^{(n/2+o(n))/\ln((m \log q/(n+m) - \log k)/\log B)}$$

*with negligible probability of failure.*

*Proof.* We use the previous lemma with  $\sigma = 2q^{(n+2)/(n+m-1)}$ , so that we have  $\beta = \Omega(q^{(m-3)/(n+m)}/k)$ . The algorithm from Theorem 4 needs  $2^{o(n)}$  samples, so the advantage of the potential distinguisher for Decision-LWE is  $2^{-(1/4+o(1))n}/q$  for  $\xi = 2^{-n/4}$ ; while the previous lemma proves it is less than  $2^{-n/2}/q$ .  $\square$

<sup>3</sup> The authors of [15] gave a short justification of a similar claim which is far from proven.

The NTRU cryptosystem [20] is based on the hardness of finding two polynomials  $f$  and  $g$  whose coefficients are bounded by 1 given  $h = f/g \bmod (X^n - 1, q)$ . Since  $hg = 0$  with an error bounded by 1, we can apply previous algorithms in this section to *heuristically* recover  $f$  and  $g$  in time  $2^{(n/2+o(n))/\ln \ln q}$ . This is the first subexponential time algorithm for this problem since it was introduced back in 1998.

**Corollary 6.** *Assume we have a Search-LWE problem with  $n \log q + \Omega(n/\log q)$  samples and Gaussian noise with  $\alpha = n^{-c}$  and  $q = n^d$ . Then, we can solve it in time  $2^{n/(2 \ln(d/(d-c)) - o(1))}$  for any failure probability in  $2^{-n^{o(1)}}$ .*

*Proof.* First, apply a secret-error switching (Theorem 1). Apply the previous corollary with  $B = n^{d-c+o(1)}$  which is a correct bound for the secret, except with probability  $2^{-n^{o(1)}}$ . Lemma 10 shows that  $k^2 \leq \log q \sigma^2$ , except with probability  $2^{-\Omega(n)}$ , so that  $\beta = n^{c+o(1)}$ . We can then use  $\sigma = \Theta(1)$  and apply Theorem 4.  $\square$

Note that this corollary can in fact be applied to a very large class of distributions, and in particular to the learning with rounding problem, while the distortion parameter is too large for a direct application of Theorem 4.

Also, if the reduction gives a fast (subexponential) algorithm, one may use  $\sigma = 2\sqrt{n}$  and assume that there is no quantum algorithm solving the corresponding lattice problem in dimension  $m$ .

Even more heuristically, one can choose  $\sigma$  to be the lowest such that if the reduction does not work, we have an algorithm faster than the best *known* algorithm for the same problem.

## References

1. Proceedings of the 25th Annual IEEE Conference on Computational Complexity, CCC 2010, Cambridge, Massachusetts, 9–12 June 2010. IEEE Computer Society (2010)
2. Albrecht, M.R., Cid, C., Faugère, J., Fitzpatrick, R., Perret, L.: Algebraic algorithms for LWE problems. IACR Cryptology ePrint Arch. **2014**, 1018 (2014)
3. Albrecht, M.R., Cid, C., Faugère, J., Fitzpatrick, R., Perret, L.: On the complexity of the BKW algorithm on LWE. Des. Codes Crypt. **74**(2), 325–354 (2015)
4. Albrecht, M.R., Faugère, J.-C., Fitzpatrick, R., Perret, L.: Lazy modulus switching for the BKW algorithm on LWE. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 429–445. Springer, Heidelberg (2014)
5. Applebaum, B., Cash, D., Peikert, C., Sahai, A.: Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In: Halevi [19], pp. 595–618
6. Arora, S., Ge, R.: New algorithms for learning in presence of errors. In: Aceto, L., Henzinger, M., Sgall, J. (eds.) ICALP 2011, Part I. LNCS, vol. 6755, pp. 403–415. Springer, Heidelberg (2011)
7. Banaszczyk, W.: New bounds in some transference theorems in the geometry of numbers. Math. Ann. **296**(1), 625–635 (1993)
8. Bernstein, D.J., Lange, T.: Never trust a bunny. In: Hoepman, J.-H., Verbauwhede, I. (eds.) RFIDSec 2012. LNCS, vol. 7739, pp. 137–148. Springer, Heidelberg (2013). <https://eprint.iacr.org/2012/355.pdf>

9. Blum, A., Kalai, A., Wasserman, H.: Noise-tolerant learning, the parity problem, and the statistical query model. *J. ACM* **50**(4), 506–519 (2003)
10. Brakerski, Z., Langlois, A., Peikert, C., Regev, O., Stehlé, D.: Classical hardness of learning with errors. In: *Symposium on Theory of Computing Conference, STOC 2013*, pp. 575–584 (2013). <http://perso.ens-lyon.fr/damien.stehle/downloads/LWE.pdf>
11. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) LWE. *SIAM J. Comput.* **43**(2), 831–871 (2014)
12. Chen, Y., Nguyen, P.Q.: BKZ 2.0: better lattice security estimates. In: Lee, D.H., Wang, X. (eds.) *ASIACRYPT 2011*. LNCS, vol. 7073, pp. 1–20. Springer, Heidelberg (2011)
13. Coster, M.J., Joux, A., LaMacchia, B.A., Odlyzko, A.M., Schnorr, C., Stern, J.: Improved low-density subset sum algorithms. *Comput. Complex.* **2**, 111–128 (1992)
14. Döttling, N., Müller-Quade, J.: Lossy codes and a new variant of the learning-with-errors problem. In: Johansson, T., Nguyen, P.Q. (eds.) *EUROCRYPT 2013*. LNCS, vol. 7881, pp. 18–34. Springer, Heidelberg (2013)
15. Duc, A., Tramèr, F., Vaudenay, S.: Better algorithms for lwe and LWR. *Cryptology ePrint Archive, Report 2015/056* (2015). <http://eprint.iacr.org/>
16. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: conceptually-simpler, asymptotically-faster, attribute-based. In: Canetti, R., Garay, J.A. (eds.) *CRYPTO 2013, Part I*. LNCS, vol. 8042, pp. 75–92. Springer, Heidelberg (2013)
17. Gray, R.M., Neuhoff, D.L.: Quantization. *IEEE Trans. Inf. Theor.* **44**(6), 2325–2383 (1998)
18. Guo, Q., Johansson, T., Löhndahl, C.: Solving LPN using covering codes. In: Sarkar, P., Iwata, T. (eds.) *ASIACRYPT 2014*. LNCS, vol. 8873, pp. 1–20. Springer, Heidelberg (2014)
19. Halevi, S. (ed.): *CRYPTO 2009*. LNCS, vol. 5677. Springer, Heidelberg (2009)
20. 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)
21. Kirchner, P.: Improved generalized birthday attack. *IACR Cryptology ePrint Arch.* **2011**, 377 (2011). <http://eprint.iacr.org/2011/377.pdf>
22. Lagarias, J.C., Odlyzko, A.M.: Solving low-density subset sum problems. *J. ACM* **32**(1), 229–246 (1985)
23. Lenstra, A., Lenstra, J.H., Lovász, L.: Factoring polynomials with rational coefficients. *Math. Ann.* **261**, 515–534 (1982)
24. Lévieux, É., Fouque, P.-A.: An improved LPN algorithm. In: De Prisco, R., Yung, M. (eds.) *SCN 2006*. LNCS, vol. 4116, pp. 348–359. Springer, Heidelberg (2006)
25. Lindner, R., Peikert, C.: Better key sizes (and attacks) for LWE-based encryption. In: Kiayias, A. (ed.) *CT-RSA 2011*. LNCS, vol. 6558, pp. 319–339. Springer, Heidelberg (2011)
26. Liu, M., Nguyen, P.Q.: Solving BDD by enumeration: an update. In: Dawson, E. (ed.) *CT-RSA 2013*. LNCS, vol. 7779, pp. 293–309. Springer, Heidelberg (2013)
27. Lyubashevsky, V., Micciancio, D.: On bounded distance decoding, unique shortest vectors, and the minimum distance problem. In: Halevi [19], pp. 577–594
28. Lyubashevsky, V., Palacio, A., Segev, G.: Public-key cryptographic primitives provably as secure as subset sum. In: Micciancio, D. (ed.) *TCC 2010*. LNCS, vol. 5978, pp. 382–400. Springer, Heidelberg (2010)
29. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. *J. ACM* **60**(6), 43 (2013)

30. Micciancio, D., Peikert, C.: Hardness of SIS and LWE with small parameters. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 21–39. Springer, Heidelberg (2013)
31. Mulder, E.D., Hutter, M., Marson, M.E., Pearson, P.: Using Bleichenbacher’s solution to the hidden number problem to attack nonce leaks in 384-bit ECDSA: extended version. *J. Crypt. Eng.* **4**(1), 33–45 (2014)
32. Peikert, C.: Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In: Mitzenmacher, M. (ed.) Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31–June 2 2009, pp. 333–342. ACM (2009)
33. Pöppelmann, T., Ducas, L., Güneysu, T.: Enhanced lattice-based signatures on reconfigurable hardware. *IACR Cryptology ePrint Arch.* **2014**, 254 (2014). <https://eprint.iacr.org/2014/254.pdf>
34. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. *J. ACM (JACM)* **56**(6), 34 (2009). <http://www.cims.nyu.edu/regev/papers/qcrypto.pdf>
35. Wagner, D.: A generalized birthday problem. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 288–303. Springer, Heidelberg (2002)

# Provably Weak Instances of Ring-LWE

Yara Elias<sup>1</sup>, Kristin E. Lauter<sup>2</sup>(✉), Ekin Ozman<sup>3</sup>, and Katherine E. Stange<sup>4</sup>

<sup>1</sup> Department of Mathematics and Statistics,  
McGill University, Montreal, QC, Canada  
yara.elias@mail.mcgill.ca

<sup>2</sup> Microsoft Research, One Microsoft Way, Redmond, WA 98052, USA  
klauter@microsoft.com

<sup>3</sup> Department of Mathematics, Faculty of Arts and Science, Bogazici University,  
34342 Bebek-Istanbul, Turkey  
ekin.ozman@boun.edu.tr

<sup>4</sup> Department of Mathematics, University of Colorado, Campux Box 395,  
Boulder, CO 80309-0395, USA  
kstange@math.colorado.edu

**Abstract.** The *ring and polynomial learning with errors* problems (Ring-LWE and Poly-LWE) have been proposed as hard problems to form the basis for cryptosystems, and various security reductions to hard lattice problems have been presented. So far these problems have been stated for general (number) rings but have only been closely examined for cyclotomic number rings. In this paper, we state and examine the Ring-LWE problem for general number rings and demonstrate *provably weak instances* of the Decision Ring-LWE problem. We construct an explicit family of number fields for which we have an efficient attack. We demonstrate the attack in both theory and practice, providing code and running times for the attack. The attack runs in time linear in  $q$ , where  $q$  is the modulus.

Our attack is based on the attack on Poly-LWE which was presented in [EHL]. We extend the EHL-attack to apply to a larger class of number fields, and show how it applies to attack Ring-LWE for a heuristically large class of fields. Certain Ring-LWE instances can be transformed into Poly-LWE instances without distorting the error too much, and thus provide the first weak instances of the Ring-LWE problem. We also provide additional examples of fields which are vulnerable to our attacks on Poly-LWE, including power-of-2 cyclotomic fields, presented using the minimal polynomial of  $\zeta_{2^n} \pm 1$ .

## 1 Introduction

Lattice-based cryptography has become a very hot research topic recently with the emergence of new applications to homomorphic encryption. The hardness of the Ring-LWE problem was related to various well-known hard lattice problems [R,MR09,MR04,LPR,BL+], and the hardness of the Poly-LWE problem was reduced to Ring-LWE in [LPR,DD]. The hardness of the Poly-LWE



problem is used as the basis of security for numerous cryptosystems, including [BV, BGV, GHS]. The hardness of Ring-LWE was also shown [SS] to form a basis for the proof of security of a variant of NTRU [HPS, IEEE].

In [EHL], the first weaknesses in the Poly-LWE problem were discovered for classes of number fields satisfying certain properties. In addition, a list of properties of number fields were identified which are sufficient to guarantee a reduction between the Ring-LWE and the Poly-LWE problems, and a search-to-decision reduction for Ring-LWE. Unfortunately, in [EHL], no number fields were found which satisfied both the conditions for the attack and for the reductions. Thus [EHL] produced only examples of number fields which were weak instances for Poly-LWE.

The contributions of this paper at a high level are as follows: In Sect. 3 we strengthen and extend the attacks presented in [EHL] in several significant ways. In Sect. 4, most importantly, we show how the attacks can be applied also to the Ring-LWE problem. In Sect. 5, we construct an explicit family of number fields for which we have an efficient attack on the Decision Ring-LWE Problem. This represents the first successful attacks on the Decision Ring-LWE problem for number fields with special properties. For Galois number fields, we also know that an attack on the decision problem gives an attack on the search version of Ring-LWE [EHL]. In addition, in Sect. 9, we present the first successful implementation of the EHL attack at cryptographic sizes and attack both Ring-LWE and Poly-LWE instances. For example for  $n = 1024$  and  $q = 2^{31} - 1$ , the attack runs in about 13 hours. Code for the attack is given in Appendix A. In Sect. 6 we give a more general construction of number fields such that heuristically a large percentage of them will be vulnerable to the attacks on Ring-LWE.

In more detail, we consider rings of integers in number fields  $K = \mathbb{Q}[x]/(f(x))$  of degree  $n$ , modulo a large prime number  $q$ , and we give attacks on Poly-LWE which work when  $f(x)$  has a root of small order modulo  $q$ . The possibility of such an attack was mentioned in [EHL] but not explored further. In Sects. 3.1 and 3.2, we give *two* algorithms for this attack, and in Sects. 7 and 7.3 we give many examples of number fields and moduli, some of cryptographic size, which are vulnerable to this attack. The most significant consequence of the attack is the construction of the number fields which are weak for the Ring-LWE problem (Sect. 6).

To understand the vulnerability of Ring-LWE to these attacks, we state and examine the Ring-LWE problem for general number rings and demonstrate *provably weak instances* of Ring-LWE. We demonstrate the attack in both theory and practice for an explicit family of number fields, providing code and running times for the attack. The attack runs in time linear in  $q$ , where  $q$  is the modulus. The essential point is that Ring-LWE instances can be mapped into Poly-LWE instances, and if the map does not distort the error too much, then the instances may be vulnerable to attacks on Poly-LWE. The distortion is governed by the spectral norm of the map, and we compute the spectral norm for the explicit family we construct in Sect. 5 and analyze when the attack will succeed. For the provably weak family which we construct, the feasibility of the attack depends on the ratio of  $\sqrt{q}/n$ . We prove that the attack succeeds when  $\sqrt{q}/n$  is above a certain bound, but in practice we find that we can attack instances where the

ratio is almost 100 times smaller than that bound. Even for Ring-LWE examples which are not taken from the provably weak family, we were able to attack in practice relatively generic instances of number fields where the spectral norm was small enough (see Sect. 9).

We investigate cyclotomic fields (even 2-power cyclotomic fields) given by an alternate minimal polynomial, which are weak instances of Poly-LWE for that choice of polynomial basis. Section 7.3 contains numerous examples of 2-power cyclotomic fields which are vulnerable to attack when instantiated using an alternative polynomial basis, thus showing the heavy dependence in the hardness of these lattice-based problems on the choice of polynomial basis. In addition, we analyze the case of cyclotomic fields to understand their potential vulnerability to these lines of attack and we explain why cyclotomic fields are immune to attacks based on roots of small order (Sect. 8). Finally, we provide code in the form of simple routines in SAGE to implement the attacks and algorithms given in this paper and demonstrate successful attacks with running times (Sect. 9).

As a consequence of our results, one can conclude that the hardness of Ring-LWE is both *dependent on special properties of the number field* and *sensitive to the particular choice of  $q$* , and some choices may be significantly weaker than others. In addition, for applications to cryptography, since our attacks on Poly-LWE run in time roughly  $O(q)$  and may be applicable to a wide range of fields, including even 2-power cyclotomic fields with a bad choice of polynomial basis, these attacks should be taken into consideration when selecting parameters for Poly-LWE-based systems such as [BV, BGV] and other variants. For many important applications to homomorphic encryption (see for example [GLN, BLN]), these attacks will not be relevant, since the modulus  $q$  is chosen large enough to allow for significant error growth in computation, and would typically be of size 128 bits up to 512 bits. For that range, the attacks presented in this paper would not run. However, in other applications of Ring-LWE to key exchange for the TLS protocol [BCNS], parameters for achieving 128-bit security are suggested where  $n = 2^{10}$  and  $q = 2^{32} - 1$ , with  $\sigma \approx 3$ , and these parameters would certainly be vulnerable to our attacks for weak choices of fields and  $q$ .

## 2 Background on Poly-LWE

Let  $f(x)$  be a monic irreducible polynomial in  $\mathbb{Z}[x]$  of degree  $n$ , and let  $q$  be a prime such that  $f(x)$  factors completely modulo  $q$ . Let  $P = \mathbb{Z}[x]/f(x)$  and let  $P_q = P/qP = \mathbb{F}_q[x]/f(x)$ . Let  $\sigma \in \mathbb{R}^{>0}$ . The uniform distribution on  $P \simeq \mathbb{Z}^n$  will be denoted  $\mathcal{U}$ . By *Gaussian distribution of parameter  $\sigma$*  we refer to a discrete Gaussian distribution of mean 0 and variance  $\sigma^2$  on  $P$ , spherical with respect to the power basis. This will be denoted  $\mathcal{G}_\sigma$ . It is important to our analysis that we assume that in practice, elements are sampled from Gaussians of parameter  $\sigma$  truncated at width  $2\sigma$ .

There are two standard Poly-LWE problems. Our attack solves the *decision* variant, but it also provides information about the secret.

*Problem 1 (Decision Poly-LWE Problem).* Let  $s(x) \in P$  be a secret. The *decision Poly-LWE problem* is to distinguish, with non-negligible advantage, between the same number of independent samples in two distributions on  $P \times P$ . The first consists of samples of the form  $(a(x), b(x) := a(x)s(x) + e(x))$  where  $e(x)$  is drawn from a discrete Gaussian distribution of parameter  $\sigma$ , and  $a(x)$  is uniformly random. The second consists of uniformly random and independent samples from  $P \times P$ .

*Problem 2 (Search Poly-LWE Problem).* Let  $s(x) \in P$  be a secret. The *search Poly-LWE problem*, is to discover  $s$  given access to arbitrarily many independent samples of the form  $(a(x), b(x) := a(x)s(x) + e(x))$  where  $e(x)$  is drawn from a Discrete Gaussian of parameter  $\sigma$ , and  $a(x)$  is uniformly random.

The polynomial  $s(x)$  is called the *secret* and the polynomials  $e_i(x)$  are called the *errors*.

## 2.1 Parameter Selection

The selection of parameters for security is not yet a well-explored topic. Generally parameter recommendations for Poly-LWE and Ring-LWE are just based on the recommendations for general LWE, ignoring the extra ring structure e.g. [PG,RV+,BCNS]. Sample concrete parameter choices have been suggested, where  $w$  is the width of the Gaussian error distribution (precisely,  $w = \sqrt{2\pi}\sigma$ ):

1.  $P_{LP1} = (n, q, w) = (192, 4093, 8.87)$ ,  $P_{LP2} = (256, 4093, 8.35)$ ,  $P_{LP3} = (320, 4093, 8.00)$  for low, medium and high security, recommended by Lindner and Peikert in [LP];
2.  $P_{GF} = (n, q, w) = (512, 12289, 12.18)$  for high security used in [GF+];
3.  $P_{BCNS} = (n, q, w) = (1024, 2^{31} - 1, 3.192)$  suggested in [BCNS] for the TLS protocol. Here,  $q = 2^{32} - 1$  was actually suggested but it is not prime. Here, the authors remark that  $q$  is taken to be large for correctness but could potentially be decreased.

## 3 Attacks on Poly-LWE

The attack we are concerned with is quite simple. It proceeds in four stages:

1. Transfer the problem to  $\mathbb{F}_q$  via a ring homomorphism  $\phi : P_q \rightarrow \mathbb{F}_q$ .
2. Loop through guesses for the possible images  $\phi(s(x))$  of the secret.
3. Obtain the values  $\phi(e_i(x))$  under the assumption that the guess at hand is correct.
4. Examine the distribution of the  $\phi(e_i(x))$  to determine if it is Gaussian or uniform.

If  $f$  is assumed to have a root  $\alpha \equiv 1 \pmod q$  or  $\alpha$  of small order modulo  $q$ , then this attack is due to Eisentraeger-Hallgren-Lauter [EHL].

The first part is to transfer the problem to  $\mathbb{F}_q$ . Write  $f(x) = \prod_{i=1}^n (x - \alpha_i)$  for the factorization of  $f(x)$  over  $\mathbb{F}_q$  which is possible by assumption. By the Chinese remainder theorem, if  $f$  has no double roots, then

$$P_q \simeq \prod_{i=1}^n \mathbb{F}_q[x]/(x - \alpha_i) \simeq \mathbb{F}_q^n$$

There are  $n$  ring homomorphisms

$$\phi : P_q \rightarrow \mathbb{F}_q[x]/(x - \alpha_i) \simeq \mathbb{F}_q, \quad g(x) \mapsto g(\alpha_i).$$

Fix one of these, by specifying a root  $\alpha = \alpha_i$  of  $f(x)$  in  $\mathbb{F}_q$ . Apply the homomorphism to the coordinates of the  $\ell$  samples  $(a_i(x), b_i(x))$ , obtaining  $(a_i(\alpha), b_i(\alpha))_{i=1, \dots, \ell}$ .

Next, loop through all  $g \in \mathbb{F}_q$ . Each value  $g$  is to be considered a guess for the value of  $s(\alpha)$ . For each guess  $g$ , assuming that it is a correct guess and  $g = s(\alpha)$ , then

$$e_i(\alpha) = b_i(\alpha) - a_i(\alpha)g = b_i(\alpha) - a_i(\alpha)s(\alpha).$$

In the case that the samples were LWE samples and the guess was correct, then this produces a collection  $(e_i(\alpha))$  of images of errors chosen according to some distribution. If the distributions  $\phi(\mathcal{U})$  and  $\phi(\mathcal{G}_\sigma)$  are distinguishable, then we can determine whether the distribution was uniform or Gaussian. Note that  $\phi(\mathcal{U})$  will of course be uniform on  $\mathbb{F}_q$ . If our guess is incorrect, or if the samples are not LWE samples, then the distribution will appear uniform.

Therefore, after looping through all guesses, if all the distributions appeared uniform, then conclude that the samples were not LWE samples; whereas if one of the guesses worked for all samples and always yielded an error distribution which appeared Gaussian, assume that particular  $g$  was a correct guess. In the latter case this also yields one piece of information about the secret:  $g = s(\alpha) \pmod q$ .

The attack *will* succeed whenever

1.  $q$  is small enough to allow looping through  $\mathbb{F}_q$ ,
2.  $\phi(\mathcal{U})$  and  $\phi(\mathcal{G}_\sigma)$  are distinguishable.

Our analysis hinges on the difficulty of distinguishing  $\phi(\mathcal{U})$  from  $\phi(\mathcal{G}_\sigma)$ , as a function of the parameters  $\sigma$ ,  $n$ ,  $\ell$ ,  $q$ , and  $f$ . Distinguishability becomes easier when  $\sigma$  is smaller (so  $\mathcal{U}$  and  $\mathcal{G}_\sigma$  are farther apart to begin with),  $n$  is smaller and  $q$  is larger (since then less information is lost in the map  $\phi$ ), and  $\ell$  is larger (since there are more samples to test the distributions). The dependence on  $f$  comes primarily as a function of its roots  $\alpha_i$  modulo  $q$ , which may have properties that make distinguishing easier.

Ideally, for higher security, one will choose parameters that make distinguishing nearly impossible, i.e. such that  $\phi(\mathcal{G}_\sigma)$  appears very close to uniform modulo  $q$ .

**Example** ([EHL]). We illustrate the attack in the simplest case  $\alpha = 1$ . Assume  $f(1) \equiv 0 \pmod q$ , and consider the distinguishability of the two distributions  $\phi(\mathcal{U})$

and  $\phi(\mathcal{G}_\sigma)$ . Given  $(a_i(x), b_i(x))$ , make a guess  $g \in \mathbb{F}_q$  for the value of  $s(1)$  and compute  $b_i(1) - g \cdot a_i(1)$ . If  $b_i$  is uniform, then  $b_i(1) - g \cdot a_i(1)$  is uniform for all  $g$ . If  $b_i = a_i s + e_i$ , then there is a guess  $g$  for which  $b_i(1) - g a_i(1) = e_i(1)$  where  $e_i(x) = \sum_{j=1}^n e_{ij} x^j$  and  $g = s(1)$ . Since  $e_i(1) = \sum_{j=1}^n e_{ij}$ , where  $e_{ij}$  are chosen from  $\mathcal{G}_\sigma$ , it follows that  $e_i(1)$  are sampled from  $\mathcal{G}_{\sqrt{n}\sigma}$  where  $n\sigma^2 \ll q$ . The attack can be described loosely as follows: for each sample, test each guess  $g$  in  $\mathbb{F}_q$  to see if  $b_i(1) - g \cdot a_i(1)$  is small modulo  $q$ , and only keep those guesses which pass the test. Repeat with the next sample and continue to keep only the guesses which pass.

### 3.1 Attack Based on a Small Set of Error Values Modulo $q$

In this section, we assume that there exists a root  $\alpha$  of  $f$  such that  $\alpha$  has small order  $r$  modulo  $q$ , that is  $\alpha^r \equiv 1 \pmod q$ . Then

$$e(\alpha) = \sum_{i=0}^{n-1} e_i \alpha^i = (e_0 + e_r + e_{2r} + \dots) + \alpha(e_1 + e_{r+1} + \dots) + \dots + \alpha^{r-1}(e_{r-1} + e_{2r-1} + \dots). \tag{1}$$

If  $r$  is small enough, then  $e(\alpha)$  takes on only a small number of values modulo  $q$ . If so, then we can efficiently distinguish whether a value modulo  $q$  belongs to that subset.

Let  $S$  be the set of possible values of  $e(\alpha)$  modulo  $q$ . We assume for simplicity that  $n$  is divisible by  $r$ . Then the coefficients  $e_j + e_{j+r} + \dots + e_{n-r+j}$  of (1) fall into a subset of  $\mathbb{Z}/q\mathbb{Z}$  of size at most  $4\sigma n/r$ . We sum over  $r$  terms, hence,  $|S| = (4\sigma n/r)^r$  residues modulo  $q$ . For  $r = 2$ , this becomes  $(2n\sigma)^2$ .

---

#### Algorithm 1. Small set of error values

---

**Input:** A collection of  $\ell$  Poly-LWE samples.

**Output:** A guess  $g$  for  $s(\alpha)$ , the value of the secret polynomial at  $\alpha$ ; or else **NOT PLWE**; or **INSUFFICIENT SAMPLES**.

The value **NOT PLWE** indicates that the collection of samples were definitely not Poly-LWE samples.

The value **INSUFFICIENT SAMPLES** indicates that there were not enough samples to determine a single guess  $s(\alpha)$ . In this case, the algorithm may be continued on a new set of samples by looping the remaining surviving guesses on the new samples.

Create an ordered list of elements of  $S$ .

Let  $G$  be an empty list.

**for**  $g$  from 0 to  $q - 1$  **do**

**for**  $(a(x), b(x))$  in the collection of samples **do**

**if**  $b(\alpha) - g a(\alpha)$  does not equal an element of  $S$  **then**

**break** (i.e. begin next value of  $g$ )

append  $g$  to  $G$  (note: occurs only if the loop of samples completed without a break)

**if**  $G$  is empty **then**

return **NOT PLWE**

**if**  $G = \{g\}$  **then**

return  $g$

**if**  $\#G > 1$  **then**

return **INSUFFICIENT SAMPLES**

---

The attack described below succeeds with high probability if  $|S| \ll q$ , that is

$$(4\sigma n/r)^r \ll q.$$

**Proposition 1.** *Assume that*

$$(4\sigma n/r)^r < q. \tag{2}$$

*Algorithm 1 terminates in time at most  $\tilde{O}(\ell q + nq)$ , where the  $\tilde{O}$  notation hides the  $\log(q)$  factors and the implied constant depends upon  $r$ . Furthermore, if the algorithm returns **NOT PLWE**, then the samples were not valid Poly-LWE samples. If it outputs anything other than **NOT PLWE**, then the samples are valid Poly-LWE samples with probability  $1 - (\frac{\#S}{q})^\ell$ . In particular, this probability tends to 1 as  $\ell$  grows.*

*Proof.* As discussed above, there are at most  $q$  possible values for the elements of  $S$  under the assumption (2). To compute each one takes  $n$  additions per coefficient (of which there are  $r$ ), combined with an additional  $r$  multiplications and  $r$  additions. (Here we have assumed the  $\alpha^i$  have been computed; this takes  $r$  multiplications.) Each addition or multiplication takes time at most  $\log q$ . Therefore, computing  $S$  takes time at most  $\tilde{O}(qnr)$ . For sorting, it is best to sort as  $S$  is computed; placing each element correctly takes  $\log q$  time.

The principal double loop takes time at most  $\tilde{O}(\ell q)$ . If  $b(\alpha)$  and  $a(\alpha)$  are precomputed, then for each guess  $g$ , the computation of  $b(\alpha) - ga(\alpha)$  only costs one multiplication and one subtraction modulo  $q$  (i.e.  $2 \log q$ ) while it requires only  $\log q$  bit comparisons to decide whether this is in the set  $S$ .

In Step 4, for later samples, only guesses which were successful in the previous samples (i.e. gave a value which was in the set  $S$ ) are considered. For a sample chosen uniformly at random, one expects the number of successful guesses to be roughly  $\frac{\#S}{q}$ . Thus for the second sample, we repeat the above test for only  $(\#S)$  guesses. At the  $\ell^{\text{th}}$  sample, retaining only guesses which were successful for all previous samples, we expect to test only  $(\frac{\#S}{q})^\ell q$  guesses, which very quickly goes to zero. Hence, if we examine  $\ell$  samples, our tolerance for false positives is proportional to  $(\frac{\#S}{q})^\ell$ .

### 3.2 Attack Based on the Size of the Error Values

In this section, we describe the most general  $\phi : P_q \rightarrow \mathbb{F}_q$  attack on the Poly-LWE problem, one which can be carried out in any situation. The rub is that the probability of success will be vanishingly small unless we are in a very special situation. Therefore our analysis actually bolsters the security of Poly-LWE.

Suppose that  $f(\alpha) \equiv 0 \pmod q$ . Let  $E_i$  be the event that  $b_i(\alpha) - ga_i(\alpha) \pmod q$  is in the interval  $[-q/4, q/4)$  for some sample  $i$  and guess  $g$  for  $s(\alpha) \pmod q$ . The main idea is to compare  $P(E_i \mid \mathcal{D} = \mathcal{U})$  and  $P(E_i \mid \mathcal{D} = \mathcal{G}_\sigma)$ . If  $\mathcal{D} = \mathcal{U}$ , then  $b_i(\alpha) - ga_i(\alpha)$  is random modulo  $q$  for all guesses  $g$ , that is,

$P(E_i | \mathcal{D} = \mathcal{U}) = \frac{1}{2}$ . If  $\mathcal{D} = \mathcal{G}_\sigma$ , then  $b_i(\alpha) - s(\alpha)a_i(\alpha) = e_i(\alpha) \pmod q$ . We consider

$$e_i(\alpha) = \sum_{j=0}^{n-1} e_{ij}\alpha^j,$$

where  $e_{ij}$  is chosen according to the distribution  $\mathcal{G}_\sigma$  (truncated at  $2\sigma$ ) and distinguish two cases:

1.  $\alpha = \pm 1$
2.  $\alpha \neq \pm 1$  and  $\alpha$  has small order  $r \geq 3$  modulo  $q$

**Case 1** ( $\alpha = \pm 1$ ).

The error  $e_i(\alpha)$  is chosen according to the distribution  $\mathcal{G}_{\sigma\sqrt{n}}$  truncated at  $2\sigma\sqrt{n}$ . Hence

$$-2\sigma\sqrt{n} \leq e_i(\alpha) \leq 2\sigma\sqrt{n}.$$

Therefore, assuming that

$$2\sigma\sqrt{n} < \frac{q}{4},$$

we obtain  $P(E_i | \mathcal{D} = \mathcal{G}_\sigma) = 1$  for  $g = s(\alpha)$ . Hence  $\mathcal{U}$  and  $\mathcal{G}_\sigma$  are distinguishable.

**Case 2** ( $\alpha \neq \pm 1$  and  $\alpha$  has small order  $r \geq 3$  modulo  $q$ ).

The error can be written as

$$e(\alpha) = \sum_{i=0}^{r-1} e_i\alpha^i = (e_0 + e_r + \dots) + \alpha(e_1 + e_{r+1} + \dots) + \dots + \alpha^{r-1}(e_{r-1} + e_{2r-1} + \dots)$$

where we assume that  $n$  is divisible by  $r$  for simplicity. For  $j = 0, \dots, r-1$ , we have that  $e_j + e_{j+r} + \dots + e_{j+n-r}$  is chosen according to the distribution  $\mathcal{G}_{\sqrt{\frac{n}{r}}\sigma}$ . As a consequence  $e(\alpha)$  is sampled from  $\mathcal{G}_{\bar{\sigma}}$  where

$$\bar{\sigma}^2 = \sum_{i=0}^{r-1} \frac{n}{r}\sigma^2\alpha^{2i} = \frac{n}{r}\sigma^2\frac{\alpha^{2r} - 1}{\alpha^2 - 1}.$$

Hence

$$-2\frac{\sqrt{n}}{\sqrt{r}}\sigma\frac{\sqrt{\alpha^{2r} - 1}}{\sqrt{\alpha^2 - 1}} \leq e(\alpha) \leq 2\frac{\sqrt{n}}{\sqrt{r}}\sigma\frac{\sqrt{\alpha^{2r} - 1}}{\sqrt{\alpha^2 - 1}}.$$

Therefore, assuming that

$$2\frac{\sqrt{n}}{\sqrt{r}}\sigma\frac{\sqrt{\alpha^{2r} - 1}}{\sqrt{\alpha^2 - 1}} < \frac{q}{4}, \tag{3}$$

we obtain  $P(E_i | \mathcal{D} = \mathcal{G}_\sigma) = 1$  for  $g = s(\alpha)$ , and uniform and Gaussian are distinguishable. Note that Hypothesis (2) implies in particular that  $\alpha^r > q$ .

In each of the two cases, we have given conditions on the size of  $\sigma$  under which  $\mathcal{U}$  and  $\mathcal{G}_\sigma$  are distinguishable and an attack is likely to succeed. We now elaborate on the algorithm that would be used.

**Algorithm 2.** Small error values

---

**Input:** A collection of  $\ell$  Poly-LWE samples.

**Output:** A guess  $g$  for  $s(\alpha)$ ; or else **NOT PLWE**; or **INSUFFICIENT SAMPLES**.

The output **INSUFFICIENT SAMPLES** indicates that more samples are needed to make a determination. In this case, the algorithm can be continued by looping through remaining surviving guesses on new samples.

Let  $G$  be an empty list.

**for**  $g$  from 1 to  $q - 1$  **do**

**for**  $(a(x), b(x))$  in the collection of samples **do**

**if** the minimal residue  $b(\alpha) - ga(\alpha)$  does not lie in  $[-q/4, q/4]$  **then**

**break** (i.e. begin next value of  $g$ )

        append  $g$  to  $G$  (note: occurs only if the loop of samples completed without a break)

**if**  $G$  is empty **then**

    return **NOT PLWE**

**if**  $G = \{g\}$  **then**

    return  $g$

**if**  $\#G > 1$  **then**

    return **INSUFFICIENT SAMPLES**

---

We denote by  $\ell$  the number of samples observed. For each guess  $g \bmod q$ , we compute  $b_i - ga_i$  for  $i = 1, \dots, \ell$ . If there is a guess  $g \bmod q$  for which the event  $E_i$  occurs for all  $i = 1, \dots, \ell$ , then the algorithm returns the guess if it is unique and **INSUFFICIENT SAMPLES** otherwise; the samples are likely valid Poly-LWE samples. Otherwise, it reports that they are certainly not valid Poly-LWE samples.

**Proposition 2.** *Assume that we are in one of the following cases:*

1.  $\alpha = \pm 1$  and

$$8\sigma\sqrt{n} < q.$$

2.  $\alpha$  has small order  $r \geq 3$  modulo  $q$ , and

$$8\sigma \frac{\sqrt{n} \sqrt{\alpha^{2r} - 1}}{\sqrt{r} \sqrt{\alpha^2 - 1}} < q.$$

Then Algorithm 2 terminates in time at most  $\tilde{O}(\ell q)$ , where the implied constant is absolute. Furthermore, if the algorithm returns **NOT PLWE**, then the samples were not valid Poly-LWE samples. If it outputs anything other than **NOT PLWE**, then the samples are valid Poly-LWE samples with probability at least  $1 - (\frac{1}{2})^\ell$ .

*Proof.* The proof is as in Proposition 1, without the first few steps.

We remark that Propositions and Algorithms 1 and 2 overlap in some cases. For  $\alpha = \pm 1$ , Algorithm 2 is more applicable (i.e. more parameter choices are susceptible), while for  $\alpha$  of other small orders, Algorithm 1 is more applicable.



## 4 Moving the Attack from Poly-LWE to Ring-LWE

We use the term Poly-LWE to refer to LWE problems generated by working in a polynomial ring, and reserve the term Ring-LWE for LWE problems generated by working with the canonical embedding of a number field as in [LPR, LPR13]. In the previous sections we have expanded upon Eisenträger, Hallgren and Lauter’s observation that for certain distributions on certain lattices given by Poly-LWE, the ring structure presents a weakness. We will now consider whether it is possible to expand that analysis to LWE instances created through Ring-LWE for number fields besides cyclotomic ones.

In particular, the necessary ingredient is that the distribution be such that under the ring homomorphisms of Sect. 3, the image of the errors is a ‘small’ subset of  $\mathbb{Z}/q\mathbb{Z}$ , either the error values themselves are small, or they form a small, identifiable subset of  $\mathbb{Z}/q\mathbb{Z}$ . Assuming a spherical Gaussian in the canonical embedding of  $R$  or  $R^\vee$ , we describe a class of number fields for which this weakness occurs. A similar analysis would apply without the assumption that the distribution is spherical in the canonical embedding.

Here, we setup the key players (a number field and its canonical embedding, etc.) for general number fields so that these definitions specialize to those in [LPR13]. There are some choices inherent in our setup: it may be possible to generalize Ring-LWE to number fields in several different ways. We consider the two most natural ways.

### 4.1 The Canonical Embedding

Let  $K$  be a number field of degree  $n$  with ring of integers  $R$  whose dual is  $R^\vee$ . We will embed the field  $K$  in  $\mathbb{R}^n$ . Note that our setup is essentially that of [DD], rather than [LPR13], but the difference is notational.

Let  $\sigma_1, \dots, \sigma_n$  be the  $n$  embeddings of  $K$ , ordered so that  $\sigma_1$  through  $\sigma_{s_1}$  are the  $s_1$  real embeddings, and the remaining  $n - s_1 = 2s_2$  complex embeddings are paired in such a way that  $\overline{\sigma_{s_1+k}} = \sigma_{s_1+s_2+k}$  for  $k = 1, \dots, s_2$  (i.e. list  $s_2$  non-pairwise-conjugate embeddings and then list their conjugates following that).

Define a map  $\theta : K \rightarrow \mathbb{R}^n$  given by

$$\theta(r) = (\sigma_1(r), \dots, \sigma_{s_1}(r), \operatorname{Re}(\sigma_{s_1+1}(r)), \dots, \operatorname{Re}(\sigma_{s_1+s_2}(r)), \operatorname{Im}(\sigma_{s_1+1}(r)), \dots, \operatorname{Im}(\sigma_{s_1+s_2}(r))).$$

The image of  $K$  is the  $\mathbb{Q}$ -span of  $\theta(\omega_i)$  for any basis  $\omega_i$  for  $K$  over  $\mathbb{Q}$ . This is not the usual Minkowski embedding, but it has the virtues that (1) the codomain is a real, not complex, vector space; and (2) the spherical or elliptical Gaussians used as error distributions in [LPR13] are, in our setup, spherical or elliptical with respect to the usual inner product. We denote the usual inner product by  $\langle \cdot, \cdot \rangle$  and the corresponding length by  $|x| = \sqrt{\langle x, x \rangle}$ . It is related to the trace pairing on  $K$ , i.e.  $\langle \theta(r), \theta(s) \rangle = \operatorname{Tr}(r\bar{s})$ .

Then  $R$  and  $R^\vee$  form lattices in  $\mathbb{R}^n$ .

## 4.2 Spherical Gaussians and Error Distributions

We define a *Ring-LWE error distribution* to be a spherical Gaussian distribution in  $\mathbb{R}^n$ . That is, for a parameter  $\sigma > 0$ , define the *continuous Gaussian distribution function*  $D_\sigma : \mathbb{R}^n \rightarrow (0, 1]$  by

$$D_\sigma(x) := (\sqrt{2\pi}\sigma)^{-n} \exp(-|x|^2/(2\sigma^2)).$$

This gives a distribution  $\Psi$  on  $K \otimes \mathbb{R}$ , via the isomorphism  $\theta$  to  $\mathbb{R}^n$ . By approximating  $K \otimes \mathbb{R}$  by  $K$  to sufficient precision, this gives a distribution on  $K$ .

From this distribution we can generate the *Ring-LWE error distribution* on  $R$ , respectively  $R^\vee$ , by taking a valid discretization  $[\Psi]_R$ , respectively  $[\Psi]_{R^\vee}$ , in the sense of [LPR13]. Now we have at hand a lattice,  $R$ , respectively  $R^\vee$ , and a distribution on that lattice. The parameters (particularly  $\sigma$ ) are generally advised to be chosen so that this instance of LWE is secure against general attacks on LWE (which do not depend on the extra structure endowed by the number theory).

## 4.3 The Ring-LWE Problems

Write  $R_q := R/qR$  and  $R_q^\vee = R^\vee/qR^\vee$ . The standard Ring-LWE problems are as follows, where  $K$  is taken to be a cyclotomic field [LPR, LPR13].

**Definition 1 (Ring-LWE Average-Case Decision [LPR]).** *Let  $s \in R_q^\vee$  be a secret. The average-case decision Ring-LWE problem, is to distinguish with non-negligible advantage between the same number of independent samples in two distributions on  $R_q \times R_q^\vee$ . The first consists of samples of the form  $(a, b := as + e)$  where  $e$  is drawn from  $\chi := [\Psi]_{R^\vee}$  and  $a$  is uniformly random, and the second consists of uniformly random and independent samples from  $R_q \times R_q^\vee$ .*

**Definition 2 (Ring-LWE Search [LPR]).** *Let  $s \in R_q^\vee$  be a secret. The search Ring-LWE problem, is to discover  $s$  given access to arbitrarily many independent samples of the form  $(a, b := as + e)$  where  $e$  is drawn from  $\chi := [\Psi]_{R^\vee}$  and  $a$  is uniformly random.*

In proposing general number field Ring-LWE, one of two avenues may be taken:

1. preserve these definitions exactly as they are stated, or
2. eliminate the duals, i.e. replace every instance of  $R^\vee$  with  $R$  in the definitions above.

To distinguish these two possible definitions, we will refer to *dual Ring-LWE* and *non-dual Ring-LWE*. Lyubashevsky, Peikert and Regev remark that for cyclotomic fields, dual and non-dual Ring-LWE lead to computationally equivalent problems [LPR, Sect. 3.3]. They go on to say that over cyclotomics, for implementation and efficiency reasons, dual Ring-LWE is superior.

Generalising dual Ring-LWE to general number fields is the most naive approach, but it presents the problem that working with the dual in a general number field may be difficult. Still, it is possible there are families of accessible number fields for which this may be the desired avenue.

We will analyse the effect of the Poly-LWE vulnerability on both of these candidate definitions. In fact, the analysis will highlight some potential differences in their security, already hinted at in the discussion in [LPR, Sect. 3.3].

#### 4.4 Isomorphisms from $\theta(R)$ to a Polynomial Ring

Suppose  $K$  is a *monogenic number field*, meaning that  $R$  is isomorphic to a polynomial ring  $P = \mathbb{Z}[X]/f(X)$  for some monic irreducible polynomial  $f$  ( $f$  is a *monogenic polynomial*). In this case, we obtain  $R = \gamma R^\vee$ , for some  $\gamma \in R$  (here,  $\gamma$  is a generator of the different ideal), so that  $\theta(R^\vee)$  and  $\theta(R)$  are related by a linear transformation. Thus a (dual or non-dual) Ring-LWE problem concerning the lattice  $\theta(R)$  or  $\theta(R^\vee)$  can be restated as a Poly-LWE problem concerning  $P$ .

Let  $\alpha$  be a root of  $f$ . Then  $R$  is isomorphic to  $P$ , via  $\alpha \mapsto X$ . An integral basis for  $R$  is  $1, \alpha, \alpha^2, \dots, \alpha^{n-1}$ . An integral basis for  $R^\vee$  is  $\gamma^{-1}, \gamma^{-1}\alpha, \gamma^{-1}\alpha^2, \dots, \gamma^{-1}\alpha^{n-1}$ . Let  $M_\alpha$  be the matrix whose columns are  $\{\theta(\alpha^i)\}$ . Let  $M_\alpha^\vee$  be the matrix whose columns are  $\{\theta(\gamma^{-1}\alpha^i)\}$ . If  $\mathbf{v}$  is a vector of coefficients representing some  $\beta \in K$  in terms of the basis  $\{\alpha^i\}$  for  $K/\mathbb{Q}$ , then  $\theta(\beta) = M_\alpha \mathbf{v}$ . In other words,  $M_\alpha : P \rightarrow \theta(R)$  is an isomorphism (where  $P$  is represented as vectors of coefficients). Similarly,  $M_\alpha^\vee : P \rightarrow \theta(R^\vee)$  is an isomorphism.

#### 4.5 The Spectral Norm

Given an  $n \times n$  matrix  $M$ , its *spectral norm*  $\rho = \|M\|_2$  is the  $\ell_2$  norm on its  $n^2$  entries. This is equal to the largest singular value of  $M$ . This is also equal to the largest radius of the image of a unit ball under  $M$ . This last interpretation allows one to bound the image of a spherical Gaussian distribution of parameter  $\sigma$  on the domain of  $M$  by another of parameter  $\rho\sigma$  on the codomain of  $M$  (in the sense that the image of the ball of radius  $\sigma$  will map into a ball of radius  $\rho\sigma$  after application of  $M$ ).

The normalized spectral norm of  $M$  is defined to be  $\rho' = \|M\|_2 / \det(M)^{1/n}$ . The condition number of  $M$  is  $k(M) = \|M\|_2 \|M^{-1}\|_2$ .

#### 4.6 Moving the Attack from Poly-LWE to Ring-LWE

Via the isomorphism  $M := M_\alpha^{-1}$  (respectively  $M := (M_\alpha^\vee)^{-1}$ ), an instance of the non-dual (respectively dual) Ring-LWE problem gives an instance of the Poly-LWE problem in which the error distribution is the image of the error distribution in  $\theta(R)$  (respectively  $\theta(R^\vee)$ ). In general, this may be an elliptic Gaussian distorted by the isomorphism. If the distortion is not too large, then it may be bounded by a spherical Gaussian which is not too large. In that

case, a solution to the Poly-LWE problem with the new spherical Gaussian error distribution may be possible. If so, it will yield a solution to the original Ring-LWE problem.

This is essentially the same reduction described in [EHL]. However, those authors assume that the isomorphism is an orthogonal linear map; we are loosening this condition. The essential question in this loosening is how much the Gaussian distorts under the isomorphism. Our contribution is an analysis of the particular basis change.

This distortion is governed by the spectral norm  $\rho$  of  $M$ . If the continuous Gaussian in  $\mathbb{R}^n$  is of parameter  $\sigma$  (with respect to the standard basis of  $\mathbb{R}^n$ ), then the new spherical Gaussian bounding its image is of parameter  $\rho\sigma$  with respect to  $P$  (in terms of the coefficient representation). The appropriate analysis for discrete Gaussians is slightly more subtle. Loosely speaking, we find that a Ring-LWE instance is weak if the following three things occur:

1.  $K$  is monogenic.
2.  $f$  satisfies  $f(1) \equiv 0 \pmod{q}$ .
3.  $\rho$  and  $\sigma$  are sufficiently small.

The first condition guarantees the existence of appropriate isomorphisms to a polynomial ring; the second and third are required for the Poly-LWE attack to apply. The purpose of the third requirement is that the discrete Gaussian distribution in  $\mathbb{R}^n$  transfers to give vectors  $e(x)$  in the polynomial ring having the property that  $e(1)$  lies in the range  $[-q/4, q/4]$  except with negligible probability; this allows Algorithm 2 and the conclusions of Proposition 2 to apply.

Let us now state our main result.

**Theorem 1.** *Let  $K$  be a number field such that  $K = \mathbb{Q}(\beta)$ , and the ring of integers of  $K$  is equal to  $\mathbb{Z}[\beta]$ . Let  $f$  be the minimal polynomial of  $\beta$  and suppose  $q$  is a prime such that  $f$  has root 1 modulo  $q$ . Finally, suppose that the spectral norm  $\rho$  of  $M_\beta^{-1}$  satisfies*

$$\rho < \frac{q}{4\sqrt{2\pi\sigma n}}.$$

*Then the non-dual Ring-LWE decision problem for  $K, q, \sigma$  can be solved in time  $\tilde{O}(\ell q)$  with probability  $1 - 2^{-\ell}$ , using a dataset of  $\ell$  samples.*

*Proof.* Sampling a discrete Gaussian with parameter  $\sigma$  results in vectors of norm at most  $\sqrt{2\pi\sigma}\sqrt{n}$  except with probability at most  $2^{-2n}$  [LPR13, Lemma 2.8]. Considering the latter to be negligible, then we can expect error vectors to satisfy  $\|\mathbf{v}\|_2 < \sqrt{2\pi\sigma}\sqrt{n}$  and their images in the polynomial ring to satisfy

$$|e(1)| = \|e(x)\|_1 < \sqrt{n}\|e(x)\|_2 < \sqrt{n}\rho\sqrt{2\pi\sigma}\sqrt{n} = \rho\sqrt{2\pi\sigma n}.$$

Therefore, if

$$\rho\sqrt{2\pi\sigma n} < q/4,$$

then we may apply the attack of Sect. 3.2 that assumes  $f(1) \equiv 0 \pmod{q}$  and that error vectors lie in  $[-q/4, q/4]$ .

In what follows, we find a family of polynomials satisfying the conditions of the theorem, and give heuristic arguments that such families are in fact very common. The other cases (other than  $\alpha = 1$ ) appear out-of-reach for now, simply because the bounds on  $\rho$  are much more difficult to attain. We will not examine them closely.

### 4.7 Choice of $\sigma$

The parameters of Sect. 2.1 are used in implementations where the Gaussian is taken over  $(\mathbb{Z}/q\mathbb{Z})^n$ , and security depends upon the proportion of this space included in the ‘bell,’ meaning, it depends upon the ratio  $q/\sigma$ . In the case of Poly-LWE, sampling is done on the coefficients, which are effectively living in the space  $(\mathbb{Z}/q\mathbb{Z})^n$ , so this is appropriate. However, in Ring-LWE, the embedding  $\theta(R)$  in  $\mathbb{R}^n$  may be very sparse (i.e.  $\theta(R^\vee)$  may be very dense). Still, the security will hinge upon the proportion of  $\theta(R)/q\theta(R)$  that is contained in the bell. We have not seen a discussion of security parameters for Ring-LWE in the literature, and so we propose that the appropriate meaning of the width of the Gaussian,  $w$ , in this case is

$$w := \sqrt{2\pi}\sigma' := \sqrt{2\pi}\sigma \det(M_\alpha)^{1/n}, \tag{4}$$

where  $\sigma'$  is defined by the above equality. The reason for this choice is that  $\theta(R)$  has covolume  $\det(M_\alpha)$ ; a very sparse lattice (corresponding to large determinant) needs a correspondingly large  $\sigma$  so that the same proportion of its vectors lie in the bell.

If  $\rho$  represents the spectral norm of  $M_\alpha^{-1}$  (which has determinant  $\det(M_\alpha)^{-1}$ ), then

$$\rho' := \rho \det(M_\alpha)^{1/n}$$

is the normalized spectral norm. Therefore  $\rho/\sigma = \rho'/\sigma'$ . Hence the bound of Theorem 1 becomes

$$\rho' < \frac{q}{4wn}. \tag{5}$$

## 5 Provably Weak Ring-LWE Number Fields

Consider the family of polynomials

$$f_{n,q}(x) = x^n + q - 1$$

for  $q$  a prime. These satisfy  $f(1) \equiv 0 \pmod{q}$ . By the Eisenstein criterion, they are irreducible whenever  $q - 1$  has a prime factor that appears to exponent 1. These polynomials have discriminant [M] given by

$$(-1)^{\frac{n^2-n}{2}} n^n (q - 1)^{n-1}.$$

**Proposition 3.** *Let  $n$  be power of a prime  $\ell$ . If  $q - 1$  is squarefree and  $\ell^2 \nmid ((1 - q)^n - (1 - q))$  then the polynomials  $f_{n,q}$  are monogenic.*

*Proof.* This is a result of Gassert in [G, Theorem 5.1.4]. As stated, Theorem 5.1.4 of [G] requires  $\ell$  to be an odd prime. However, for the monogenicity portion of the conclusion, the proof goes through for  $p = 2$ .

**Proposition 4.** *Suppose that  $f_{n,q}$  is irreducible, and the associated number field has  $r_2$  complex embeddings. Then  $r_2 = n/2$  or  $(n-1)/2$  (whichever is an integer), and the normalized spectral norm of  $M_\alpha^{-1}$  is exactly*

$$2^{-r_2/n} \sqrt{(q-1)^{1-\frac{1}{n}}}.$$

*Proof.* Let  $a$  be a positive real  $n$ -th root of  $q-1$ . Then the roots of the polynomial are exactly  $a\zeta_{2n}^j$  for  $j$  odd such that  $1 \leq j < 2n$ . The embeddings take  $a\zeta_{2n}$  to each of the other roots. There is  $r_1 = 1$  real embedding if  $n$  is odd (otherwise  $r_1 = 0$ ), and the rest are  $r_2$  complex conjugate pairs, so that  $n = r_1 + 2r_2$ . Then the dot product of the  $r$ -th and  $s$ -th columns of  $M_\alpha^{-1}$  is

$$\sum_{k=0}^{n-1} a^{r+s} \zeta_{2n}^{(r-s)(2k+1)} = 0$$

Therefore, the columns of the matrix are orthogonal to one another. Hence, the matrix is diagonalizable, and its eigenvalues are the lengths of its column vectors, which is for the  $r$ -th column,

$$\left( \sum_{k=0}^{n-1} \|a^r \zeta_{2n}^{2k+1}\|^2 \right)^{1/2} = \sqrt{n} a^r$$

Therefore the smallest singular value of  $M_\alpha$  is  $\sqrt{n}$  and the largest is  $\sqrt{n} a^{n-1}$ . Correspondingly, the largest singular values of  $M_\alpha^{-1}$  is  $1/\sqrt{n}$ .

A standard result of number theory relates the determinant of  $M_\alpha$  to the discriminant of  $K$  via

$$\det(M_\alpha) = 2^{-r_2} \sqrt{\text{disc}(f_{n,q})},$$

where  $r_2 \leq \frac{n}{2}$  is the number of complex embeddings of  $K$ . Combining the smallest singular value with this determinant (the discriminant is given explicitly at the beginning of this section) gives the result.

**Theorem 2.** *Suppose  $q$  is prime,  $n$  is an integer and  $f = f_{n,q}$  satisfies*

1.  $n$  is a power of the prime  $\ell$ ,
2.  $q-1$  is squarefree,
3.  $\ell^2 \nmid ((1-q)^n - (1-q))$ ,
4. we have  $\tau > 1$ , where

$$\tau := \frac{q}{2\sqrt{2}wn(q-1)^{\frac{1}{2}-\frac{1}{2n}}}.$$

Then the non-dual Ring-LWE decision problem for  $f$  and  $w$  (defined by (4)) can be solved in time  $\tilde{O}(\ell q)$  with probability  $1 - 2^{-\ell}$ , using a dataset of  $\ell$  samples.

*Proof.* Under the stated conditions,  $f$  has a root 1 modulo  $q$ , and therefore Poly-LWE is vulnerable to the attack specified in Algorithm 2. The other properties guarantee the applicability of Theorem 1 via Propositions 3 and 4.

Under the assumption that  $q - 1$  is infinitely often squarefree, this provides a family of examples which are susceptible to attack (taking, for example,  $n$  as an appropriate power of 2; note that in this case item (3) is automatic).

Interestingly, their susceptibility increases as  $q$  increases relative to  $n$ . It is the ratio  $\sqrt{q}/n$ , rather than their overall size, which controls the vulnerability (at least as long as  $q$  is small enough to run a loop through the residues modulo  $q$ ).

The quantity  $\tau$  can be considered a measure of security against this attack; it should be small to indicate higher security. For the various parameters indicated in Sect. 2.1, the value of  $\tau$  is:

Parameters	$P_{LP1}$	$P_{LP2}$	$P_{LP3}$	$P_{GF}$	$P_{BCNS}$
$\tau$	0.0136	0.0108	0.0090	0.0063	5.0654

The bound on  $\tau$  in Theorem 1 is stronger than what is required in practice for the attack to succeed. In particular, the spectral norm of the transformation  $M_\alpha^{-1}$  does not accurately reflect the average behaviour; it is worst case. As  $n$  increases, it is increasingly unlikely that error samples happen to lie in just the right direction from the origin to be inflated by the full spectral norm. Furthermore, we assumed in the analysis of Theorem 1 an overly generous bound on the error vectors.

The proof is in the pudding: in Sect. 9 we have successfully attacked parameters for which  $\tau < 0.02$ , including  $P_{LP1}$ .

## 6 Heuristics on the Prevalence of Weak Ring-LWE Number Fields

In this section, we argue that many examples satisfying Theorem 1 are very likely to exist. In fact, each of the individual conditions is fairly easy to attain. We will see in what follows that given a random monogenic number field, there is with significant probability at least one prime  $q$  for which Ring-LWE is vulnerable (i.e. the bound (5) is attained) for parameters comparable to those of  $P_{BCNS}$ . Note that in this parameter range, the spectral norm is expensive to compute directly.

### 6.1 Monogenicity

Monogenic fields are expected to be quite common in the following sense. If  $f$  of degree  $n \geq 4$  is taken to be a random polynomial (i.e. its coefficients are chosen randomly), then it is conjecturally expected that with probability

$\gtrsim 0.307$ ,  $P$  will be the ring of integers of a number field  $[K]$ . In particular, if  $f$  has squarefree discriminant, this will certainly happen. Furthermore, cyclotomic fields are monogenic, as are the families described in the last section.

However, at degrees  $n \sim 2^{10}$ , the discriminant of  $f$  is too large to test for squarefreeness, so testing for monogenicity may not be feasible. Kedlaya has developed a method for constructing examples of arbitrary degree  $[K]$ .

## 6.2 Examples, $n = 2^{10}$ , $q \sim 2^{32}$

Consider the following examples:

$$\begin{aligned} f(x) &= x^{1024} + (2^{31} + 14)x + 2^{31}, & q &= 4294967311, \\ f(x) &= x^{1024} + (2^{31} + 2^{30} + 22)x + (2^{31} + 2^{30}), & q &= 6442450967, \\ f(x) &= x^{1024} + (2^{31} + 2^{30} + 29)x + (2^{31} + 2^{30} + 5), & q &= 6442450979. \end{aligned}$$

These examples are discussed at greater length in Sect. 7.2, where the method for constructing them is explained. In each case,  $f(1) \equiv 0 \pmod{q}$ .

In this size range, we were not able to compute the spectral norm of  $K$  directly in a reasonable amount of time. In the next few sections we will make persuasive heuristic arguments that it can be expected to have  $\rho'$  well within the required bound (5), i.e.  $\rho' < 2^{17}$ . That is, we expect these examples and others like them to be vulnerable.

## 6.3 Heuristics for the Spectral Norm

To find large  $q$  requires taking more complex polynomials  $f$ , which in turn may inflate the spectral norm, so the complexity of  $f$  must be balanced.

One approach is to consider polynomials of the form  $f(x) = x^n + ax + b$ . Let us recall a standard result of number theory. For a number field  $K$  with  $r_1$  real embeddings and  $r_2$  conjugate pairs of complex embeddings, the determinant of the canonical embedding is  $\sqrt{\Delta_K} 2^{-r_2}$ . Therefore, if  $\Delta_K > 2^{2r_2}$  (call this Assumption A), we obtain  $\det(M_f) > 1$ . Then  $\|M_f\|_2 > 1$ . We are interested in the spectral norm of the inverse:

$$\|M_f^{-1}\|_2 \leq k(M_f) / \|M_f\|_2 \leq k(M_f),$$

where  $k(M_f)$  represents the condition number of  $M_f$ . Now,

$$\rho' = \|M_f^{-1}\|_2 \det(M_f)^{1/n}.$$

As mentioned above,  $\det(M_f)$  is given in terms of  $\Delta_K$ . Under the assumption that  $\mathbb{Z}[X]/f(X)$  is indeed a ring of integers,  $\Delta_K = \text{Disc}(f)$  (call this Assumption B). By [M],

$$\text{Disc}(f) = (n-1)^{n-1} a^n + (-1)^{n-1} n^n (b+1)^{n-1}.$$



It is evident that by judicious choice of  $a$  and  $b$ , it is possible to obtain a range of discriminant sizes. We can expect there to be plenty of examples in the range  $n^2 < \Delta_K < n^3$  (in this range, Assumption A is satisfied). Then we obtain

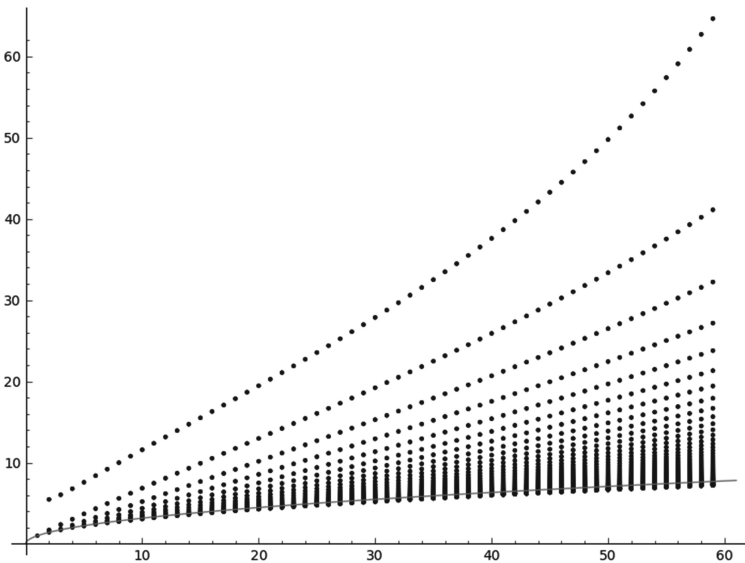
$$\rho' \leq 2k(M_f).$$

The condition number of  $M_f$  is hard to access theoretically, but heuristically, for random perturbations of any fixed matrix, most perturbations are well-conditioned (having small condition number) [TV]. The matrix  $M_f$  is a perturbation of  $M_p$  for  $p = x^n + 1$ . The extent of this perturbation can be bounded in terms of the coefficients  $a$  and  $b$ , since the perturbation is controlled by the perturbation in the roots of the polynomial. It is a now-standard result in numerical analysis, due to Wilkinson, that roots may be ill-conditioned in this sense, but the condition number can be bounded in terms of the coefficients  $a$  and  $b$ . This implies that, heuristically,  $k(M_f)$  is likely to be small quite frequently.

In conclusion, we expect to find that many  $f(x)$  will have  $\rho'$  quite small.

### 6.4 Experimental Evidence for the Spectral Norm

We only ran experiments in a small range due to limitations of our Sage implementation [S]. The polynomials  $x^{32} + ax + b$ ,  $-60 \leq a, b \leq 60$  were plotted on a  $\max\{a, b\}$ -by- $\rho'$  plane. The result is as follows:



There are some examples with quite high  $\rho'$ , but the majority cluster low. The grey line is  $y = \sqrt{x}$ . Therefore, we may conjecture based on this experiment, that we may expect to find plenty of  $f$  satisfying  $\rho' < \sqrt{\max\{a, b\}}$ .

Experimentally, we may guess that the examples of Sect. 6.2, for which  $n = 2^{10}$  and  $\max\{a, b\} \leq 2^{30}$ , will frequently satisfy  $\rho' < 2^{15}$ , which is the range required by Theorem 1. (Note that the coefficients cannot be taken smaller if  $f$  is to have root 1 modulo a prime  $q \sim 2^{31}$ .)

## 7 Weak Poly-LWE Number Fields

### 7.1 Finding $f$ and $q$ with Roots of Small Order

It is relatively easy to generate polynomials  $f$  and primes  $q$  for which  $f$  has a root of given order modulo  $q$ . There are two approaches: given  $f$ , find suitable  $q$ ; and given  $q$ , find suitable  $f$ . Since there are other conditions one may require for other reasons (particularly on  $f$ ), we focus on the first of these.

Given  $f$ , in order to find  $q$  such that  $f$  has a root of small order (this includes the cases  $\alpha = \pm 1$ ), the following algorithm can be applied.

---

**Algorithm 3.** Finding primes  $q$  such that  $f(x)$  has a root of small order modulo  $q$

---

**Input:** A non-cyclotomic irreducible polynomial  $f(X) \in \mathbb{Z}[X]$ ; and an integer  $m \geq 1$ .

**Output:** A prime  $q$  such that  $f(X)$  has a root of order  $m$  modulo  $q$ .

1. Let  $\Phi_m(X)$  be the cyclotomic polynomial of degree  $m$ . Apply the extended Euclidean algorithm to  $f(X)$  and  $\Phi_m(X)$  over the ring  $\mathbb{Q}[X]$  to obtain  $a(X)$ ,  $b(X)$  such that

$$a(X)f(X) + b(X)\Phi_m(X) = 1.$$

(Note that 1 is the GCD of  $f(X)$  and  $\Phi_m(X)$  by assumption.)

2. Let  $d$  be the least common multiple of all the denominators of the coefficients of  $a$  and  $b$ .
  3. Factor  $d$ .
  4. Return the largest prime factor of  $d$ .
- 

It is also possible to generate examples by first choosing  $q$  and searching for appropriate  $f$ . For example, taking  $f(x) = \Phi_m(x)g(x) + q$  where  $g(x)$  is monic of degree  $m - n$  suffices. Both methods can be adapted to find  $f$  having any specified root modulo  $q$ .

### 7.2 Examples, $n \sim 2^{10}$ , $q \sim 2^{32}$

For the range  $n \sim 2^{10}$ , we hope to find  $q \sim 2^{32}$ . Examples were found by applying Algorithm 3 to polynomials  $f(x)$  of the form  $x^n + ax + b$  for  $a, b$  chosen from a likely range. Examples are copious and not difficult to find (see Appendix A.2 for code).

**Case  $\alpha = 1$ .** A few typical examples of irreducible  $f$  with 1 as a root modulo  $q$  are:

$$\begin{aligned} f(x) &= x^{1024} + (2^{31} + 14)x + 2^{31}, & q &= 4294967311, \\ f(x) &= x^{1024} + (2^{31} + 2^{30} + 22)x + (2^{31} + 2^{30}), & q &= 6442450967, \\ f(x) &= x^{1024} + (2^{31} + 2^{30} + 29)x + (2^{31} + 2^{30} + 5), & q &= 6442450979. \end{aligned}$$

These examples satisfy condition 1 of Proposition 2 with  $\sigma = 3$ , hence are vulnerable.

**Case  $\alpha = -1$ .** Here is an irreducible  $f$  with root  $-1$ :

$$f(x) = x^{1024} + (2^{31} + 9)x - (2^{31} + 7), \quad q = 4294967311 \sim 2^{32}.$$

This example similarly satisfies condition 1 of Proposition 2 and so is vulnerable.

**Case  $\alpha$  Small Order.** Here is an irreducible  $f$  with a root of order 3:

$$f(x) = x^{1024} + (2^{16} + 2)x - 2^{16}, \quad q = 1099514773507 \sim 2^{40}.$$

This example has  $q \sim 2^{40}$ ; taking this larger  $q$  allows us to satisfy (2) of Proposition 1 and hence it is vulnerable to Algorithm 1.

### 7.3 Examples of Weak Poly-LWE Number Fields with Additional Properties

In this section we will give examples of number fields  $K = \mathbb{Q}[x]/(f(x))$  which are vulnerable to our attack on Poly-LWE. They will be vulnerable by satisfying one of the following two possible conditions:

**R**  $f(1) \equiv 0 \pmod{q}$ .

**R'**  $f$  has a root of small order modulo  $q$ .

We must also require:

**Q** The prime  $q$  can be chosen suitably large.

The examples we consider are cyclotomic fields and therefore Galois and monogenic. One should note that guaranteeing these two conditions together is nontrivial in general. In addition to these, there are additional conditions for the attack explained in [EHL]. The desirable conditions are:

**G**  $K$  is Galois.

**M**  $K$  is monogenic.

**S** The ideal  $(q)$  splits completely in the ring of integers  $R$  of  $K$ , and  $q \nmid [R : \mathbb{Z}[\beta]]$ .

**O** The transformation between the canonical embedding of  $K$  and the power basis representation of  $K$  is given by a scaled orthogonal matrix.

Conditions G and S are needed for the Search-to-Decision reduction and Conditions M and O are needed for the Ring-LWE to Poly-LWE reduction in [EHL].

Note that checking the splitting condition for fields of cryptographic size is not computationally feasible in general. However, we are able to give a sufficient condition for certain splittings which is quite fast to check.

**Proposition 5.** *Using the notation as above, if  $f(2) \equiv 0 \pmod{q}$  then  $q$  splits in  $R$ .*

*Proof.* Since  $2^{2^{k-1}} \equiv -1 \pmod{q}$ , it follows that  $(2^\alpha)^{2^{k-1}} \equiv (-1)^\alpha \equiv -1 \pmod{q}$  for all odd  $\alpha$  in  $\mathbb{Z}$ . We'll show that  $2, 2^3, 2^5, \dots, 2^m$  where  $m = 2^k - 1$  are all distinct mod  $q$ , hence showing that  $f(x)$  has  $2^{k-1}$  distinct roots mod  $q$  i.e.  $f(x)$  splits mod  $q$ . Assume that  $2^i \equiv 2^j \pmod{q}$  for some  $1 \leq i < j \leq 2^k - 1$ . Then  $2^{j-i} \equiv 1 \pmod{q}$ , which means that the order of 2 modulo  $q$  divides  $j - i$ . However, by the fact below (Lemma 1), the order of 2 mod  $q$  is  $2^k$ , which is a contradiction since  $j - i < 2^k$ .

**Lemma 1.** *Let  $q$  be a prime such that  $2^{2^{k-1}} \equiv -1 \pmod{q}$  for some integer  $k$ . Then the order of 2 modulo  $q$  is  $2^k$ .*

*Proof.* Let  $a$  be the order of 2 modulo  $q$ . By assumption  $(2^{2^{k-1}})^2 \equiv 2^{2^k} \equiv 1 \pmod{q}$ . Then  $a|2^k$  i.e.  $a = 2^\alpha$  for some  $\alpha \leq k$ . Say  $\alpha \leq k - 1$ . Then  $1 = (2^{2^\alpha})^{2^{k-1-\alpha}} = 2^{2^{k-1}} \equiv -1 \pmod{q}$ , a contradiction.

The converse of Proposition 5 does not hold. For instance, let  $K$  be the splitting field of the polynomial  $x^8 + 1$  and  $q = 401$ . Then  $q$  splits in  $R$ . However  $f(2) = 257 \not\equiv 0 \pmod{q}$ .

We now present a family of examples for which  $\alpha = -1$  is a root of  $f$  of order two. Conditions G, M, S, R' (order 2) and Q are all satisfied. The field  $K$  is the cyclotomic number field of degree  $\phi(2^k) = 2^{k-1}$ , but instead of the cyclotomic polynomial we take the minimal polynomial of  $\zeta_{2^k} + 1$ . In each case,  $q$  is obtained by factoring  $2^{2^{k-1}} + 1$  for various values of  $k$  and splitting is verified using Proposition 5.

k 2	3	4	5	6	7	7	8	8
q 5	17	257	65537 ~ 2 <sup>16</sup>	6700417 ~ 2 <sup>22</sup>	274177 ~ 2 <sup>18</sup>	q <sub>5</sub> ~ 2 <sup>45</sup>	q <sub>6</sub> ~ 2 <sup>55</sup>	q <sub>1</sub> ~ 2 <sup>72</sup>
k 9	9	10	10	10	11	11	11	
q q <sub>7</sub> ~ 2 <sup>50</sup>	q <sub>2</sub> ~ 2 <sup>205</sup>	2424833 ~ 2 <sup>21</sup>	q <sub>3</sub> ~ 2 <sup>162</sup>	q <sub>4</sub> ~ 2 <sup>328</sup>	q <sub>8</sub> ~ 2 <sup>25</sup>	q <sub>9</sub> ~ 2 <sup>32</sup>	q <sub>10</sub> ~ 2 <sup>131</sup>	

Several of these examples are of cryptographic size<sup>1</sup>, i.e. the field has degree  $2^{10}$  and the prime is of size  $\sim 2^{32}$  or greater. These provide examples which are weak against our Poly-LWE attack, by Proposition 2.

## 8 Cyclotomic (in)vulnerability

One of our principal observations is that the cyclotomic fields, used for Ring-LWE, are uniquely protected against the attacks presented in this paper. The next proposition states that the polynomial ring of the  $m$ -th cyclotomic polynomial  $\Phi_m$  will never be vulnerable to the attack based on a root of small order.

<sup>1</sup>  $q_1 = 5704689200685129054721, q_2 = 93461639715357977769163558199606896584051237541638188580280321,$   
 $q_3 = 7455602825647884208337395736200454918783366342657, q_5 = 67280421310721,$   
 $q_6 = 59649589127497217$   
 $q_4 = 74164006262753080152478714190193747405994078109751902390582131614441$   
 $5759504705008092818711693940737$   
 $q_7 = 1238926361552897, q_8 = 45592577, q_9 = 6487031809, q_{10} = 46597757852200185$   
 $43264560743076778192897.$

**Proposition 6.** *The roots of  $\Phi_m$  have order  $m$  modulo every split prime  $q$ .*

*Proof.* Consider the field  $\mathbb{F}_q$ ,  $q$  prime. Since  $\mathbb{F}_q$  is perfect, the cyclotomic polynomial  $\Phi_m(x)$  has  $\phi(m)$  roots in an extension of  $\mathbb{F}_q$ . This polynomial has no common factor with  $x^k - 1$  for  $k < m$ . However, it divides  $x^m - 1$ . Therefore its roots have order dividing  $m$ , but not less than  $m$ . That is, its roots are all of order exactly  $m$  in the field in which they live. Now, if we further assume that  $\Phi_m(x)$  splits modulo  $q$ , then its  $\phi(m)$  roots are all elements of order  $m$  modulo  $q$ , so in particular,  $m \mid q - 1$ . The roots of  $\Phi_m(x)$  are all elements of  $\mathbb{Z}/q\mathbb{Z}$  of order exactly  $m$ .

The question remains whether there is another polynomial representation for the ring of cyclotomic integers for which  $f$  does have a root of small order. This may in fact be the case, but the error distribution is transformed under the isomorphism to this new basis, so this does not guarantee a weakness in Poly-LWE for  $\Phi_m$ .

However, it is not necessary to search for all such representations to rule out the possibility that this provides an attack. The ring  $R_q \cong \mathbb{F}_q^n$  has exactly  $n = \phi(m)$  homomorphisms to  $\mathbb{Z}/q\mathbb{Z}$ . If  $R_q$  can be represented as  $(\mathbb{Z}/q\mathbb{Z})[X]/f(X)$  with  $f(\alpha) = 0$ , then the map  $R_q \rightarrow \mathbb{Z}/q\mathbb{Z}$  is given by  $p \mapsto p(\alpha)$  is one of these  $n$  maps. It suffices to write down these  $n$  maps (in terms of any representation!) and verify that the errors map to all of  $\mathbb{Z}/q\mathbb{Z}$  instead of a small subset. It is a special property of the cyclotomics that these  $n$  homomorphisms coincide. Thus we are reduced to the case above.

## 9 Successfully Coded Attacks

The following table documents Ring-LWE and Poly-LWE parameters that were successfully attacked on a Thinkpad X220 laptop with Sage Mathematics Software [S], together with approximate timings. For code, see Appendix A. The first row indicates that cryptographic size is attackable in Poly-LWE. The second row indicates that a generic example attackable by Poly-LWE is also susceptible to Ring-LWE (see Sect. 6). We were unable to test the Ring-LWE attack for  $n > 256$  only because Sage’s built-in Discrete Gaussian Sampler was not capable of initializing (thus we were unable to produce samples to test). The last two rows illustrate the  $\tau$  of Theorem 1 that is required for security in practice (approximately  $\tau < 0.013$  instead of  $\tau < 1$  in theory). In the Ring-LWE rows, parameters were chosen to illustrate the boundary of feasibility for a fixed  $n$ . Since the feasibility of the attack depends on the ratio  $\sqrt{q}/n$ , there is no reason to think larger  $n$  are invulnerable (provided  $q$  also grows), but we were unable to produce samples to test against. The Poly-LWE example illustrates that runtime for large  $q$  is feasible (runtimes for Poly-LWE and Ring-LWE are the same; it is only the samples which differ).

Case	$f$	$q$	$w$	$\tau$	Samples per run	Successful runs	Time per run
Poly-LWE	$x^{1024} + 2^{31} - 2$	$2^{31} - 1$	3.192	N/A	40	1 of 1	13.5 hrs
Ring-LWE	$x^{128} + 524288x + 524285$	524287	8.00	N/A	20	8 of 10	24 sec
Ring-LWE	$x^{192} + 4092$	4093	8.87	0.0136	20	1 of 10	25 sec
Ring-LWE	$x^{256} + 8189$	8190	8.35	0.0152	20	2 of 10	44 sec

**Acknowledgments.** The authors are indebted to the organizers of the research conference Women in Numbers 3 (Rachel Pries, Ling Long and the fourth author), as well as to the Banff International Research Station, for bringing together this collaboration. The authors would also like to thank Martin Albrecht for help with Sage.

## A Appendix: Code

### A.1 Proof of Concept for Ring-LWE and Poly-LWE Attacks

The following Sage Mathematical Software [S] code verifies that Algorithm 2 succeeds on the Poly-LWE and Ring-LWE examples of Sect. 9. Note that Algorithm 1 is a minor modification of Algorithm 2.

This code relies on `DiscreteGaussianDistributionLatticeSampler`, an inbuilt package in Sage. The sampler is incapable of initializing in sufficiently large dimension to fully test the attacks in this paper. See the related trac ticket <http://trac.sagemath.org/ticket/17764>.

Built into the code are several error checks that will be triggered if insufficient precision is not used.

This code is available in electronic form at <http://math.colorado.edu/~kstange/scripts.html>.

```
#####
# RING-LWE ATTACK #
#####

# General preparation of Sage: Create a polynomial ring and import GaussianSampler, Timer
P.<y> = PolynomialRing(RationalField(), 'y')
from sage.stats.distributions.discrete_gaussian_lattice import DiscreteGaussianDistributionLatticeSampler
RP = RealField(300) # this sets the precision; if it is insufficient, the implementation won't be valid
from sage.doctest.util import Timer

# Give the Minkowski lattice for a given ring determined by a polynomial.
# Also gives a key to which are real embeddings.
def cmatrix(): # returns a matrix, columns basis 1, x, x^2, x^3, ... given in the canonical embedding
    global N, a
    N.<a> = NumberField(f)
    fdeg = f.degree()
    key = [0 for i in range(fdeg)] # 0 = real, 1 = real part of complex emb, 2 = imaginary part
    embs = N.embeddings(CC)
    M = matrix(RP, fdeg, fdeg)
    print"Preparing an embedding matrix: computing powers of the root."
    apows = [ a^j for j in range(n) ]
    print"Finished computing the powers of the root."
    i = 0
    while i < n:
        em = embs[i]
        if Mod(i,20)==Mod(0,20) or Mod(i,20)==Mod(1,20):
            print"Embedding matrix: ", i, " rows out of ", n, " complete."
```

```

if em(a).imag() == 0:
    key[i] = 0
    for j in range(n):
        M[i,j] = em(apows[j]).real()
    i = i + 1
else:
    key[i] = 1
    key[i+1] = 2
    for j in range(n):
        M[i,j] = em(apows[j]).real()
        M[i+1,j] = (em(apows[j])*I).real()
    i = i + 2
return M, key

# Produce a random vector from (Z/qZ)^n
def random_vec(q, dim):
    return vector([ZZ.random_element(0,q) for i in range(dim)])

# Useful function for real numbers modulo q
def modq(r,q):
    s = r/q
    t = r/q - floor(r/q)
    return t*q

# Call sampler
def call_sampler():
    e = sampler().change_ring(RP)
    return e

# Create samples using a lattice (given by latmat and its inverse),
# a Gaussian sampler on that lattice, secret, prime
def get_sample(latmat, latmatinv, sec, qval, keyval):
    e = call_sampler() # create error, in R^n
    dim = latmat.dimensions()[0] # detect dimension of lattice
    pre_a = random_vec(qval, dim) # create a uniformly randomly in terms of basis in cm
    a = latmat*pre_a # create a, in R^n
    b = vecmul_poly(a,sec,latmat,latmatinv) + e # create b, in R^n
    pre_b = latmatinv*b # move to basis in cm in order to reduce mod q
    pre_b_red = vector([modq(c,qval) for c in pre_b])
    b = latmat*pre_b_red
    return [a, b]

# Global choices: setup a field and prime, sampler.
# Set to dummy values that will be altered when an attack is run
q = 1
n = 1
sig = 1/sqrt(2*pi)
Zq = IntegerModRing(q)
R.<x> = PolynomialRing(Zq)
f = y + 1
N.<a> = NumberField(f)
S.<z> = R.quotient(f) # This is P_q
cm,key = cmatrix()
cmi = cm.inverse()
cm
cm53 = cm.change_ring(RealField(10))
cmqq = cm53.change_ring(QQ)
sampler = DiscreteGaussianDistributionLatticeSampler(cmqq.transpose(), sig)

# Set the parameters for the attack
def setup_params(fval,qval,sval):
    global q,n,sig,f,S,x,Z,Zq
    f = fval
    n = f.degree()
    q = qval
    Zq = IntegerModRing(q)
    R.<x> = PolynomialRing(Zq)
    sig = sval/sqrt(2*pi)
    S.<z> = R.quotient(f)
    print"Setting up parameters, polynomial = ", f, " and prime = ", q, " and sigma = ", sig
    print"Verifying properties: "
    print"Prime?", q.is_prime()

```

```

print"Irreducible? ", f.is_irreducible()
print"Value at 1 modulo q?", Mod(f.subs(y=1),q)
return True

# Compute the lattices in Minkowski space
def prepare_matrices():
    global cm, key, cmi, cmqq
    print"Preparing matrices."
    cm,key = cmatrix()
    print"Embedding matrix prepared."
    cmi = cm.inverse()
    print"Inverse matrix found."
    cm53 = cm.change_ring(RealField(10))
    cmqq = cm53.change_ring(QQ)
    print"All matrices prepared."
    return True

# Make a vector in R^n into a polynomial, given change of basis matrix and variable to use
def make_poly(a,matchchange,var):
    coeffs = matchchange*a #coefficients of the polynomial are given by the change of basis matrix
    pol = 0
    for i in range(n):
        pol = pol + ZZ(round(coeffs[i]))*var^i # var controls where it will live (what poly ring)
    return pol

# Make a polynomial into a vector in Minkowski space
def make_vec(fval,matchchange):
    if fval == 0:
        coeffs = [0 for i in range(n)]
    else:
        coeffs = [0 for i in range(n)]
        colist = lift(fval).coefficients()
        for i in range(len(colist)):
            coeffs[i] = ZZ(colist[i])
    return matchchange*vector(coeffs)

# Multiplication in the Minkowski space via moving to polynomial ring
def vecmul_poly(u,v,mat,matinv):
    poly_u = make_poly(u,matinv,z)
    poly_v = make_poly(v,matinv,z)
    poly_prod = poly_u*poly_v
    return make_vec(poly_prod,mat)

# Create the sampler on the lattice embedded in R^n
def initiate_sampler():
    global sampler
    print"Initiating Sampler."
    sampler = DiscreteGaussianDistributionLatticeSampler(cmqq.transpose(), sig)
    print"Sampler initiated with sigma", RDF(sig)
    return True

# Produce error vectors, just a test to see how they look
def error_test(num):
    print"Testing the error vector production by producing ", num," errors."
    errorlist = [sampler().norm().n() for _ in range(num)]
    meannorm = mean(errorlist) # average norm
    maxnorm = max(errorlist) # maximum norm
    print"The average error norm is ", RDF(meannorm/( sqrt(n)*sampler.sigma*sqrt(2*pi) )), " times sqrt(n)*s."
    maxratio = RDF(maxnorm/( sqrt(n)*sampler.sigma*sqrt(2*pi) ))
    print"The maximum error norm is ", maxratio," times sqrt(n)*s."
    if maxratio > 1:
        print"----- ERROR -----"
        print"The errors do not satisfy a proven upper bound in norm."
    return True

# Create the secret
secret = 0
def create_secret():
    global secret
    secret = cm*random_vec(q,n)
    return True

# Produce samples

```



```

samps = []
numsamps = 1
def create_samples(numsampsval):
    global samps, numsamps
    samps = []
    print"Creating samples"
    for i in range(numsampsval):
        print"Creating sample number ", i
        samp = get_sample(cm, cmi, secret, q, key)
        samps.append(samp)
    numsamps = len(samps)
    print"Done creating ", numsamps,"samples."
    return True

# Function for going down to q
def go_to_q(a,matchange,x):
    pol = make_poly(a,matchange,x)
    #print"debug got pol:", pol
    pol_eval = pol.subs(x=1)
    #print"debug eval'd to:", pol_eval," and then ", Zq((pol_eval))
    return Zq(pol_eval)

# Check to make sure moving to q preserves product -- the last two lines should be equal
def sanity_check():
    print"Initiating sanity check"
    mat = cmi
    pvec1 = random_vec(q,n)
    vec1 = cm*pvec1
    pvec2 = random_vec(q,n)
    vec2 = cm*pvec2
    vprod2 = vecmul_poly(vec1,vec2,cm,cmi)
    first_thing = go_to_q(vprod2,mat)
    second_thing = go_to_q(vec1,mat)*go_to_q(vec2,mat)
    if first_thing == second_thing:
        print"Sanity confirmed."
    else:
        print"----- ERROR -----"
        print"Sanity problem:", first_thing," is not equal to ", second_thing, "."
        print"Are you sure your ring has root 1 mod q?"
    return True

# Given a list of elements of  $\mathbb{Z}/q\mathbb{Z}$ , make a histogram and zero count
def histq(data):
    hist = [0 for i in range(10)] # empty histogram
    zeroct=0 # count of zeroes mod q
    for datum in data:
        e = datum
        if e == 0:
            zeroct = zeroct+1
        histbit = floor(ZZ(e)*10/q)
        hist[histbit]=hist[histbit]+1
    return [hist, zeroct]

# Given a list of vectors in  $R^n$ , create a histogram of their
# values in  $\mathbb{Z}/q\mathbb{Z}$  under make_poly, together with a zero count
def histo(data,cmi):
    return histq([go_to_q(datum,cmi) for datum in data])

# Create a histogram of error vectors, transported to polynomial ring
def histogram_of_errors():
    print"Creating a histogram of errors mod q."
    errs = []
    for i in range(80):
        errs.append(sampler())
    hist = histo(errs,cmi)
    print"The number of error vectors that are zero:", hist[1]
    bar_chart(hist[0], width=1).show(figsize=2)
    return True

# Create a histogram of the a's in the samples, transported to polynomial ring
def histogram_of_as():
    print"Creating a histogram of sample a's mod q."
    a_vals = [samp[0] for samp in samps]

```

```

hist = histo(a_vals,cmi)
print"The number of a's that are zero:", hist[1]
bar_chart(hist[0], width=1).show(figsize=2)
return True

# Create a histogram of errors by correct guess
def histogram_of_errors_2():
    print"Creating a histogram of supposed errors if sample is guessed, mod q."
    hist = histoq([ lift(Zq(go_to_q(sample[i],cmi) - go_to_q(sample[0],cmi)*go_to_q(secret,cmi))) for sample in samps])
    print"The number of such that are zero:", hist[1]
    bar_chart(hist[0], width=1).show(figsize=2)
    return True

# Create the secret mod q
lift_s = 0
def secret_mod_q():
    global lift_s
    lift_s = go_to_q(secret,cmi)
    print"Storing the secret mod q. The secret is ", secret," which becomes ", lift_s
    return True

# Algorithm 2
# reportrate controls how often it updates the status of the loop; larger = less frequently
# quickflag = True will run only the secret and a few other values to give a quick idea if it works
def alg2(reportrate, quickflag = False):
    print"Beginning algorithm 2."
    numsamps = len(samps)
    a = [ 0 for i in range(numsamps)]
    b = [ 0 for i in range(numsamps)]
    print"Moving samples to F_q."
    for i in range(numsamps):
        sample = samps[i]
        a[i] = go_to_q(sample[0],cmi)
        b[i] = go_to_q(sample[1],cmi)
    possibles = []
    winner = [],0
    print"Samples have been moved to F_q."
    for i in range(2):
        if i == 0:
            print"!!!! ROUND 1: !!!!! First, checking how many samples the secret survives (peeking ahead)."
            iterat = [lift_s]
        if i == 1:
            print"!!!! ROUND 2: !!!!! Now, running the attack naively."
            possibles = []
            if quickflag:
                print"We are doing it quickly (not a full test)."
                iterat = xrange(1000)
            else:
                iterat = xrange(q)
        for g in iterat:
            if Mod(g,reportrate) == Mod(0,reportrate):
                print"Currently checking residue ", g
                g = Zq(g)
                potential = True
                ctr = 0
                while ctr < numsamps and potential:
                    e = abs(lift(Zq(b[ctr]-g*a[ctr])))
                    if e > q/4 and e < 3*q/4:
                        potential = False
                        if ctr == winner[1]:
                            winner[0].append(g)
                            print"We have a new tie for longest chain:", g," has survived ", ctr," rounds."
                        if ctr > winner[1]:
                            winner = [[g],ctr]
                            print"We have a new longest chain of samples survived:", g," has survived ", ctr," rounds."
                ctr = ctr + 1
            if potential == True:
                print"We found a potential secret: ", g
                possibles.append(g)
        if g == lift_s:
            if i == 0:
                print"The real secret survived ", ctr,"samples."

```

```

        #break
print"Full list of survivors of the ", numamps, " samples:", possibles
print"The real secret mod q was: ", lift_s
if len(possibles) == 1 and possibles[0] == lift_s:
    print"Success!"
    return True
else:
    print"Failure!"
    return False

# Run a simulation.
def shebang(fval,qval,sval,numampsval,numtrials,quickflag=False):
    global sig
    print"Welcome to the Ring-LWE Attack."
    n = fval.degree()
    print"The attack should theoretically work if the following quantity is greater than 1."
    print"Quantity: ", RDF( qval/( 2*sqrt(2)*sval*n*(qval-1)^( (n-1)/2/n ) ) )
    timer = Timer()
    timer2 = Timer()
    timer.start()
    print"***** PHASE 1: SETTING UP SYSTEM "
    setup_params(fval,qval,sval)
    prepare_matrices()
    print"Computing the adjustment factor for s."
    cemps = (n - len(N.embeddings(RR)))/2
    detscale = RP( ( 2^(-cemps)*sqrt(abs(f.discriminant())) )^(1/n) ) # adjust the sigma,s
    sval = sval*detscale
    sig = sig*detscale
    print"Adjusted s for use with this embedding, result is ", sval
    initiate_sampler()
    print"The sampler has been created with sigma = ", sampler.sigma
    print"Sampled vectors will have expected norm ", RDF(sqrt(n)*sampler.sigma)
    error_test(5)
    print"Time for Phase 1: ", timer.stop()
    timer.start()
    count_successes = 0
    timer2.start()
    for trialnum in range(numtrials):
        print"*-*-*-*-*-*-*-*-*-*-*-*-*-*-* TRIAL NUMBER ", trialnum,"*-*-*-*-*-*-*-*-*-*-*-*-*-*-*"
        print"***** PHASE 2: CREATE SECRET AND SAMPLES"
        create_secret()
        create_samples(numampsval)
        sanity_check()
        print"Time for Phase 2: ", timer.stop()
        timer.start()
        print"***** PHASE 3: HISTOGRAMS"
        histogram_of_errors()
        print"The histogram of errors (above) should be clustered at edges for success."
        histogram_of_as()
        print"The histogram of a's (above) should be fairly uniform."
        histogram_of_errors_2()
        print"The histogram of sample errors (above) should be clustered at edges for success."
        print"Time for Phase 3: ", timer.stop()
        timer.start()
        print"***** PHASE 4: ATTACK ALGORITHM"
        secret_mod_q()
        result = alg2(10000,quickflag)
        print"Result of Algorithm 2:", result
        print"Time for Phase 4: ", timer.stop()
        if result == True:
            count_successes = count_successes + 1
            print"*-*-*-*-*-*-*-*-*-*-*-*-*-*-* ", count_successes," out of ", trialnum+1," successes so far. *-*-*-*-*-*"
    totaltime = timer2.stop()
    print"Total time for ", trialnum+1," trials was ", totaltime
    return count_successes

```

## A.2 Sage Code for Algorithm 3

The following Sage Mathematics Software [S] algorithm returns the largest prime  $q$  for which a polynomial  $f$  has a root of order  $m$  modulo  $q$ .

```

x = PolynomialRing(RationalField(), 'x').gen()
def findq(f,m):
    g = x^m-1
    xg = f.xgcd(g)
    cofs = xg[2].coefficients()
    dens = [ a.denominator() for a in cofs ]
    facts = lcm(dens).factor()
    return max([fac[0] for fac in facts ])

```

## References

- IEEE. P1363.1: Standard Specifications for Public-Key Cryptographic Techniques Based on Hard Problems over Lattices, December 2008. <http://grouper.ieee.org/groups/1363/>
- BCNS. Bos, J.W., Costello, C., Naehrig, M., Stebila, D.: Post-quantum key exchange for the TLS protocol from the ring learning with errors problem. In: 36th IEEE Symposium on Security and Privacy 2015 (2015). <http://eprint.iacr.org/2014/599.pdf>
- BLN. Bos, J.W., Lauter, K., Naehrig, M.: Private predictive analysis on encrypted medical data. *J. Biomed. Inform.* **54**, 234–243 (2014)
- BL+. Brakerski, Z., Langlois, A., Peikert, C., Regev, O., Stehlé, D.: Classical hardness of learning with errors. In: *STOC 2013 Proceedings of the 2013 ACM Symposium on Theory of Computing*, pp. 575–584. ACM, New York (2013)
- BV. Brakerski, Z., Vaikuntanathan, V.: Fully homomorphic encryption from ring-LWE and security for key dependent messages. In: Rogaway, P. (ed.) *CRYPTO 2011*. LNCS, vol. 6841, pp. 505–524. Springer, Heidelberg (2011)
- BGV. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: Fully homomorphic encryption without bootstrapping. *ACM Trans. Comput. Theor.* **6**(3), 36 (2014). Article No 13
- DD. Ducas, L., Durmus, A.: Ring-LWE in polynomial rings. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) *PKC 2012*. LNCS, vol. 7293, pp. 34–51. Springer, Heidelberg (2012)
- EHL. Eisenträger, K., Hallgren, S., Lauter, K.: Weak instances of PLWE. In: Joux, A., Youssef, A. (eds.) *SAC 2014*. LNCS, vol. 8781, pp. 183–194. Springer, Heidelberg (2014)
- G. Gassert, T.A.: Prime decomposition in iterated towers and discriminant formulae. Ph.D. thesis, University of Massachusetts, Amherst (2014)
- GF+. Göttert, N., Feller, T., Schneider, M., Buchmann, J., Huss, S.: On the design of hardware building blocks for modern lattice-based encryption schemes. In: Prouff, E., Schaumont, P. (eds.) *CHES 2012*. LNCS, vol. 7428, pp. 512–529. Springer, Heidelberg (2012)
- GHS. Gentry, C., Halevi, S., Smart, N.P.: Fully homomorphic encryption with polylog overhead. In: Pointcheval, D., Johansson, T. (eds.) *EUROCRYPT 2012*. LNCS, vol. 7237, pp. 465–482. Springer, Heidelberg (2012)
- GLN. Graepel, T., Lauter, K., Naehrig, M.: ML confidential: machine learning on encrypted data. In: Kwon, T., Lee, M.-K., Kwon, D. (eds.) *ICISC 2012*. LNCS, vol. 7839, pp. 1–21. Springer, Heidelberg (2013)
- HPS. 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)

- K. Kedlaya, K.: A construction of polynomials with squarefree discriminants. *Proc. Am. Math. Soc.* **140**, 3025–3033 (2012)
- LP. Lindner, R., Peikert, C.: Better key sizes (and attacks) for LWE-based encryption. In: Kiayias, A. (ed.) *CT-RSA 2011*. LNCS, vol. 6558, pp. 319–339. Springer, Heidelberg (2011)
- LPR. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Gilbert, H. (ed.) *EUROCRYPT 2010*. LNCS, vol. 6110, pp. 1–23. Springer, Heidelberg (2010)
- LPR13. Lyubashevsky, V., Peikert, C., Regev, O.: A toolkit for ring-LWE cryptography. In: Johansson, T., Nguyen, P.Q. (eds.) *EUROCRYPT 2013*. LNCS, vol. 7881, pp. 35–54. Springer, Heidelberg (2013)
- M. Masser, D.W.: 3136. The discriminants of special equations. *Math. Gaz.* **50**(372), 158–160 (1966)
- MR04. Micciancio, D., Regev, O.: Worst-case to average-case reductions based on Gaussian measure. *SIAM J. Comput.* **37**(1), 267–302 (2007). Preliminary version in FOCS 2004
- MR09. Micciancio, D., Regev, O.: Lattice-based cryptography. In: Bernstein, D.J., Buchmann, J., Dahmen, E. (eds.) *Post Quantum Cryptography*, pp. 147–191. Springer, Heidelberg (2009)
- PG. Pöppelmann, T., Güneysu, T.: Towards practical lattice-based public-key encryption on reconfigurable hardware. In: Lange, T., Lauter, K., Lisoněk, P. (eds.) *SAC 2013*. LNCS, vol. 8282, pp. 68–86. Springer, Heidelberg (2014)
- R. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. *J. ACM* **56**(6), 1–40 (2009). Preliminary version STOC 2005
- RV+. Roy, S.S., Vercauteren, F., Mentens, N., Chen, D.D., Verbauwhede, I.: Compact ring-LWE cryptoprocessor. In: Batina, L., Robshaw, M. (eds.) *CHES 2014*. LNCS, vol. 8731, pp. 371–391. Springer, Heidelberg (2014)
- S. Stein, W.A., et al.: Sage Mathematics Software (Version 6.4.1), The Sage Development Team (2014). <http://www.sagemath.org>
- SS. Stehlé, D., Steinfeld, R.: Making NTRUEncrypt and NTRUSign as secure as worst-case problems over ideal lattices. In: Paterson, K.G. (ed.) *EUROCRYPT 2011*. LNCS, vol. 6632, pp. 27–47. Springer, Heidelberg (2011)
- TV. Tao, T., Vu, V.: Smooth analysis of the condition number and the least singular value. *Math. Comput.* **79**(272), 2333–2352 (2010)

# **Cryptanalytic Insights**

# Links Among Impossible Differential, Integral and Zero Correlation Linear Cryptanalysis

Bing Sun<sup>1,3</sup>, Zhiqiang Liu<sup>2,3</sup>(✉), Vincent Rijmen<sup>3</sup>(✉), Ruilin Li<sup>4</sup>(✉),  
Lei Cheng<sup>1</sup>, Qingju Wang<sup>2,3</sup>, Hoda Alkhzaimi<sup>5</sup>, and Chao Li<sup>1</sup>

<sup>1</sup> College of Science, National University of Defense Technology, Changsha 410073, Hunan, China

happy\_come@163.com

<sup>2</sup> Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China

ilu\_zq@sjtu.edu.cn

<sup>3</sup> Department of Electrical Engineering (ESAT), KU Leuven and iMinds, Leuven, Belgium

vincent.rijmen@esat.kuleuven.be

<sup>4</sup> College of Electronic Science and Engineering, National University of Defense Technology, Changsha 410073, Hunan, China

securitylrl@163.com

<sup>5</sup> Technical University of Denmark, Kongens Lyngby, Denmark

**Abstract.** As two important cryptanalytic methods, impossible differential and integral cryptanalysis have attracted much attention in recent years. Although relations among other cryptanalytic approaches have been investigated, the link between these two methods has been missing. The motivation in this paper is to fix this gap and establish links between impossible differential cryptanalysis and integral cryptanalysis.

Firstly, by introducing the concept of structure and dual structure, we prove that  $a \rightarrow b$  is an impossible differential of a structure  $\mathcal{E}$  if and only if it is a zero correlation linear hull of the dual structure  $\mathcal{E}^\perp$ . Meanwhile, our proof shows that the automatic search tool presented by Wu and Wang could find all impossible differentials of both Feistel structures with SP-type round functions and SPN structures. Secondly, by establishing some boolean equations, we show that a zero correlation linear hull always indicates the existence of an integral distinguisher. With this observation we improve the number of rounds of integral distinguishers of Feistel structures, CAST-256, SMS4 and Camellia. Finally, we conclude that an  $r$ -round impossible differential of  $\mathcal{E}$  always leads to an  $r$ -round integral distinguisher of the dual structure  $\mathcal{E}^\perp$ . In the case that  $\mathcal{E}$  and  $\mathcal{E}^\perp$  are linearly equivalent, we derive a direct link between impossible differentials and integral distinguishers of  $\mathcal{E}$ .

Our results could help to classify different cryptanalytic tools and facilitate the task of evaluating security of block ciphers against various cryptanalytic approaches.

---

The work in this paper is supported by the National Natural Science Foundation of China (No: 61202371, 61402515, 61472250), and National Basic Research Program of China (973 Program) (2013CB338002, 2013CB338004).

© International Association for Cryptologic Research 2015

R. Gennaro and M. Robshaw (Eds.): CRYPTO 2015, Part I, LNCS 9215, pp. 95–115, 2015.

DOI: 10.1007/978-3-662-47989-6\_5

**Keywords:** Impossible differential · Integral · Zero correlation linear · Feistel · SPN · Camellia · CAST-256 · SMS4 · PRESENT · PRINCE · ARIA

## 1 Introduction

Block ciphers are considered vital elements in constructing many symmetric cryptographic schemes such as encryption algorithms, hash functions, authentication schemes and pseudo-random number generators. The core security of these schemes depends on the resistance of the underlying block ciphers to known cryptanalytic techniques. So far a variety of cryptanalytic techniques have been proposed such as impossible differential cryptanalysis [1, 2], integral cryptanalysis [3], zero correlation linear cryptanalysis [4], etc.

Impossible differential cryptanalysis was independently proposed by Knudsen [1] and Biham [2]. One of the most popular impossible differentials is called a truncated impossible differential. It is independent of the choices of the  $S$ -boxes. Several approaches have been proposed to derive truncated impossible differentials of a block cipher/structure effectively such as the  $\mathcal{U}$ -method [5],  $UID$ -method [6] and the extended tool of the former two methods generalized by Wu and Wang in Indocrypt 2012 [7]. Integral cryptanalysis [3] was first proposed by Knudsen and Wagner, and a number of these ideas have been exploited, such as square attack [8], saturation attack [9], multi-set attack [10], and higher order differential attack [11, 12]. With some special inputs, we check whether the sum of the corresponding ciphertexts is zero or not. Usually, we do not need to investigate the details of the  $S$ -boxes and only view the  $S$ -boxes as some bijective transformations over finite fields. Zero correlation linear cryptanalysis, proposed by Bogdanov and Rijmen in [4], tries to construct some linear hulls with correlation exactly zero. In most cases, as in impossible differential and integral cryptanalysis, we do not need to investigate the details of the  $S$ -boxes. Generally, though there has been lots of work concentrating on the design and cryptanalysis of  $S$ -boxes [13], most cryptanalytic results by using impossible differential, integral and zero correlation linear cryptanalysis are independent of the choices of the  $S$ -boxes. If we choose some other  $S$ -boxes in a cipher, the corresponding cryptanalytic results will remain almost the same.

Along with the growing of the list of cryptanalytic tools, the question whether there are direct links or any connections among different tools has drawn much attention of the cryptographic research community, since such relations can be used to compare the effectiveness of different tools as well as to improve cryptanalytic results on block ciphers.

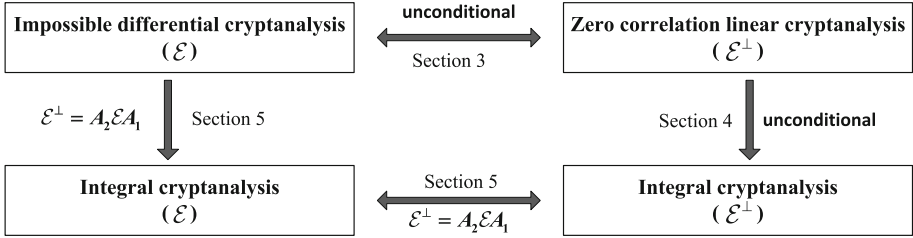
Efforts to find and build the links among different cryptanalytic techniques were initiated by Chabaud and Vaudenay in [14], where a theoretical link between differential and linear cryptanalysis was presented. After that, many attempts have been made to establish further relations among various cryptanalytic tools. In [15], Sun *et al.* proved that from an algebraic view, integral cryptanalysis can be seen as a special case of the interpolation attack. In [16], Leander stated that



statistical saturation distinguishers are averagely equivalent to multidimensional linear distinguishers. In [17], Bogdanov *et al.* showed that an integral implies a zero correlation linear hull unconditionally, a zero correlation linear hull indicates an integral distinguisher under certain conditions, and a zero correlation linear hull is actually a special case of multidimensional linear distinguishers. In [18], Blondeau and Nyberg further analyzed the link between differential and linear cryptanalysis and demonstrated some new insights on this link to make it more applicable in practice. They established new formulas between the probability of truncated differentials and the correlation of linear hulls. This link was later applied in [19] to provide an exact expression of the bias of a differential-linear approximation. Moreover, they claimed that the existence of a zero correlation linear hull is equivalent to the existence of an impossible differential in some specific cases [18]. As shown in [20], this link is usually not practical for most known impossible differential or zero correlation linear distinguishers, since the sum of the dimensions of input and output of each distinguisher is always the block size of the cipher, which means if the dimension parameter for one type is small, it will be infeasibly large for the other type. Blondeau *et al.* proposed a practical relation between these two distinguishers for Feistel-type and Skipjack-type ciphers and showed some equivalence between impossible differentials and zero correlation linear hulls with respect to Feistel-type and Skipjack-type ciphers [20]. In [21], Blondeau and Nyberg gave the link between truncated differential and multidimensional linear approximation, and then applied this link to explore the relations between the complexities of chosen-plaintext and known-plaintext distinguishing/key recovery attacks of differential and linear types. Moreover, they showed that statistical saturation cryptanalysis is indeed equivalent to truncated differential cryptanalysis, which could be used to estimate the data requirement of the statistical saturation key recovery attack.

**Contributions.** Although there have been intriguing results with respect to the relations among some important cryptanalytic approaches, the link between impossible differential cryptanalysis and integral cryptanalysis is still missing. In this paper, we aim to explore the link between these two cryptanalytic methods. Since the fundamental step in statistical cryptanalysis of block ciphers is to construct effective distinguishers, we focus on building the links among impossible differential, zero correlation linear and integral cryptanalysis from the aspect of distinguishers. Our main contributions are as follows (see Fig. 1).

1. We characterize what “being independent of the choices of  $S$ -boxes” means by proposing the definition of *structure*  $\mathcal{E}$ , which is a set containing some ciphers that are “similar” to each other. Then, by introducing the *dual structure*  $\mathcal{E}^\perp$ , we prove that  $a \rightarrow b$  is an impossible differential of  $\mathcal{E}$  if and only if it is a zero correlation linear hull of  $\mathcal{E}^\perp$ . More specifically, let  $P^T$  and  $P^{-1}$  denote the transpose and inverse of  $P$  respectively. Then for a Feistel structure with  $SP$ -type round functions where  $P$  is invertible, denoted as  $\mathcal{F}_{SP}$ , constructing an  $r$ -round zero correlation linear hull is equivalent to constructing an impossible differential of  $\mathcal{F}_{SP^T}$ , which is the same structure as  $\mathcal{F}_{SP}$  with  $P^T$  instead



**Fig. 1.** Links among Impossible Differential, Integral and Zero Correlation Linear Cryptanalysis, where  $\mathcal{E}$  is a structure and  $\mathcal{E}^\perp$  is the dual structure of  $\mathcal{E}$ ,  $A_1$  and  $A_2$  are linear transformations applied before the input and after the output of  $\mathcal{E}$ .

- of  $P$ ; For an SPN structure  $\mathcal{E}_{SP}$ , constructing an  $r$ -round zero correlation linear hull of  $\mathcal{E}_{SP}$  is equivalent to constructing an impossible differential of  $\mathcal{E}_{S(P^{-1})^T}$ , which is the same structure as  $\mathcal{E}_{SP}$  with  $(P^{-1})^T$  instead of  $P$ . Based on this result, we find 8-round zero correlation linear hulls of Camellia without  $FL/FL^{-1}$  layer and 4-round zero correlation linear hulls of ARIA.
2. We show that the automatic search tool, presented by Wu and Wang in Indocrypt 2012, could find all impossible differentials of a cipher that are independent of the choices of the  $S$ -boxes. This can be used in provable security of block ciphers against impossible differential cryptanalysis.
  3. We find that a zero correlation linear hull always implies the existence of an integral distinguisher, which means the conditions used for deriving integral distinguisher from zero correlation linear hull in [17] can be removed. Meanwhile, we observe that the statement “*integral unconditionally implies zero correlation linear hull*” in [17] is correct only under the definition that integral property is a balanced vectorial boolean function, while it does not hold for the general case. For example, up to date we cannot use the integral distinguisher for 4-round AES (with extra MixColumns) [4, 8] to construct a zero correlation linear hull.
  4. Following the results given above, we build the link between impossible differential cryptanalysis and integral cryptanalysis, i.e., an  $r$ -round impossible differential of a structure  $\mathcal{E}$  always implies the existence of an  $r$ -round integral distinguisher of  $\mathcal{E}^\perp$ . Moreover, in the case that  $\mathcal{E}^\perp = A_2\mathcal{E}A_1$  where  $A_1$  and  $A_2$  are linear transformations, we could get direct links between impossible differential cryptanalysis and integral cryptanalysis of  $\mathcal{E}$ . Specifically, an  $r$ -round impossible differential of SPN structure which adopts bit permutation as the linear layer, always leads to an  $r$ -round integral distinguisher.
  5. We improve the integrals of Feistel structures by 1 round, build a 24-round integral of CAST-256, present a 12-round integral of SMS4 which is 2-round longer than previously best known ones, and construct an 8-round integral for Camellia without  $FL/FL^{-1}$  layers. These distinguishers could not be obtained by the known methods for constructing integral distinguishers or by using the link given in [17]. As an example, the best known key recovery attack on reduced round CAST-256 in non-weak key model is given to show the effectiveness of the newly constructed distinguishers.

**Organization.** The remainder of this paper is organized as follows. Section 2 introduces the notations and concepts that will be used throughout the paper. In Sect. 3, we establish the new links between impossible differential and zero correlation linear cryptanalysis. Section 4 shows the refined link between integral and zero correlation linear cryptanalysis. The link between impossible differential and integral cryptanalysis is presented in Sect. 5. Then in Sect. 6, we give some examples to show the effectiveness of the newly established links in constructing new distinguishers of block ciphers. Finally, Sect. 7 concludes this paper.

## 2 Preliminaries

### 2.1 Boolean Functions

This section recalls the notations and concepts [22] which will be used throughout this paper. Let  $\mathbb{F}_2$  denote the finite field with two elements, and  $\mathbb{F}_2^n$  be the vector space over  $\mathbb{F}_2$  with dimension  $n$ . Let  $a = (a_1, \dots, a_n), b = (b_1, \dots, b_n) \in \mathbb{F}_2^n$ . Then

$$a \cdot b \triangleq a_1 b_1 \oplus \dots \oplus a_n b_n$$

denotes the *inner product* of  $a$  and  $b$ . Note that the inner product of  $a$  and  $b$  can be written as  $ab^T$  where  $b^T$  stands for the *transpose* of  $b$  and the multiplication is defined as matrix multiplication. Given a function  $G : \mathbb{F}_2^n \rightarrow \mathbb{F}_2$ , the *correlation* of  $G$  is defined by

$$c(G(x)) \triangleq \frac{\#\{x \in \mathbb{F}_2^n | G(x) = 0\} - \#\{x \in \mathbb{F}_2^n | G(x) = 1\}}{2^n} = \frac{1}{2^n} \sum_{x \in \mathbb{F}_2^n} (-1)^{G(x)}.$$

Given a vectorial function  $H : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^k$ , the *correlation* of the linear approximation for a  $k$ -bit output mask  $b$  and an  $n$ -bit input mask  $a$  is defined by

$$c(a \cdot x \oplus b \cdot H(x)) \triangleq \frac{1}{2^n} \sum_{x \in \mathbb{F}_2^n} (-1)^{a \cdot x \oplus b \cdot H(x)}.$$

If  $c(a \cdot x \oplus b \cdot H(x)) = 0$ , then  $a \rightarrow b$  is called a *zero correlation linear hull* of  $H$  [4]. This definition can be extended as follows: Let  $A \subseteq \mathbb{F}_2^n, B \subseteq \mathbb{F}_2^k$ . If for all  $a \in A, b \in B, c(a \cdot x \oplus b \cdot H(x)) = 0$ , then  $A \rightarrow B$  is called a *zero correlation linear hull* of  $H$ . In the case that  $H$  is a permutation on  $\mathbb{F}_2^n$ , for any  $b \neq 0, c(b \cdot H(x)) = 0$  and for any  $a \neq 0, c(a \cdot x) = 0$ . We call  $0 \rightarrow b$  and  $a \rightarrow 0$  trivial zero correlation linear hulls of  $H$  where  $a \neq 0$  and  $b \neq 0$ . Let  $A \subseteq \mathbb{F}_2^n$ . If the size of the set

$$H_A^{-1}(y) \triangleq \{x \in A | H(x) = y\}$$

is independent of  $y \in \mathbb{F}_2^k$ , we say  $H$  is *balanced on  $A$* . Specifically, if  $A = \mathbb{F}_2^n$ , we say  $H$  is a *balanced function*. If the sum of all images of  $H$  is 0, i.e.

$$\sum_{x \in \mathbb{F}_2^n} H(x) = 0,$$

we say  $H$  has an *integral-balanced (zero-sum)* property [3]. Let  $\delta \in \mathbb{F}_2^n$  and  $\Delta \in \mathbb{F}_2^k$ . The differential probability of  $\delta \rightarrow \Delta$  is defined as

$$p(\delta \rightarrow \Delta) \triangleq \frac{\#\{x \in \mathbb{F}_2^n | H(x) \oplus H(x \oplus \delta) = \Delta\}}{2^n}.$$

If  $p(\delta \rightarrow \Delta) = 0$ , then  $\delta \rightarrow \Delta$  is called an *impossible differential* of  $H$  [1, 2]. Let  $A \subseteq \mathbb{F}_2^n$ ,  $B \subseteq \mathbb{F}_2^k$ . If for all  $a \in A$  and  $b \in B$ ,  $p(a \rightarrow b) = 0$ ,  $A \rightarrow B$  is called an *impossible differential* of  $H$ . We recall the following property of balanced boolean functions: a function  $G : \mathbb{F}_2^n \rightarrow \mathbb{F}_2$  is balanced if and only if  $c(G(x)) = 0$ .

## 2.2 Block Ciphers

**Feistel Ciphers.** An  $r$ -round Feistel cipher  $E$  is defined as follows: Let  $(L_0, R_0) \in \mathbb{F}_2^{2n}$  be the input of  $E$ . Iterate the following transformation  $r$  times:

$$\begin{cases} L_{i+1} = F_i(L_i) \oplus R_i \\ R_{i+1} = L_i \end{cases} \quad 0 \leq i \leq r-1,$$

where  $L_i, R_i \in \mathbb{F}_2^n$ . The output of the  $r$ -th iteration is defined as the output of  $E$ . In this paper, we will focus on the case that  $F_i$ 's are SP-type functions which will be defined in the following.

**SPN Ciphers.** The SPN structure is widely used in constructing cryptographic primitives. It iterates some SP-type round functions to achieve confusion and diffusion. Specifically, the SP-type function  $f : \mathbb{F}_2^{s \times t} \rightarrow \mathbb{F}_2^{s \times t}$  used in this paper is defined as follows: Assume the input  $x$  is divided into  $t$  pieces  $x = (x_0, \dots, x_{t-1})$ , and each of the  $x_i$ 's is an  $s$ -bit word. Then apply the nonlinear transformation  $S_i$  to  $x_i$  and let  $y = (S_0(x_0), \dots, S_{t-1}(x_{t-1})) \in \mathbb{F}_2^{s \times t}$ . At last, apply a linear transformation  $P$  to  $y$ , and  $Py$  is the output of  $f$ .

The following strategies are popular in designing the diffusion layer  $P$  of a cipher:

- (1)  $P$  is a bit-wise permutation of  $\mathbb{F}_2^{s \times t}$  as in PRESENT [23]. PRESENT adopts bit permutation as the diffusion layer  $P$ , which can be defined as a permutation matrix  $P = (P_{i,j})_{64 \times 64}$ :

$$P_{i,j} = \begin{cases} 1 & \text{if } j = 16i \bmod 63 \\ 0 & \text{otherwise.} \end{cases}$$

- (2) Each bit of  $Py$  is a sum of some bits of  $y$  as in PRINCE [24]. Firstly, we will define  $SR$  and  $M'$  as follows:

$SR$  permutes the 16 nibbles, therefore it is a permutation of 64 bits and we could write  $SR$  as a permutation matrix in  $\mathbb{F}_2^{64 \times 64}$ .

To construct  $M'$ , we first define

$$\hat{M}^{(0)} = \begin{pmatrix} M_0 & M_1 & M_2 & M_3 \\ M_1 & M_2 & M_3 & M_0 \\ M_2 & M_3 & M_0 & M_1 \\ M_3 & M_0 & M_1 & M_2 \end{pmatrix}, \quad \hat{M}^{(1)} = \begin{pmatrix} M_1 & M_2 & M_3 & M_0 \\ M_2 & M_3 & M_0 & M_1 \\ M_3 & M_0 & M_1 & M_2 \\ M_0 & M_1 & M_2 & M_3 \end{pmatrix}$$

where

$$M_0 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, M_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, M_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, M_3 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix},$$

and then we define  $M' = \text{diag}(\hat{M}^{(0)}, \hat{M}^{(1)}, \hat{M}^{(1)}, \hat{M}^{(0)})$ , which is a  $64 \times 64$  block diagonal matrix.

$M'$  is used as the linear transformation of the middle round. The transformations  $M = SR \circ M'$  and  $M^{-1}$  are used before and after the middle round, respectively.

- (3) Each word of  $Py$  is a sum of some words of  $y$  as in Camellia [25] and ARIA [26]. The block cipher Camellia was recommended in the NESSIE block cipher portfolio in 2003 and selected as a new international standard by ISO/IEC in 2005. ARIA is a 128-bit block cipher established as a Korean Standard by the Ministry of Commerce, Industry and Energy in 2004. The linear transformations  $P_C$  and  $P_A$  of Camellia and ARIA could be written as follows:

$$P_C = \begin{pmatrix} E & 0 & E & E & 0 & E & E & E \\ E & E & 0 & E & E & 0 & E & E \\ E & E & E & 0 & E & E & 0 & E \\ 0 & E & E & E & E & E & E & 0 \\ E & E & 0 & 0 & 0 & E & E & E \\ 0 & E & E & 0 & E & 0 & E & E \\ 0 & 0 & E & E & E & E & 0 & E \\ E & 0 & 0 & E & E & E & E & 0 \end{pmatrix} \quad P_A = \begin{pmatrix} 0 & 0 & 0 & E & E & 0 & E & 0 & E & E & 0 & 0 & 0 & E & E & 0 \\ 0 & 0 & E & 0 & 0 & E & 0 & E & E & E & 0 & 0 & E & 0 & 0 & E \\ 0 & E & 0 & 0 & E & 0 & E & 0 & 0 & 0 & E & E & E & 0 & 0 & E \\ E & 0 & 0 & 0 & 0 & E & 0 & E & 0 & 0 & E & E & 0 & E & E & 0 \\ E & 0 & E & 0 & 0 & E & 0 & 0 & E & 0 & 0 & E & 0 & 0 & E & E \\ 0 & E & 0 & E & E & 0 & 0 & 0 & E & E & 0 & 0 & 0 & E & E \\ E & 0 & E & 0 & 0 & 0 & E & 0 & E & E & 0 & E & E & 0 & 0 & 0 \\ 0 & E & 0 & E & 0 & 0 & E & 0 & E & 0 & 0 & E & E & E & 0 & 0 \\ E & E & 0 & 0 & E & 0 & 0 & E & 0 & 0 & E & 0 & 0 & E & 0 & E \\ E & E & 0 & 0 & E & E & 0 & 0 & 0 & 0 & E & E & 0 & E & 0 & E \\ 0 & 0 & E & E & 0 & E & E & 0 & E & 0 & 0 & 0 & 0 & E & 0 & E \\ 0 & 0 & E & E & E & 0 & 0 & E & 0 & E & 0 & 0 & E & 0 & E & 0 \\ 0 & E & E & 0 & 0 & 0 & E & E & 0 & E & 0 & E & E & 0 & 0 & 0 \\ E & 0 & 0 & E & 0 & 0 & E & E & E & 0 & E & 0 & 0 & E & 0 & 0 \\ E & 0 & 0 & E & E & E & 0 & 0 & 0 & E & 0 & E & 0 & 0 & E & 0 \\ 0 & E & E & 0 & E & E & 0 & 0 & E & 0 & E & 0 & 0 & 0 & 0 & E \end{pmatrix}$$

where  $E$  and  $0$  denote  $8 \times 8$  identity and zero matrices, respectively.

- (4) Each word of  $Py$ , seen as an element of some extension fields of  $\mathbb{F}_2$ , is a linear combination of some other words of  $y$  as in the AES. In the following, we will use the matrix expression of finite fields to show how to write the linear layer of AES as a  $128 \times 128$  binary matrix:

Since ShiftRows is a permutation on 16 bytes, it is also a permutation on 128 bits. Therefore, as in the discussion above, we can represent ShiftRows as a permutation matrix  $M_{SR}$  in  $\mathbb{F}_2^{128 \times 128}$ . Let  $\mathbb{F}_{2^8} = \mathbb{F}_2[x] / \langle f(x) \rangle$  where  $\mathbb{F}_2[x]$  is the polynomial ring over  $\mathbb{F}_2$ ,  $f(x) = x^8 + x^4 + x^3 + x + 1 \in \mathbb{F}_2[x]$  is the defining polynomial of  $\mathbb{F}_{2^8}$ . Then  $1 = (00000001) \in \mathbb{F}_{2^8}$  can be written as the  $8 \times 8$  identity matrix  $E$ ,  $2 = (00000010) \in \mathbb{F}_{2^8}$  can be written as the following  $8 \times 8$  matrix:

$$M_2 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

and the matrix representation of  $3 = (00000011)$  is  $M_3 = E \oplus M_2$ . If we substitute 1, 2 and 3 in MixColumns by  $E$ ,  $M_2$  and  $M_3$ , respectively, we get a  $128 \times 128$  binary matrix  $M_{MC}$  and the linear layer of AES can be written as  $M_{MC}M_{SR}$  which is a  $128 \times 128$  matrix over  $\mathbb{F}_2$ .

Generally, no matter which linear transformation a cipher adopts, it is always linear over  $\mathbb{F}_2$ . Therefore,  $P$  can always be written as a multiplication by a matrix which leads to the following definition:

**Definition 1.** Let  $P$  be a linear transformation over  $\mathbb{F}_2^m$  for some positive integer  $m$ . The matrix representation of  $P$  over  $\mathbb{F}_2$  is called the primitive representation of  $P$ .

### 2.3 Structure and Dual Structure

In many cases, when constructing impossible differentials and zero correlation linear hulls, we are only interested in detecting whether there is a difference (mask) of an  $S$ -box or not, regardless of the value of this difference (mask). For example, the truncated impossible differential and zero correlation linear hull of AES in [4, 27] and Camellia in [28, 29]. In other words, if these ciphers adopt some other  $S$ -boxes, these distinguishers still hold. This leads to the following definition:

**Definition 2.** Let  $E : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$  be a block cipher with bijective  $S$ -boxes as the basic non-linear components.

- (1) A structure  $\mathcal{E}^E$  on  $\mathbb{F}_2^n$  is defined as a set of block ciphers  $E'$  which is exactly the same as  $E$  except that the  $S$ -boxes can take all possible bijective transformations on the corresponding domains.
- (2) Let  $a, b \in \mathbb{F}_2^n$ . If for any  $E' \in \mathcal{E}^E$ ,  $a \rightarrow b$  is an impossible differential (zero correlation linear hull) of  $E'$ ,  $a \rightarrow b$  is called an impossible differential (zero correlation linear hull) of  $\mathcal{E}^E$ .

**Note.** In the definition of  $\mathcal{E}^E$ , if  $E$  uses bijective  $S$ -boxes, then the  $S$ -boxes in  $\mathcal{E}^E$  should be bijective. However, if  $S$ -boxes used in  $E$  are not necessarily bijective, then  $\mathcal{E}^E$  could be defined as a set of block ciphers  $E'$  which is exactly the same as  $E$  except that the  $S$ -boxes can take all possible transformations on the corresponding domains. As discussed above, the truncated impossible differentials and zero correlation linear hulls of AES and Camellia found so far are actually the impossible differentials and zero correlation linear hulls of  $\mathcal{E}^{\text{AES}}$  and  $\mathcal{E}^{\text{Camellia}}$ .

**Definition 3.** Let  $\mathcal{F}_{SP}$  be a Feistel structure with  $SP$ -type round function, and let the primitive representation of the linear transformation be  $P$ . Let  $\sigma$  be the operation that exchanges the left and right halves of a state. Then the dual structure  $\mathcal{F}_{SP}^\perp$  of  $\mathcal{F}_{SP}$  is defined as  $\sigma \circ \mathcal{F}_{PTS} \circ \sigma$ .

Let  $\mathcal{E}_{SP}$  be an SPN structure with primitive representation of the linear transformation being  $P$ . Then the dual structure  $\mathcal{E}_{SP}^\perp$  of  $\mathcal{E}_{SP}$  is defined as  $\mathcal{E}_{S(P^{-1})^T}$ .

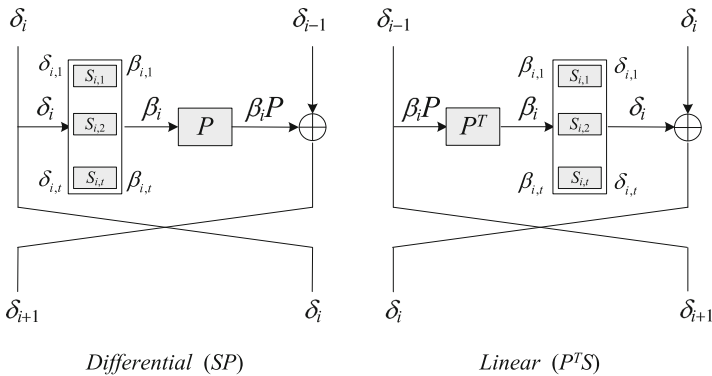
### 3 Links Between Impossible Differential and Zero Correlation Linear Cryptanalysis

In this section, we will show the equivalence between impossible differentials and zero correlation linear hulls of a structure, which will be used to establish the link between impossible differential and integral cryptanalysis in Sect. 5. The next theorem is stated without proof in [17].

**Theorem 1.**  $a \rightarrow b$  is an  $r$ -round impossible differential of  $\mathcal{F}_{SP}$  if and only if it is an  $r$ -round zero correlation linear hull of  $\mathcal{F}_{SP}^\perp$ .

*Proof.* The proof can be divided into the following two parts (See Fig. 2):

**Part (I).** We prove that for  $(\delta_0, \delta_1) \rightarrow (\delta_r, \delta_{r+1})$ , if one can find  $E \in \mathcal{F}_{SP}^\perp$  such that  $c((\delta_0, \delta_1) \cdot x \oplus (\delta_r, \delta_{r+1}) \cdot E(x)) \neq 0$ , then one can find  $E' \in \mathcal{F}_{SP}$  such that  $p((\delta_1, \delta_0) \rightarrow (\delta_{r+1}, \delta_r)) > 0$ .



**Fig. 2.** Differential Propagation of  $\mathcal{F}_{SP}$  and Linear Propagation of  $\mathcal{F}_{SP}^\perp$

Assume that  $(\delta_0, \delta_1) \rightarrow (\delta_r, \delta_{r+1})$  is a linear hull with non-zero correlation for some  $E \in \mathcal{F}_{SP}^\perp$ , and the input to the round function could be divided into  $t$  pieces, each of which is an  $s$ -bit word. Then there exists a linear characteristic with non-zero correlation:

$$(\delta_0, \delta_1) \rightarrow \cdots (\delta_{i-1}, \delta_i) \rightarrow \cdots \rightarrow (\delta_r, \delta_{r+1}),$$

where  $\delta_i \in (\mathbb{F}_2^s)^t$ . In this characteristic, the output mask of  $S_i = (S_{i,1}, \dots, S_{i,t})$  is  $\delta_i = (\delta_{i,1}, \dots, \delta_{i,t}) \in (\mathbb{F}_2^s)^t$ , and let the input mask of  $S_i$  be  $\beta_i = (\beta_{i,1}, \dots, \beta_{i,t}) \in (\mathbb{F}_2^s)^t$ . Since for  $\gamma \neq \beta_i P$ ,  $c(\gamma \cdot x \oplus \beta_i \cdot (xP^T)) = 0$ ,  $\delta_{i+1} = \delta_{i-1} \oplus \beta_i P$ .

In the following, for any  $(x_L, x_R) = (x_{L,1}, \dots, x_{L,t}, x_{R,1}, \dots, x_{R,t}) \in (\mathbb{F}_2^s)^t \times (\mathbb{F}_2^s)^t$ , we will construct an  $r$ -round cipher  $E_r \in \mathcal{F}_{SP}$ , such that  $E_r(x_L, x_R) \oplus E_r(x_L \oplus \delta_1, x_R \oplus \delta_0) = (\delta_{r+1}, \delta_r)$ .

If  $r = 1$ , for  $j \in \{1, \dots, t\}$ : if  $\delta_{1,j} = 0$ , we can define  $S_{1,j}$  as any possible transformation on  $\mathbb{F}_2^s$ , and if  $\delta_{1,j} \neq 0$ , we can define

$$S_{1,j}(x_{L,j}) = x_{L,j}, \quad S_{1,j}(x_{L,j} \oplus \delta_{1,j}) = x_{L,j} \oplus \beta_{1,j},$$

then for  $E_1 \in \mathcal{F}_{SP}$  which adopts such  $S$ -boxes,

$$E_1(x_L, x_R) \oplus E_1(x_L \oplus \delta_1, x_R \oplus \delta_0) = (\delta_0 \oplus \beta_1 P, \delta_1) = (\delta_2, \delta_1).$$

Suppose that we have constructed  $E_{r-1}$  such that  $E_{r-1}(x_L, x_R) \oplus E_{r-1}(x_L \oplus \delta_1, x_R \oplus \delta_0) = (\delta_r, \delta_{r-1})$ . Denote by  $(y_L, y_R) = (y_{L,1}, \dots, y_{L,t}, y_{R,1}, \dots, y_{R,t})$  the output of  $E_{r-1}(x_L, x_R)$ . Then in the  $r$ -th round, if  $\delta_{r,j} = 0$ , we can define  $S_{r,j}$  as any possible transformation on  $\mathbb{F}_2^s$ , otherwise, define  $S_{r,j}$  as follows:

$$S_{r,j}(y_{L,j}) = y_{L,j}, \quad S_{r,j}(y_{L,j} \oplus \delta_{r,j}) = y_{L,j} \oplus \beta_{r,j}.$$

Therefore  $E_r(x_L, x_R) \oplus E_r(x_L \oplus \delta_1, x_R \oplus \delta_0) = (\delta_{r-1} \oplus \beta_r P, \delta_r) = (\delta_{r+1}, \delta_r)$ .

**Part (II).** We prove that for  $(\delta_1, \delta_0) \rightarrow (\delta_{r+1}, \delta_r)$ , if one can find some  $E \in \mathcal{F}_{SP}$  such that  $p((\delta_1, \delta_0) \rightarrow (\delta_{r+1}, \delta_r)) > 0$ , one can find some  $E' \in \mathcal{F}_{SP}^\perp$  such that  $c((\delta_0, \delta_1) \cdot x \oplus (\delta_r, \delta_{r+1}) \cdot E'(x)) \neq 0$ .

Assume that  $(\delta_1, \delta_0) \rightarrow (\delta_{r+1}, \delta_r)$  is a differential of  $E \in \mathcal{F}_{SP}$ . Then there exists a differential characteristic with positive probability:

$$(\delta_1, \delta_0) \rightarrow \cdots (\delta_{i+1}, \delta_i) \rightarrow \cdots \rightarrow (\delta_{r+1}, \delta_r),$$

where  $\delta_i \in (\mathbb{F}_2^s)^t$ . In this characteristic, the input difference of  $S_i = (S_{i,1}, \dots, S_{i,t})$  is  $\delta_i = (\delta_{i,1}, \dots, \delta_{i,t}) \in (\mathbb{F}_2^s)^t$ , and let the output difference of  $S_i$  be  $\beta_i = (\beta_{i,1}, \dots, \beta_{i,t}) \in (\mathbb{F}_2^s)^t$ , then  $\delta_{i+1} = \delta_{i-1} \oplus (\beta_i P)$ .

Taking the following fact into consideration: for  $(\delta_{i,j}, \beta_{i,j})$ , where  $\delta_{i,j} \neq 0$ , there always exists an  $s \times s$  binary matrix  $M_{i,j}$  such that  $\beta_{i,j} = \delta_{i,j} M_{i,j}^T$ , then for  $S_{i,j}(x) = x M_{i,j}$ ,  $c(\beta_{i,j} \cdot x \oplus \delta_{i,j} \cdot S_{i,j}(x)) = 1$ .

Now we construct an  $r$ -round cipher  $E_r \in \mathcal{F}_{SP}^\perp$  such that  $c((\delta_0, \delta_1) \cdot x \oplus (\delta_r, \delta_{r+1}) \cdot E_r(x)) \neq 0$ . If  $r = 1$ , let  $S_{1,j}(x) = x M_{1,j}$  for  $\delta_{1,j} \neq 0$  and any linear transformation on  $\mathbb{F}_2^s$  otherwise. Then all operations in  $E_1 \in \mathcal{F}_{SP}^\perp$  are linear



over  $\mathbb{F}_2$ , which implies that there exists a  $2st \times 2st$  binary matrix  $M_1$  such that  $E_1(x) = xM_1$ , and

$$c((\delta_0, \delta_1) \cdot x \oplus (\delta_1, \delta_2) \cdot E_1(x)) = 1.$$

Assume that we have constructed  $E_{r-1}(x) = xM_{r-1}$  with  $M_{r-1}$  being a  $2st \times 2st$  binary matrix such that

$$c((\delta_0, \delta_1) \cdot x \oplus (\delta_{r-1}, \delta_r) \cdot E_{r-1}(x)) = 1,$$

and we can define  $S_{r,j}(x)$  in the  $r$ -th round similarly, then  $E_r(x) = xM_r$  for some  $2st \times 2st$  binary matrix  $M_r$ , and

$$c((\delta_0, \delta_1) \cdot x \oplus (\delta_r, \delta_{r+1}) \cdot E_r(x)) = 1,$$

which ends our proof.  $\square$

**Note.** In the proof of Theorem 1, the  $S$ -boxes we constructed are not necessarily bijective. If we add the bijective condition, Theorem 1 still holds. Since for a bijective  $S$ -box, if the correlation is non-zero,  $\delta_{1,j} \neq 0$  implies  $\beta_{1,j} \neq 0$ . Therefore, in Part(I) of the proof, we can further define  $S_{1,j}$  as

$$S_{1,j}(x) = \begin{cases} x_{L,j} \oplus \delta_{1,j} & x = x_{L,j} \oplus \beta_{1,j}, \\ x_{L,j} \oplus \beta_{1,j} & x = x_{L,j} \oplus \delta_{1,j}, \\ x & \text{others,} \end{cases}$$

and a similar definition can also be given to  $S_{r,j}$ . In this case, the  $S$ -boxes are invertible. Moreover, for a bijective  $S$ -box, if the differential probability is positive,  $\delta_{i,j} \neq 0$  implies  $\beta_{i,j} \neq 0$ , thus in Part (II) of the proof, we can always find a non-singular binary matrix  $M_{i,j}$  such that  $\beta_{i,j} = \delta_{i,j}M_{i,j}^T$ .

Similarly, we can prove the following theorem:

**Theorem 2.**  $a \rightarrow b$  is an  $r$ -round impossible differential of  $\mathcal{E}_{SP}$  if and only if it is an  $r$ -round zero correlation linear hull of  $\mathcal{E}_{SP}^\perp$ .

Definition 2 implies that the ‘‘impossibility’’ of an impossible differential of a structure can be caused only by a differential  $\delta_1 \rightarrow \delta_2$  where either  $\delta_1 = 0$  or  $\delta_2 = 0$  (but not both) over an invertible  $S$ -box, or by a differential  $0 \rightarrow \delta_2$  over a non-invertible  $S$ -box. Otherwise, according to the proof of Theorem 1, we can always find an  $S$ -box such that  $\delta_1 \rightarrow \delta_2$  is a possible differential. Therefore, we have the following corollary:

**Corollary 1.** The method presented in [7] finds all impossible differentials of  $\mathcal{F}_{SP}$  and  $\mathcal{E}_{SP}$ .

As a matter of fact, this corollary can be used in the provable security of block ciphers against impossible differential cryptanalysis, since with the help of this corollary, the longest impossible differentials of a given structure could be given.

In case  $P$  is invertible, according to the definition of equivalent structures given in [30], we have

$$\mathcal{F}_{P^T S} = ((P^T)^{-1}, (P^T)^{-1}) \mathcal{F}_{SP^T} (P^T, P^T), \tag{1}$$

which indicates:

**Corollary 2.** *Let  $\mathcal{F}_{SP}$  be a Feistel structure with SP-type round function, and let the primitive representation of the linear transformation be  $P$ . If  $P$  is invertible, finding zero correlation linear hulls of  $\mathcal{F}_{SP}$  is equivalent to finding impossible differentials of  $\mathcal{F}_{SP^T}$ .*

*Example 1. (8-Round Zero Correlation Linear Hull of Camellia Without FL/FL<sup>-1</sup>).* Let Camellia\* denote the cipher which is exactly the same as Camellia without  $FL/FL^{-1}$  layer except that  $P^T$  is used instead of  $P$ . Then we find that, for example:

$$((0, 0, 0, 0, 0, 0, 0, 0), (0, 0, 0, 0, a, 0, 0, 0)) \rightarrow ((0, 0, 0, 0, 0, 0, 0, h), (0, 0, 0, 0, 0, 0, 0, 0))$$

is an 8-round impossible differential of Camellia\*, where  $a$  and  $h$  denote any non-zero values. Therefore, we can derive an 8-round zero correlation linear distinguisher of Camellia without  $FL/FL^{-1}$  layer as shown below:

$$((a, a, 0, 0, a, 0, a, a), (0, 0, 0, 0, 0, 0, 0, 0)) \rightarrow ((0, 0, 0, 0, 0, 0, 0, 0), (h, 0, 0, h, 0, h, h, h)).$$

Furthermore, if  $\mathcal{F}_{SP} = \mathcal{F}_{SP^T}$  and  $\mathcal{E}_{SP} = \mathcal{E}_{S(P^{-1})^T}$ , we have:

**Corollary 3.** *For a Feistel structure  $\mathcal{F}_{SP}$  with SP-type round function, if  $P$  is invertible and  $P = P^T$ , there is a one-to-one correspondence between impossible differentials and zero correlation linear hulls.*

*For an SPN structure  $\mathcal{E}_{SP}$ , if  $P^T P = E$ ,  $a \rightarrow b$  is an impossible differential if and only if it is a zero correlation linear hull.*

*Example 2. (4-Round Zero Correlation Linear Hull of ARIA).* Since the linear layer  $P$  of ARIA satisfies  $P^T P = E$ , any impossible differential of  $\mathcal{E}^{\text{ARIA}}$  is automatically a zero correlation linear hull of  $\mathcal{E}^{\text{ARIA}}$ . Therefore, the impossible differentials of 4-round ARIA shown in [28] are also zero correlation linear hulls of 4-round ARIA.

## 4 Links Between Integral and Zero Correlation Linear Cryptanalysis

Firstly, we give two fundamental statements that give links between integral cryptanalysis and zero correlation linear cryptanalysis:

**Lemma 1.** *Let  $A$  be a subspace of  $\mathbb{F}_2^n$ ,  $A^\perp = \{x \in \mathbb{F}_2^n \mid a \cdot x = 0, a \in A\}$  be the dual space of  $A$  and  $F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$  be a function on  $\mathbb{F}_2^n$ . For any  $\lambda \in \mathbb{F}_2^n$ ,  $T_\lambda : A^\perp \rightarrow \mathbb{F}_2^n$  is defined as  $T_\lambda(x) = F(x \oplus \lambda)$ , then for any  $b \in \mathbb{F}_2^n$ ,*

$$\sum_{a \in A} (-1)^{a \cdot \lambda} c(a \cdot x \oplus b \cdot F(x)) = c(b \cdot T_\lambda(x)).$$

*Proof.*

$$\begin{aligned}
 \sum_{a \in A} (-1)^{a \cdot \lambda} c(a \cdot x \oplus b \cdot F(x)) &= \sum_{a \in A} (-1)^{a \cdot \lambda} \frac{1}{2^n} \sum_{x \in \mathbb{F}_2^n} (-1)^{a \cdot x \oplus b \cdot F(x)} \\
 &= \frac{1}{2^n} \sum_{x \in \mathbb{F}_2^n} (-1)^{b \cdot F(x)} \sum_{a \in A} (-1)^{a \cdot (\lambda \oplus x)} = \frac{1}{2^n} \sum_{x \in \mathbb{F}_2^n} (-1)^{b \cdot F(x)} |A| \delta_{A^\perp}(\lambda \oplus x) \\
 &= \frac{1}{|A^\perp|} \sum_{y \in A^\perp} (-1)^{b \cdot T_\lambda(y)} = c(b \cdot T_\lambda(x)),
 \end{aligned}$$

where  $\delta_{A^\perp}(x) = \begin{cases} 1 & x \in A^\perp \\ 0 & x \notin A^\perp. \end{cases}$  □

**Lemma 2.** *Let  $A$  be a subspace of  $\mathbb{F}_2^n$ ,  $F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ , and let  $T_\lambda : A^\perp \rightarrow \mathbb{F}_2^n$  be defined as  $T_\lambda(x) = F(x \oplus \lambda)$  where  $\lambda \in \mathbb{F}_2^n$ . Then for any  $b \in \mathbb{F}_2^n$ ,*

$$\frac{1}{2^n} \sum_{\lambda \in \mathbb{F}_2^n} (-1)^{b \cdot F(\lambda)} c(b \cdot T_\lambda(x)) = \sum_{a \in A} c^2(a \cdot x \oplus b \cdot F(x)).$$

The proof of Lemma 2 is given in the full version of this paper [31]. The conclusion of [17] that integral unconditionally implies zero correlation linear hull, is correct only under their definition of integral, which requires that  $c(b \cdot T_\lambda(x)) = 0$ . Under the original, more general definition for an integral distinguisher [3], this conclusion may not hold.

From Lemma 1, we can deduce the following:

**Corollary 4.** *Let  $F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$  be a function on  $\mathbb{F}_2^n$ , and let  $A$  be a subspace of  $\mathbb{F}_2^n$  and  $b \in \mathbb{F}_2^n \setminus \{0\}$ . Suppose that  $A \rightarrow b$  is a zero correlation linear hull of  $F$ , then for any  $\lambda \in \mathbb{F}_2^n$ ,  $b \cdot F(x \oplus \lambda)$  is balanced on  $A^\perp$ .*

This corollary states that if the input masks of a zero correlation linear hull form a subspace, then a zero correlation linear hull implies an integral distinguisher. Furthermore, the condition that input masks form a subspace can be removed, which leads to the following result:

**Theorem 3.** *A nontrivial zero correlation linear hull of a block cipher always implies the existence of an integral distinguisher.*

*Proof.* Assume that  $A \rightarrow B$  is a non-trivial zero correlation linear hull of a block cipher  $E$ . Then we can choose  $0 \neq a \in A, 0 \neq b \in B$ , such that  $\{0, a\} \rightarrow b$  is also a zero correlation linear hull of  $E$ .

Since  $V = \{0, a\}$  forms a subspace on  $\mathbb{F}_2$ , according to Corollary 4,  $b \cdot E(x)$  is balanced on  $V^\perp$ . This implies an integral distinguisher of  $E$ . □

Moreover, in the proof of Theorem 3, we can always assume that  $0 \in A$ . Then

1. If  $A$  forms a subspace, an integral distinguisher can be constructed from  $A \rightarrow b$ ;
2. If  $A$  does not form a subspace, we can choose some  $A_1 \subset A$  such that  $A_1$  forms a subspace, then an integral distinguisher can be constructed from  $A_1 \rightarrow b$ .

It was stated in [17] that a zero correlation linear hull indicates the existence of an integral distinguisher under certain conditions, while Theorem 3 shows that these conditions can be removed. This results in a more applicable link between zero correlation linear cryptanalysis and integral cryptanalysis.

It can be seen that Theorem 3 also gives us a new approach to find integral distinguishers of block ciphers. More specifically, an  $r$ -round zero correlation linear hull can be used to construct an  $r$ -round integral distinguisher.

## 5 Links Between Impossible Differential and Integral Cryptanalysis

According to the links given in the previous sections, we establish a link between impossible differential cryptanalysis and integral cryptanalysis:

**Theorem 4.** *Let  $\mathcal{E} \in \{\mathcal{F}_{SP}, \mathcal{E}_{SP}\}$ . Then an impossible differential of  $\mathcal{E}$  always implies the existence of an integral of  $\mathcal{E}^\perp$ .*

*Proof.* This can be deduced from the following facts:

- A zero correlation linear hull of  $\mathcal{E}^\perp$  always implies the existence of an integral of  $\mathcal{E}^\perp$ ;
- A zero correlation linear hull of  $\mathcal{E}^\perp$  could be constructed by constructing an impossible differential of  $\mathcal{E}$ .  $\square$

In case  $\mathcal{E}^\perp = A_2 \mathcal{E} A_1$  where  $A_1$  and  $A_2$  are linear transformations, we get the direct links between impossible differential and integral cryptanalysis:

**Corollary 5.** *Let  $\mathcal{F}_{SP}$  be a Feistel structure with SP-type round function, and let the primitive representation of the linear transformation be  $P$ . If  $P$  is invertible and there exists a permutation  $\pi$  on  $t$  elements such that for any  $(x_0, \dots, x_{t-1}) \in \mathbb{F}_2^{s \times t}$ ,*

$$P(x_0, \dots, x_{t-1}) = \pi^{-1} P^T \pi(x_0, \dots, x_{t-1}),$$

*then for  $\mathcal{F}_{SP}$ , an impossible differential always implies the existence of an integral distinguisher.*

*Example 3.* SNAKE(2) is a Feistel cipher proposed by Lee and Cha at JW-ISC'97, please refer to [32, 33] for details. According to [30], the round function

of SNAKE(2) can be seen as an SP-type one with the primitive presentation of the matrix being defined as

$$P = \begin{pmatrix} E & E & E & E \\ E & 0 & E & E \\ E & 0 & 0 & E \\ E & 0 & 0 & 0 \end{pmatrix},$$

where  $E$  and  $0$  are the identity and zero matrices of  $\mathbb{F}_2^{8 \times 8}$ , respectively. Let

$$\pi = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}.$$

Then we have  $P = \pi^{-1}P^T\pi$ , therefore, an impossible differential of SNAKE(2), which is independent of the details of the  $S$ -boxes, always implies the existence of an integral distinguisher of SNAKE(2).

**Corollary 6.** *Let  $\mathcal{E}_{SP}$  be an SPN structure with the primitive representation of the linear transformation being  $P$ . If  $P^T P = \text{diag}(Q_1, \dots, Q_t)$ , where  $Q_i \in \mathbb{F}_2^{s \times s}$ , then for  $\mathcal{E}_{SP}$ , an impossible differential always implies the existence of an integral distinguisher.*

*Proof.* Firstly, according to Theorem 4, if  $P^T P = E$ , an impossible differential of  $\mathcal{E}_{SP}$  always implies the existence of an integral.

Secondly, for the  $S$ -layer of  $\mathcal{E}_{SP}$ , if we substitute  $S$  by applying  $Q_i$  to the  $i$ -th  $S$ -box, according to definition 2, the structure stays identical. Since

$$P \circ (\text{diag}(Q_1, \dots, Q_t) \circ S) = (P \circ \text{diag}(Q_1, \dots, Q_t)) \circ S,$$

an SPN structure  $\mathcal{E}_{SP}$  is equivalent to an SPN structure  $\mathcal{E}_{S(P \circ \text{diag}(Q_1, \dots, Q_t))}$ .

Based on the above two points, we can get the conclusion.  $\square$

To show applications of these links, we recall that, an  $n \times n$  matrix  $P$  is called *orthogonal* if and only if  $P^T P = E$ , where  $E$  is the  $n \times n$  identity matrix.

*Example 4.* We can check that,  $SR$  and  $M'$  used in PRINCE are orthogonal matrices, therefore

$$M^T M = (SR \circ M')^T (SR \circ M') = E,$$

where  $E$  is the  $64 \times 64$  identity matrix. So all the linear layers used in different rounds of PRINCE are orthogonal based on which we could conclude that any  $r$ -round impossible differential of PRINCE which is independent of the choices of the  $S$ -boxes implies the existence of an  $r$ -round integral distinguisher.

*Example 5.* Since the linear layer  $P$  of ARIA is both symmetric and involutorial, e.g.  $P = P^{-1} = P^T$ , any impossible differential of ARIA which is independent of the choices of  $S$ -boxes implies the existence of an integral distinguisher.

*Example 6.* We can check that  $P$  used in PRESENT satisfies  $P = (P^{-1})^T$ , therefore, an impossible differential, which is independent of the details of the  $S$ -boxes, always leads to the existence of an integral distinguisher. In fact, since a permutation matrix  $P$  is always orthogonal, we have the following Corollary:

**Corollary 7.** *For an SPN structure which adopts bit permutation as the diffusion layer, the existence of an  $r$ -round impossible differential implies the existence of an  $r$ -round integral distinguisher.*

## 6 New Integrals for Block Ciphers/Structures

### 6.1 New Integrals for Feistel Structures

Let  $\mathcal{E}_r$  be an  $r$ -round Feistel structure  $\mathcal{F}_{SP}$ . Then for any  $a \neq 0, b \neq a, (a, 0) \rightarrow (0, b)$  is a zero correlation linear hull of  $\mathcal{E}_3$ ; and if the round functions are bijective, then for any  $a \neq 0, (a, 0) \rightarrow (0, a)$  is a zero correlation linear hull of  $\mathcal{E}_5$ .

So far the longest integral distinguisher known for a Feistel structure with bijective round functions counts 4 rounds, and the longest integral distinguisher for a Feistel structure with general round functions counts 2 rounds. We improve these distinguishers by 1 round using Theorem 3.

**Proposition 1.** *Let  $\mathcal{E}_r$  be an  $r$ -round Feistel structure defined on  $\mathbb{F}_2^{2n}$ . Then*

1. *If the  $F_i$ 's are bijective, then for any  $c \in \mathbb{F}_2^n, c \neq 0, c \cdot R_5$  is balanced on  $\{(0, 0), (c, 0)\}^\perp$  with respect to  $\mathcal{E}_5$ .*
2. *If the  $F_i$ 's are not necessarily bijective, then let  $\{\alpha_0, \dots, \alpha_{n-1}\}$  be a base of  $\mathbb{F}_2^n$  over  $\mathbb{F}_2$ . Then  $\alpha_{n-1} \cdot R_3$  is balanced on  $\{(0, \sum_{i=0}^{n-2} c_i \alpha_i) | c_i \in \mathbb{F}_2\}^\perp$  with respect to  $\mathcal{E}_3$ .*

As a matter of fact, for any  $c \in \mathbb{F}_2^n, c \neq 0, (c, 0) \rightarrow (0, c)$  is a zero correlation linear hull of  $\mathcal{E}_5$ . Thus according to Theorem 3, we can construct an integral distinguisher of  $\mathcal{E}_5$ , i.e., let  $(L_0, R_0)$  take all values in  $\{(0, 0), (c, 0)\}^\perp$ , then  $c \cdot R_5$  is balanced.

### 6.2 24-Round Integral for CAST-256

The block cipher CAST-256 was proposed as a first-round AES candidate, and we refer to [34] for details. Firstly, we recall the following zero correlation linear property given in [17].

*Property 1.*  $(0, 0, 0, L_1) \rightarrow (0, 0, 0, L_2)$  is a zero correlation linear hull of the 24-round CAST-256 (from the 13-th round to the 36-th round of CAST-256), where  $L_1 \neq 0, L_2 \neq 0$  and  $L_1 \neq L_2$ .

Let  $L_1^* = \{(l_1, l_2, \dots, l_{31}, 0) | l_i \in \mathbb{F}_2\}$  and  $L_2 = (0, \dots, 0, 1)$ . Then we obtain a zero correlation linear hull  $(0, 0, 0, L_1^*) \rightarrow (0, 0, 0, L_2)$  for the 24-round CAST-256. According to Theorem 3, we can get the following result:

**Proposition 2.** *Let  $V = \{(x_1, x_2, x_3, 0^{31}y) | x_i \in \mathbb{F}_2^{32}, y \in \mathbb{F}_2\}$ . If the input takes all values in  $V$ , and let the output of the 24-round be  $(C_0, C_1, C_2, C_3) \in \mathbb{F}_2^{32 \times 4}$  (from the 13-th round to 36-th round). Then  $(0, \dots, 0, 1) \cdot C_3$  is balanced.*

Based on this integral distinguisher, we present a key recovery attack on 28-round CAST-256 which is the best known attack on CAST-256 in the non-weak key model. The details of the attack are listed the full version of this paper [31].

### 6.3 12-Round Integral for SMS4

The SMS4 [35] block cipher is designed by the Chinese government as part of their WAPI standard for wireless networks. Up to date, the longest known integral distinguisher of SMS4 covers 10 rounds [36]. The details of SMS4 and the proof of the following propositions are listed in the full version of this paper [31].

**Proposition 3.** *Let  $V = \{v \in (\mathbb{F}_2^8)^4 | HW(v\mathcal{L}^T) = 1\}$ , where  $HW(x_1, x_2, x_3, x_4) = \#\{x_i \neq 0, i = 1, 2, 3, 4\}$ . For any  $d \in V$ ,  $(0, 0, 0, d) \rightarrow (d, 0, 0, 0)$  is a 12-round zero correlation linear hull of SMS4.*

**Proposition 4.** *Let  $V = \{v \in (\mathbb{F}_2^8)^4 | HW(v\mathcal{L}^T) = 1\}$ ,  $V_d = \{w \in (\mathbb{F}_2^{32})^4 | (0, 0, 0, d) \cdot w = 0\}$ , and let  $(c_0, c_1, c_2, c_3)$  be the output of 12-round SMS4. Then for any  $d \in V$ , when the input takes all possible values in  $V_d$ , we have*

$$\#\{d \cdot c_0 = 0\} = \#\{d \cdot c_0 = 1\}.$$

Note that most of the known integral distinguishers are independent of the choices of the  $S$ -boxes. However, the integral distinguisher presented above is highly related with the  $S$ -boxes, since for different  $S$ -boxes, we would find different zero correlation linear hulls which lead to different integral distinguishers of SMS4.

### 6.4 8-Round Integral for Camellia Without $FL/FL^{-1}$ Layer

Based on the 8-round zero correlation linear hull presented in Example 1, we get the following 8-round integral of Camellia without  $FL/FL^{-1}$  layer:

**Proposition 5.** *Let  $V$  be defined as*

$$V = \{((x_1, \dots, x_8), (x_9, \dots, x_{16})) | x_1 \oplus x_2 \oplus x_5 \oplus x_7 \oplus x_8 = 0, x_i \in \mathbb{F}_2^8\}.$$

*For any  $h \in \mathbb{F}_2^8$ ,  $h \neq 0$ ,  $(h, 0, 0, h, 0, h, h, h) \cdot R_{i+8}$  is balanced on  $V$  with respect to 8-round Camellia without  $FL/FL^{-1}$  layer.*

## 7 Conclusion

In this paper, we have investigated the link between impossible differential and integral cryptanalysis. To do this, we have introduced the concept of *structure*  $\mathcal{E}$  and *dual structure*  $\mathcal{E}^\perp$  and established the link in the following steps:

- We derived the relation between impossible differential of  $\mathcal{E}$  and zero correlation linear hull of  $\mathcal{E}^\perp$ . We have shown that for a Feistel structure  $\mathcal{F}_{SP}$  with  $SP$ -type round functions where  $P$  is invertible, constructing a zero correlation linear hull of  $\mathcal{F}_{SP}$  is equivalent to constructing an impossible differential of  $\mathcal{F}_{SP^T}$ , which is the same structure as  $\mathcal{F}_{SP}$  with  $P^T$  instead of  $P$ . For an SPN structure  $\mathcal{E}_{SP}$ , constructing a zero correlation linear hull of  $\mathcal{E}_{SP}$  is equivalent to constructing an impossible differential of  $\mathcal{E}_{S(P^{-1})^T}$ , which is the same structure as  $\mathcal{E}_{SP}$  with  $(P^{-1})^T$  instead of  $P$ .
- We presented the relation between zero correlation linear hull and integral distinguisher of block ciphers. As proven in Sect. 4, a zero correlation linear hull always implies the existence of an integral distinguisher, while such statement only holds under certain conditions in [17]. Meanwhile, we have observed that the statement “*integral unconditionally implies zero correlation linear hull*” in [17] is correct only under the definition that integral property is a balanced vectorial boolean function, while it does not hold for the general case (i.e., integral defined in [3] is a zero-sum property).
- We built the link between impossible differential of  $\mathcal{E}$  and integral distinguisher of  $\mathcal{E}^\perp$ . We have demonstrated that an  $r$ -round impossible differential of  $\mathcal{E}$  always leads to an  $r$ -round integral distinguisher of  $\mathcal{E}^\perp$ . In the case that  $\mathcal{E}$  and  $\mathcal{E}^\perp$  are linearly equivalent, we obtained some direct links between impossible differential and integral distinguisher of  $\mathcal{E}$ . Specifically, an  $r$ -round impossible differential of an SPN structure, which adopts bit permutation as the linear layer, always indicates the existence of an  $r$ -round integral distinguisher.

The results and links presented in this paper not only allow to achieve a better understanding and classifying of impossible differential cryptanalysis, integral cryptanalysis and zero correlation linear cryptanalysis, but also provide some new insights with respect to these cryptanalytic approaches as shown below:

- The automatic search tool presented by Wu and Wang in Indocrypt 2012 finds all impossible differentials of both Feistel structures with  $SP$ -type round functions and SPN structures, which is useful in provable security of block ciphers against impossible differential cryptanalysis.
- Our statement “*zero correlation linear hull always implies the existence of an integral distinguisher*” provides a novel way for constructing integral distinguisher of block ciphers. With this observation, we have improved the integral of Feistel structures by 1 round, built a 24-round integral of CAST-256, proposed a 12-round integral of SMS4 which is 2-round longer than previously best known ones, and present an 8-round integral of Camellia without  $FL/FL^{-1}$  layers. These distinguishers could not be obtained by either the previously known methods for constructing integral distinguishers or by using



the link given in [17]. Moreover, we have presented the best known key recovery attack on CAST-256 in non-weak key model to show that the new links can also be used to improve cryptanalytic results of some concrete ciphers.

By using the matrix representation given in [37], the concept of dual structure can be extended to generalized Feistel structures, and we can get similar results for these structures. Furthermore, we have focused on the links among the distinguishers used in impossible differential, integral and zero correlation linear cryptanalysis since distinguishers are the essential points in the evaluation of security margins of a block cipher against various cryptanalytic tools, and our results can be helpful in designing a block cipher from this point of view.

## References

1. Knudsen, L.R.: DEAL – A 128-bit Block Cipher. Department of Informatics, University of Bergen, Norway. Technical report (1998)
2. Biham, E., Biryukov, A., Shamir, A.: Cryptanalysis of skipjack reduced to 31 rounds using impossible differentials. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 12–23. Springer, Heidelberg (1999)
3. Knudsen, L.R., Wagner, D.: Integral cryptanalysis. In: Daemen, J., Rijmen, V. (eds.) FSE 2002. LNCS, vol. 2365, pp. 112–127. Springer, Heidelberg (2002)
4. Bogdanov, A., Rijmen, V.: Linear hulls with correlation zero and linear cryptanalysis of block ciphers. *Des. Codes Crypt.* **70**(3), 369–383 (2014)
5. Kim, J., Hong, S., Lim, J.: Impossible differential cryptanalysis using matrix method. *Discrete Math.* **310**(5), 988–1002 (2010)
6. Luo, Y., Lai, X., Wu, Z., Gong, G.: A unified method for finding impossible differentials of block cipher structures. *Inf. Sci.* **263**(1), 211–220 (2014)
7. Wu, S., Wang, M.: Automatic search of truncated impossible differentials for word-oriented block ciphers. In: Galbraith, S., Nandi, M. (eds.) INDOCRYPT 2012. LNCS, vol. 7668, pp. 283–302. Springer, Heidelberg (2012)
8. Daemen, J., Knudsen, L.R., Rijmen, V.: The block cipher SQUARE. In: Biham, E. (ed.) FSE 1997. LNCS, vol. 1267, pp. 149–165. Springer, Heidelberg (1997)
9. Lucks, S.: The saturation attack - a bait for twofish. In: Matsui, M. (ed.) FSE 2001. LNCS, vol. 2355, pp. 1–15. Springer, Heidelberg (2002)
10. Biryukov, A., Shamir, A.: Structural cryptanalysis of SASAS. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 394–405. Springer, Heidelberg (2001)
11. Lai, X.: Higher order derivatives and differential cryptanalysis. In: Blahut, R.E., Costello Jr., D.J., Maurer, U., Mittelholzer, T. (eds.) *Communications and Cryptography: Two Sides of One Tapestry*, vol. 276, pp. 227–233. Springer, USA (1994)
12. Knudsen, L.R.: Truncated and higher order differentials. In: Preneel, Bart (ed.) FSE 1994. LNCS, vol. 1008, pp. 196–211. Springer, Heidelberg (1995)
13. Picek, S., Batina, L., Jakobović, D., Ege, B., Golub, M.: S-box, SET, match: a toolbox for S-box analysis. In: Naccache, D., Sauveron, D. (eds.) WISTP 2014. LNCS, vol. 8501, pp. 140–149. Springer, Heidelberg (2014)
14. Chabaud, F., Vaudenay, S.: Links between differential and linear cryptanalysis. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 356–365. Springer, Heidelberg (1995)

15. Sun, B., Li, R., Qu, L., Li, C.: SQUARE attack on block ciphers with low algebraic degree. *Sci. China Inf. Sci.* **53**(10), 1988–1995 (2010)
16. Leander, G.: On linear hulls, statistical saturation attacks, PRESENT and a cryptanalysis of PUFFIN. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 303–322. Springer, Heidelberg (2011)
17. Bogdanov, A., Leander, G., Nyberg, K., Wang, M.: Integral and multidimensional linear distinguishers with correlation zero. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 244–261. Springer, Heidelberg (2012)
18. Blondeau, C., Nyberg, K.: New links between differential and linear cryptanalysis. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 388–404. Springer, Heidelberg (2013)
19. Blondeau, C., Leander, G., Nyberg, K.: Differential-linear cryptanalysis revisited. In: Cid, C., Rechberger, C. (eds.) FSE 2014. LNCS, vol. 8540, pp. 411–430. Springer, Heidelberg (2015)
20. Blondeau, C., Bogdanov, A., Wang, M.: On the (In)equivalence of impossible differential and zero-correlation distinguishers for Feistel- and Skipjack-type ciphers. In: Boureanu, I., Owesarski, P., Vaudenay, S. (eds.) ACNS 2014. LNCS, vol. 8479, pp. 271–288. Springer, Heidelberg (2014)
21. Blondeau, C., Nyberg, K.: Links between truncated differential and multidimensional linear properties of block ciphers and underlying attack complexities. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 165–182. Springer, Heidelberg (2014)
22. Carlet, C.: Boolean Functions for Cryptography and Error Correcting Codes. Cambridge University Press, Cambridge (2006)
23. Bogdanov, A.A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M., Seurin, Y., Vikkelsoe, C.: PRESENT: an ultra-lightweight block cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 450–466. Springer, Heidelberg (2007)
24. Borghoff, J., Canteaut, A., Güneysu, T., Kavun, E.B., Knezevic, M., Knudsen, L.R., Leander, G., Nikov, V., Paar, C., Rechberger, C., Rombouts, P., Thomsen, S.S., Yalçın, T.: PRINCE – a low-latency block cipher for pervasive computing applications. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 208–225. Springer, Heidelberg (2012)
25. Aoki, K., Ichikawa, T., Kanda, M., Matsui, M., Moriai, S., Nakajima, J., Tokita, T.: *Camellia*: a 128-bit block cipher suitable for multiple platforms - design and analysis. In: Stinson, D.R., Tavares, S. (eds.) SAC 2000. LNCS, vol. 2012, pp. 39–56. Springer, Heidelberg (2001)
26. Kwon, D., et al.: New block cipher: ARIA. In: Lim, Jong-In, Lee, Dong-Hoon (eds.) ICISC 2003. LNCS, vol. 2971, pp. 432–445. Springer, Heidelberg (2004)
27. Mala, H., Dakhilalian, M., Rijmen, V., Modarres-Hashemi, M.: Improved impossible differential cryptanalysis of 7-round AES-128. In: Gong, G., Gupta, K.C. (eds.) INDOCRYPT 2010. LNCS, vol. 6498, pp. 282–291. Springer, Heidelberg (2010)
28. Wu, W., Zhang, W., Feng, D.: Impossible differential cryptanalysis of round-reduced ARIA and camellia. *J. Comput. Sci. Technol.* **22**(3), 449–456 (2007)
29. Bogdanov, A., Geng, H., Wang, M., Wen, L., Collard, B.: Zero-correlation linear cryptanalysis with FFT and improved attacks on ISO standards camellia and CLEFIA. In: Lange, T., Lauter, K., Lisoněk, P. (eds.) SAC 2013. LNCS, vol. 8282, pp. 306–323. Springer, Heidelberg (2014)
30. Lei, D., Chao, L., Feng, K.: New observation on camellia. In: Preneel, B., Tavares, S. (eds.) SAC 2005. LNCS, vol. 3897, pp. 51–64. Springer, Heidelberg (2006)

31. Sun, B., Liu, Z., Rijmen, V., Li, R., Cheng, L., Wang, Q., Alkhzaimi, H., Li, C.: Links among Impossible Differential, Integral and Zero Correlation Linear Cryptanalysis. <http://eprint.iacr.org/2015/181.pdf>
32. Lee, C., Cha, Y.: The block cipher: SNAKE with provable resistance against DC and LC attacks. In: Proceedings of 1997 Korea-Japan Joint Workshop on Information Security and Cryptology (JW-ISC 1997), pp. 3–17 (1997)
33. Moriai, S., Shimoyama, T., Kaneko, T.: Interpolation attacks of the block cipher: SNAKE. In: Knudsen, L.R. (ed.) FSE 1999. LNCS, vol. 1636, pp. 275–289. Springer, Heidelberg (1999)
34. First AES Candidate Conference. <http://csrc.nist.gov/archive/aes/round1/conf1/aes1conf.htm>
35. Specification of SMS4, Block Cipher for WLAN Products – SMS4 (in Chinese). <http://www.oscca.gov.cn/UpFile/200621016423197990.pdf>
36. Zhang, W., Su, B., Wu, W., Feng, D., Wu, C.: Extending higher-order integral: An efficient unified algorithm of constructing integral distinguishers for block ciphers. In: Bao, F., Samarati, P., Zhou, J. (eds.) ACNS 2012. LNCS, vol. 7341, pp. 117–134. Springer, Heidelberg (2012)
37. Berger, T.P., Minier, M., Thomas, G.: Extended generalized feistel networks using matrix representation. In: Lange, T., Lauter, K., Lisoněk, P. (eds.) SAC 2013. LNCS, vol. 8282, pp. 289–305. Springer, Heidelberg (2014)

# On Reverse-Engineering S-Boxes with Hidden Design Criteria or Structure

Alex Biryukov<sup>(✉)</sup> and Léo Perrin

University of Luxembourg, SnT, Walferdange, Luxembourg  
{alex.biryukov,leo.perrin}@uni.lu

**Abstract.** S-Boxes are the key components of many cryptographic primitives and designing them to improve resilience to attacks such as linear or differential cryptanalysis is well understood. In this paper, we investigate techniques that can be used to reverse-engineer S-box design and illustrate those by studying the S-Box  $F$  of the Skipjack block cipher whose design process so far remained secret. We first show that the linear properties of  $F$  are far from random and propose a design criteria, along with an algorithm which generates S-Boxes very similar to that of Skipjack. Then we consider more general S-box decomposition problems and propose new methods for decomposing S-Boxes built from arithmetic operations or as a Feistel Network of up to 5 rounds. Finally, we develop an S-box generating algorithm which can fix a large number of DDT entries to the values chosen by the designer. We demonstrate this algorithm by embedding images into the visual representation of S-box's DDT.

**Keywords:** S-box design criteria · Skipjack · Linearity · Functional decomposition problem · Efficient implementation

## 1 Introduction

Non-linearity in cryptographic primitives is usually provided by so-called S-Boxes, functions which map a few inputs bits to a few output bits and which are often specified as look-up tables. These have been a topic of intensive research since their properties are crucial for resilience of a cipher against differential [1–3] and linear [4, 5] attacks. Further, the structure or the method used to build the S-Box can provide other benefits.

Indeed, the structure of an S-Box can be leveraged for instance to improve the implementation of a primitive using it. The hash function Whirlpool [6] and the block ciphers Khazad [7], Fantomas, Robin [8] and Zorro [9] among others use  $8 \times 8$  bits S-Boxes built from smaller  $4 \times 4$  ones, since storing several  $4 \times 4$  permutations as tables of 16 4-bits nibbles is more memory efficient than storing one  $8 \times 8$  permutation as a table of 256 bytes. Except for implementation advantage, knowledge of the internal structure helps to produce more efficient masked implementations against side-channel attacks, a notable example here being the AES [10] with its algebraic S-box based on a power function.

In some cases the design process of an S-Box might be kept secret for the purpose of implementing white-box cryptography, as described e.g. in [11]. In this paper, Biryukov et al. describe a memory-hard white-box encryption scheme based on a Substitution-Permutation Network where the S-Boxes are very large and are built using a so-called ASASA or ASASASA structure where “A” denotes an affine layer and “S” a non-linear S-Box layer. Preventing an adversary from decomposing these S-Boxes into their “A” and “S” layers is at the core of the security claims for this scheme.

Moreover such memory-hard white-box implementations with hidden structure of components can be of use in crypto-currencies, for example in cases where an entity is interested in issuing a crypto-currency of its own. One of the dangers is that powerful adversaries may launch a 51 % attack taking control of the mining process. Memory hard S-Boxes with hidden structure can offer a distinct advantage in such setting since efficient implementation of the proof-of-work function may be kept secret by the owners of the currency.

Examples of algorithms for which the components are known but the rationale behind their choice is not (at least at the time of release), are the block ciphers designed by or with the help of the US National Security Agency (NSA), namely the DES [12], Skipjack [13], SIMON and SPECK [14] (the last two do not use S-Boxes though). Although the design criteria for the S-Boxes of DES were later released [15] they were kept secret for 20 years in order to hide the existence of differential cryptanalysis, a technique only known by IBM and NSA at the time. Skipjack also uses an S-Box, denoted  $F$ , which is a permutation of  $\{0, 1\}^8$ . However, nothing was known so far about how this S-Box was chosen.

*Our Contribution.* Different methods can be used to recover the hidden structure of an S-Box. We propose that a cryptanalyst follows the strategy given below to try and decompose an unknown S-Box  $S$ :

1. Draw the “Pollock” visual representation of the LAT and DDT of  $S$  (see Sect. 4).
2. Check whether the linear and differential properties of  $S$  are compatible with a random function/permutation (see Sect. 2).
3. Compute the signature  $\sigma(S)$  of  $S$ .
4. If  $\sigma(S)$  is even, you may:
  - (a) Try an attack on SASAS [16],
  - (b) Try to distinguish  $S$  from a Feistel Network with XOR, using the distinguishers in [17],
  - (c) If one of the Feistel Network distinguishers worked, run `DecomposeFeistel` ( $S, R, \oplus$ ) for an appropriate  $R$  (see Sect. 3.2).
5. Regardless of  $\sigma(S)$ , run `DecomposeFeistel`( $S, R, \boxplus$ ) for  $R \in [2, 5]$  (see Sect. 3.2).
6. Regardless of  $\sigma(S)$ , run `BreakArithmetic`( $S$ ) (see Sect. 3.1).

We study in Sect. 2 the seemingly average linear properties of  $F$ . After a careful investigation and despite the fact that these properties are not impressive, we show that the probability for a random permutation of  $\{0, 1\}^8$  to have linear

properties at least as good as those of  $F$  is negligible. This implies three things. First,  $F$  was not chosen uniformly at random. Second,  $F$  is very unlikely to have been picked among random candidates according to some criteria. Third, the method used to build it improved the linear properties. We also provide a candidate algorithm which can be used to generate S-Boxes with very similar differential and linear properties.

In Sect. 3 we consider a general problem of decomposition of an S-box with hidden structure and describe two algorithms which can be used to decompose S-Boxes based on: (a) multiple iterations of simple arithmetic operations (for ex. like those found in a typical microprocessor) and (b) Feistel Networks with up to five independent rounds. The first algorithm is an optimised tree-search and the second one involves a SAT-solver.

Finally, we show in Sect. 4 how visual representations of the difference distribution table (DDT) or the linear approximation table (LAT) of an S-Box can help a cryptographer to spot non-randomness at a glance. As a bonus, we present an algorithm which generates non-bijective S-Boxes such that large set of entries in their DDT are set according to the designer's choices. We illustrate it by embedding images in the visual representation of the S-Box's DDT.

## 2 Partially Reverse-Engineering the S-Box of Skipjack

### 2.1 Overview of the S-Box of Skipjack and Useful Definitions

Skipjack is a block cipher with a block size of 64 bits and key size of 80 bits. The interested reader may refer to the official specification [13] or to the best attack on the cipher [18], an impossible differential attack leveraging its particular round structure. Further analysis trying to discover the design criteria of Skipjack is given in [19, 20].

Skipjack's specification contains an  $8 \times 8$  bit bijective S-box which is called "F-Table" and which is given as a lookup table (we list it in the Appendix A). In order to study it we need to introduce the following concepts.

**Definition 1 (Permutations Set).** We denote  $\mathfrak{S}_{2^n}$  the set of all the permutations of  $\{0, 1\}^n$ .

**Definition 2 (Difference Distribution Table).** Let  $s : \{0, 1\}^n \rightarrow \{0, 1\}^n$  be a function. Its difference distribution table (DDT) is a  $2^n \times 2^n$  matrix where the number at line  $i$  and column  $j$  is

$$d_{i,j} = \#\{x \in \{0, 1\}^n \mid s(x \oplus i) \oplus s(x) = j\}.$$

The maximum coefficient in this table (minus the first line and column) is the differential uniformity of  $s$  which we denote  $\Delta(s)$ :  $\Delta(s) = \max_{i>0, j>0} (d_{i,j})$ .

Differential cryptanalysis relies on finding differential transitions with high probabilities, i.e. pairs  $(a, b)$  such that  $s(x \oplus a) \oplus s(x) = b$  has many solutions which is equivalent to  $d_{a,b}$  being high. Therefore, cryptographers usually attempt

to use S-Boxes  $s$  with as low a value of  $\Delta(s)$  as possible. A function differentially 2-uniform, the best possible, is called *Almost Perfect Nonlinear* (APN). The existence of APN permutations of  $GF(2^n)$  for even  $n$  was only proved recently by Browning<sup>1</sup> et al. [21] in the case  $n = 6$ , while the case  $n = 8$  and beyond still remains an open problem. Hence, the differential uniformity of the S-Boxes of the AES [10] and of most modern S-Box based ciphers is equal to 4.

The distribution of the coefficients in the DDT of Skipjack is summarized in Table 1 along with the theoretical distribution identified in [22] for a random permutation of  $GF(2^8)$ . As we can see it is differentially 12-uniform, the same as you would expect from a random permutation, which is surprising since minimizing the differential uniformity is usually one of the corner stones of provable resilience against differential attacks.

**Table 1.** Distribution of the coefficients in the DDT of  $F$ .

Coefficient	Number	Proportion (%) in $F$	Poisson(1/2) (%)
0	39104	60.14	60.65
2	20559	31.62	30.33
4	4855	7.467	7.582
6	686	1.055	1.264
8	69	0.106	0.158
10	5	0.008	0.016
12	2	0.003	0.002

We briefly mention the linear properties of  $F$  before studying them thoroughly in Sect. 2.2. In particular, we define the Linear Approximations Table of an S-Box.

**Definition 3 (Linear Approximations Table).** Let  $s : \{0, 1\}^n \rightarrow \{0, 1\}^n$  be a function. Its linear approximations table (LAT) is a  $2^n \times 2^n$  matrix where the number at line  $i$  and column  $j$  is

$$c_{i,j} = \#\{x \in \{0, 1\}^n \mid x \cdot i = s(x) \cdot j\} - 2^{n-1} = \frac{1}{2} \sum_{x \in \{0, 1\}^m} (-1)^{i \cdot x \oplus j \cdot s(x)}$$

with “ $\cdot$ ” denoting the scalar product. The maximum absolute value of the  $c_{i,j}$  is the linearity of  $s$ ,  $\Lambda(s)$ , where  $\Lambda(s) = \max_{i>0, j>0}(|c_{i,j}|)$ .

The quantity  $c_{i,j}$  has different names in the literature. It is called “*bias*” or “*Imbalance*” of the Boolean function  $x \mapsto i \cdot x \oplus j \cdot s(x)$  in, for example, [22]. In papers from the Boolean functions community, it is more often defined in terms of *Walsh Spectrum*, the Walsh Spectrum of a Boolean function being the

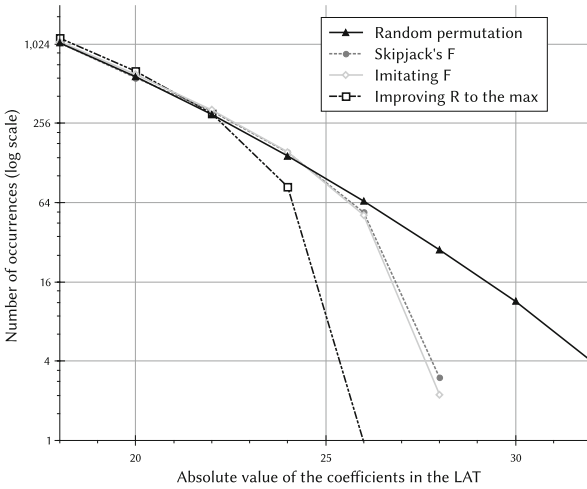
<sup>1</sup> The fact that Browning works at the NSA shows that this agency values theoretical considerations, which makes the simplicity of  $F$  all the stranger.

multiset  $\{c_{i,j}/2\}_{i \geq 0, j \geq 0}$ . The maximum coefficient in the LAT of  $F$  is  $\Lambda(F) = 28$  and it occurs in absolute value 3 times.

For the sake of completeness, we also give the sizes of the cycles in which  $F$  can be decomposed: 2, 10, 45, 68, 131.

### 2.2 The Linear Properties are Too Good to be True

Figure 1 contains the distribution of the value of the coefficients of the LAT (minus the first line and column) along with the theoretical proportions for a random permutation of  $GF(2^8)$  described below.



**Fig. 1.** Coefficients of the LAT of  $F$ , random permutations and some outputs of  $\text{Improve-}R(s)$ .

The probability distribution for the coefficients  $c_{i,j}$  in the LAT of a permutation of  $\mathfrak{S}_{2^n}$  is described in [23]:

$$P[c_{i,j} = 2z] = \frac{\binom{2^{n-1}}{2^{n-2}+z}^2}{\binom{2^n}{2^{n-1}}}.$$

Using Sect. 3.4 of [22], we derived that  $\Lambda(s)$  has a mean over all permutations  $s \in \mathfrak{S}_{2^8}$  of approximately 34.8 which is notably larger than for  $F$  since  $\Lambda(F) = 28$ .

Given the probability distribution of the coefficients of the LAT, it is easy to compute the probability that  $\Lambda(f) \leq 28$  assuming that  $f$  is a permutation chosen uniformly at random and that the coefficients' values correspond to independent sample of the same distribution. Note that there are only  $(2^8 - 1)^2$  such trials because the first line and column are ignored here.

$$P[\Lambda(f) \leq 28] = \left( \sum_{j=-14}^{14} P[c_{i,j} = 2j] \right)^{(2^8-1)^2} \approx 2^{-25.62}.$$



This probability is low but it would be feasible to generate a set of about  $2^{26}$  random permutations from  $\mathfrak{S}_{2^8}$  and compute the LAT for each of them. In such a set, the best S-Box  $s$  should verify  $\Lambda(s) = 28$ . However, we must also take into account that in order to resist linear cryptanalysis it is not only best to have a low maximum value, it is also better to have a low number of occurrences of it. In this regard,  $F$  and its only three occurrences of 28 could almost be considered as having a maximum value of 26 for which  $P[\Lambda(f) = 26] = 2^{-66.4}$ .

More rigorously, we compute the probability to have at most  $q$  coefficients equal to 28 in the LAT of a permutation picked uniformly at random from  $\mathfrak{S}_{2^8}$ . If we let  $p(2i) = P[c_{i,j} = 2i]$ , then this probability is equal to  $P_{28,q}$  where

$$P_{28,q} = \sum_{j=0}^q \left[ \binom{(2^8 - 1)^2}{j} (p(28) + p(-28))^j \left( \sum_{k=-13}^{13} p(2k) \right)^{(2^8 - 1)^2 - j} \right].$$

Unsurprisingly, we find that this probability is equal to  $2^{-66.4}$  for  $q = 0$ , i.e. the probability to have  $\Lambda(s) \leq 26$ . It also converges to  $2^{-25.6} = P[\Lambda(s) \leq 28]$  when  $q$  increases. For  $q = 3$ , the case of Skipjack's  $F$ , we find:

$$P_{28,3} = 2^{-54.4}.$$

The probability for a random permutation to have linear properties comparable to those of Skipjack's  $F$  is thus at most  $2^{-54.4}$ . Hence, we claim:

- $F$  was not chosen uniformly at random in  $\mathfrak{S}_{2^8}$ ,
- the designers of Skipjack did not generate many random permutation to then pick the best according to some criteria as they would need to have generated at least about  $2^{55}$  S-Boxes,
- the method used to build  $F$  improved its linear properties.

### 2.3 A Possible Design Criteria

We tried to create an algorithm capable of generating S-Boxes with linear and differential properties similar to those of  $F$ . It turns out that such an algorithm is easy to write. First, we introduce a quantity we denote  $R(f)$  and define as follows:

$$R(f) = \sum_{\ell \geq 0} N_\ell \cdot 2^\ell,$$

where  $N_\ell$  counts coefficients with absolute value  $\ell$  in the LAT of  $f$ :  $N_\ell = \#\{c_{i,j} \in (\text{LAT of } f), |c_{i,j}| = \ell\}$ .

Algorithm 1 starts from a random permutation  $s$  of  $\mathfrak{S}_{2^8}$  and returns a new permutation  $s'$  such that  $R(s') < R(s)$  and such that  $s'$  is identical to  $s$  except for two entries  $x$  and  $y$  which are swapped:  $s'(x) = s(y)$  and  $s'(y) = s(x)$ . It works by identifying one of the highest coefficient in the LAT, removing it through swapping two entries and checking whether  $R(s)$  was actually improved. This algorithm can be used in two different ways: either we keep iterating it until it

**Algorithm 1.** Improve- $R(s)$ 


---

```

 $c :=$  LAT of  $s$ 
Find  $a, b$  such that  $|c_{a,b}| = \Lambda(s)$ 
 $L :=$  empty list
for all  $x \in \{0, 1\}^s$  do
  if  $a \cdot x = b \cdot f(x)$  then
    Append  $x$  to  $L$ 
  end if
end for
for all  $(x, y) \in L^2, x \neq y$  do
   $s' = s$  ;  $s'(x) = s(y)$  ;  $s'(y) = s(x)$ 
  if  $R(s') < R(s)$  then
    return  $s'$ 
  end if
end for
return Fail

```

---

reaches a point at which no swap can improve  $R(s)$  or we stop as soon as  $R(s)$  is below an arbitrary threshold.

We implemented both variants. For the second one, we stop when  $R(s) < 10^{10}$  because  $R(F) \approx 10^{9.92}$ . We denote  $N_\ell$  the average number of coefficient with absolute value  $\ell$  in the LAT or the DDT of the S-Boxes obtained. For the LAT,  $\log_2(N_\ell)$  is given in Table 2 and in Fig. 1; for the DDT it is in Table 3. “Random” corresponds to the average over 200 S-Boxes picked uniformly at random in  $\mathfrak{S}_{2s}$ ; “ $F$ ” to the distribution for the S-Box of Skipjack; “ $F$ -like” to the average over 100 S-Boxes obtained using Improve- $R()$  and stopping when  $R(s) < 10^{10}$ ; “best” to the average over 100 S-Boxes obtained using Improve- $R()$  and stopping only when it fails.

**Table 2.** Distribution of  $\log_2(N_\ell)$  in the LAT of different S-Boxes.

$\ell$	Random	$F$	$F$ -like	best $R()$
20	9.164	9.147	9.230	9.311
22	8.220	8.308	8.336	8.247
24	7.173	7.267	7.280	6.400
26	6.041	5.755	5.688	0.000
28	4.826	1.585	1.157	-
30	3.506	-	-	-
32	2.146	-	-	-
34	0.664	-	-	-

Using Improve- $R()$  with an appropriate threshold allows us to create S-Boxes with both linear and differential properties very close to  $F$ . However, in order to achieve this, we need to choose a threshold value computed from  $F$

**Table 3.** Distribution of  $\log_2(N_\ell)$  in the DDT of different S-Boxes.

$\ell$	Random	$F$	$F$ -like	Best $R()$
0	15.265	15.246	15.250	15.227
2	14.270	14.327	14.314	14.380
4	12.277	12.245	12.257	12.210
6	9.693	9.422	9.492	9.126
8	6.701	6.109	6.198	5.265
10	3.374	2.322	2.287	0.714
12	-0.059	1.000	-1.786	-5.059
14	-4.059	-	-5.059	-

and which does not correspond to anything specific. In fact, to the best of our knowledge, the quantity  $R(s)$  does not have any particular importance unlike for instance the linearity  $\Lambda(s)$ . Still, replacing  $R(s)$  by the linearity  $\Lambda(s)$  or a pair  $(\Lambda(s), \#\{(i, j), c_{i,j} = \Lambda(s)\})$  yields S-Boxes which are very different from  $F$ . Such S-Boxes indeed have a value of  $N_{\Lambda(s)-2}$  much higher than in the random case, which is not the case for  $F$ .

While our definition of  $R(s)$  may seem arbitrary, it is the only one we could find that leads to linear properties similar to those of  $F$ . For instance it may have been tempting to base  $R(s)$  on the square of  $\ell$  which is used when computing the correlation potential of a linear trail, a quantity useful when looking for linear attacks. We would thus define  $R(s) = \sum_{\ell \geq 0} N_\ell \ell^2$ . However this quantity is worthless as an optimization criteria since it is constant: Parseval's equality on the Walsh spectrum of a Boolean function imposes that the sum of the  $(c_{i,j})^2$  over each column is equal to  $2^{2n-2}$ .

To conclude: we have found new non-random properties of the S-box of Skipjack which are improving its strength against linear cryptanalysis and we developed an algorithm which could be used to generate such S-boxes.

## 2.4 Public Information About the Design of Skipjack

The only information indirectly published by the NSA on Skipjack corresponds to an ‘‘Interim Report’’ [24] written by external cryptographers and it contains no information on the specifics of the design. The most relevant parts of this report as far as the S-Box is concerned are the following ones.

SKIPJACK was designed to be evaluatable [...]. In summary, SKIPJACK is based on some of NSA's best technology. Considerable care went into its design and evaluation in accordance with the care given to algorithms that protect classified data.

Furthermore, after the “leakage” of an alleged version of Skipjack to usenet<sup>2</sup>, Schneier replied with a detailed analysis of the cipher [26] which contained in particular the following quote indicating that the S-box was changed in August 1992.

The only other thing I found [through documents released under FOIA] was a SECRET memo. [...] The date is 25 August 1992. [...] [P]aragraph 1 reads:

1. (U) The enclosed Informal Technical Report revises the F-table in SKIPJACK
2. No other aspect of the algorithm is changed.

Note also that the first linear cryptanalysis of DES [4] had not been published yet in August 1992 when the F-Table was changed. Gilbert et al. suggested at CRYPTO’90 [27] to use linear equation to help with key guessing in differential attack to attack FEAL. This block cipher was later attacked at CRYPTO’91 [28] and EUROCRYPT’92 [29] using directly some linear equations involving plaintext, ciphertext and key bits. We can but speculate about a connection between these papers and the change of S-Box of Skipjack.

### 3 Algorithm Decomposing Particular Structures

A powerful tool able to discard quickly some possible structures for an S-Box is its *signature*, as shown in Lemma 1.

**Definition 4 (Permutation Signature).** *A permutation  $s$  of  $\{0, 1\}^n$  has an odd signature if and only if it can be decomposed into an odd number of transpositions, a transposition being a function permuting two elements of  $\{0, 1\}^n$ . Otherwise, its signature is even.*

*The signature of  $f \circ g$  is even if and only if  $f$  and  $g$  have the same signature.*

**Lemma 1.** *The following  $b \times b$  permutations always have an even signature:*

- Feistel Networks using XOR to combine the output of the Feistel function with the other branch,
- Substitution-Permutation Networks for which the diffusion layer is linear in  $GF(2)^b$  or can be decomposed into a sequence of permutations ignoring a fraction of the internal state.

*Proof.* Let  $b$  be the block size of the block ciphers considered. The proof for the case of Feistel Networks with XOR can be found in [30].

<sup>2</sup> An anonymous member of `sci.crypt` posted what they claimed to be Skipjack at a time when this algorithm was still classified [25]. Although the algorithm described, “S-1”, turned out to be different from Skipjack as we know it, it used similar notations — the S-Box is called “F-Table” — and the key-schedule leads to identical round keys being used every 5 rounds, just like in the actual Skipjack.

Let us look at substitution permutation networks. An S-Box layer consists in the parallel application of several invertible S-Boxes operating on  $n$  bits, with  $n$  dividing  $b$ . This operation can be seen as the successive application of the S-Box on each  $n$  bit block, one after another. Such an operation ignores  $2^{b-n}$  bits, meaning that its cycle decomposition consists in  $2^{b-n}$  replicas of the same set of cycles. Since  $2^{b-n}$  is even, the application of each S-Box is even; which in turn implies that the successive application of the S-Box on each block is even. More generally, any permutation which can be decomposed into a sequence of sub-permutations ignoring a fraction of the internal state is even. The fact that permutations linear in  $GF(2)^b$  are even is showed in the proof of Lemma 2 in [31].  $\square$

The restriction put on the diffusion layer of SPN's is usually not important, e.g. the diffusion layer of the AES fits the requirement. However, for small block sizes, it must be taken into account.

So far, we have proved that  $F$  has been built in contrast to being picked out of a set of random S-Boxes according to some criteria. The signature of  $F$  is odd so Lemma 1 implies that  $F$  cannot be a Feistel Network with XOR. The generic attack on the SASAS structure [16] fails on  $F$ , meaning that it is not a simple SPN either. Finally,  $F$  is not affine equivalent to a monomial of  $GF(2^n)$  like for instance the S-Box of the AES. Indeed, such functions have the same coefficients in the lines of their DDT, only the order is different. This observation lead to the definition of the differential spectrum by Blondeau et al. [32]. It also implies that, for a monomial, the number of coefficients equal to  $d$  in its DDT must divide  $2^n - 1$ . As it is not the case for  $F$ , we can also rule out this structure.

However, this is not sufficient to conclude that  $F$  does not have a particular structure. It could be based on simple operations such as rotations, addition modulo  $2^n$  and multiplication available in a typical microprocessor (thus offering the designer a benefit of memory-efficient implementation) or on a Feistel Network which uses modular addition to combine the output of the Feistel function with the other branch. We study these two possibilities in this section by first describing an algorithm capable of decomposing S-Boxes built from multiple simple arithmetic operations and then by presenting a new attack recovering all Feistel functions of a small Feistel Network of up to 5-rounds regardless of whether XOR or modular addition is used.

The purpose of the algorithms we present in this section can be linked to the more general *Functional Decomposition Problem (FDP)* tackled notably over two rounds in [33]. In this paper, Faugère et al. introduce a general algorithm capable of decomposing  $h = (h_1, \dots, h_u)$  into  $(f_1(g_1, \dots, g_n), \dots, f_u(g_1, \dots, g_n))$  where the  $h_i$ 's,  $f_i$ 's and  $g_i$ 's are polynomials of  $n$  variables. The time complexity of this algorithm (see Theorem 3 of [33]) is lower bounded by  $O(n^{3 \cdot (d_f d_g - 1)})$  where  $d_f$  (respectively  $d_g$ ) is the maximum algebraic degree of the  $f_i$ 's (respectively the  $g_i$ 's). Note that this lower bound on the time complexity is not tight. In fact, the ratio  $n/u$  of the number of input variables over the number of coordinates of  $h$  is also of importance, the lower being the better.

### 3.1 Iterated Simple Arithmetic Permutation

A plausible assumption for an efficient yet compact S-box design is that the S-box is constructed using a formula containing basic instructions available in the microprocessor. Indeed, a simple code:

```
for (i = 0; i < 3; i++) {
    y = a * (ROTL8((b * y) ^ c, d)) ^ e;
}
```

generates an S-box which may have a differential uniformity better than Skipjack's  $F$ 's for a proper choice of constants  $a, b, c, d$  and  $e$ .

We introduce **BreakArithmetic**( $s$ ), an optimized tree-search capable of recovering the simple operations used to create such an S-Box constructed as an arbitrary sequence of basic processor instructions. It is based on the following observation. Suppose that  $s = \phi_r \circ \dots \circ \phi_1$ , where the  $\phi_i$ 's are one of the following algebraic operations: constant XOR, constant addition modulo  $2^n$ , multiplication by a constant modulo  $2^n$  and bit rotation by a constant. Then  $s \circ \phi_1^{-1} = (\phi_r \circ \dots \circ \phi_1) \circ \phi_1^{-1} = \phi_r \circ \dots \circ \phi_2$ , meaning that  $s \circ \phi_1^{-1}$  is "less complex", "closer from the identity" than  $s$  itself. The aim of this algorithm is to peel of the  $\phi_i$ 's one after another by performing a tree-search among all possible simple operations which selects operations to consider first based on how closer they get us to the identity.

In order for this to work, we need to capture the concept of "distance to the identity" using an actual metric which can be implemented efficiently. We chose to base this metric on the DDT since it is less expensive to compute than the LAT<sup>3</sup>. We define the following metric:  $M(s) = \sum_{\ell \geq 2} N_\ell (\ell - 2)^2$ . Our tree-search privileges candidates  $\phi_1$  such that  $M(s \circ \phi_1^{-1})$  is closer from  $M(\text{Id})$ , where  $\text{Id}$  is the identity function.

Our implementation of this algorithm is for example capable of recovering the decomposition of  $s : x \mapsto \psi(\psi(\psi(x)))$  with  $\psi : x \mapsto 0xa7 \cdot ((3 \cdot x \oplus 0x53) \gg \gg 4) \oplus 0x8b$ . However, our algorithm could not find any such decomposition for Skipjack's  $F$  despite running for 96 hours on a CUDA computer with more than 1000 cores for fast computation of the DDT.

### 3.2 Decomposing Feistel Structures

Another possible structure for  $F$  which is compatible with its having an odd signature is a Feistel Network where the XOR is replaced by a modular addition. In this section, we describe an algorithm which uses a SAT-solver to recover the Feistel functions of small Feistel Networks which use either XOR or modular addition. We describe below the key idea of this attack, namely the encoding of the truth table of each Feistel function using Boolean variables and then how we can use this encoding to actually decompose a small Feistel Network.

<sup>3</sup> One can also notice that linear operations do not alter the DDT profile of the permutation and thus one has to recompute the metric only after non-linear operations.

Methods to distinguish Feistel Networks from random permutations have been actively investigated, notably in the work by Luby and Rackoff [34] as well as by Patarin [35,36]. Here, we present a method which goes beyond distinguishing: it actually recovers all the Feistel functions for up to 5-rounds of Feistel Networks with low branch width.

**Encoding of the Feistel Function.** Let  $f : \{0, 1\}^n \rightarrow \{0, 1\}^n$  be an unknown function. We associate to each of its output bits  $i$  on each possible input  $x$  a unique variable  $z_i^x$ . The truth-table of  $f$  is thus as shown in Table 4 for  $n = 3$ . We encode the fact that a vector of Boolean variables  $y_i, i \in [0, n - 1]$  is the output of  $f$  given input variables  $x_i, i \in [0, n - 1]$  using the truth-table of  $f$  by building a CNF<sup>4</sup> involving  $\{x_i\}_{i < n}, \{y_i\}_{i < n}$  and  $\{z_i^x\}_{i < n, x < 2^n}$  which is true if and only if  $(y_{n-1}, \dots, y_0) = f(x_{n-1}, \dots, x_0)$ .

**Table 4.** The variables used to encode an unknown function  $f : \{0, 1\}^3 \rightarrow \{0, 1\}^3$ , where  $(y_2, y_1, y_0) = f(x_2, x_1, x_0)$ .

$x_2$	$x_1$	$x_0$	$y_2$	$y_1$	$y_0$
0	0	0	$z_2^0$	$z_1^0$	$z_0^0$
0	0	1	$z_2^1$	$z_1^1$	$z_0^1$
...	...	...	...	...	...
1	1	1	$z_2^7$	$z_1^7$	$z_0^7$

We denote  $\text{bit}_i(b)$  the  $i$ -th of the binary expansion of any integer  $b < 2^n$  in little-endian notation so that  $b = \sum_{i < n} \text{bit}_i(n)2^{n-i}$ . We also denote  $a^1$  the variable  $a$  itself and  $a^0$  its negation. The procedure used to build this CNF is based on the following implication: if  $\{x_i\}_{i < n}$  corresponds to the binary expansion of an integer  $x < 2^n$  and  $\{y_i\}_{i < n}$  to the binary expansion of the integer  $y = f(x)$ , then  $y_i \oplus z_i^x = 0$  for all  $i < n$ . Using the notations we just introduced, this idea can be written as  $n$  implications, the conjunction of which for  $j < n$  must hold:

$$\left( \bigwedge_{i < n} x_i^{\text{bit}_i(x)} \right) \implies (y_j \oplus z_j^x = 0).$$

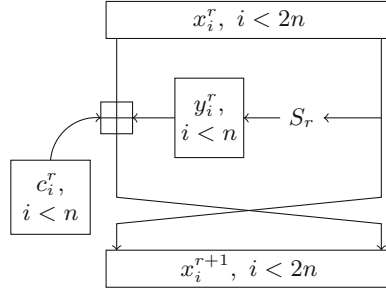
Each of these can be turned into a CNF made of two clauses using that  $(a \implies b) \equiv (a^0 \vee b^1)$ , that  $(a \oplus b = 0) \equiv ((a^1 \vee b^0) \wedge (a^0 \vee b^1))$  and basic linear algebra as follows:

$$\left( \left( \bigvee_{i < n} x_i^{1-\text{bit}_i(x)} \right) \vee y_j^1 \vee z_j^0 \right) \wedge \left( \left( \bigvee_{i < n} x_i^{1-\text{bit}_i(x)} \right) \vee y_j^0 \vee z_j^1 \right).$$

If we concatenate the CNF generated in this way for all values of  $x < 2^n$ , we obtain a CNF which we denote “CNF( $f, \{x_i\}, \{y_i\}$ )” with  $2n2^n$  clauses involving  $n2^n + 2n$  variables. It holds if and only if the assignment of the variables  $\{x_i\}_{i < n}$  and  $\{y_i\}_{i < n}$  is such that  $(y_{n-1}, \dots, y_0) = f(x_{n-1}, \dots, x_0)$ .

<sup>4</sup> A formula in *Conjunctive Normal Form* is the conjunction of multiple *clauses*, each of them being the disjunction of some possibly negated variables.

**Generating the Full CNF and Solving.** Using  $\text{CNF}(f, \{x_i\}, \{y_i\})$ , a SAT-solver and the full codebook of a S-Box  $S : \{0, 1\}^{2n} \rightarrow \{0, 1\}^{2n}$ , we can recover the Feistel functions used to generate  $S$  if it was indeed generated using a Feistel network or prove that it was not constructed in this fashion using  $\text{DecomposeFeistel}(S, R, \text{operation})$  (see Algorithm 2). To describe it, we introduce variables  $\{x_i^r\}_{i < 2n}$ ,  $\{y_i^r\}_{i < n}$  and, if the combining function is a modular addition instead of a XOR,  $\{c_i^r\}_{i < n}$  for  $r < R$  where  $R$  is the number of rounds we consider were used. These are summarized in Fig. 2.



**Fig. 2.** The variables used to encode round  $r$  of a Feistel Network operating on blocks of  $2n$  bits.

The general idea consists in building the CNF representation of the fact that  $S(p) = c$  for each input/output pair  $(p, c)$  separately, concatenate these CNF's and then have a SAT-solver solve the CNF obtained in this fashion. To each Feistel functions is associated a unique set of  $n2^p$  variables as described in the previous section. These are used when encoding that half of the internal state at round  $r + 1$  of the Feistel Network goes through the corresponding Feistel function. The only difficulty left is the combination of the left branch with the output of the Feistel function. In the case where a XOR is used, we can simply encode that  $x_i^{r+1} = y_i^r \oplus x_{i+n}^r$  separately for each bit  $i$ . However, in the case of a modular addition, we need to introduce a new set of variables for each evaluation of the addition corresponding to the carry bits:  $\{c_i^r\}_{i < n}$ . The addition is then encoded into a CNF using the CNF encoding of the following equations:

$$\begin{aligned} x_{i+1}^r &= c_i^r \oplus x_{i+n}^r \oplus y_i^r, \\ c_{i+1}^r &= (c_i^r \wedge x_{i+n}^r) \vee (y_i^r \wedge x_{i+n}^r) \vee (c_i^r \wedge y_i^r). \end{aligned}$$

A useful heuristic when trying to decompose more than 4 rounds is to look for decompositions with particular patterns in the sequence of the Feistel functions. For instance, decomposing a 5-rounds Feistel Network with round functions  $(S_a, S_b, S_c, S_d, S_a)$  is easier than decomposing a similar structure with round functions  $(S_a, S_b, S_c, S_d, S_e)$  if this knowledge is hard-coded in the CNF by using the same sets of variables to encode both  $S_e$  and  $S_a$ . In this case,



**Algorithm 2.** `DecomposeFeistel( $S, R, \text{operation}$ )`


---

```

 $C := \text{empty CNF}$ 
for all  $p \in [0, 2^n - 1]$  do
  for all  $r \in [0, R - 1]$  do
     $\{x_{i+n}^{p,r+1}\}_{i < n} = \{x_i^{p,r}\}_{i < n}$ 
     $C := \text{Concatenation of CNF}(S_r, \{x_i^{p,r}\}_{i < n}, \{y_i^{p,r}\})$  and  $C$ 
    if operation is  $\oplus$  then
      Append CNF repr. of  $\{x_i^{p,r+1}\}_{i < n} = \{y_i^{p,r}\}_{i < n} \oplus \{x_{i+n}^{p,r}\}_{i < n}$  to  $C$ 
    else
      Append CNF repr. of  $\{x_i^{p,r+1}\}_{i < n} = \{y_i^{p,r}\}_{i < n} \boxplus \{x_{i+n}^{p,r}\}_{i < n}$  to  $C$ 
    end if
  end for
  for all  $i \in [0, n - 1]$  do
    Append clause only made of literal  $(x_i^{p,0})^{\text{bit}_i(p)}$  to  $C$ 
    Append clause only made of literal  $(x_i^{p,R})^{\text{bit}_i(S(p))}$  to  $C$ 
  end for
end for
Run SAT-solver on  $C$ 
if  $C$  is satisfiable then
  Extract truth-table of all  $S_r$ 's from the variable assignment
  return "Feistel Network with  $R$  rounds"
else
  return "Not a Feistel Network with  $R$  rounds"
end if

```

---

`DecomposeFeistel( $S, R, \text{operation}$ )` also takes the assumed sequence of the S-Boxes as an additional input.

Another improvement comes from the observation that constants can be XOR-ed (or added/subtracted) in the input of Feistel functions in the first  $R - 2$  rounds — provided they are cancelled by XOR-ing (or adding/subtracting) in the later rounds — without changing the output of the function. Using this, we can arbitrarily decide that the first Feistel functions all map, say, 0 to 0. This simplification of the CNF helps the SAT-solver a lot and is actually necessary to attack 5 independent rounds.

We implemented Algorithm 2 and used the SAT-solver Minisat [37] to solve the CNF formula generated. The time taken to decompose S-Boxes actually made of small Feistel Networks is smaller than the time taken to discard an S-Box which is not based on such a structure. Decomposing  $8 \times 8$  S-Boxes built using 4-rounds Feistel Networks, regardless of whether  $\oplus$  or  $\boxplus$  is used, takes less than a second on a regular desktop PC<sup>5</sup> and discarding S-Boxes built in other ways requires about 5 seconds. Decomposing 5-rounds requires a bit less than a minute but discarding this structure takes longer, for instance 3 min to prove that  $F$  is not a 5-rounds  $\oplus$ -Feistel and 23 min to show that is it not a 5-rounds  $\boxplus$ -Feistel. It is also possible to attack larger instances provided enough RAM is

<sup>5</sup> The PC used for the experiments has a Intel(R) Core(TM) i7-3770 CPU (3.40GHz) for a cpu and 8 Go of RAM.

available. A 4-rounds Feistel Network corresponding to a  $14 \times 14$  S-Box can be broken in about 2 hours using up to about 38 Go of RAM<sup>6</sup>.

The CNF formulas equivalent to  $F$  being a Feistel Network with 3,4 or 5 rounds, using either  $\oplus$  or  $\boxplus$  are all unsatisfiable, meaning that  $F$  is not a Feistel Network with at most 5 rounds.

For the sake of completeness, we mention the existence of another time efficient attack on 5-round Feistel Networks by Gaëtan Leurent based on a boomerang-like property [38]. Indeed one of the open problems is how far crypt-analytic techniques can go in analysis of ciphers with small block, where the full code-book is available to the attacker.

## 4 From an S-Box to a Picture and Back Again

In order to distinguish an S-Box from a random one we propose a new method which we call *Pollock's Pattern Recognition*<sup>7</sup>. It is based on turning the DDT and the LAT of the S-Box into a picture and then use the natural pattern finding power of the human eye to identify not-random properties. We also describe a method to perform (partially) the inverse operation: *Seurat's Steganography*<sup>8</sup>. It creates an S-Box such that an image is embedded in the picture representation of its DDT.

### 4.1 Pollock's Pattern Recognition

As is clear from Sect. 2, the distribution of the coefficients in the LAT of an S-Box provides a powerful tool to distinguish a random-looking S-Box from a permutation chosen uniformly at random from the set of all permutations. We suggest here another method for looking at these coefficients which can also be applied to the DDT. The idea is to look at the whole table at once, be it a DDT or LAT, and then rely on the pattern matching capabilities of the pair human eye/human brain to possibly discard that the S-Box was chosen uniformly at random. In order to look at the whole table, we associate to the values of the coefficients different colors. Exactly which color scale to use is a question which can only be answered by trying different ones. As an illustration of the power of this method, we provide pictures allowing us to discard the randomness of 4 S-Boxes using merely a quick glance in Appendix B.

**Zorro.** The S-Box of this cipher [9] is based on a 4-rounds Feistel Network with a complex diffusion layer. As a consequence, the algorithm presented in

<sup>6</sup> This experiment was performed on a single core of a dedicated server with 500 Go of RAM.

<sup>7</sup> The pictures obtained in this fashion have a strong abstract feel to them, hence a name referring to the painter Jackson Pollock for this algorithm.

<sup>8</sup> As will be explained later, this algorithm works by drawing the image to embed point after point just like in a *pointillist* painting, hence the name of the painter who invented this method.

Sect. 3.2 fails on it. The picture representation of its LAT, given in Fig. 4a, contains “stripes”. These correspond to coefficients equal to 6 (orange) and 2 (green). These never appear for half of the input masks according to a repeating pattern. Such a behaviour is not expected from a random permutation. The color scheme was chosen so as to highlight this property. We note that the congruence modulo  $2^k$  for some  $k$  of the coefficients of the LAT is related to the algebraic degree of  $i \cdot x \oplus j \cdot S(x)$  as explained for example in [39] (Proposition 6.1).

**CLEFIA.** This block cipher [40] uses two distinct S-Boxes. The one denoted  $S_0$  has a particular structure based on smaller  $4 \times 4$  S-Boxes. The LAT of this S-Box is given in Fig. 4b: note the “dents” on the top and left side of the picture as well as the low number of colors compared to Fig. 4c which also depicts a LAT and uses the same color-scale. This low number of colors is a consequence of the fact that no coefficient in the LAT is congruent to 2 modulo 4 which in turn is related to this S-Box having an algebraic degree equal to 6 on all of its coordinates. Neither this nor the “dents” are expected from a random permutation.

**SAFER+.** This block cipher [41] uses an S-Box based on exponentiation in  $\mathbb{Z}/256\mathbb{Z}$ . Its LAT is given in Fig. 4c; note in particular the vertical lines which appear in this representation.

**Arithmetic.** The DDT can also be used in the same fashion. For example, we can look at the DDT of an S-Box generated using a simple algebraic expression similar to those discussed in Sect. 3.1, namely  $s : x \mapsto \psi(\psi(x))$  with  $\psi : x \mapsto 3 \cdot ((3 \cdot x \oplus 0x53) \gg \gg 4) \oplus 0x8b$ . The representation of its DDT is in Fig. 4d. Note the white rectangles corresponding to subsets of impossible differentials and the loose similarity between the top left and bottom right quadrants on one hand and the top right and bottom left quadrants on the other hand. None of these characteristics are expected from the DDT of a random permutation. Note that with 3 iterations of  $\phi$  this S-box becomes reasonably good.

We however were not able to spot any particular pattern in the Pollock representation of neither the DDT nor the LAT of Skipjack’s  $F$ . Such representations are given respectively in Figs. 3a and b in Appendix B. We used the function `matrix_plot` from the SAGE [42] software package to draw the Pollock representations.

## 4.2 Seurat’s Steganography

In this section, we present an algorithm allowing the creation of a non-bijective S-Box such that the picture representation of its DDT contains a particular image. Since we draw this image dot after dot like in *pointillism* and since it hides said image, we call the method we present below *Seurat’s Steganography*. The pictures we embed are black and white, the white parts corresponding to places where differentials are impossible and black parts to places where the differentials have non-zero probability.

**The Algorithm.** We define white and black equations as those giving the corresponding pixel color in the Pollock representation of the DDT of an S-Box.

**White Equations.**  $W_{a,b} : \forall x \in \{0, 1\}^m, S(x + a) + S(x) \neq b$ .

**Black Equations.**  $B_{a,b} : \exists x \in \{0, 1\}^m, S(x + a) + S(x) = b$ .

The inputs considered in this Section are:

$B$  The complete list of the black equations.

$T_w$  A table of booleans of size  $u \times v$  (the dimensions of the image) where  $T_w[a, b]$  is false if and only if the pixel at  $(a, b)$  cannot be white.

$S$  A partially unspecified S-Box such that all equations  $B_j$  for  $j < i$  hold and such that none of the  $W_j$  has a solution for any  $j$ .

$i$  The index of the equation in  $B$  for which we need to find a solution.

We first need a sub-routine checking if adding an entry  $S(x) = y$  to a partially assigned S-Box, i.e. an S-Box for which some of the outputs are unspecified, leads to at least one of the white equations not holding anymore. It is described in Algorithm 3.

---

**Algorithm 3.** `checkW( $S, x, y, T_w$ )`.

---

```

for all  $a \in \{0, 1\}^m$ , if  $S(x + a)$  is specified, do
  if  $T_w[a, S(x + a) + S(x)]$  is false then return false
end for
return true

```

---

We now describe Seurat's Steganography, namely Algorithm 4, which uses two lists of equations to iteratively build an S-Box such that a particular picture appears in its DDT. It works by first making a list  $L$  of all the ways entries could be added to the S-Box in order to satisfy the black equation  $B_i$ . If none are found, the function fails. The function is finally called recursively on the candidates found to look for a solution for the next equation. If no solution are found for the next equation, the function fails.

Some optimizations are possible. First of all, it is not necessary to write this algorithm using recursion. It is also not necessary to let  $L$  be as large as possible. In fact  $|L| \leq 2$  is sufficient, although  $|L| = 1$  does not work unless the picture is very simple. It is also possible to allow some noise by tweaking `CheckW( $S, x, y, T_w$ )` to return `true` with low probability for pairs  $(x, y)$  even if they blacken a white pixel.

Two outputs of this algorithm are presented in Appendix C: the S-Boxes are given along with the Pollock representation of their DDT which clearly show the pictures we chose to embed in them. The differential and linear properties of the S-Box described in Table 6 are close from what would be expected from a random function (differential uniformity of 14, linearity of 39), meaning that it could be used in a context where a  $8 \times 8$  random function would be sufficient.

---

**Algorithm 4.** Seurat( $S, B, T_w, i$ ).

---

```

 $\delta_{\text{in}}$  := input difference in  $B_i$ 
 $\delta_{\text{out}}$  := output difference in  $B_i$ 
 $L$  := empty list of S-Boxes
if  $B_i$  is already satisfied by  $S$  then
    Append  $S$  to  $L$  and return  $L$ 
end if
for all  $x \in \{0, 1\}^m$  do
    if  $S(x)$  is not defined but  $S(x + \delta_{\text{in}})$  is defined then
         $y = S(x + \delta_{\text{in}}) + \delta_{\text{out}}$ 
        if CheckW( $S, x, y, T_w$ ) then
             $S' = S$  ;  $S'(x) = y$ 
            Append  $S'$  to  $L$ 
        end if
    else if  $S(x + \delta_{\text{in}})$  is not defined either then
        for all  $y \in \{0, 1\}^n$  do
            if CheckW( $S, x, y, T_w$ ) and CheckW( $S, x + \delta_{\text{in}}, y + \delta_{\text{out}}, T_w$ ) then
                 $S' = S$  ;  $S'(x) = y$  ;  $S'(x + \delta_{\text{in}}) = y + \delta_{\text{out}}$ 
                Append  $S'$  to  $L$ 
            end if
        end for
    end if
end for
end for
If  $L$  is still empty then return Fail
for all  $S' \in L$  do
    If Seurat( $S', B, T_w, i + 1$ ) does not fail then return  $S'$ 
end for
return Fail

```

---

**Counting Possible S-Boxes.** Let  $S$  be a random function from  $\{0, 1\}^m$  to  $\{0, 1\}^n$ . Then  $W_{a,b}$  holds if and only if  $d_{a,b} = 0$ , which happens with probability  $P[d_{a,b} = 0] = \exp(-2^{m-n-1})$  because the coefficients in the DDT of a random function follow approximately a Poisson distribution with parameter  $1/2$  (see [22]). Hence, if we have  $b$  black equations,  $w$  white ones and if we consider that their having solutions are independent events, then the probability that an S-Box has the correct image at the center of its DDT is  $P_{\text{success}} = (\exp(-2^{m-n-1}))^w \times (1 - \exp(-2^{m-n-1}))^b$ . In the case where  $m = n$ , we use that  $\log_2(\exp(-1/2)) \approx -1.35$  and that  $\log_2(1 - \exp(-1/2)) \approx -0.72$  to approximate this probability by

$$P_{\text{success}} = 2^{-(0.72 \cdot w + 1.35 \cdot b)}.$$

As there are  $2^{n^2}$  possible  $n \times n$  S-Boxes, we expect to have very roughly the following amount of solutions:

$$N_{\text{Solutions}} = 2^{n^2 - (0.72 \cdot w + 1.35 \cdot b)}.$$

Therefore, we need  $0.72 \cdot w + 1.35 \cdot b < n2^n$  in order to have a non-empty set of S-Box with the image we want inside their DDT. Black pixels are about twice as expensive as white ones according to this model. However, in practice, it is only possible to build a S-Box such that its DDT contains a black square of size  $22 \times 22$  or a white one of size  $62 \times 62$  without any noise, meaning that black pixels are, from our algorithm's point of view, about 8 times more expensive. Stirling's equation gives an approximate number of  $2^{(n-1.44) \cdot 2^n}$  permutations of  $\{0, 1\}^n$ , so we need that  $0.72 \cdot w + 1.35 \cdot b < (n - 1.44)2^n$  in order for permutations with the correct black/white pixels to exist with non negligible probability. However, our algorithm will require significant changes in order to search for permutations.

Since our algorithm does not require the pixels to be organised inside a square, we can also use it to force white or black pixels to appear anywhere in the DDT of an S-Box. This could be used to place a sort of trapdoor by for instance ensuring that a truncated differential compatible with the general structure of a cipher is present. Another possible use could be to "sign" a S-Box: Alice would agree with Bob to generate a S-Box for him and tell him before hand where some black/white pixels will be. Bob can then check that those are placed as agreed.

## 5 Conclusion

Knowledge of the internal structure of an S-box gives clear advantages to the designer of a cipher in terms of efficient or side-channel resistant implementation. It is also crucial in the white-box or crypto-currency setting. Hiding the S-box's structure can be also a way to hide superior cryptanalysis techniques or trapdoors.

In this paper we have introduced several approaches and algorithms to decompose an S-Box with unknown structure and we illustrated them by studying the S-Box of the NSA's block cipher Skipjack. This allowed us to rule out some possible structure, and to prove that its linear properties are too unlikely to have happened at random. We also provided an algorithm capable of generating very similar S-Boxes (Table 5).

An open problem related to this work is the study of block ciphers with small block sizes: how far can cryptanalysis go given a whole codebook? How many rounds of small-block Feistel Network or SPN is it feasible to break?

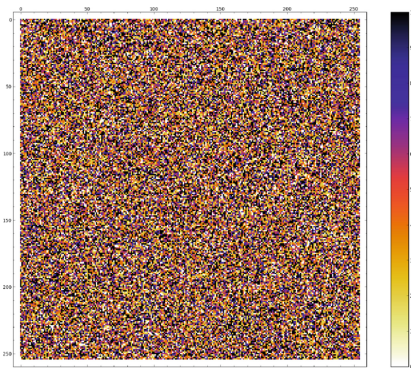
**Acknowledgement.** We thank the CRYPTO reviewers for their helpful comments. We also thank Anne Canteaut for pointing out the connection between algebraic degree and congruence of the coefficient of the LAT modulo  $2^k$ . The work of Léo Perrin is supported by the CORE ACRYPT project (ID C12-15-4009992) funded by the *Fonds National de la Recherche* (Luxembourg).

## A The S-Box of Skipjack

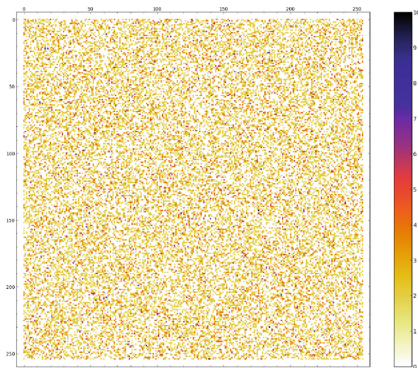
**Table 5.** Skipjack’s S-Box,  $F$ , in hexadecimal notation. For example,  $F(7a) = d6$ .

	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.a	.b	.c	.d	.e	.f
0.	a3	d7	09	83	f8	48	f6	f4	b3	21	15	78	99	b1	af	f9
1.	e7	2d	4d	8a	ce	4c	ca	2e	52	95	d9	1e	4e	38	44	28
2.	0a	df	02	a0	17	f1	60	68	12	b7	7a	c3	e9	fa	3d	53
3.	96	84	6b	ba	f2	63	9a	19	7c	ae	e5	f5	f7	16	6a	a2
4.	39	b6	7b	0f	c1	93	81	1b	ee	b4	1a	ea	d0	91	2f	b8
5.	55	b9	da	85	3f	41	bf	e0	5a	58	80	5f	66	0b	d8	90
6.	35	d5	c0	a7	33	06	65	69	45	00	94	56	6d	98	9b	76
7.	97	fc	b2	c2	b0	fe	db	20	e1	eb	d6	e4	dd	47	4a	1d
8.	42	ed	9e	6e	49	3c	cd	43	27	d2	07	d4	de	c7	67	18
9.	89	cb	30	1f	8d	c6	8f	aa	c8	74	dc	c9	5d	5c	31	a4
a.	70	88	61	2c	9f	0d	2b	87	50	82	54	64	26	7d	03	40
b.	34	4b	1c	73	d1	c4	fd	3b	cc	fb	7f	ab	e6	3e	5b	a5
c.	ad	04	23	9c	14	51	22	f0	29	79	71	7e	ff	8c	0e	e2
d.	0c	ef	bc	72	75	6f	37	a1	ec	d3	8e	62	8b	86	10	e8
e.	08	77	11	be	92	4f	24	c5	32	36	9d	cf	f3	a6	bb	ac
f.	5e	6c	a9	13	57	25	b5	e3	bd	a8	3a	01	05	59	2a	46

## B Picture Representation of the DDT and LAT of Some S-Boxes

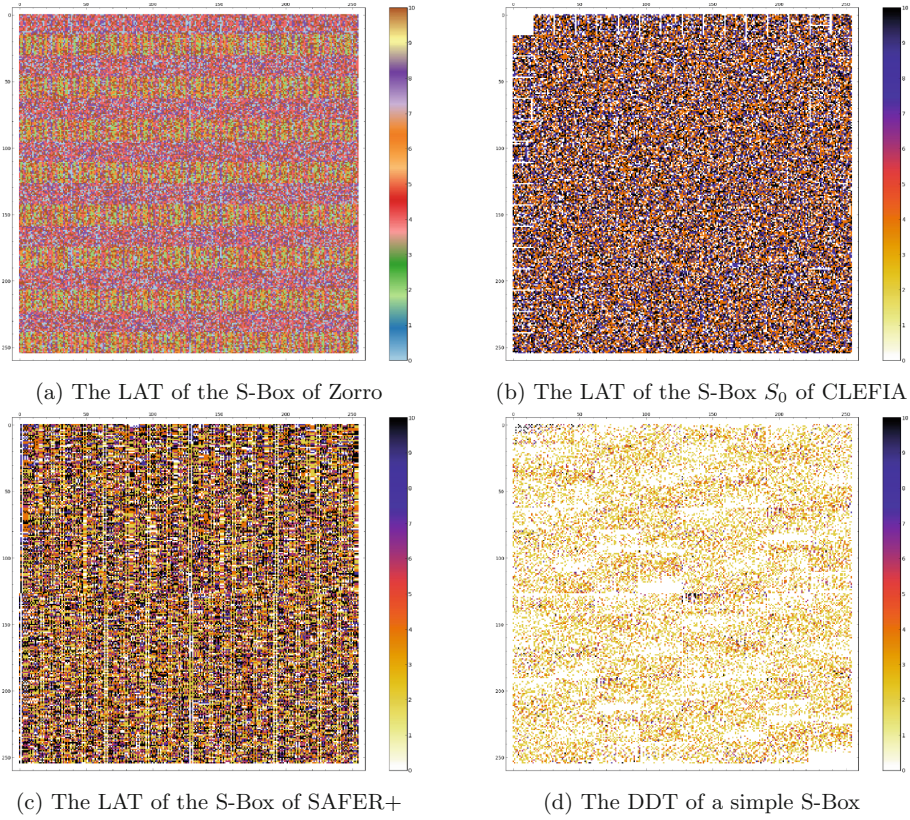


(a) The LAT of Skipjack’s  $F$

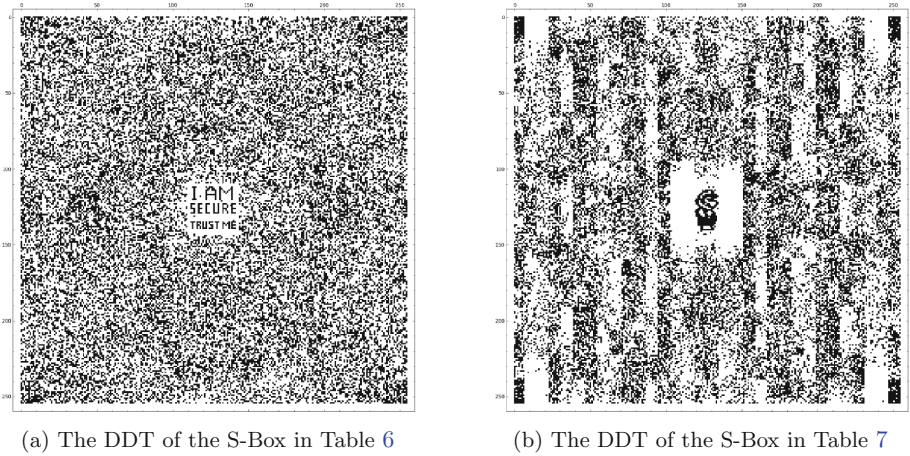


(b) The DDT of Skipjack’s  $F$

**Fig. 3.** The Pollock representations of the LAT and DDT of Skipjack’s  $F$ . For both, 0 is in white and anything equal to or above 10 is black.



**Fig. 4.** The Pollock representation of the LAT or DDT of different S-Boxes. The scales all go from 0 to 10 (anything above 10 is treated as equal to 10).



**Fig. 5.** The DDT of some outputs of Seurat’s Steganography:  $|d_{i,j}| = 0$  is in white,  $|d_{i,j}| \geq 2$  is in black.



## C S-Boxes Built from Pictures

The S-Box described in Tables 6 and 7 were built using the method described in Sect. 4.2. Note that these are not bijections. The picture representations of their DDT are given in Figs. 5a and b.

**Table 6.** An output of the algorithm described in Sect. 4.2.

	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.a	.b	.c	.d	.e	.f
0.	7a	b3	b9	b6	53	b1	26	6e	b9	43	86	ec	94	4b	9e	43
1.	5d	83	d5	57	16	4c	44	d5	5d	81	7f	79	b3	8d	e6	f8
2.	0d	59	b3	8d	04	4c	8d	ec	d9	ff	7f	7a	7e	9a	92	61
3.	05	fc	e3	1a	ed	12	1e	52	1a	e5	30	34	ef	e5	97	e5
4.	9e	69	29	d6	29	cd	b8	3a	d2	c4	1b	d1	1c	17	c3	3b
5.	44	ba	bd	19	57	0c	5a	5f	bb	55	b7	4a	5e	3f	a6	fe
6.	7f	c8	7e	65	be	1e	b3	bf	8b	85	83	83	87	12	b2	26
7.	a6	b4	bc	ef	9e	9d	6c	9e	90	5e	68	25	30	97	9f	71
8.	bf	64	65	9a	77	18	da	60	05	97	58	b2	88	d5	25	a1
9.	58	00	db	85	ca	9f	8d	42	db	bc	b2	b6	e7	85	44	78
a.	ac	be	5b	21	45	e9	40	4d	73	5f	af	93	4b	bd	45	42
b.	55	37	e2	c8	c8	20	d1	ee	7e	36	c5	28	32	37	2f	d4
c.	86	21	79	70	08	b6	91	89	e3	e5	10	e5	c6	cf	02	ca
d.	cc	b9	e1	9a	8c	8c	f3	70	ec	13	0f	00	17	7e	57	5c
e.	09	27	27	85	a0	87	3f	53	74	e3	b1	bd	de	b1	8d	61
f.	4b	84	9c	f3	72	04	7e	9c	25	3e	98	9e	43	8d	b2	9d

**Table 7.** An output of the algorithm described in Sect. 4.2.

	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.a	.b	.c	.d	.e	.f
0.	1b	1e	e7	1b	00	1b	4f	e7	07	a8	b7	1c	00	06	1c	1c
1.	30	a9	ab	af	54	50	36	57	65	01	17	7c	53	99	fb	65
2.	86	b5	33	78	c9	80	f5	7f	79	7d	87	7a	4d	14	49	2b
3.	66	d5	c8	54	a9	57	54	ab	aa	98	a8	a8	32	17	d2	cb
4.	d4	e7	73	1b	51	b3	af	50	51	68	ac	6b	d7	52	1b	d5
5.	71	75	8a	97	c8	36	37	33	74	ce	75	4a	77	88	8f	77
6.	1b	ff	e4	b5	ff	1f	1e	fa	b3	4a	b1	4c	fd	fc	4b	01
7.	ca	c8	a0	5b	5e	a1	5b	a6	9d	c8	98	84	cb	31	ca	cb
8.	33	ca	33	cc	7b	83	98	cb	a2	7f	a3	ce	34	33	cb	cd
9.	e7	fd	ff	03	7f	2d	00	b5	05	e5	ff	02	03	06	fc	06
a.	88	8e	74	8b	8c	8e	8c	51	c9	03	88	c9	8a	c9	70	fc
b.	94	2b	d4	29	ae	69	6b	af	b7	91	b7	b7	8b	89	d4	75
c.	d1	c9	98	99	61	ab	aa	61	99	66	12	65	15	2d	2d	33
d.	b3	b3	7c	86	83	7a	7f	78	cf	98	81	30	7e	cf	c9	c9
e.	01	a9	57	ad	e3	80	ad	61	56	53	53	28	56	a8	c8	ae
f.	18	1d	00	06	df	52	52	af	1d	61	e2	60	e2	e6	fa	e2

## References

1. Biham, E., Shamir, A.: Differential cryptanalysis of DES-like cryptosystems. *J. Cryptology* **4**(1), 3–72 (1991)
2. Nyberg, K.: Differentially uniform mappings for cryptography. In: Helleseht, T. (ed.) *EUROCRYPT 1993*. LNCS, vol. 765, pp. 55–64. Springer, Heidelberg (1994)
3. Blondeau, C., Nyberg, K.: Perfect nonlinear functions and cryptography. *Finite Fields Appl.* **32**, 120–147 (2015). Special Issue: Second Decade of FFA
4. Matsui, M.: Linear cryptanalysis method for DES cipher. In: Helleseht, T. (ed.) *EUROCRYPT 1993*. LNCS, vol. 765, pp. 386–397. Springer, Heidelberg (1994)
5. Nyberg, K.: Linear approximation of block ciphers. In: De Santis, A. (ed.) *EUROCRYPT 1994*. LNCS, vol. 950, pp. 439–444. Springer, Heidelberg (1995)
6. Barreto, P., Rijmen, V.: The whirlpool hashing function. In: *First open NESSIE Workshop*, Leuven, Belgium, vol. 13, p. 14 (2000)
7. Barreto, P., Rijmen, V.: The khazad legacy-level block cipher. Primitive submitted to NESSIE 97 (2000)
8. Grosso, V., Leurent, G., Standaert, F.-X., Varıcı, K.: LS-Designs: bitslice encryption for efficient masked software implementations. In: Cid, C., Rechberger, C. (eds.) *FSE 2014*. LNCS, vol. 8540, pp. 18–37. Springer, Heidelberg (2015)
9. Gérard, B., Grosso, V., Naya-Plasencia, M., Standaert, F.-X.: Block ciphers that are easier to mask: how far can we go? In: Bertoni, G., Coron, J.-S. (eds.) *CHES 2013*. LNCS, vol. 8086, pp. 383–399. Springer, Heidelberg (2013)
10. Daemen, J., Rijmen, V.: *The Design of Rijndael: AES-the Advanced Encryption Standard*. Springer, Heidelberg (2002)

11. Biryukov, A., Bouillaguet, C., Khovratovich, D.: Cryptographic schemes based on the ASASA structure: black-box, white-box, and public-key (extended abstract). In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8873, pp. 63–84. Springer, Heidelberg (2014)
12. U.S. DEPARTMENT: OF COMMERCE/National Institute of Standards and Technology: Data encryption standard. Publication, Federal Information Processing Standards (1999)
13. National Security Agency, N.S.A.: SKIPJACK and KEA Algorithm Specifications (1998)
14. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The simon and speck families of lightweight block ciphers. IACR Cryptology ePrint Archive 2013, 404 (2013)
15. Coppersmith, D.: The Data Encryption Standard (DES) and its strength against attacks. IBM J. Res. Dev. **38**(3), 243–250 (1994)
16. Biryukov, A., Shamir, A.: Structural cryptanalysis of SASAS. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 394–405. Springer, Heidelberg (2001)
17. Patarin, J.: Luby-Rackoff: 7 rounds are enough for formula  $2^{n(1-\epsilon)}$  security. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 513–529. Springer, Heidelberg (2003)
18. Biham, E., Biryukov, A., Shamir, A.: Cryptanalysis of skipjack reduced to 31 rounds using impossible differentials. J. Cryptology **18**(4), 291–311 (2005)
19. Knudsen, L.R., Robshaw, M., Wagner, D.: Truncated differentials and skipjack. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 165–180. Springer, Heidelberg (1999)
20. Knudsen, L., Wagner, D.: On the structure of skipjack. Discrete Appl. Math. **111**(1), 103–116 (2001)
21. Browning, K., Dillon, J., McQuistan, M., Wolfe, A.: An apn permutation in dimension six. Finite Fields: Theory Appl. **518**, 33–42 (2010)
22. Daemen, J., Rijmen, V.: Probability distributions of correlation and differentials in block ciphers. J. Math. Cryptology JMC **1**(3), 221–242 (2007)
23. O’Connor, L.: Properties of linear approximation tables. In: Preneel, B. (ed.) FSE 1995. LNCS, vol. 1008, pp. 131–136. Springer, Heidelberg (1995)
24. Brickell, E.F., Denning, D.E., Kent, S.T., Maher, D.P., Tuchman, W.: Skipjack review: Interim report (1993)
25. Anonymous: This looks like it might be interesting. sci.crypt (usenet), August 1995. <https://groups.google.com/forum/#!msg/sci.crypt/vLtuBDoqPfc/jm6MshFbomgJ>
26. Schneier, B.: The S-1 Algorithm. mail to the cypherpunk mailing list (1995). <http://cypherpunks.venona.com/date/1995/09/msg00315.html>
27. Gilbert, H., Chassé, G.: A statistical attack of the FEAL-8 cryptosystem. In: Menezes, A., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 22–33. Springer, Heidelberg (1991)
28. Tardy-Corffdir, A., Gilbert, H.: A known plaintext attack of FEAL-4 and FEAL-6. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 172–182. Springer, Heidelberg (1992)
29. Matsui, M., Yamagishi, A.: A new method for known plaintext attack of FEAL cipher. In: Rueppel, R.A. (ed.) EUROCRYPT 1992. LNCS, vol. 658, pp. 81–91. Springer, Heidelberg (1993)
30. Patarin, J.: Generic attacks on feistel schemes. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 222–238. Springer, Heidelberg (2001)

31. Wernsdorf, R.: The round functions of RIJNDAEL generate the alternating group. In: Daemen, J., Rijmen, V. (eds.) FSE 2002. LNCS, vol. 2365, pp. 143–148. Springer, Heidelberg (2002)
32. Blondeau, C., Canteaut, A., Charpin, P.: Differential properties of power functions. *Int. J. Inf. Coding Theory* **1**(2), 149–170 (2010)
33. Jean-Charles, F., Perret, L.: An efficient algorithm for decomposing multivariate polynomials and its applications to cryptography. *J. Symbolic Comput.* **44**(12), 1676–1689 (2009)
34. Luby, M., Rackoff, C.: How to construct pseudorandom permutations from pseudorandom functions. *SIAM J. Comput.* **17**(2), 373–386 (1988)
35. Patarin, J.: New results on pseudorandom permutation generators based on the DES scheme. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 301–312. Springer, Heidelberg (1992)
36. Patarin, J.: Security of random feistel schemes with 5 or more rounds. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 106–122. Springer, Heidelberg (2004)
37. Een, N., Sörensson, N.: Minisat: A sat solver with conflict-clause minimization. *Sat* **5** (2005)
38. Biryukov, A., Leurent, G., Perrin, L.: ESC 2015 S-box Reverse-Engineering Challenge. In: Early Symmetric Crypto, ESC 2015, pp. 104–107 (2015)
39. Canteaut, A.: Analyse et Conception de Chiffrements à Clef Secrète. Habilitation à diriger des recherches, Institut National de Recherche en Informatique et Automatique, Rocquencourt, September 2006
40. Shirai, T., Shibutani, K., Akishita, T., Moriai, S., Iwata, T.: The 128-Bit blockcipher CLEFIA (extended abstract). In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 181–195. Springer, Heidelberg (2007)
41. Massey, J.L.: Safer k-64: A byte-oriented block-ciphering algorithm. In: Anderson, R. (ed.) FSE 1993. LNCS, vol. 809, pp. 1–17. Springer, Heidelberg (1994)
42. Stein, W., et al.: Sage Mathematics Software (Version 5.10). The Sage Development Team (2013). <http://www.sagemath.org>

# Capacity and Data Complexity in Multidimensional Linear Attack

Jialin Huang<sup>1,2(✉)</sup>, Serge Vaudenay<sup>3</sup>, Xuejia Lai<sup>2</sup>, and Kaisa Nyberg<sup>4</sup>

<sup>1</sup> Technische Universität Darmstadt, Darmstadt, Germany  
jhuang.cn@gmail.com

<sup>2</sup> Shanghai Jiao Tong University, Shanghai, China

<sup>3</sup> EPFL, Lausanne, Switzerland  
serge.vaudenay@epfl.ch

<sup>4</sup> Aalto University, Espoo, Finland

**Abstract.** Multidimensional linear attacks are one of the most powerful variants of linear cryptanalytic techniques now. However, there is no knowledge on the key-dependent capacity and data complexity so far. Their values were assumed to be close to the average value for a vast majority of keys. This assumption is not accurate. In this paper, under a reasonable condition, we explicitly formulate the capacity as a Gamma distribution and the data complexity as an Inverse Gamma distribution, in terms of the average linear probability and the dimension. The capacity distribution is experimentally verified on the 5-round PRESENT.

Regarding to complexity, we solve the problem of estimating the average data complexity, which was difficult to estimate because of the existence of zero correlations. We solve the problem of using the median complexity in multidimensional linear attacks, which is an open problem since proposed in Eurocrypt 2011. We also evaluate the difference among the median complexity, the average complexity and a lower bound of the average complexity – the reciprocal of average capacity. In addition, we estimate more accurately the key equivalent hypothesis, and reveal the fact that the average complexity only provides an accurate estimate for less than half of the keys no matter how many linear approximations are involved.

Finally, we revisit the so far best attack on PRESENT based on our theoretical result.

**Keywords:** Multidimensional linear attack · Capacity · Data complexity · Linear hull effect · Linear probability

## 1 Introduction

Block ciphers are used as basic building primitives in symmetric cryptography for encryption, authentication, construction of hash functions and so on. Evaluation

---

J. Huang—This work was finished when the author affiliated to Shanghai Jiao Tong University and was visiting EPFL.

of their practical security has been a hot research issue over the decades, giving rise to different analysis techniques. Statistical attacks exploit non-uniform behaviors of the plaintext-ciphertext data to find information about the key. One of the most prominent statistical attacks is linear cryptanalysis. Previously, linear trails were assumed to behave equally for each key [3, 4, 17, 20]. Then, by considering many trails in one approximation [24, 25], the linear hull effect raises interesting discussions about fixed-key behaviors in single linear approximations [21, 22]. Daemen et al. gave a fixed-key probability distribution for single linear correlations [13], leading to subsequent works on e.g., fundamental assumptions [9], the effect of key schedules [1] and measures for data complexity [19], all for single linear attacks. However, we still do not understand the situation in multidimensional linear cryptanalysis.

A collection of linear approximations has a *capacity* which measures their bias to the uniform distribution. One important open problem in multidimensional linear cryptanalysis is to estimate the capacity and data complexity when a large number of different keys are considered. In previous work, the capacity was assumed to hold an average value constantly for most of the keys, and the data complexity was usually measured by reciprocal of the average capacity. However, neither is correct. As we know, the key equivalent hypothesis has been questioned for single linear approximations and differential trails [5, 9, 12]. Now this hypothesis also requires adjustment in multidimensional linear setting.

Also, it has always been difficult to compute average data complexity over the keys in linear cryptanalysis. Using Jensen’s inequality, Murphy [22] points out that the Fundamental Theorem [24] can only give a lower bound for the average data complexity when a collection of linear trails in a linear approximation is used. Leander shows that in single linear attacks we should focus on median complexity instead of average complexity since the latter usually turns to infinity [19]. Both Murphy’s and Leander’s concerns haven’t been addressed yet in the scenario of multidimensional linear attacks.

As one of the most powerful variants of linear attacks, multidimensional linear attacks notably benefit the data complexity, both in theory and in practice [10, 11, 15, 16, 23]. Moreover, the multidimensional linear distinguisher has been discovered to have connections with other statistical distinguishers, e.g., truncated differential distinguishers [6], statistical saturation distinguishers [19], and integral distinguishers [8]. All the above suggests the importance of multidimensional linear cryptanalysis, hence, the lack of knowledge on fundamental aspects of this attack is especially surprising, and deserves more attention.

*Our Contributions.* In this paper, we point out that under a reasonable assumption, the distribution of key-dependent capacity can be explicitly formulated with a Gamma distribution, depending on average linear probability and dimension (Sect. 3). This distribution is verified experimentally on the round-reduced PRESENT cipher. Then, we derive the distribution of data complexity, an Inverse Gamma distribution based on the same parameters (Sect. 4). Our results allow a more accurate measurement for multidimensional linear attacks.

With these distributions, in Sect. 5 we discuss three well-known measures when considering the data complexity of multidimensional linear attacks: the reciprocal of average capacity, the average and the (general) median complexity. The following fundamental questions in single linear attacks are then generalized to multidimensional linear attacks and solved.

Firstly, we consider the standard key equivalence hypothesis. We discover that instead of holding for a majority of keys, the average capacity actually holds for less than half of the keys, no matter how many linear approximations are used. Hence, we modify the hypothesis in a way which is more in line with the practical situation.

Secondly, as we know, the average data complexity of single linear attacks is difficult to calculate, since the linear hull effect may result in zero correlation for some keys. However, we show that the situation changes when multiple linear approximations are involved, and in this case the average data complexity can be easily calculated from the Inverse Gamma distribution. Then, by generalizing Murphy's idea from the case of linear hulls to the case of multiple linear approximations, the reciprocal of average capacity is proved to be only a lower bound of the average data complexity. We also figure out the exact difference between this lower bound and the average data complexity.

Thirdly, we solve the open problem proposed by Leander in [19] by developing the usage of median complexity to multidimensional linear attacks. Finally, all measures of data complexity are compared under different dimensions. An interesting observation is that, the median complexity infinitely approaches to the average one as the dimension increases.

In Sect. 6, we revisit Cho's 25 rounds of multidimensional linear attack on PRESENT [10], which targets the most rounds of PRESENT with data complexity less than the whole codebook. As an application of our theoretical analysis, we can directly estimate the average capacity, instead of making a complex proof like [10]. Our results are very close to Cho's. Moreover, the exact knowledge of the capacity distribution allows us to compute the ratio of weak keys precisely. Using Cho's attack method by changing some parameters in the attack,  $2^{123.24}$  weak keys for 26 rounds PRESENT can be recovered with no more than  $2^{62.5}$  plaintext-ciphertext pairs.

## 2 Preliminaries

### 2.1 Block Ciphers and Linear Cryptanalysis

Let  $\mathbb{F}_2$  be the binary field with two elements and  $\mathbb{F}_2^n$  be the  $n$ -dimensional vector space over  $\mathbb{F}_2$ . The inner product on  $\mathbb{F}_2^n$  is defined by  $a \cdot b = \sum_{i=1}^n a_i b_i$ , where  $a, b \in \mathbb{F}_2^n$ .

A *block cipher* is a mapping  $E : \mathbb{F}_2^n \times \mathbb{F}_2^k \rightarrow \mathbb{F}_2^n$  with  $E_k(\cdot) \stackrel{\text{def}}{=} E(k, \cdot)$  for each  $k \in \mathbb{F}_2^k$ . If  $y = E_k(x)$ ,  $x$ ,  $y$  and  $k$  are referred to as the plaintext, the ciphertext and the master key, respectively. A *key-alternating cipher* is a block cipher consisting of an alternating sequence of unkeyed rounds and simple bitwise key additions.

Linear cryptanalysis uses a linear relation between bits from  $x$ ,  $y$  and  $k$ . A *linear approximation*  $(u, v)$  is a probabilistic linear relation expressed as a boolean function of these bits, i.e.,

$$B(k) \stackrel{\text{def}}{=} u \cdot x \oplus v \cdot E_k(x), \quad (1)$$

where  $(u, v)$  is called the *text mask*.  $B(k)$  is a boolean random variable characterized by

$$p(k) \stackrel{\text{def}}{=} \Pr_{x \in \mathbb{F}_2^n} (B(k) = 0).$$

We call  $c(k) = 2p(k) - 1$  the fixed-key *correlation* of the linear approximation  $(u, v)$ . The *linear probability* (LP) of approximation  $(u, v)$  is defined as  $LP(k) = c(k)^2$ . Both  $c(k)$  and  $LP(k)$  vary over different keys, and can be regarded as real-value random variables over the whole key space.

In a linear approximation  $(u, v)$ , there may be many paths with different intermediate masks, but sharing the same input and output mask  $(u, v)$ . A path that considers linear relation round by round is called as *linear trail* (or *linear characteristic*). Note that in a key-alternating cipher, the LP of a linear trail<sup>1</sup> is independent of the subkeys.

## 2.2 Multidimensional Linear Approximations and Data Complexity

Multidimensional linear attacks use  $m$  approximations with linearly independent text masks, called *base approximations*, to construct an  $m$ -dimensional vectorial boolean function  $f$ . Let  $p = (p_0, p_1, \dots, p_{2^m-1})$  be the probability distribution of  $f$ . It can be computed by the following lemma.

**Lemma 1.** ([15, Corollary 1]) *Let  $f : \mathbb{F}_2^n \mapsto \mathbb{F}_2^m$  be a vectorial boolean function with the probability distribution  $p$ . Then, we have*

$$c_a = \sum_{\eta \in \mathbb{F}_2^m} (-1)^{a \cdot \eta} p_\eta, \text{ for all } a \in \mathbb{F}_2^m$$

and

$$p_\eta = 2^{-m} \sum_{a \in \mathbb{F}_2^m} (-1)^{a \cdot \eta} c_a, \text{ for all } \eta \in \mathbb{F}_2^m.$$

Here,  $c_a$  is the correlation of the boolean function  $a \cdot f$ ,  $a \in \mathbb{F}_2^m$ .

In multidimensional linear attack,  $c_a$  is indeed the correlation of the approximation that combines the base approximations linearly.

Let  $q = (q_0, \dots, q_{2^m-1})$  be another discrete probability distribution of an  $m$ -bit random variable. Then, the capacity of  $p$  and  $q$  is defined as follows.

<sup>1</sup> Hereafter, whether the LP is of a linear approximation or of a linear trail will be clear from the context.



**Definition 1.** *The capacity between two probability distributions  $p$  and  $q$  is defined by*

$$C(p, q) = \sum_{\eta=0}^{2^m-1} (p_\eta - q_\eta)^2 q_\eta^{-1}.$$

The capacity of multidimensional linear approximations with probability distribution  $p$  is  $C(p) = C(p, \theta)$ , where  $\theta$  is the uniform distribution.

**Lemma 2.** ([15, Corollary 2]) *Given an  $m$ -dimensional vectorial boolean function  $f$  with the probability distribution  $p$ , the capacity is*

$$C(p) = \sum_{a \in \mathbb{F}_2^m, a \neq 0} c_a^2.$$

Thus, the capacity of multidimensional linear approximations is computed from  $m$  base approximations and other  $2^m - 1 - m$  approximations that are XOR sum of the  $m$  base approximations. These  $2^m - 1 - m$  approximations, denoted as *combined approximations*, are linearly spanned from the  $m$  base approximations.

To estimate the data complexity of multidimensional linear cryptanalysis, the Chernoff information  $D^*$  can be considered [2].

**Theorem 1.** ([2, Theorem 1]) *Let  $BestAdv_N(p, q)$  be the best advantage for distinguishing probability distribution  $p$  from probability distribution  $q$ , using  $N$  samples. We have*

$$1 - BestAdv_N(p, q) = 2^{-ND^*(p, q) + o(N)}.$$

Hence, the data complexity is  $N \approx \frac{1}{D^*(p, q)}$ . When  $q$  is the uniform distribution and  $p$  is close to  $q$ , the Chernoff information can be approximated by the capacity  $C(p)$ , [2, Theorem 7], by

$$D^*(p, q) \simeq \frac{C(p)}{8 \ln 2}.$$

In this case, when the optimal distinguisher based on LLR-statistic (or  $\chi^2$ -statistic) is used, the data complexity is given as  $\frac{\lambda}{C(p)}$ , where  $\lambda$  depends on the success probability of the distinguisher.

The probability distribution  $p$  of an  $m$ -dimensional linear approximation actually varies over different keys, so does the capacity (as we will show later). Hereafter, instead of using  $C(p(k))$ , we use  $C(k)$  to represent the variable of key-dependent capacity.

### 2.3 Related Distributions and Assumptions

**Note 3.** *Let  $\mathcal{N}(\mu, \sigma^2)$  be the normal distribution with mean  $\mu$  and variance  $\sigma^2$ . Let  $\Gamma(\alpha, \theta)$  be the Gamma distribution under the shape-scale parametrization, with mean  $\alpha\theta$ , the probability density function  $g$  and the cumulative distribution function  $\mathcal{G}$ . If  $X \sim \mathcal{N}(0, \sigma^2)$ , then  $X^2 \sim \Gamma(1/2, 2\sigma^2)$ .  $Inv\text{-Gamma}(\alpha, \beta)$  denotes the inverse-Gamma distribution with mean  $\frac{\beta}{\alpha-1}$  for  $\alpha > 1$ . If  $X \sim \Gamma(\alpha, \theta)$ , then  $\frac{1}{X} \sim Inv\text{-Gamma}(\alpha, \theta^{-1})$ .*

Daemen et al. give the distribution of the fixed-key LP of linear approximations when linear hull effect is considered [13].

**Approximation 4.** [13, Theorem 22] *Given a key-alternating cipher with independent round-keys, when the number of linear trails of  $(u, v)$  is large enough and their LP are small compared to  $ELP(u, v)$ , the fixed-key correlation of  $(u, v)$ ,  $c(k)$ , which is a real-value random variable, follows*

$$c(k) \sim \mathcal{N}(0, ELP(u, v)).$$

The fixed-key  $LP(k)$  follows the distribution of  $\Gamma(\frac{1}{2}, 2ELP(u, v))$ , with mean  $ELP(u, v)$  and variance  $2ELP(u, v)^2$ , where  $ELP(\cdot)$  is the average linear probability of the approximation over all keys.

The  $ELP(u, v)$  can be denoted as  $\overline{c^2}$  and computed by the following proposition for key-alternating ciphers.

**Proposition 1.** [12, 24] *Let  $E$  be a key-alternating block cipher and assume that all subkeys are independent. The average LP of a linear approximation is the sum of all LP of the linear trails  $t_j$ ,  $LPT(t_j)$ , between the input and output mask of this approximation, i.e.,*

$$ELP(u, v) = \sum_{t_j \in (u, v)} LPT(t_j).$$

### 3 Key-Dependent Capacity in Multidimensional Linear Approximations

In this section, we study the distribution of key-dependent capacity. Let  $c(k)$  (resp.  $LP(k)$ ) be a real-value random variable representing the fixed-key correlation (resp. linear probability) of the linear approximation and we can know  $c(k)$  and  $LP(k)$  from Approximation 4. When multiple linear approximations are used, we use  $i$  in the subscript to denote the index of linear approximations, e.g., denote  $c_i(k)$  as the fixed-key correlation of the  $i$ th linear approximation. W.l.o.g, we use  $i = 1, \dots, m$  to represent the subscript of  $m$  base approximations.

In [16], the authors claim that in practical experiments the probability distributions vary a lot with the keys while the capacity remains rather constant. However, in this section we point out that the capacity also varies over different keys from the theoretical point and give experimental verification. We focus on dealing with two cases, both existing in practical block ciphers. These two cases are shown in Propositions 2 and 3, respectively.

**Proposition 2.** *In an  $m$ -dimensional linear attack using  $m$  base approximations with correlations  $c_i(k)$  i.i.d. to  $\mathcal{N}(0, \overline{c^2})$  over the keys, where  $\overline{c^2}$  is the average LP. If for each fixed key, the binary random variables associated to the base approximations are statistically independent, the fixed-key capacity of this  $m$ -dimensional linear approximation,  $C(k)$ , approximately follows Gamma-distribution  $\Gamma(\frac{m}{2}, 2\overline{c^2})$ .*

*Proof.* Let  $f_1(k), \dots, f_m(k)$  be  $m$  linearly independent base approximations to construct the  $m$ -dimensional approximation  $f(k)$ , and  $f(k) = (f_1(k), \dots, f_m(k))$  is an  $m$ -dimensional vectorial boolean function with the probability distribution  $p(k) = \{p_\eta(k)\}$ , where  $\eta \in \mathbb{F}_2^m$  and  $p_\eta(k)$  is the probability that  $f(k) = \eta$ . Indeed,  $f_i(k)$  is a binary random variable with correlation  $c_i(k)$ . Since  $f_i(k)$  are statistically independent each other for each fixed key  $k$ ,

$$p_\eta(k) = \prod_{i=1}^m \left( \frac{1}{2} + (-1)^{f_i(k)} \frac{c_i(k)}{2} \right), \eta \in \mathbb{F}_2^m$$

According to Definition 1,

$$\begin{aligned} C(k) &= \sum_{\eta \in \mathbb{F}_2^m} (p_\eta(k) - 2^{-m})^2 / 2^{-m} = 2^m \sum_{\eta \in \mathbb{F}_2^m} (p_\eta(k) - 2^{-m})^2 \\ &= 2^m \sum_{\eta \in \mathbb{F}_2^m} \left( \prod_{i=1}^m \left( \frac{1}{2} + (-1)^{f_i(k)} \frac{c_i(k)}{2} \right) - 2^{-m} \right)^2 \end{aligned}$$

For each fixed key,  $c_i(k) \cdot c_j(k) \ll c_i(k)$ ,

$$\begin{aligned} C(k) &= 2^m \sum_{\eta \in \mathbb{F}_2^m} \left[ \sum_{i=1}^m (-1)^{f_i(k)} \frac{c_i(k)}{2 \cdot 2^{m-1}} \right]^2 \\ &= 2^m \sum_{\eta \in \mathbb{F}_2^m} \left[ \frac{1}{2^{2m-2}} \left( \sum_{i=1}^m \left( \frac{c_i(k)}{2} \right)^2 + 2 \sum_{i \neq j} (-1)^{f_i(k) + f_j(k)} \frac{c_i(k)}{2} \frac{c_j(k)}{2} \right) \right] \end{aligned}$$

Since  $\sum_{\eta \in \mathbb{F}_2^m} \sum_{i \neq j} (-1)^{f_i(k) + f_j(k)} \frac{c_i(k)}{2} \frac{c_j(k)}{2} = 0$ ,

$$C(k) = \frac{2^m}{2^{2m-2}} \sum_{\eta \in \mathbb{F}_2^m} \sum_{i=1}^m \left( \frac{c_i(k)}{2} \right)^2 = \sum_{i=1}^m c_i(k)^2 = \sum_{i=1}^m LP_i(k)$$

Since  $c_i(k)$  are i.i.d. to  $\mathcal{N}(0, \overline{c^2})$ ,  $LP_i(k)$  are i.i.d to  $\Gamma(\frac{1}{2}, 2\overline{c^2})$ ,  $i = 1, \dots, m$ . Thus,  $C(k)$  is the sum of  $m$  independent Gamma distribution  $\Gamma(\frac{1}{2}, 2\overline{c^2})$ . Hence,  $C(k) \sim \Gamma(\frac{m}{2}, 2\overline{c^2})$ .  $\square$

Recall that for one-dimensional linear approximations,  $\overline{c^2}$  can be calculated by Proposition 1 when the dominant trails in a linear approximation are known.

Proposition 2 considers the scenario where the LP of base approximations are dominant. In this case, we approximate the capacity by summing the LP of base approximations and ignoring the LP of combined approximations (see Lemma 2). To show the reasonableness of this approximated capacity, we also bound the error of our approximation. For this part of analysis, please see Appendix B.

In the other hand, Proposition 3 considers the case that not only  $m$  base approximations but also  $2^m - 1 - m$  combined approximations have non-negligible contribution to the capacity. In this case, the correlations of  $2^m - 1 - m$  combined approximations are not independent any more. Thus, we derive the capacity in this case under another hypothesis.

**Proposition 3.** *In an  $m$ -dimensional linear attack using the  $m$ -dimensional linear approximation with the probability distribution  $p_\eta(k)$  i.i.d to a normal distribution  $\mathcal{N}(2^{-m}, \sigma^2)$ ,  $\eta \in \mathbb{F}_2^m$ , the fixed-key capacity of this  $m$ -dimensional linear approximation,  $C(k)$ , follows Gamma-distribution  $\Gamma(\frac{2^m-1}{2}, 2 \cdot 2^m \sigma^2)$ .*

*Proof.* Since  $p_\eta(k)$  are i.i.d. to  $\mathcal{N}(2^{-m}, \sigma^2)$ ,

$$Q = \sum_{\eta=0}^{2^m-1} \frac{(p_\eta(k) - 2^{-m})^2}{\sigma^2} \sim \chi^2(2^m - 1) = \Gamma(\frac{2^m - 1}{2}, 2)$$

According to the definition of capacity,

$$C(k) = \sum_{\eta=0}^{2^m-1} \frac{(p_\eta(k) - 2^{-m})^2}{2^{-m}} = 2^m \sigma^2 Q = \Gamma(\frac{2^m - 1}{2}, 2 \cdot 2^m \sigma^2) \quad \square$$

Compared with Proposition 2 which considers only  $m$  base approximations with equally dominant correlations, Proposition 3 indeed addresses the situation where the correlation  $c_a(k)$  of  $2^m - 1$  approximations are identically distributed (for the proof please refer to Appendix A). Thus, the average LP of  $2^m - 1$  approximations are equal, denoted as  $c^2$  again. As we know, the average capacity is the sum of the average LP of involved approximations, i.e.,  $(2^m - 1) \cdot 2^m \sigma^2 = (2^m - 1) \overline{c^2}$ , the distribution of capacity in Proposition 3 can also be represented as  $\Gamma(\frac{2^m-1}{2}, 2\overline{c^2})$ .

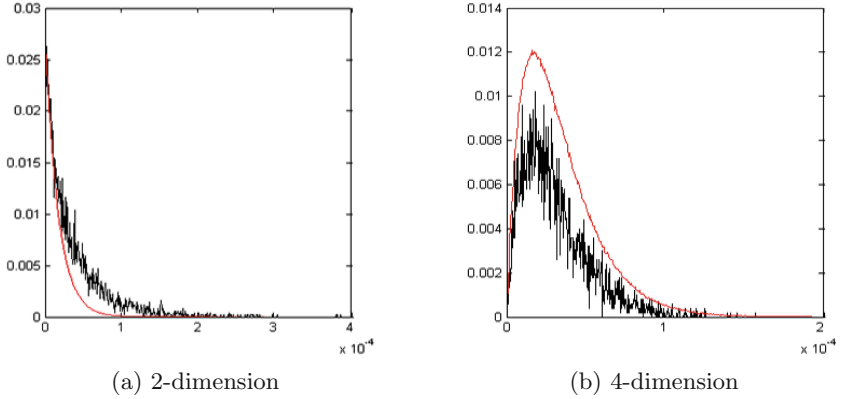
**Experimental Verification.** In order to verify that the above analysis reflects the reality with reasonable accuracy, we have experimentally computed the capacity distributions sampled from 5000 randomly chosen keys for 5-round PRESENT. A set of usable one-dimensional linear approximations is discovered in [26], with theoretical average LP computed as  $2^{-16.83}$ . Thus, the correlation distributions of these approximations are  $\mathcal{N}(0, 2^{-16.83})$ , and the LP distributions are  $\Gamma(\frac{1}{2}, 2^{-15.83})^2$ .

We can select linearly independent approximations from this set as the base approximations. Here we examine the 2-dimensional and 4-dimensional linear approximations for the case of Proposition 2.

In this case, the base approximations with input masks from different S-boxes in the first round and output masks from different S-boxes in the last round are chosen. According to Proposition 2, the theoretical distribution of 2-dimensional capacity is  $\Gamma(1, 2^{-15.83})$  and of 4-dimensional capacity is  $\Gamma(2, 2^{-15.83})$ . The experimental distributions of 2-dimensional and 4-dimensional capacity sampled over 5000 keys are as (a) and (b) of Fig. 1, respectively.

As illustrated in Fig. 1, the experimental distribution of capacity follows the theoretical estimate closely. The scattering of data points occurs due to the fact that we basically use a histogram, and deal with raw data instead of averaging.

<sup>2</sup> For more details about the approximations used in our experiments, please refer to [26].



**Fig. 1.** Experimental (black) and theoretical (red) distributions of the capacity for the 2 and 4-dimensional approximation of the first case (Color figure online)

## 4 Distribution of Data Complexity

With the knowledge of capacity distribution, the distribution of data complexity, which approximates to  $\lambda$  times the reciprocal of capacity, can be obtained formally. Hereafter we focus on the case mentioned in Proposition 2. The case of Proposition 3 can be deduced in a similar way.

**Corollary 1.** *If the fixed-key capacity of the multidimensional linear approximation follows  $C(k) \sim \Gamma(\frac{m}{2}, 2c^2)$ , then the fixed-key data complexity of the corresponding multidimensional attack follows  $N(k) \sim \text{Inv-Gamma}(\frac{m}{2}, \frac{\lambda}{2c^2})$ .*

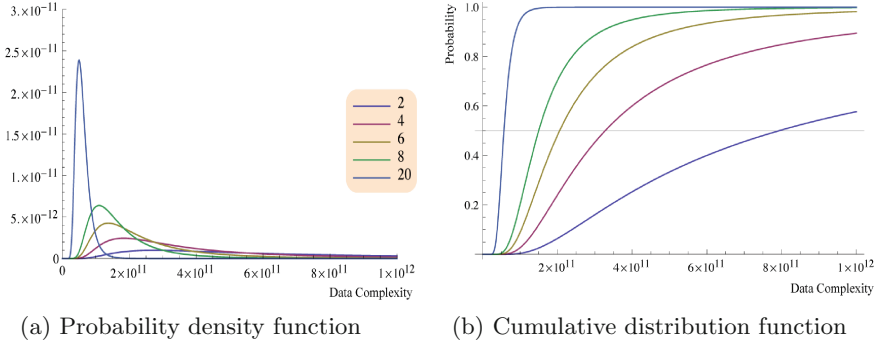
Corollary 1 is derived directly from Proposition 2 (also refer to Note 3), and addresses the case that  $m$  correlations of base approximations play a prominent role in the capacity. Since  $\lambda$  is a constant for any fixed success probability in an attack, w.l.o.g. hereafter we study the above data complexity distribution as Inv-Gamma( $\frac{m}{2}, \frac{1}{2c^2}$ ). For each key  $k$ ,  $N(k)$  is asymptotically inversely proportional to  $C(k)$ . The average data complexity over all keys is denoted by  $N$ ,  $N = E_k[N(k)]$ , which is proportional to

$$E_k \left[ \frac{1}{C(k)} \right] = \frac{1}{|\mathcal{K}|} \sum_{k \in \mathcal{K}} \frac{1}{C(k)},$$

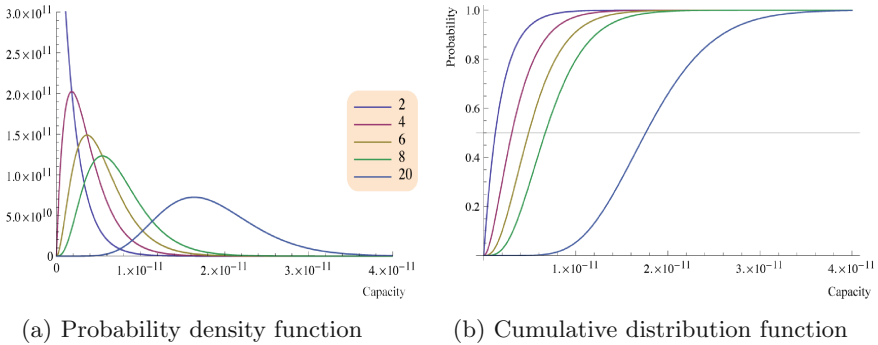
where  $\mathcal{K}$  denotes the whole key space, and  $E_k(\cdot)$  means an expected value taken over the whole key space. According to Corollary 1 and the mean of inverse Gamma distribution (see Note 3), the average data complexity is  $E_k \left[ \frac{1}{C(k)} \right] = \frac{1}{2c^2(m/2-1)} = \frac{1}{mc^2-2c^2}$ .

*Remark.* The data complexity distribution in Corollary 1 also holds for single linear attacks where  $m = 1$ . In the case of  $m = 1$ , the average data complexity is infinite as pointed out by [19]<sup>3</sup>, which corresponds to the fact that the mean of the distribution

<sup>3</sup> In fact, the data complexity should be upper-bounded by the size of the codebook.



**Fig. 2.** Distributions of the data complexity for  $m = 2, 4, 6, 8, 20$ .



**Fig. 3.** Distributions of the capacity for  $m = 2, 4, 6, 8, 20$ .

$\text{Inv-Gamma}(\frac{1}{2}, \frac{1}{2c^2})$  doesn't exist. When  $m$  is equal to 2, the mean of the inverse Gamma distribution also doesn't exist because there are always values going to infinite according to the distribution.

Similarly, the average capacity over the keys

$$E_k[C(k)] = \frac{1}{|\mathcal{K}|} \sum_{k \in \mathcal{K}} C(k)$$

is equal to  $m\overline{c^2}$ , derived from the mean of the Gamma distribution in Proposition 2 (see Note 3).

**Example 5.** For clearer explanation, hereafter a simple example which quite meets real situations in practical ciphers is used in our analysis. We take  $c^2$  as  $2^{-40}$ , which roughly equates the case in 15-round PRESENT, and take different  $m$  as 2, 4, 6, 8, 20 respectively. In this example, the distribution functions of data complexity are shown in Fig. 2, and the distribution functions of capacity are shown in Fig. 3.

## 5 Evaluation of the Data Complexity

In practical attacks,  $E_k[\frac{1}{C(k)}]$  and  $\frac{1}{E_k[C(k)]}$  are highly related to the evaluation of data complexity. Since  $E_k[\frac{1}{C(k)}]$  is hard to estimate, the complexity is usually measured by  $\frac{1}{E_k[C(k)]}$ . In this section, we firstly propose a refined key equivalent hypothesis for  $E_k[C(k)]$  (Sect. 5.1). With the exact description of data complexity distributions, the difficulty of evaluating  $E_k[\frac{1}{C(k)}]$  is overcome, and a basic issue about the relation of average capacity and average data complexity is studied (Sect. 5.2). We also extend Leander's idea of exploiting median data complexities [19] to multidimensional linear attacks (Sect. 5.3). Finally, all measures are compared.

### 5.1 Adjusted Key Equivalence Hypothesis

In regard to the connection between the fixed-key capacity and the average capacity in a multidimensional linear system, the traditional key equivalence hypothesis indicates that the fixed-key capacity does not deviate significantly from its average value [14, 18]. This key equivalence hypothesis can be interpreted as follows:  $C(k) \approx E_k[C(k)]$ , for almost all keys  $k$ . As we have shown, the capacity is actually Gamma distributed so that this hypothesis does not hold. Thus, two questions arise: which value is suitable for the evaluation of the attack complexity? Is that average value enough and correct? We start with the following conjecture to show that the average capacity is far from being able to represent the majority of keys.

*Conjecture 1.* There are always less than half of the keys having a capacity larger than the average capacity. That is,  $|\{k^* \in \mathcal{K} | C(k^*) \geq E_k[C(k)]\}| < \frac{1}{2}|\mathcal{K}|$ . Hence, less than half of the right keys can be recovered with a data complexity of  $\frac{\lambda}{E_k[C(k)]}$ , where  $\mathcal{K}$  is the whole key space.

This conjecture is illustrated in Table 1 with Example 5. With the increase of  $m$ , the ratio of keys that have a capacity larger than the average capacity approximates to  $\frac{1}{2}$ , but cannot equal to  $\frac{1}{2}$ . This is because, for such a skew Gamma distribution as in Proposition 2, the median value is always smaller than the mean. It can be concluded that, using the number of cipher texts equal to  $\frac{\lambda}{E_k[C(k)]}$ , more than half of the keys cannot be recovered successfully with a reasonable probability. Thus, the average capacity is not enough to bring a sound estimation of attack complexities for most keys, especially when  $m$  is not large enough.

Since the capacity is highly dependent on the choice of the key, we concern that with how many data texts the multidimensional attacks can succeed for a majority

**Table 1.** The ratio of keys that have a capacity larger than the average capacity

$m$	2	4	6	8	20
ratio(%)	36.79 %	40.6 %	42.32 %	43.35 %	45.79 %

of keys. A natural way to adjust the hypothesis is to consider the upper bound of data complexity for, e.g. 90 %, of the keys, meaning that for these 90 % keys the amount of data texts can guarantee a successful attack with high probability, even for some of these keys this data complexity is overestimated.

**Hypothesis 6.** (Adjusted Key Equivalence Hypothesis) *If the capacity distribution of an  $m$ -dimensional linear attack satisfies Proposition 2, then 90 % of the keys in the key space have a capacity no smaller than  $\mathcal{G}^{-1}(0.1)$ , where  $\mathcal{G}$  is the cumulative distribution function of  $\Gamma(\frac{m}{2}, 2c^2)$ . Using  $\frac{\lambda}{\mathcal{G}^{-1}(0.1)}$  data is enough for recovering 90 % of the keys in the key space.*

## 5.2 On Average Data Complexity

**Why the Average Data Complexity is Calculable?** It is known that in the classical single linear attacks considering linear hull effect, the average data complexity is hard to derive and usually infinite because of the existence of zero correlation. This difficulty now can be solved in the situation of  $m$ -dimensional linear attacks, since the average value can be easily derived from the accurate distributions of data complexity, when  $m$  is larger than 2. From the point of capacity distributions, we can understand more about the reason why the average data complexity is calculable in multidimensional attacks.

In the single linear setting, the keys with zero  $C(k)$  may make the average complexity infinite, thus, this part of keys should be focused on. Here, we point out that by taking multiple linear approximations simultaneously into consideration instead of only one, the number of keys with zero capacity can be very tiny so that the average complexity turns out to be computable.

We compare the ratio of keys bringing  $C(k)$  between zero and  $\epsilon$ , where  $\epsilon$  is a fixed value very close to zero. From (b) of Fig. 3, it is obvious that with the increase of  $m$ , the ratio of keys with capacity going to zero decreases. This ratio for several fixed  $\epsilon$  is shown in Table 2. From Table 2 we can see that as the increase of  $m$ , the ratio of keys with capacity close to zero decreases dramatically. This is because as the number of approximations grows, for each key there is higher probability that at least one approximation brings a non-zero LP, so that a non-zero capacity. Hence, for a fixed  $\epsilon$ , the more base approximations are used, the fewer the number of keys which bring infinite data complexities becomes. When  $\epsilon$  is small enough and  $m$  has a reasonable size, this ratio can be negligible in the whole key space. In this case it is sound to assume that there is no key causing a zero capacity, so that the average data complexity is computable.

**A Difference Between  $E_k[\frac{1}{C(k)}]$  and  $\frac{1}{E_k[C(k)]}$ .** The problem discussed here is firstly found in the context of linear hull effect by Murphy [22]. We extend it to multidimensional linear attacks and make further investigation.

In some attack analysis, e.g. [10], the reduction in data complexity given by multiple approximations is based on the assertion that the data complexity  $N$  is proportional to  $\frac{1}{E_k[C(k)]}$ . Like the effectiveness issue of linear hull effect studied



**Table 2.** The ratio of keys with capacity close to zero for different  $m$  and  $\epsilon$ 

$\epsilon \setminus m$	2	4	6	8	20
$10^{-16}$	$5.5 \times 10^{-5}$	$1.5 \times 10^{-9}$	$2.77 \times 10^{-14}$	$3.8 \times 10^{-19}$	$6.95 \times 10^{-50}$
$10^{-20}$	$5.5 \times 10^{-9}$	$1.5 \times 10^{-17}$	$2.77 \times 10^{-26}$	$3.8 \times 10^{-35}$	$6.95 \times 10^{-90}$
$10^{-25}$	$5.5 \times 10^{-14}$	$1.5 \times 10^{-27}$	$2.77 \times 10^{-41}$	$3.8 \times 10^{-55}$	$6.95 \times 10^{-140}$

in [22], there is also a difference between  $\frac{1}{E_k[\overline{C(k)}]}$  and the actual average data complexity. According to Jensen's Inequality and the fact that reciprocal of positive real numbers is a convex function, we have

$$E_k \left[ \frac{1}{C(k)} \right] \geq \frac{1}{E_k[\overline{C(k)}]}.$$

Thus, the  $\frac{1}{E_k[\overline{C(k)}]}$  can only be used to give a lower bound to the average data complexity.

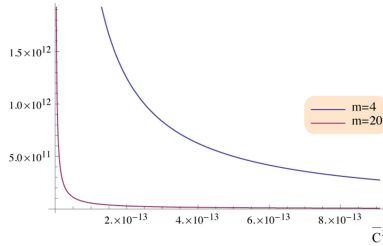
Jensen's Inequality gives a general comparison without considering the details of the variables. When the distributions of both  $C(k)$  and  $\frac{1}{C(k)}$  are known,  $E_k[\frac{1}{C(k)}]$  and  $\frac{1}{E_k[\overline{C(k)}]}$  can be derived as in Sect. 4. Their difference is formulated as  $\frac{1}{m\overline{c^2} - 2\overline{c^2}} - \frac{1}{m\overline{c^2}} = \frac{2}{m(m-2)\overline{c^2}}$ . Therefore, in fact the equality will never hold for  $m$  larger than 2, i.e.,  $E_k[\frac{1}{C(k)}]$  is always larger than  $\frac{1}{E_k[\overline{C(k)}]}$ . The difference can be ignored only when  $m$  is large enough. Figure 4 shows the difference for  $m = 4$  and  $m = 20$ . For small  $m$  the difference is much more non-negligible, and  $\frac{1}{E_k[\overline{C(k)}]}$  does not reflect the real average data complexity. As more approximations are involved, the difference has a quicker trend to be small. For a fixed  $m$ , the smaller the average LP is, the larger the difference becomes. That is, as  $\overline{c^2}$  decreases, which is a typical case since cryptanalysts always try to break as many rounds of the cipher as possible, the difference between  $E_k[\frac{1}{C(k)}]$  and  $\frac{1}{E_k[\overline{C(k)}]}$  turns to be huge.

### 5.3 On Median Data Complexity

Leander proposed a way to overcome the problem of infinite data complexities for single linear attacks [19]. Namely, instead of studying the average complexity, he studied the median complexities  $\tilde{N}$  such that for half of the keys the data complexity of an attack is less than or equal to  $\tilde{N}$ . So far the usage of median complexity in multidimensional linear attacks remains unsolved, which we will discuss in this section. A general definition of  $N_p$  is as follows, where  $\tilde{N} = N_{1/2}$ .

**Definition 2.** ([19, Definition 1])  $N_p$  is defined as the complexity such that the probability that for a given key the attack complexity is lower than  $N_p$ , is  $p$ .

Although Leander gave this general definition, he focused on the case of  $N_{1/2}$  in single linear attacks. With the knowledge of accurate distributions of data complexity, we generalize Leander's Theorem 2 in [19] not only under the multidimensional linear model but also from  $N_{1/2}$  to  $N_p$ .



**Fig. 4.** The difference between  $E_k[\frac{1}{C(k)}]$  and  $\frac{1}{E_k[C(k)]}$  with  $\overline{c^2}$  ranging from  $2^{-60}$  to  $2^{-40}$ .

**Theorem 2.** *Assuming independent subkeys in an  $m$ -dimensional linear attack using  $m$  base approximations with the i.i.d. LP that is  $\Gamma(\frac{1}{2}, 2\overline{c^2})$ ,  $p$  percent of the keys yield to a capacity of at least  $\mathcal{G}^{-1}(1-p)$ , where  $\mathcal{G}$  is the cumulative distribution function of  $\Gamma(\frac{m}{2}, 2\overline{c^2})$ . Thus, the complexity of this  $m$ -dimensional linear attack is less than  $\frac{\lambda}{\mathcal{G}^{-1}(1-p)}$  with the probability  $p$ .*

Leander’s Theorem 2 is a special case of Theorem 2 taking  $m$  as 1 and  $p$  as  $\frac{1}{2}$ , when the noisy linear trails are ignored in the linear hull effect (If the noisy trails are considered, the ratio of keys reduces by a factor of 2). If we explain Leander’s Theorem 2 in our context, we use the fact that  $F^{-1}(1/2) = 0.46\overline{c^2}$ , where  $F$  is the cumulative distribution function of  $\Gamma(\frac{1}{2}, 2\overline{c^2})$  (see [19] for more details).

As illustrated in (b) of Fig. 3, for the Y-axis at 1/2, the median capacity increases with the increment of  $m$ . That is, when the LP of base approximations are i.i.d., the more approximations we use, the lower data complexity we require for the same ratio of weak keys. Given a fixed capacity (so that a fixed data complexity), the ratio of keys causing a larger capacity than the fixed one increases when more base approximations are used. Thus, the ratio of weak keys resulting in a data complexity lower than the fixed one also increases.

Considering Example 5 again, we take different  $p$ , and fix the same  $\lambda$  (as 1 w.l.o.g.) for each  $m$ . The highest data complexity required for different  $m$ -dimensional linear attacks for  $p$  percent of keys is shown in Table 3.

When the general median complexity  $N_p$  is applied, there is such a question: which  $p$  is more suitable for measuring and comparing the strength of a linear attack. Obviously, it is meaningless to compare  $N_{1/3}$  and  $N_{2/3}$  directly. A natural and simple way is to consider the value of  $\frac{N_p}{p}$  because the division of  $p$  can unify the disparity for different  $N_p$  to a reasonably great extent. For example, if the attack complexity is lower than  $N_{1/3}$  with probability 1/3, then the attack requires to be repeated 3 times for a sufficiently sound success rate. This should be equivalently compared with the case that, let’s say, an attack with complexity lower than  $N_{1/2}$  has to be repeated twice. By confirming the existence of the minimal  $\frac{N_p}{p}$ , we can evaluate different multidimensional linear attacks with the value of  $\min_p \frac{N_p}{p}$ . The results are shown in Table 4.

**Table 3.** The highest data complexity for different  $m$  and different ratios of keys

$m$	2	4	6	8	20
$\log_2(N_{1/3})$	38.864	37.805	37.22	36.813	35.532
$\log_2(N_{1/2})$	39.529	38.253	37.581	37.123	35.727
$\log_2(N_{2/3})$	40.302	38.75	37.974	37.457	35.931

Moreover, comparing  $E_k[\frac{1}{C(k)}]$ ,  $\frac{1}{E_k[C(k)]}$  and the median complexity, we observe that the average complexity is always larger than the median one, and the median complexity is always larger than the reciprocal of average capacity. As  $m$  increases, the difference between these three values decreases. When  $m$  is large enough, these values are approximately equal (see Table 4), since the Gamma and Inverse Gamma distribution turn to be normal distributions.

## 6 Application to Cho's Multidimensional Attack on PRESENT

### 6.1 Cho's Attack on 25-Round PRESENT

The structure of PRESENT [7] makes it vulnerable for a multidimensional attack: there are several strong one-dimensional approximations. The linear hull of each such approximation with non-negligible correlations consists of several equally strong single-bit trails, whose intermediate masks have Hamming weight one. The average LP  $\overline{c^2}$  of all such approximations are  $2^{2(-2r)}L(r)$  [26], where  $L(r)$  is the number of  $r$ -round trails in each approximation. The so far best result for PRESENT is proposed by Cho aiming to 25 rounds [10]. Nine 23-round  $m$ -dimensional linear approximations are used simultaneously, and each of them has the dimension  $m = 8$  starting at one of the S-boxes  $S_i$ ,  $i = 5, 9$  or  $13$  and ending at one of the S-boxes  $S_j$ ,  $j = 5, 6$  or  $7$ . They recover 16 bits of key in the first round and 16 bits of key in the last round. Please refer to [10] for more details of this attack. Cho proved that the average capacity is  $2^{-52.77}$ , and gave the formula of data complexity as in [10]:

**Table 4.** Comparison of the average data complexity, the median data complexity, the reciprocal of average capacity, and  $\min_p \frac{N_p}{p}$ .

$m$	2	4	6	8	20
$\log_2(E_k[\frac{1}{C(k)}])$	$\infty$	39	38	37.41	35.83
$\log_2(N_{1/2})$	39.529	38.253	37.581	37.123	35.727
$\log_2(\frac{1}{E_k[C(k)]})$	39	38	37.41	37	35.68
$\log_2(\min_p \frac{N_p}{p})$	40.44	39.25	38.55	38.04	36.46

$$N = (\sqrt{\text{advantage} \cdot 4 \cdot M} + 4(\Phi^{-1}(2P_s - 1))^2)/C = \lambda/C \quad (2)$$

where  $\Phi$  is the cumulative distribution function of the normal distribution,  $P_s$  is the success probability,  $C(p)$  is the capacity,  $M$  is the number of linear approximations used in the attack. In Eq. (2), if the advantage is equal to  $a$  bits, then the right key candidate should be within the position of  $2^{\ell-a}$ , where  $\ell$  is the number of targeted key bits. Cho chose the  $\lambda = 2^{9.08}$  (advantage is 32 bits,  $M = 9 \cdot (2^8 - 1)$ ,  $P_s = 0.95$ )<sup>4</sup>, and estimated the average data complexity about  $2^{61.85}$ .

## 6.2 Our Investigation on Cho's Attack

We give a simpler but close estimation on the capacity and data complexity of Cho's attack. The authors in [16] claimed that Cho observes in practical experiments that the probability distribution of multidimensional linear approximations varies a lot with the keys, while the capacity remains rather constant. We have shown that the capacity also varies for different keys from theoretical and experimental viewpoints.

In order to attack 25-round PRESENT, 23-round approximations are used, thus  $r = 23$ . According to [26],  $L(23) = 367261713$ , thus  $\overline{c^2} = 2^{-63.55}$ . With Propositions 2 and 3, the fixed-key capacity of 9 8-dimensional approximations is estimated to be  $\Gamma(9 \cdot \frac{2^8-1}{2}, 2^{-62.55})$ . Hence, the average capacity is  $2^{-52.39}$ . With the same  $\lambda$  as Cho, we obtain the data complexity  $N = \frac{2^{9.08}}{C(k)} \sim \text{Inv-Gamma}(9 \cdot \frac{2^8-1}{2}, 2^{71.63})$ . The average data complexity is  $2^{61.47}$ . This result is very close to the estimate in Cho's attack, but easier to compute.

In the same way, we compute the capacity distribution used for 26-round PRESENT, which approximates to  $\Gamma(9 \cdot \frac{2^8-1}{2}, 2^{-65.16})$ . With the knowledge of distributions, we can derive the exact number of weak keys corresponding to different attack scenarios. Using Cho's attack method by taking  $\lambda = 2^{7.58}$  (advantage is 4 bits,  $P_s = 0.8$ ), there are now  $2^{123.24}$  (3.7% in the whole key space) weak keys with capacity larger than  $2^{-54.92}$ . That means, for  $2^{123.24}$  keys out of  $2^{128}$  keys, 26-round PRESENT can be attacked using less than  $2^{62.5}$  plaintext/ciphertext pairs, with success probability 0.8.

## 7 Conclusion and Further Work

In this paper, we deal with the multidimensional linear attacks using  $m$  base approximations with i.i.d. correlations (linear probability). We focus more on the case where the base linear approximations can be regarded as statistically independent. In this case, we point out that the capacity of multidimensional linear approximations satisfies a Gamma distribution, which also leads to an exact Inverse Gamma distribution for the data complexity. Both distributions are parametrized by the dimension and the average linear probability of each approximation. These theoretical results have been verified by experiments on PRESENT. We establish

<sup>4</sup> This result is slightly different from [10], since Eq. (2) is slightly corrected in [16] and our computation uses the corrected formula.

an explicit connection between the fixed-key behaviour and the average behaviour. Based on the distributions, several fundamental issues are discussed in more detail. Multidimensional linear attacks not only benefit from data complexity, but also offer more convenience for measuring the average data complexity due to the fact that the ratio of keys with capacity going to zero decreases with the increase of dimension. The relation of the median and average data complexity, as well as the inverse of average capacity is derived. When the dimension is large enough, these three values are infinitely close. We also propose a modified key equivalent hypothesis that is more suitable for practical situations. Finally, the multidimensional linear attack on 25- and 26-round PRESENT is analyzed based on our theoretical result.

In future work, more complicated cases about the relations of LP distributions should be studied, which may bring more precise evaluation on multidimensional attacks. The measure of  $\frac{N_p}{p}$  can be extended to single linear attacks. Moreover, given the close relation between statistical saturation attacks and multidimensional linear attacks, our results may allow a clearer understanding for the capacity of statistical saturation attacks, whose key-dependent performance still lacks accurate measurement.

## A Appendix - Proof of Lemma 7

**Lemma 7.** *For an  $m$ -dimensional linear approximation with the probability distribution  $p_\eta(k)$  i.i.d. to the normal distribution  $\mathcal{N}(2^{-m}, \sigma^2)$ ,  $\eta = 0, \dots, 2^m - 1$ , the correlations  $c_a(k)$  ( $a \in \mathbb{F}_2^m$ ,  $a \neq 0$ ) of the involved  $2^m - 1$  approximations are identically distributed.*

*Proof.* According to Lemma 1, for  $a \neq 0$ ,

$$\begin{aligned} c_a(k) &= \sum_{\eta \in \mathbb{F}_2^m} (-1)^{a \cdot \eta} p_\eta(k) \\ &= \sum_{\eta \in \mathbb{F}_2^m} (-1)^{a \cdot \eta} (p_\eta(k) - 2^{-m} + 2^{-m}) \\ &= \sum_{\eta \in \mathbb{F}_2^m} (-1)^{a \cdot \eta} (p_\eta(k) - 2^{-m}) + \sum_{\eta \in \mathbb{F}_2^m} (-1)^{a \cdot \eta} 2^{-m} \end{aligned}$$

As  $p_\eta(k)$  are i.i.d. to the normal distribution  $\mathcal{N}(2^{-m}, \sigma^2)$ ,  $p_\eta(k) - 2^{-m}$  are i.i.d. to  $\mathcal{N}(0, \sigma^2)$ . Thus,

$$\sum_{\eta \in \mathbb{F}_2^m} (-1)^{a \cdot \eta} (p_\eta(k) - 2^{-m}) \sim \mathcal{N}(0, 2^m \sigma^2)$$

As  $\sum_{\eta \in \mathbb{F}_2^m} (-1)^{a \cdot \eta} 2^{-m}$  is equal to 0,  $c_a(k)$  are identically distributed to the normal distribution  $\mathcal{N}(0, 2^m \sigma^2)$ , where  $a \in \mathbb{F}_2^m$  and  $a \neq 0$ .  $\square$

## B Appendix - Error Bound of Proposition 2

In Proposition 2, the binary random variables associated to the base approximations are statistically independent, for each fixed key. According to Piling-up Lemma, the LP of combined approximations is equal to the multiplication of the corresponding base LPs. Thus, the accurate capacity is the summation of LP of all base and combined approximations (see Lemma 2):

$$\begin{aligned}
 C(k) &= LP_1(k) + \dots + LP_m(k) + LP_1(k) \times LP_2(k) + \dots + LP_1(k) \times LP_2(k) \times \dots \times LP_m(k) \\
 &= \prod_{i=1}^m (LP_i(k) + 1) - 1,
 \end{aligned}$$

while our approximated capacity in Proposition 2 is  $\sum_{i=1}^m LP_i(k)$ . Their difference is

$$\begin{aligned}
 &\prod_{i=1}^m (LP_i(k) + 1) - 1 - \sum_{i=1}^m LP_i(k) \\
 &< \left( \frac{\sum_{i=1}^m LP_i(k) + m}{m} \right)^m - 1 - \sum_{i=1}^m LP_i(k)
 \end{aligned}$$

In practical attacks,  $LP_i(k) \ll 1$  is natural and reasonable. Denote  $\sum_{i=1}^m LP_i(k)$  as  $A$ , and  $A \ll 1$ . The above formula can be written as

$$\begin{aligned}
 \left( \frac{A + m}{m} \right)^m - 1 - A &= \left( \frac{A}{m} + 1 \right)^m - 1 - A = 1 + \sum_{i=1}^m \frac{C_m^i}{m^i} A^i - 1 - A \\
 &= \sum_{i=2}^m \frac{C_m^i}{m^i} A^i < \sum_{i=2}^m A^i < (m - 1)A^2
 \end{aligned}$$

In our case,  $A$  is a random variable distributed to  $\Gamma(\frac{m}{2}, 2\bar{c}^2)$ . The expected value of  $A$ ,  $E(A)$ , is  $m\bar{c}^2$ . The variance of  $A$ ,  $D(A)$ , is  $m/2 \times (2\bar{c}^2)^2$ . The expected value of  $A^2$ ,  $E(A^2)$ , is equal to  $D(A) + [E(A)]^2$ , i.e.,

$$\begin{aligned}
 E(A^2) &= D(A) + [E(A)]^2 \\
 &= m(m + 2)(\bar{c}^2)^2
 \end{aligned}$$

Thus, the expected value of the error is less than  $(m - 1)m(m + 2)(\bar{c}^2)^2$ , which is reasonably smaller than the expected value of our approximated capacity,  $m\bar{c}^2$ . As we target towards attacking more and more rounds of the cipher, in average  $\bar{c}^2$  tends to be close to the inverse of the message space, for example,  $2^{-64}$ , meaning that the error is negligible in this case.

## References

1. Abdelraheem, M.A., Ågren, M., Beelen, P., Leander, G.: On the distribution of linear biases: three instructive examples. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 50–67. Springer, Heidelberg (2012)

2. Baignères, T., Vaudenay, S.: The complexity of distinguishing distributions (invited talk). In: Safavi-Naini, R. (ed.) ICITS 2008. LNCS, vol. 5155, pp. 210–222. Springer, Heidelberg (2008)
3. Biham, E.: On matsui’s linear cryptanalysis. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 341–355. Springer, Heidelberg (1995)
4. Biryukov, A., De Cannière, C., Quisquater, M.: On multiple linear approximations. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 1–22. Springer, Heidelberg (2004)
5. Blondeau, C., Bogdanov, A., Leander, G.: Bounds in shallows and in miseries. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 204–221. Springer, Heidelberg (2013)
6. Blondeau, C., Nyberg, K.: New links between differential and linear cryptanalysis. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 388–404. Springer, Heidelberg (2013)
7. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsoe, C.: PRESENT: an ultra-lightweight block cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 450–466. Springer, Heidelberg (2007)
8. Bogdanov, A., Leander, G., Nyberg, K., Wang, M.: Integral and multidimensional linear distinguishers with correlation zero. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 244–261. Springer, Heidelberg (2012)
9. Bogdanov, A., Tischhauser, E.: On the wrong key randomisation and key equivalence hypotheses in matsui’s algorithm 2. In: Moriai, S. (ed.) FSE 2013. LNCS, vol. 8424, pp. 19–38. Springer, Heidelberg (2014)
10. Cho, J.Y.: Linear cryptanalysis of reduced-round PRESENT. In: Pieprzyk, J. (ed.) CT-RSA 2010. LNCS, vol. 5985, pp. 302–317. Springer, Heidelberg (2010)
11. Cho, J.Y., Hermelin, M., Nyberg, K.: A new technique for multidimensional linear cryptanalysis with applications on reduced round serpent. In: Lee, P.J., Cheon, J.H. (eds.) ICISC 2008. LNCS, vol. 5461, pp. 383–398. Springer, Heidelberg (2009)
12. Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. Information Security and Cryptography. Springer (2002)
13. Daemen, J., Rijmen, V.: Probability distributions of correlation and differentials in block ciphers. *J. Math. Cryptology* **1**, 221–242 (2007)
14. Harpes, C., Kramer, G.G., Massey, J.L.: A generalization of linear cryptanalysis and the applicability of matsui’s piling-up lemma. In: Guillou, L.C., Quisquater, J.-J. (eds.) EUROCRYPT 1995. LNCS, vol. 921, pp. 24–38. Springer, Heidelberg (1995)
15. Hermelin, M., Cho, J.Y., Nyberg, K.: Multidimensional linear cryptanalysis of reduced round serpent. In: Mu, Y., Susilo, W., Seberry, J. (eds.) ACISP 2008. LNCS, vol. 5107, pp. 203–215. Springer, Heidelberg (2008)
16. Hermelin, M., Nyberg, K.: Linear cryptanalysis using multiple linear approximations. In: Junod, P., Canteaut, A. (eds.) Advanced Linear Cryptanalysis of Block and Stream Ciphers, IOS Press (2011)
17. Kaliski Jr., B.S., Robshaw, M.J.B.: Linear cryptanalysis using multiple approximations. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 26–39. Springer, Heidelberg (1994)
18. Lai, X., Massey, J.L.: Markov ciphers and differential cryptanalysis. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 17–38. Springer, Heidelberg (1991)
19. Leander, G.: On linear hulls, statistical saturation attacks, PRESENT and a cryptanalysis of PUFFIN. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 303–322. Springer, Heidelberg (2011)

20. Matsui, M.: Linear cryptanalysis method for DES cipher. In: Helleseht, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 386–397. Springer, Heidelberg (1994)
21. Murphy, S.: The independence of linear approximations in symmetric cryptanalysis. *IEEE Trans. Inf. Theory* **52**, 5510–5518 (2006)
22. Murphy, S.: The effectiveness of the linear hull effect. *J. Math. Cryptology* **6**, 137–148 (2012)
23. Nguyen, P.H., Wei, L., Wang, H., Ling, S.: On multidimensional linear cryptanalysis. In: Steinfeld, R., Hawkes, P. (eds.) ACISP 2010. LNCS, vol. 6168, pp. 37–52. Springer, Heidelberg (2010)
24. Nyberg, K.: Linear approximation of block ciphers. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 439–444. Springer, Heidelberg (1995)
25. Nyberg, K.: Correlation theorems in cryptanalysis. *Discrete Appl. Math.* **111**, 177–188 (2001)
26. Ohkuma, K.: Weak keys of reduced-round PRESENT for linear cryptanalysis. In: Jacobson Jr., M.J., Rijmen, V., Safavi-Naini, R. (eds.) SAC 2009. LNCS, vol. 5867, pp. 249–265. Springer, Heidelberg (2009)



# Observations on the SIMON Block Cipher Family

Stefan Kölbl<sup>1</sup>, Gregor Leander<sup>2</sup>(✉), and Tyge Tiessen<sup>1</sup>

<sup>1</sup> DTU Compute, Technical University of Denmark, Lyngby, Denmark  
`{stek,tyti}@dtu.dk`

<sup>2</sup> Horst Görtz Institute for IT Security,  
Ruhr-Universität Bochum, Bochum, Germany  
`gregor.leander@rub.de`

**Abstract.** In this paper we analyse the general class of functions underlying the SIMON block cipher. In particular, we derive efficiently computable and easily implementable expressions for the exact differential and linear behaviour of SIMON-like round functions.

Following up on this, we use those expressions for a computer aided approach based on SAT/SMT solvers to find both optimal differential and linear characteristics for SIMON. Furthermore, we are able to find all characteristics contributing to the probability of a differential for SIMON32 and give better estimates for the probability for other variants.

Finally, we investigate a large set of SIMON variants using different rotation constants with respect to their resistance against differential and linear cryptanalysis. Interestingly, the default parameters seem to be not always optimal.

**Keywords:** SIMON · Differential cryptanalysis · Linear cryptanalysis · Block cipher · Boolean functions

## 1 Introduction

Lightweight cryptography studies the deployment of cryptographic primitives in resource-constrained environments. This research direction is driven by a demand for cost-effective, small-scale communicating devices such as RFID tags that are a cornerstone in the Internet of Things. Most often the constrained resource is taken to be the chip-area but other performance metrics such as latency [7], code-size [2] and ease of side-channel protection [12] have been considered as well. Some of these criteria were already treated in NOEKEON [9].

The increased importance of lightweight cryptography and its applications has lately been reflected in the NSA publishing two dedicated lightweight cipher families: SIMON and SPECK [5]. Considering that this is only the third time within four decades that the NSA has published a block cipher, this is quite remarkable. Especially as NIST has started shortly after this publication to investigate the possibilities to standardise lightweight primitives, SIMON and

SPECK certainly deserve a thorough investigation. This is emphasised by the fact that, in contrast to common practice, neither a security analysis nor a justification of the daesign choices were published by the NSA. This lack of openness necessarily gives rise to curiosity and caution.

In this paper we focus on the SIMON family of block ciphers; an elegant, innovative and very efficient set of block ciphers. There exists already a large variety of papers, mainly focussed on evaluating SIMON’s security with regard to linear and differential cryptanalysis. Most of the methods used therein are rather ad-hoc, often only using approximative values for the differential round probability and in particular for the linear square correlation of one round.

**Our Contribution.** With this study, we complement the existing work three-fold. Firstly we develop an exact closed form expression for the differential probability and a  $\log(n)$  algorithm for determining the square correlation over one round. Their accuracy is proven rigorously. Secondly we use these expressions to implement a model of differential and linear characteristics for SAT/SMT solvers which allows us to find the provably best characteristics for different instantiations of SIMON. Furthermore we are able to shed light on how differentials in SIMON profit from the collapse of many differential characteristics. Thirdly by generalising the probability expressions and the SAT/SMT model, we are able to compare the quality of different parameter sets with respect to differential and linear cryptanalysis.

As a basis for our goal to understand both the security of SIMON as well as the choice of its parameter set, we rigorously derive formulas for the differential probabilities and the linear square correlations of the SIMON-like round function that can be evaluated in constant time and time linear in the word size respectively. More precisely, we study differential probabilities and linear correlations of functions of the form

$$S^a(x) \odot S^b(x) + S^c(x)$$

where  $S^i(x)$  corresponds to a cyclic left shift of  $x$  and  $\odot$  denotes the bitwise AND operation.

We achieve this goal by first simplifying this question by considering equivalent descriptions both of the round function as well as the whole cipher (cf. Sect. 2.4). These simplifications, together with the theory of quadratic boolean functions, result in a clearer analysis of linear and differential properties (cf. Sects. 3 and 4). Importantly, the derived simple equations for computing the probabilities of the SIMON round function can be evaluated efficiently and, more importantly maybe, are conceptually very easy. This allows them to be easily used in computer-aided investigations of differential and linear properties over more rounds. It should be noted here that the expression for linear approximations is more complex than the expression for the differential case. However, with respect to the running time of the computer-aided investigations this difference is negligible.

We used this to implement a framework based on SAT/SMT solvers to find the provably best differential and linear characteristics for various instantiations of SIMON (cf. Sect. 5, in particular Table 1). Furthermore we are able to shed light on how differentials in SIMON profit from the collapse of many differential characteristics by giving exact distributions of the probabilities of these characteristics for chosen differentials. The framework is open source and publicly available to encourage further research [13].

In Sect. 6 we apply the developed theory and tools to investigate the design space of SIMON-like functions. In particular, using the computer-aided approach, we find that the standard SIMON parameters are not optimal with regard to the best differential and linear characteristics.

As a side result, we improve the probabilities for the best known differentials for several variants and rounds of SIMON. While this might well lead to (slightly) improved attacks, those improved attacks are out of the scope of our work.

Interestingly, at least for SIMON32 our findings indicate that the choices made by the NSA are good but not optimal under our metrics, leaving room for further investigations and questions. To encourage further research, we propose several alternative parameter choices for SIMON32. Here, we are using the parameters that are optimal when restricting the criteria to linear, differential and dependency properties. We encourage further research on those alternative choices to shed more light on the undisclosed design criteria.

We also like to point out that the SIMON key-scheduling was not part of our investigations. Its influence on the security of SIMON is left as an important open question for further investigations. In line with this, whenever we investigate multi-round properties of SIMON in our work, we implicitly assume independent round keys in the computation of probabilities.

Finally, we note that most of our results can be applied to more general constructions, where the involved operations are restricted to AND, XOR, and rotations.

**Related Work.** There are various papers published on the cryptanalysis of SIMON [1, 3, 6, 17–19]. The most promising attacks so far are based on differential and linear cryptanalysis, however a clear methodology of how to derive the differential probabilities and square correlations seems to miss in most cases. Biryukov, Roy and Velichkov [6] derive a correct, but rather involved method to find the differential probabilities. Abed, List, Lucks and Wenzel [1] state an algorithm for the calculation of the differential probabilities but without further explanation. For the calculation of the square correlations an algorithm seems to be missing all together.

Previous work also identifies various properties like the strong differential effect and give estimate of the probability of differentials.

The concept behind our framework was previously also applied on the ARX cipher Salsa20 [14] and the CAESAR candidate NORX [4]. In addition to the applications proposed in previous work we extend it for linear cryptanalysis, examine the influence of rotation constants and use it to compute the distribution of characteristics corresponding to a differential.

## 2 Preliminaries

In this section, we start by defining our notation and giving a short description of the round function. We recall suitable notions of equivalence of Boolean functions that allow us to simplify our investigations of SIMON-like round functions. Most of this section is generally applicable to AND-RX constructions, i.e. constructions that only make use of the bitwise operations AND, XOR, and rotations.

### 2.1 Notation

We denote by  $\mathbb{F}_2$  the field with two elements and by  $\mathbb{F}_2^n$  the  $n$ -dimensional vector space over  $\mathbb{F}_2$ . By  $\mathbf{0}$  and  $\mathbf{1}$  we denote the vectors of  $\mathbb{F}_2^n$  with all 0s and all 1s respectively. The Hamming weight of a vector  $a \in \mathbb{F}_2^n$  is denoted as  $\text{wt}(a)$ . By  $\mathbb{Z}_n$  we denote the integers modulo  $n$ .

The addition in  $\mathbb{F}_2^n$ , i.e. bit-wise XOR, is denoted by  $+$ . By  $\odot$  we denote the AND operation in  $\mathbb{F}_2^n$ , i.e. multiplication over  $\mathbb{F}_2$  in each coordinate:

$$x \odot y = (x_i y_i)_i.$$

By  $\vee$  we denote the bitwise OR operation. By  $\bar{x}$  we denote the bitwise negation of  $x$ , i.e.  $\bar{x} := (x + \mathbf{1})$ . We denote by  $S^i : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$  the left circular shift by  $i$  positions. We also note that any arithmetic of bit indices is always done modulo the word size  $n$ .

In this paper we are mainly concerned with functions of the form

$$f_{a,b,c}(x) = S^a(x) \odot S^b(x) + S^c(x) \tag{1}$$

and we identify such functions with its triple  $(a, b, c)$  of parameters.

For a vectorial Boolean function on  $n$  bits,  $f : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ , we denote by

$$\widehat{f}(\alpha, \beta) = \sum_x \mu(\langle \beta, f \rangle + \langle \alpha, x \rangle)$$

the Walsh (or Fourier) coefficient with input mask  $\alpha$  and output mask  $\beta$ . Here we use  $\mu(x) = (-1)^x$  to simplify the notation.

The corresponding squared correlation of  $f$  is given by

$$C^2(\alpha \rightarrow \beta) = \left( \frac{\widehat{f}(\alpha, \beta)}{2^n} \right)^2.$$

For differentials we similarly denote by  $\text{Pr}(\alpha \rightarrow \beta)$  the probability that a given input difference  $\alpha$  results in a given output difference  $\beta$ , i.e.

$$\text{Pr}(\alpha \rightarrow \beta) = \frac{|\{x \mid f(x) + f(x + \alpha) = \beta\}|}{2^n}.$$

Furthermore,  $\text{Dom}(f)$  is the domain of a function  $f$ ,  $\text{Im}(f)$  is its image.

### 2.2 Description of SIMON

SIMON is a family of lightweight block ciphers with block sizes 32, 48, 64, 96, and 128 bits. The constructions are Feistel ciphers using a word size  $n$  of 16, 24, 32, 48 or 64 bits, respectively. We will denote the variants as SIMON $2n$ . The key size varies between of 2, 3, and 4  $n$ -bit words. The round function of SIMON is composed of AND, rotation, and XOR operations on the complete word (see Fig. 1). More precisely, the round function in SIMON corresponds to

$$S^8(x) \odot S^1(x) + S^2(x),$$

that is to the parameters (8, 1, 2) for  $f$  as given in Eq. (1). As we are not only interested in the original SIMON parameters, but in investigating the entire design space of SIMON-like functions, we denote by

$$\text{SIMON}[a, b, c]$$

the variant of SIMON where the original round function is replaced by  $f_{a,b,c}$  (cf. Eq. (1)).

As it is out of scope for our purpose, we refer to [5] for the description of the key-scheduling.

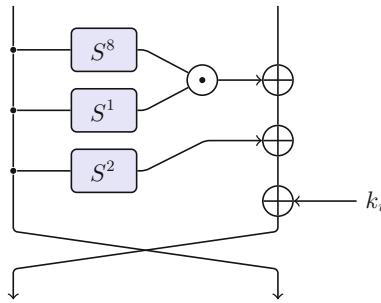


Fig. 1. The round function of SIMON

### 2.3 Affine Equivalence of Boolean Functions

Given two (vectorial) Boolean functions  $f_1$  and  $f_2$  on  $\mathbb{F}_2^n$  related by

$$f_1(x) = (A \circ f_2 \circ B)(x) + C(x)$$

where  $A$  and  $B$  are affine permutations and  $C$  is an arbitrary affine mapping on  $\mathbb{F}_2^n$  we say that  $f_1$  and  $f_2$  are *extended affine equivalent* (cf. [8] for a comprehensive survey).

With respect to differential cryptanalysis, if  $f_1$  and  $f_2$  are extended affine equivalent then the differential  $\alpha \xrightarrow{f_1} \beta$  over  $f_1$  has probability  $p$  if and only if the differential

$$B(\alpha) \xrightarrow{f_2} A^{-1}(\beta + C(\alpha))$$

over  $f_2$  has probability  $p$  as well.

For linear cryptanalysis, a similar relation holds for the linear correlation. If  $f_1$  and  $f_2$  are related as defined above, it holds that

$$\widehat{f}_1(\alpha, \beta) = \widehat{f}_2\left((C \circ B^{-1})^T \beta + (B^{-1})^T \alpha, A^T \beta\right).$$

Thus up to linear changes we can study  $f_2$  instead of  $f_1$  directly. Note that, for an actual attack, these changes are usually critical and can certainly not be ignored. However, tracing the changes is, again, simple linear algebra.

For differential and linear properties of SIMON-like functions of the form

$$f_{a,b,c}(x) = S^a(x) \odot S^b(x) + S^c(x)$$

this implies that it is sufficient to find the differential and linear properties of the simplified variant

$$f(x) = x \odot S^d(x)$$

and then transfer the results back by simply using linear algebra.<sup>1</sup>

## 2.4 Structural Equivalence Classes in AND-RX Constructions

AND-RX constructions, i.e. constructions that make only use of the operations AND ( $\odot$ ), XOR ( $+$ ), and rotations ( $S^r$ ), exhibit a high degree of symmetry. Not only are they invariant under rotation of all input words, output words and constants, they are furthermore structurally invariant under any affine transformation of the bit-indices. As a consequence of this, several equivalent representations of the SIMON variants exist.

Let  $T$  be a permutation of the bits of an  $n$ -bit word that corresponds to an affine transformation of the bit-indices. Thus there are  $s \in \mathbb{Z}_n^*$  and  $t \in \mathbb{Z}_n$  such that bit  $i$  is renamed to  $s \cdot i + t$ . As the AND and XOR operations are bitwise,  $T$  clearly commutes with these:

$$\begin{aligned} Tv \odot Tw &= T(v \odot w) \\ Tv + Tw &= T(v + w) \end{aligned}$$

where  $v$  and  $w$  are  $n$ -bit words. A rotation to the left by  $r$  can be written bitwise as  $S^r(v)_i = v_{i-r}$ . For a rotation, we thus get the following bitwise relation after transformation with  $T$

$$S^r(v)_{s \cdot i + t} = v_{s \cdot (i-r) + t} = v_{s \cdot i + t - s \cdot r}.$$

Substituting  $s \cdot i + t$  with  $j$  this is the same as

$$S^r(v)_j = v_{j-s \cdot r}.$$

---

<sup>1</sup> Note that we can transform the equation  $f(x) = S^a(x) \odot S_b(x) + S^c(x)$  to the equation  $S^{-a}(f(x)) + S^{c-a}(x) = x \odot S^{b-a}(x)$ .

Thus the rotation by  $r$  has been changed to a rotation by  $s \cdot r$ . Thus we can write

$$TS^r v = S^{s \cdot r} T v.$$

Commuting the linear transformation of the bit-indices with a rotation thus only changes the rotation constant by a factor. In the special case where all input words, output words and constants are rotated, which corresponds to the case  $s = 1$ , the rotation constants are left untouched.

To summarize the above, when applying such a transformation  $T$  to all input words, output words and constants in an AND-RX construction, the structure of the constructions remains untouched apart from a multiplication of the rotation constants by the factor  $s$ .

This means for example for SIMON32 that changing the rotation constants from  $(8, 1, 2)$  to  $(3 \cdot 8, 3 \cdot 1, 3 \cdot 2) = (8, 3, 6)$  and adapting the key schedule accordingly gives us the same cipher apart from a bit permutation. As  $s$  has to be coprime to  $n$ , all  $s$  with  $\gcd(s, n) = 1$  are allowed, giving  $\varphi(n)$  equivalent tuples of rotation constants in each equivalence class where  $\varphi$  is Euler's phi function.

Together with the result from Sect. 2.3, this implies the following lemma.

**Lemma 1.** *Any function  $f_{a,b,c}$  as defined in Eq. (1) is extended affine equivalent to a function*

$$f(x) = x \odot S^d(x)$$

where  $d|n$  or  $d = 0$ .

When looking at differential and square correlations of SIMON-like round functions this means that it is sufficient to investigate this restricted set of functions. The results for these functions can then simply be transferred to the general case.

### 3 Differential Probabilities of SIMON-like Round Functions

In this section, we derive a closed expression for the differential probability for all SIMON-like round functions, i.e. all functions as described in Eq. (1). The main ingredients here are the derived equivalences and the observation that any such function is quadratic. Being quadratic immediately implies that its derivative is linear and thus the computation of differential probabilities basically boils down to linear algebra (cf. Theorem 1). However, to be able to efficiently study multiple-round properties and in particular differential characteristics, it is important to have a simple expression for the differential probabilities. Those expressions are given for  $f(x) = x \odot S^1(x)$  in Theorem 2 and for the general case in Theorem 3.

#### 3.1 A Closed Expression for the Differential Probability

The following statement summarises the differential properties of the  $f$  function.

**Theorem 1.** *Given an input difference  $\alpha$  and an output difference  $\beta$  the probability  $p$  of the corresponding differential (characteristic) for the function  $f(x) = x \odot S^a(x)$  is given by*

$$p_{\alpha,\beta} = \begin{cases} 2^{-(n-d)} & \text{if } \beta + \alpha \odot S^a(\alpha) \in \text{Im}(L_\alpha) \\ 0 & \text{else} \end{cases}$$

where

$$d = \dim \ker(L_\alpha)$$

and

$$L_\alpha(x) = x \odot S^a(\alpha) + \alpha \odot S^a(x)$$

*Proof.* We have to count the number of solutions to the equation

$$f(x) + f(x + \alpha) = \beta.$$

This simplifies to

$$L_\alpha(x) = x \odot S^a(\alpha) + \alpha \odot S^a(x) = \beta + \alpha \odot S^a(\alpha)$$

As this is an affine equation, it either has zero solutions or the number of solutions equals the kernel size, i.e. the number of elements in the subspace

$$\{x \mid x \odot S^a(\alpha) + \alpha \odot S^a(x) = \mathbf{0}\}.$$

Clearly, the equation has solutions if and only if  $\beta + \alpha \odot S^a(\alpha)$  is in the image of  $L_\alpha$ . □

Next we present a closed formula to calculate the differential probability in the case where  $a = 1$ . Furthermore we restrict ourselves to the case where  $n$  is even.

**Theorem 2.** *Let*

$$\text{varibits} = S^1(\alpha) \vee \alpha$$

and

$$\text{doublebits} = \alpha \odot \overline{S^1(\alpha)} \odot S^2(\alpha).$$

*Then the probability that difference  $\alpha$  goes to difference  $\beta$  is*

$$P(\alpha \rightarrow \beta) = \begin{cases} 2^{-n+1} & \text{if } \alpha = \mathbf{1} \text{ and } \text{wt}(\beta) \equiv 0 \pmod{2} \\ 2^{-\text{wt}(\text{varibits}+\text{doublebits})} & \text{if } \alpha \neq \mathbf{1} \text{ and } \beta \odot \overline{\text{varibits}} = \mathbf{0} \\ & \text{and } (\beta + S^1(\beta)) \odot \text{doublebits} = \mathbf{0} \\ 0 & \text{else} \end{cases}$$



*Proof.* According to Theorem 1, we need to prove two things. Firstly we need to prove that the rank of  $L_\alpha$  (i.e.  $n - \dim \ker L_\alpha$ ) is  $n - 1$  when  $\alpha = \mathbf{1}$ , and  $\text{wt}(\text{varibits} + \text{doublebits})$  otherwise. Secondly we need to prove that  $\beta + \alpha \odot S^1(\alpha) \in \text{lmg}(L_\alpha)$  iff  $\text{wt}(\beta) \equiv 0 \pmod 2$  when  $\alpha = \mathbf{1}$ , and that  $\beta + \alpha \odot S^1(\alpha) \in \text{lmg}(L_\alpha)$  iff  $\beta \odot \text{varibits} = \mathbf{0}$  and  $(\beta + S^1(\beta)) \odot \text{doublebits} = \mathbf{0}$  when  $\alpha \neq \mathbf{1}$ .

We first consider the first part. Let us write  $L_\alpha(x)$  in matrix form and let us take  $x$  to be a column vector.  $S^1(\alpha) \odot x$  can be written as  $M_{S^1(\alpha) \odot} x$  with

$$M_{S^1(\alpha) \odot} = \begin{pmatrix} \alpha_{n-1} & \dots & 0 \\ \vdots & \alpha_0 & \vdots \\ \vdots & & \ddots & \vdots \\ 0 & \dots & \alpha_{n-2} \end{pmatrix}. \quad (2)$$

Equivalently we can write  $\alpha \odot x$  and  $S^1(x)$  with matrices as  $M_{\alpha \odot} x$  and  $M_{S^1} x$  respectively where

$$M_{\alpha \odot} = \begin{pmatrix} \alpha_0 & \dots & 0 \\ \vdots & \alpha_1 & \vdots \\ \vdots & & \ddots & \vdots \\ 0 & \dots & \alpha_{n-1} \end{pmatrix} \quad \text{and} \quad M_{S^1} = \begin{pmatrix} 0_{1,n-1} & I_{1,1} \\ I_{n-1,n-1} & 0_{n-1,1} \end{pmatrix}, \quad (3)$$

i.e.  $M_{S^1}$  consists of two identity and two zero submatrices. The result of  $M_{S^1(\alpha) \odot} + M_{\alpha \odot} M_{S^1}$  can now be written as

$$\begin{pmatrix} \alpha_{n-1} & 0 & 0 & \dots & \alpha_0 \\ \alpha_1 & \alpha_0 & 0 & \dots & 0 \\ 0 & \alpha_2 & \alpha_1 & & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \alpha_{n-1} & \alpha_{n-2} \end{pmatrix} \quad (4)$$

Clearly the rank of the matrix is  $n - 1$  when all  $\alpha_i$  are 1. Suppose now that not all  $\alpha_i$  are 1. In that case, a set of non-zero rows is linearly dependent iff there exist two identical rows in the set. Thus to calculate the rank of the matrix, we need to calculate the number of unique non-zero rows.

By associating the rows in the above matrix with the bits in **varibits**, we can clearly see that the number of non-zero rows in the matrices corresponds to the number of 1s in  $\text{varibits} = S^1(\alpha) \vee \alpha$ .

To count the number of non-unique rows, first notice that a nonzero row can only be identical to the row exactly above or below. Suppose now that a non-zero row  $i$  is identical to the row  $(i - 1)$  above. Then  $\alpha_{i-1}$  has to be 0 while  $\alpha_i$  and  $\alpha_{i-2}$  have to be 1. But then row  $i$  cannot simultaneously be identical to row  $(i + 1)$  below. Thus it is sufficient to calculate the number of non-zero rows minus the number of rows that are identical to the row above it to find the rank of the matrix. Noting that row  $i$  is non-zero iff  $\alpha_i \alpha_{i-1}$  and that  $\alpha_i \bar{\alpha}_{i-1} \alpha_{i-2}$  is only

equal 1 when row  $i$  is non-zero and equal to the row above it. Thus calculating the number of  $i$  for which

$$\alpha_i \alpha_{i-1} + \alpha_i \overline{\alpha_{i-1}} \alpha_{i-2}$$

is equal 1 gives us the rank of  $L_\alpha$ . This corresponds to calculating  $\text{wt}(\mathbf{varibits} + \mathbf{doublebits})$ .

For the second part of the proof, we need to prove the conditions that check whether  $\beta + \alpha \odot S^1(\alpha) \in \text{Img}(L_\alpha)$ . First notice that  $\alpha \odot S^1(\alpha)$  is in the image of  $L_\alpha$  (consider for  $x$  the vector with bits alternately set to 0 and 1). Thus it is sufficient to test whether  $\beta$  is in  $\text{Img}L_\alpha$ . Let  $y = L_\alpha(x)$ . Let us first look at the case of  $\alpha = \mathbf{1}$ . Then  $L_\alpha(x) = x + S^1(x)$ . We can thus deduce from bit  $y_i$  whether  $x_i = x_{i-1}$  or  $x_i \neq x_{i-1}$ . Thus the bits in  $y$  create a chain of equalities/inequalities in the bits of  $x$  which can only be fulfilled if there the number of inequalities is even. Hence in that case  $\beta \in \text{Img}L_\alpha$  iff  $\text{wt}(\beta) \equiv 0 \pmod 2$ .

For the case that  $\alpha \neq \mathbf{1}$ , we first note that  $y_i$  has to be zero if the corresponding row  $i$  in the matrix of Eq. (4) is all zeroes. Furthermore following our discussion of this matrix earlier, we see that  $y_i$  is independent of the rest of  $y$  if the corresponding row is linearly independent of the other rows and that  $y_i$  has to be the same as  $y_{i-1}$  if the corresponding rows are identical. Thus we only need to check that the zero-rows of the matrix correspond to zero bits in  $\beta$  and that the bits in  $\beta$  which correspond to identical rows in the matrix are equal. Thus  $\beta$  is in the image of  $L_\alpha$  iff  $\beta \odot \overline{\mathbf{varibits}} = \mathbf{0}$  and  $(\beta + S^1(\beta)) \odot \mathbf{doublebits} = \mathbf{0}$ .  $\square$

### 3.2 The Full Formula for Differentials

Above we treated only the case for the simplified function  $f(x) = x \cdot S^1(x)$ . As mentioned earlier, the general case where  $\text{gcd}(a - b, n) = 1$  can be deduced from this with linear algebra. When  $\text{gcd}(d, n) \neq 1$  though, the function  $f(x) = x \odot S^d(x)$  partitions the output bits into independent classes. This not only raises differential probabilities (worst case  $d = 0$ ), it also makes the notation for the formulas more complex and cumbersome, though not difficult. We thus restrict ourselves to the most important case when  $\text{gcd}(a - b, n) = 1$ . The general formulas are then

**Theorem 3.** *Let  $f(x) = S^a(x) \odot S^b(x) + S^c(x)$ , where  $\text{gcd}(n, a - b) = 1$ ,  $n$  even, and  $a > b$  and let  $\alpha$  and  $\beta$  be an input and an output difference. Then with*

$$\mathbf{varibits} = S^a(\alpha) \vee S^b(\alpha)$$

and

$$\mathbf{doublebits} = S^b(\alpha) \odot \overline{S^a(\alpha)} \odot S^{2a-b}(\alpha)$$

and

$$\gamma = \beta + S^c(\alpha)$$

we have that the probability that difference  $\alpha$  goes to difference  $\beta$  is

$$P(\alpha \rightarrow \beta) = \begin{cases} 2^{-n+1} & \text{if } \alpha = \mathbf{1} \text{ and } \text{wt}(\gamma) \equiv 0 \pmod 2 \\ 2^{-\text{wt}(\text{varibits}+\text{doublebits})} & \text{if } \alpha \neq \mathbf{1} \text{ and } \overline{\gamma \odot \text{varibits}} = \mathbf{0} \\ & \text{and } (\gamma + S^{a-b}(\gamma)) \odot \text{doublebits} = \mathbf{0} \\ 0 & \text{else.} \end{cases}$$

For a more intuitive approach and some elaboration on the differential probabilities, we refer to the ePrint version of this paper.

### 4 Linear Correlations of SIMON-like Round Functions

As in the differential case, for the study of linear approximations, we also build up on the results from Sects. 2.3 and 2.4. We will thus start with studying linear approximations for the function  $f(x) = x \odot S^a(x)$ . Again, the key point here is that all those functions are quadratic and thus their Fourier coefficient, or equivalently their correlation, can be computed by linear algebra (cf. Theorem 4). Theorem 5 is then, in analogy to the differential case, the explicit expression for the linear correlations. It basically corresponds to an explicit formula for the dimension of the involved subspace.

The first result is the following:

**Theorem 4.**

$$\widehat{f}(\alpha, \beta)^2 = \begin{cases} 2^{n+d} & \text{if } \alpha \in U_\beta^\perp \\ 0 & \text{else} \end{cases}$$

where

$$d = \dim U_\beta$$

and

$$U_\beta = \{y \mid \beta \odot S^a(y) + S^{-a}(\beta \odot y) = \mathbf{0}\}$$

*Proof.* We compute

$$\begin{aligned} \widehat{f}(\alpha, \beta)^2 &= \sum_{x,y} \mu(\langle \beta, f(x) + f(y) \rangle + \langle \alpha, x + y \rangle) \\ &= \sum_{x,y} \mu(\langle \beta, f(x) + f(x + y) \rangle + \langle \alpha, y \rangle) \\ &= \sum_{x,y} \mu(\langle \beta, x \odot S^a(x) + (x + y) \odot S^a(x + y) \rangle + \langle \alpha, y \rangle) \\ &= \sum_y \mu(\langle \beta, f(y) \rangle + \langle \alpha, y \rangle) \sum_x \mu(\langle \beta, x \odot S^a(y) + y \odot S^a(x) \rangle) \\ &= \sum_y \mu(\langle \beta, f(y) \rangle + \langle \alpha, y \rangle) \sum_x \mu(\langle x, \beta \odot S^a(y) + S^{-a}(\beta \odot y) \rangle). \end{aligned}$$

Now for the sum over  $x$  only two outcomes are possible,  $2^n$  or zero. More precisely, it holds that

$$\sum_x \mu(\langle x, \beta \odot S^a(y) + S^{-a}(\beta \odot y) \rangle) = \begin{cases} 2^n & \text{if } \beta \odot S^a(y) + S^{-a}(\beta \odot y) = \mathbf{0} \\ 0 & \text{else.} \end{cases}$$

Thus, defining

$$U_\beta = \{y \mid \beta \odot S^a(y) + S^{-a}(\beta \odot y) = \mathbf{0}\}$$

we get

$$\widehat{f}(\alpha, \beta)^2 = 2^n \sum_{y \in U_\beta} \mu(\langle \beta, f(y) \rangle + \langle \alpha, y \rangle).$$

Now as

$$\langle \beta, f(y) \rangle = \langle \beta, y \odot S^a(y) \rangle \tag{5}$$

$$= \langle \mathbf{1}, y \odot \beta \odot S^a(y) \rangle \tag{6}$$

$$= \langle \mathbf{1}, y \odot S^{-a}(\beta \odot y) \rangle \tag{7}$$

Now, the function  $f_\beta := \langle \beta, f(y) \rangle$  is linear over  $U_\beta$  as can be easily seen by the definition of  $U_\beta$ . Moreover, as  $f_\beta$  is unbalanced for all  $\beta$ , it follows that actually  $f_\beta$  is constant zero on  $U_\beta$ . We thus conclude that

$$\widehat{f}(\alpha, \beta)^2 = 2^n \sum_{y \in U_\beta} \mu(\langle \alpha, y \rangle).$$

With a similar argument as above, it follows that  $\widehat{f}(\alpha, \beta)^2$  is non-zero if and only if  $\alpha$  is contained in  $U_\beta^\perp$ . □

Let us now restrict ourselves to the case where  $f(x) = x \odot S^1(x)$ . The general case can be deduced analogously to the differential probabilities. For simplicity we also restrict ourselves to the case where  $n$  is even.

First we need to introduce some notation. Let  $x \in \mathbb{F}_2^n$  with not all bits equal to 1. We now look at blocks of consecutive 1s in  $x$ , including potentially a block that “wraps around” the ends of  $x$ . Let the lengths of these blocks, measured in bits, be denoted as  $c_0, \dots, c_m$ . For example, the bitstring 100101111011 has blocks of length 1, 3, and 4. With this notation define  $\theta(x) := \sum_{i=0}^m \lceil \frac{c_i}{2} \rceil$ .

Noting that the linear square correlation of  $f$  is  $\frac{\widehat{f}(\alpha, \beta)^2}{2^{2n}}$ , we then have the following theorem:

**Theorem 5.** *With the notation from above it holds that the linear square correlation of  $\alpha \xrightarrow{f} \beta$  can be calculated as*

$$C(\alpha \rightarrow \beta) = \begin{cases} 2^{-n+2} & \text{if } \beta = \mathbf{1} \text{ and } \alpha \in U_\beta^\perp \\ 2^{-\theta(\beta)} & \text{if } \beta \neq \mathbf{1} \text{ and } \alpha \in U_\beta^\perp \\ 0 & \text{else.} \end{cases}$$

*Proof.* Define  $L_\beta(x) := \beta \odot S^1(x) + S^{-1}(\beta \odot x)$ . Clearly  $L_\beta$  is linear. Also  $U_\beta = \ker L_\beta(x)$ . Let us determine the rank of this mapping. Define the matrices  $M_\beta$ ,  $M_{S^1}$ , and  $M_{S^{-1}}$  as

$$M_\beta = \begin{pmatrix} \beta_0 & \dots & \dots & 0 \\ \vdots & \beta_1 & & \vdots \\ \vdots & & \ddots & \vdots \\ 0 & \dots & \dots & \beta_{n-1} \end{pmatrix} \quad M_{S^1} = \begin{pmatrix} 0_{1,n-1} & I_{1,1} \\ I_{n-1,n-1} & 0_{n-1,1} \end{pmatrix} \quad (8)$$

$$M_{S^{-1}} = \begin{pmatrix} 0_{n-1,1} & I_{n-1,n-1} \\ I_{1,1} & 0_{1,n-1} \end{pmatrix}$$

We can then write  $L_\beta$  in matrix form as

$$\begin{pmatrix} 0 & \beta_1 & 0 & \dots & 0 & \beta_0 \\ \beta_1 & 0 & \beta_2 & 0 & \dots & 0 \\ 0 & \beta_2 & 0 & \beta_3 & \ddots & \vdots \\ \vdots & & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & \ddots & 0 & \beta_{n-1} \\ \beta_0 & 0 & \dots & 0 & \beta_{n-1} & 0 \end{pmatrix} \quad (9)$$

Clearly, if  $\beta$  is all 1s, the rank of the matrix is  $n - 2$  as  $n$  is even.<sup>2</sup> Let us therefore now assume that  $\beta$  is not all 1s. When we look at a block of 1s in  $\beta$  e.g.,  $\beta_{i-1} = 0$ ,  $\beta_i, \beta_{i+1}, \dots, \beta_{i+l-1} = 1$ , and  $\beta_l = 0$ . Then clearly the  $l$  rows  $i, i + 1, \dots, i + l - 1$  are linearly independent when  $l$  is even. When  $l$  is odd though, the sum of rows  $i, i + 2, i + 4$ , up to row  $i + l - 3$  will equal row  $i + l - 1$ . In that case there are thus only  $l - 1$  linearly independent rows. As the blocks of 1s in  $\beta$  generate independent blocks of rows, we can summarise that the rank of the matrix is exactly  $\theta(\beta)$ .  $\square$

Analogously to the differential probabilities, the linear probabilities in the general case can be derived from this. It is likewise straightforward to derive how to determine whether  $\alpha \in U_\beta^\perp$ . As an explicit formulation of this is rather tedious, we instead refer to the implementation in Python given in the Appendix A where both is achieved in the case where  $\gcd(a - b, n) = 1$  and  $n$  is even.

For a more intuitive approach and some elaboration on the linear probabilities, we refer to the ePrint version of this paper.

## 5 Finding Optimal Differential and Linear Characteristics

While there are various methods for finding good characteristics, determining optimal differential or linear characteristics remains a hard problem in general. The formulas derived for both differential and linear probabilities enable us to apply an algebraic approach to finding the best characteristics. A similar technique has been applied to the ARX cipher Salsa20 [14] and the CAESAR candidate NORX [4]. For finding the optimal characteristics for SIMON we implemented

<sup>2</sup> The rank is  $n - 1$  when  $n$  is odd.

an open source tool [13] based on the SAT/SMT solvers CryptoMiniSat [15] and STP [11].

In the next section we will show how SIMON can be modeled to find both the best differential and linear characteristics in this framework and how this can be used to solve cryptanalytic problems.

### 5.1 Model for Differential Cryptanalysis of SIMON

First we define the variables used in the model of SIMON. We use two  $n$ -bit variables  $x_i, y_i$  to represent the XOR-difference in the left and right halves of the state for each round and an additional variable  $z_i$  to store the XOR-difference of the output of the AND operation.

For representing the  $\log_2$  probability of the characteristic we introduce an additional variable  $w_i$  for each round. The sum over all probabilities  $w_i$  then gives the probability of the corresponding differential characteristic. The values  $w_i$  are computed according to Theorem 3 as

$$w_i = \text{wt}(\text{varibits} + \text{doublebits}) \quad (10)$$

where  $\text{wt}(x)$  is the Hamming weight of  $x$  and

$$\begin{aligned} \text{varibits} &= S^a(x_i) \vee S^b(x_i) \\ \text{doublebits} &= S^b(x_i) \odot \overline{S^a(x_i)} \wedge S^{2a-b}(x_i) \end{aligned}$$

Therefore, for one round of SIMON we get the following set of constraints:

$$\begin{aligned} y_{i+1} &= x_i \\ 0 &= (z_i \odot \text{varibits}) \\ 0 &= (z_i + S^{a-b}(z_i)) \odot \text{doublebits} \\ x_{i+1} &= y_i + z_i + S^c(x_i) \\ w_i &= \text{wt}(\text{varibits} + \text{doublebits}) \end{aligned} \quad (11)$$

A model for linear characteristics, though slightly more complex, can be implemented in a similar way. A description of this model can be found in the implementation of our framework. Despite the increase in complexity, we could not observe any significant impact on the solving time for the linear model.

### 5.2 Finding Optimal Characteristics

We can now use the previous model for SIMON to search for optimal differential characteristics. This is done by formulating the problem of finding a valid characteristic, with respect to our constraints, for a given probability  $w$ . This is important to limit the search space and makes sense as we are usually more interested in differential characteristics with a high probability as they are more promising to lead to attacks with a lower complexity. Therefore, we start with a high probability and check if such a characteristic exists. If not we lower the probability.

The procedure can be described in the following way:

- For each round of the cipher add the corresponding constraints as defined in (11). This system of constraints then exactly describes the form of a valid characteristic for the given parameters.
- Add a condition which accumulates the probabilities of each round as defined in (10) and check if it is equal to our target probability  $w$ .
- Query if there exists an assignment of variables which is satisfiable under the constraints.
- Decrement the probability  $w$  and repeat the procedure.

One of the main advantages compared to other approaches is that we can prove an upper bound on the probability of characteristics for a given cipher and number of rounds. If the solvers determines the set of conditions unsatisfiable, we know that no characteristic with the specified probability exists. We used this approach to determine the characteristics with optimal probability for different variants of SIMON. The results are given in Table 1.

**Table 1.** Overview of the optimal differential (on top) and linear characteristics for different variants of SIMON. The probabilities are given as  $\log_2(p)$ , for linear characteristic the squared correlation is used.

Rounds:	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
<b>Differential</b>															
SIMON32	-2	-4	-6	-8	-12	-14	-18	-20	-25	-30	-34	-36	-38	-40	-42
SIMON48	-2	-4	-6	-8	-12	-14	-18	-20	-26	-30	-35	-38	-44	-46	-50
SIMON64	-2	-4	-6	-8	-12	-14	-18	-20	-26	-30	-36	-38	-44	-48	-54
<b>Linear</b>															
SIMON32	-2	-4	-6	-8	-12	-14	-18	-20	-26	-30	-34	-36	-38	-40	-42
SIMON48	-2	-4	-6	-8	-12	-14	-18	-20	-26	-30	-36	-38	-44	-46	-50
SIMON64	-2	-4	-6	-8	-12	-14	-18	-20	-26	-30	-36	-38	-44	-48	-54

**Upper Bound for the Characteristics.** During our experiments we observed that it seems to be an easy problem for the SMT/SAT solver to prove the absence of differential characteristics above  $w_{\max}$ . This can be used to get a lower bound on the probability of characteristics contributing to the differential. The procedure is similar to finding the optimal characteristics.

- Start with a very low initial probability  $w_i$ .
- Add the same system of constraints which were used for finding the characteristic.
- Add a constraint fixing the variables  $(x_0, y_0)$  to  $\Delta_{\text{in}}$  and  $(x_r, y_r)$  to  $\Delta_{\text{out}}$ .
- Query if there is a solution for this weight.
- Increase the probability  $w_i$  and repeat the procedure until a solution is found.

### 5.3 Computing the Probability of a Differential

Given a differential characteristic it is of interest to determine the probability of the associated differential  $\Pr(\Delta_{\text{in}} \xrightarrow{f^r} \Delta_{\text{out}})$  as it might potentially have a much higher probability than the single characteristic. It is often assumed that the probability of the best characteristic can be used to approximate the probability of the best differential. However, this assumption only gives an inaccurate estimate in the case of SIMON.

Similarly to the previous approach for finding the characteristic, we can formalise the problem of finding the probability of a given differential in the following way:

- Add the same system of constraints which were used for finding the characteristic.
- Add a constraint fixing the variables  $(x_0, y_0)$  to  $\Delta_{\text{in}}$  and  $(x_r, y_r)$  to  $\Delta_{\text{out}}$ .
- Use a SAT solver to find **all** solutions  $s_i$  for the probability  $w$ .
- Decrement the probability  $w$  and repeat the procedure.

The probability of the differential is then given by

$$\Pr(\Delta_{\text{in}} \xrightarrow{f^r} \Delta_{\text{out}}) = \sum_{i=w_{\min}}^{w_{\max}} s_i \cdot 2^{-i} \quad (12)$$

where  $s_i$  is the number of characteristics with a probability of  $2^{-i}$ .

We used this approach to compute better estimates for the probability of various differentials (see Table 2). In the case of SIMON32 we were able to find *all* characteristics contributing to the differentials for 13 and 14 rounds. The distribution of the characteristics and accumulated probability of the differential is given in Fig. 2. It is interesting to see that the distribution of  $w$  in the range [55, 89] is close to uniform and therefore the probability of the corresponding differential improves only negligibly and converges quickly towards the measured probability<sup>3</sup>.

The performance of the whole process is very competitive compared to dedicated approaches. Enumerating all characteristics up to probability  $2^{-46}$  for the 13-round SIMON32 differential takes around 90 seconds on a single CPU core and already gives a better estimate compared to the results in [6]. A complete enumeration of all characteristics for 13-round SIMON32 took close to one core month using CryptoMiniSat4 [15]. The computational effort for other variants of SIMON is comparable given the same number of rounds. However, for these variants we can use differentials with a lower probability covering more rounds due to the increased block size. In this case the running time increases due to the larger interval  $[w_{\min}, w_{\max}]$  and higher number of rounds.

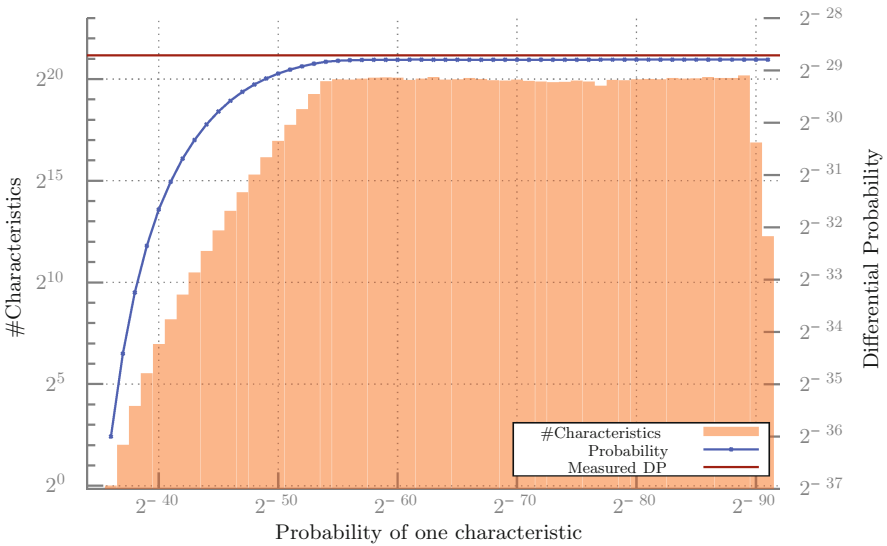
For SIMON48 and SIMON64 we are able to improve the estimate given in [16]. Additionally we found differentials which can cover 17 rounds for SIMON48 and

<sup>3</sup> We encrypted all  $2^{32}$  possible texts under 100 random keys to obtain the estimate of the probability for 13-round SIMON32.



22 rounds for SIMON64 which might have potential to improve previous attacks. Our results are also closer to the experimentally obtained estimates given in [10] but give a slightly lower probability. This can be due to the limited number of characteristics we use for the larger SIMON variants or the different assumptions on the independence of rounds.

Our results are limited by the available computing power and in general it seems to be difficult to count all characteristics for weights in  $[w_{\min}, w_{\max}]$ , especially for the larger variants of SIMON. However the whole process is embarrassingly parallel, as one can split up the computation for each probability  $w_i$ . Furthermore, the improvement that one gets decreases quickly. For all differentials we observed that the distribution of differential characteristics becomes flat after a certain point.



**Fig. 2.** Distribution of the number of characteristics for the differential  $(0, 40) \rightarrow (4000, 0)$  for 13-round SIMON32 and the accumulated probability. A total of  $\approx 2^{25.21}$  characteristics contribute to the probability.

## 6 Analysis of the Parameter Choices

The designers of SIMON so far gave no justification for their choice of the rotation constants. Here we evaluate the space of rotation parameters with regard to different metrics for the quality of the parameters. Our results are certainly not a definite answer but are rather intended as a starting point to evaluating the design space and reverse engineering the design choices. We consider all possible sets of rotation constants  $(a, b, c)$ <sup>4</sup> and checked them for diffusion properties and the optimal differential and linear characteristics.

<sup>4</sup> Without lack of generality, we assume though that  $a \geq b$ .

**Table 2.** Overview of the differentials and the range  $[w_{\min}, w_{\max}]$  of the  $\log_2$  probabilities of the characteristics contributing to the differential. For computing the lower bound  $\log_2(p)$  of the probability of the differentials, we used all characteristics with probabilities in the range from  $w_{\min}$  up to the values in brackets in the  $w_{\max}$  column.

Cipher	Rounds	$\Delta_{\text{in}}$	$\Delta_{\text{out}}$	$w_{\min}$	$w_{\max}$	$\log_2(p)$
SIMON32	13	(0, 40)	(4000, 0)	36	91 (91)	-28.79
SIMON32	14	(0, 8)	(800, 0)	38	120 (120)	-30.81
SIMON48	15	(20, 800088)	(800208, 2)	46	219 (79)	-41.02
SIMON48	16	(800000, 220082)	(800000, 220000)	50	256 (68)	-44.33
SIMON48	17	(80, 222)	(222, 80)	52	269 (85)	-46.32
SIMON64	21	(4000000, 11000000)	(11000000, 4000000)	68	453 (89)	-57.57
SIMON64	22	(440, 1880)	(440, 100)	72	502 (106)	-61.32

**Table 3.** The number of rounds after which full diffusion is reached for the standard SIMON parameters in comparison to the whole possible set of parameters.

Block size	32	48	64	96	128
Standard parameters	7	8	9	11	13
Median	8	10	11	13	14
First quartile	7	9	9	11	12
Best possible	6	7	8	9	10
Rank	2nd	2nd	2nd	3rd	4th

### 6.1 Diffusion

As a very simple measure to estimate the quality of the rotation constants, we measure the number of rounds that are needed to reach full diffusion. Full diffusion is reached when every state bit principally depends on all input bits. Compared to computing linear and differential properties it is an easy task to determine the dependency.

In Table 3 we give a comparison to how well the standard SIMON rotation parameters fare within the distribution of all possible parameter sets. The exact distributions for all SIMON variants can be found in the appendix in Table 8.

### 6.2 Differential and Linear

As a second criteria for our parameters, we computed for all  $a > b$  and  $\text{gcd}(a - b, n) = 1$  the optimal differential and linear characteristics for 10 rounds of SIMON32, SIMON48 and SIMON64. A list of the parameters which are optimal for all three variants of SIMON can be found in Appendix C.

It is important here to note that there are also many parameter sets, including the standard choice, for which the best 10-round characteristics of SIMON32 have a probability of  $2^{-25}$  compared to the optimum of  $2^{-26}$ . However, this difference by a factor of 2 does not seem to occur for more than 10 rounds and also not any larger variants of SIMON.

### 6.3 Interesting Alternative Parameter Sets

As one result of our investigation we chose three exemplary sets of parameters that surpass the standard parameters with regards to some metrics. Those variants are SIMON[12, 5, 3], SIMON[7, 0, 2] and SIMON[1, 0, 2].

SIMON[12, 5, 3] has the best diffusion amongst the parameters which have optimal differential and linear characteristics for 10 rounds. The two other choices are both restricted by setting  $b = 0$  as this would allow a more efficient implementation in software. Among those SIMON[7, 0, 2] has the best diffusion and the characteristics behave similar to the standard parameters. Ignoring the diffusion SIMON[1, 0, 2] seems also an interesting choice as it is optimal for the differential and linear characteristics.

If we look though at the differential corresponding to the best differential characteristic of SIMON[7, 0, 2] and SIMON[1, 0, 2], then we can see the number of characteristics contributing to it is significantly higher than for the standard parameters (see Appendix Table 6). However, for SIMON[12, 5, 3] the differential shows a surprisingly different behaviour and the probability of the differential is much closer to the probability of the characteristic. On the other side, the characteristics seem to be worse for the larger variants as can be seen in Table 7. Furthermore it might be desirable to have at least one rotation parameter that corresponds to a byte length, something that the standard parameter set features.

## 7 Conclusion and Future Work

In this work we analysed the general class of functions underlying the SIMON block cipher. First we rigorously derived efficiently computable and easily implementable expressions for the exact differential and linear behaviour of SIMON-like round functions.

Building upon this, we used those expressions for a computer aided approach based on SAT/SMT solvers to find both optimal differential and linear characteristics for SIMON. Furthermore, we were able to find all characteristics contributing to the probability of a differential for SIMON32 and gave better estimates for the probability for other variants.

Finally, we investigated the space of SIMON variants using different rotation constants with respect to diffusion, and the optimal differential and linear characteristics. Interestingly, the default parameters seem to be not always optimal.

This work opens up for further investigations. In particular, the choice and justifications of the NSA parameters for SIMON remains unclear. Besides our first

progress concerning the round function, the design of the key schedule remains largely unclear and further investigation is needed here.

**Acknowledgments.** First of all, we wish to thank Tomer Ashur. Both the method to check whether a linear input mask gives a correlated or uncorrelated linear 1-round characteristic for a given output mask as well as the first version of the SMT/SAT model for linear characteristics in SIMON were an outcome of our discussions. We furthermore wish to thank the reviewers for comments that helped to improve the paper.

## A Python Code to Calculate Linear and Differential Probabilities

In the following, code for calculating the differential and linear probabilities are given in Python. Restrictions are that the constants need to fulfil  $\gcd(a-b, n) = 1$  and  $n$  has to be even. We assume that the functions  $S^a(x)$  and  $\text{wt}(x)$  have been implemented as well as a function `parity` that calculates the parity  $\text{wt}(x) \bmod 2$  of a bit vector  $x$ .  $a$ ,  $b$ , and  $c$  have to be defined in the program as well.

The differential probability of  $\alpha \xrightarrow{f} \beta$  can then be calculated with the following function:

```
def pdiff (alpha,beta):
    gamma = beta ^ S(alpha,c)
    if alpha == 2**n-1:
        if hw(gamma)
            return 2**(n-1)
        else:
            return 0
    varibits = S(alpha, a) | S(alpha,b)
    if gamma & ~varibits != 0:
        return 0
    doublebits = S(alpha,2*a-b) & ~S(alpha,a) & S(alpha,b)
    if (gamma ^ S(gamma,a-b)) & doublebits != 0:
        return 0
    return 2**(-hw(varibits^doublebits))
```

The squared correlation of  $\alpha \xrightarrow{f} \beta$  can be calculated with the following function:

```
def plin (alpha,beta):
    alpha ^= S(beta,-c)
    if ((S(beta,-a) | S(beta,-b)) ^ alpha) & alpha != 0:
        return 0
    if beta == 2**n-1:
        t, v = lin, 0
        while t != 0:
            v ^= t & 3
            t >>= 2
        if v != 0:
            return 0
        else:
            return 2**(-n+2)
    tmp = beta
    abits = beta
    while tmp != 0:
        tmp = beta & S(tmp, -(a-b))
        abits ^= tmp
    sbits = S(beta, -(a-b)) & ~beta & ~S(abits, -(a-b))
```

```

sbits = S(sbits, -b)
pbits = 0
while sbits != 0:
    pbits ^= sbits & alpha
    sbits = S(sbits, (a-b)) & S(beta, -b)
    sbits = S(sbits, (a-b))
    pbits = S(pbits, 2*(a-b))
if pbits != 0:
    return 0
return 2**(-2*hw(abits))

```

## B Additional Differential Bounds

In Table 4 resp. 5 we give the distributions for the characteristics contributing to a differential up to the bound we computed them.

**Table 4.** Number of differential characteristics for the differential  $(80, 222) \xrightarrow{f^{17}} (222, 80)$  for SIMON48.

$\log_2(p)$	#Characteristics	$\log_2(p)$	#Characteristics
-52	1	-69	20890
-53	6	-70	38837
-54	15	-71	72822
-55	46	-72	133410
-56	100	-73	240790
-57	208	-74	353176
-58	379	-75	279833
-59	685	-76	235071
-60	1067	-77	259029
-61	1607	-78	225836
-62	2255	-79	256135
-63	2839	-80	252193
-64	3476	-81	252654
-65	4088	-82	198784
-66	5032	-83	229843
-67	7063	-84	208757
-68	11481	-85	253112

**Table 5.** Number of differential characteristics for the differential  $(4000000, 11000000) \xrightarrow{f^{21}} (11000000, 4000000)$  for SIMON64.

$\log_2(p)$	#Characteristics	$\log_2(p)$	#Characteristics
-68	2	-83	185709
-69	14	-84	173860
-70	70	-85	171902
-71	276	-86	171302
-72	951	-87	168190
-73	2880	-88	164694
-74	8101	-89	163141
-75	21062	-90	161089
-76	52255	-91	159354
-77	123206	-92	155804
-78	238297	-93	150954
-79	239305	-94	145061
-80	171895	-95	141914
-81	170187	-96	138480
-82	165671	-97	132931

## C Optimal Parameters for Differential Characteristics

The following sets of rotation constants  $(a, b, c)$  are optimal for 10 rounds regarding differential characteristics for SIMON32, SIMON48, and SIMON64

$(1, 0, 2), (1, 0, 3), (2, 1, 3), (4, 3, 5), (5, 0, 10), (5, 0, 15), (5, 4, 3), (7, 0, 14), (7, 6, 5)$   
 $(8, 1, 3), (8, 3, 14), (8, 7, 5), (10, 5, 15), (11, 6, 1), (12, 1, 7), (12, 5, 3), (12, 7, 1)$   
 $(13, 0, 10), (13, 0, 7), (13, 8, 2)$

Similar to the experiments for the default parameters, we used our framework to evaluate the quality of various rotation constants. In Table 7 we give an overview of the best differential characteristics for variants of SIMON using a different set of rotation constants. Table 6 shows that a carefully chosen set of constants can have a very strong effect on the differentials.

**Table 6.** Distribution of the characteristics for a 13-round differential for SIMON32 using different set of constants

$\log_2(p)$	[8, 1, 2]	[12, 5, 3]	[7, 0, 2]	[1, 0, 2]
-36	1	1	4	1
-37	4	2	16	6
-38	15	3	56	27
-39	46	2	144	88
-40	124	1	336	283
-41	288	0	744	822
-42	673	0	1644	2297
-43	1426	0	3420	6006
-44	2973	0	6933	14954
-45	5962	0	13270	34524
-46	11661	1	24436	73972
-47	21916	3	43784	150272
-48	40226	14	76261	292118
-49	72246	32	130068	/
-50	126574	54	218832	/
-51	218516	83	362284	/

**Table 7.** Overview of the optimal differential characteristics for SIMON variants

Rounds:	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
<b>Differential (12, 5, 3)</b>															
SIMON32	-2	-4	-6	-8	-12	-14	-18	-20	-26	-28	-34	-36	-42	-44	-47
SIMON48	-2	-4	-6	-8	-12	-14	-18	-20	-26	-30	-36	-36	-38	-40	-42
SIMON64	-2	-4	-6	-8	-12	-14	-18	-20	-26	-30	-35	-37	-43	-47	/
<b>Differential (1, 0, 2)</b>															
SIMON32	-2	-4	-6	-8	-12	-14	-18	-20	-26	-30	-36	-36	-38	-40	-42
SIMON48	-2	-4	-6	-8	-12	-14	-18	-20	-26	-30	-36	-38	-44	-48	-54
SIMON64	-2	-4	-6	-8	-12	-14	-18	-20	-26	-30	-36	-38	-44	-48	-54
<b>Differential (7, 0, 2)</b>															
SIMON32	-2	-4	-6	-8	-12	-14	-18	-20	-25	-30	-35	-36	-38	-40	-42
SIMON48	-2	-4	-6	-8	-12	-14	-18	-20	-26	-30	-35	-38	-44	-48	-53
SIMON64	-2	-4	-6	-8	-12	-14	-18	-20	-26	-30	-36	-38	-44	-48	/

**Table 8.** For each SIMON variant and each possible number of rounds, the number of possible combinations of rotation constants  $(a, b, c)$  with  $a \geq b$  is given that reaches full diffusion.

SIMON32	Rounds	6	7	8	9	10	11	17	$\infty$				
	$\#(a, b, c)$	48	600	528	88	144	128	64	576				
SIMON48	Rounds	7	8	9	10	11	13	14	15	25	$\infty$		
	$\#(a, b, c)$	48	1392	1680	792	528	344	144	128	64	2080		
SIMON64	Rounds	8	9	10	11	12	13	15	17	18	19	33	$\infty$
	$\#(a, b, c)$	384	4800	2112	2256	1152	608	512	48	288	256	128	4352
SIMON96	Rounds	9	10	11	12	13	14	15	16	17			
	$\#(a, b, c)$	336	4272	13920	7104	5568	3456	912	1152	800			
		19	21	25	26	27	49	$\infty$					
		1568	640	48	288	256	128	16000					
SIMON128	Rounds	10	11	12	13	14	15	16	17	18	19	20	
	$\#(a, b, c)$	768	10944	26112	25536	9024	6912	7488	2496	192	1824	2304	
		21	23	24	25	33	34	35	65	$\infty$			
		1792	1024	960	512	96	576	512	256	33792			

## References

1. Abed, F., List, E., Lucks, S., Wenzel, J.: Differential cryptanalysis of round-reduced SIMON and SPECK. In: Cid, C., Rechberger, C. (eds.) FSE 2014. LNCS, vol. 8540, pp. 525–545. Springer, Heidelberg (2015)
2. Albrecht, M.R., Driessen, B., Kavun, E.B., Leander, G., Paar, C., Yalçın, T.: Block ciphers – focus on the linear layer (feat. PRIDE). In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 57–76. Springer, Heidelberg (2014)
3. Alizadeh, J., Alkhzaimi, H.A., Aref, M.R., Bagheri, N., Gauravaram, P., Kumar, A., Lauridsen, M.M., Sanadhya, S.K.: Cryptanalysis of SIMON variants with connections. In: Sadeghi, A.-R., Saxena, N. (eds.) RFIDSec 2014. LNCS, vol. 8651, pp. 90–107. Springer, Heidelberg (2014)
4. Aumasson, J.-P., Jovanovic, P., Neves, S.: Analysis of NORX: investigating differential and rotational properties. In: Aranha, D.F., Menezes, A. (eds.) LATIN-CRYPT 2014. LNCS, vol. 8895, pp. 306–323. Springer, Heidelberg (2015)
5. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The SIMON and SPECK families of lightweight block ciphers. Cryptology ePrint Archive, Report 2013/404 (2013). <http://eprint.iacr.org/>



6. Biryukov, A., Roy, A., Velichkov, V.: Differential analysis of block ciphers SIMON and SPECK. In: Cid, C., Rechberger, C. (eds.) FSE 2014. LNCS, vol. 8540, pp. 546–570. Springer, Heidelberg (2015)
7. Borghoff, J., et al.: PRINCE – a low-latency block cipher for pervasive computing applications - extended abstract. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 208–225. Springer, Heidelberg (2012)
8. Carlet, C.: Vectorial boolean functions for cryptography. In: Crama, Y., Hammer, P.L. (eds.) Boolean Models and Methods in Mathematics, Computer Science, and Engineering. Encyclopedia of Mathematics and its Applications, vol. 134, pp. 398–469. Cambridge University Press, Cambridge (2010)
9. Daemen, J., Peeters, M., Assche, G.V., Rijmen, V.: The NOEKEON block cipher. Submission to the NESSIE project (2000)
10. Dinur, I., Dunkelman, O., Gutman, M., Shamir, A.: Improved top-down techniques in differential cryptanalysis. Cryptology ePrint Archive, Report 2015/268 (2015). <http://eprint.iacr.org/>
11. Ganesh, V., Hansen, T., Soos, M., Liew, D., Govostes, R.: STP constraint solver (2014). <https://github.com/stp/stp>
12. Grosso, V., Leurent, G., Standaert, F.-X., Varıcı, K.: LS-designs: bitslice encryption for efficient masked software implementations. In: Cid, C., Rechberger, C. (eds.) FSE 2014. LNCS, vol. 8540, pp. 18–37. Springer, Heidelberg (2015)
13. Kölbl, S.: CryptoSMT: An easy to use tool for cryptanalysis of symmetric primitives (2015). <https://github.com/kste/cryptosmt>
14. Mouha, N., Preneel, B.: Towards finding optimal differential characteristics for ARX: Application to Salsa20. Cryptology ePrint Archive, Report 2013/328 (2013). <http://eprint.iacr.org/>
15. Soos, M.: CryptoMiniSat SAT solver (2014). <https://github.com/msoos/cryptominisat/>
16. Sun, S., Hu, L., Wang, M., Wang, P., Qiao, K., Ma, X., Shi, D., Song, L., Fu, K.: Towards finding the best characteristics of some bit-oriented block ciphers and automatic enumeration of (related-key) differential and linear characteristics with predefined properties. Cryptology ePrint Archive, Report 2014/747 (2014). <http://eprint.iacr.org/>
17. Sun, S., Hu, L., Wang, M., Wang, P., Qiao, K., Ma, X., Shi, D., Song, L., Fu, K.: Constructing mixed-integer programming models whose feasible region is exactly the set of all valid differential characteristics of SIMON. Cryptology ePrint Archive, Report 2015/122 (2015). <http://eprint.iacr.org/>
18. Sun, S., Hu, L., Wang, P., Qiao, K., Ma, X., Song, L.: Automatic security evaluation and (related-key) differential characteristic search: application to SIMON, PRESENT, LBlock, DES(L) and other bit-oriented block ciphers. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8873, pp. 158–178. Springer, Heidelberg (2014)
19. Wang, Q., Liu, Z., Varıcı, K., Sasaki, Y., Rijmen, V., Todo, Y.: Cryptanalysis of reduced-round SIMON32 and SIMON48. In: Meier, W., Mukhopadhyay, D. (eds.) INDOCRYPT 2014. LNCS, vol. 8885, pp. 143–160. Springer, Heidelberg (2014)

# **Modes and Constructions**

# Tweaking Even-Mansour Ciphers

Benoît Cogliati<sup>1</sup>, Rodolphe Lampe<sup>1</sup>, and Yannick Seurin<sup>2</sup>(✉)

<sup>1</sup> University of Versailles, Versailles, France

benoitcogliati@hotmail.fr, rodolphe.lampe@gmail.com

<sup>2</sup> ANSSI, Paris, France

yannick.seurin@m4x.org

**Abstract.** We study how to construct efficient tweakable block ciphers in the Random Permutation model, where all parties have access to public random permutation oracles. We propose a construction that combines, more efficiently than by mere black-box composition, the CLRW construction (which turns a traditional block cipher into a tweakable block cipher) of Landecker *et al.* (CRYPTO 2012) and the iterated Even-Mansour construction (which turns a tuple of public permutations into a traditional block cipher) that has received considerable attention since the work of Bogdanov *et al.* (EUROCRYPT 2012). More concretely, we introduce the (one-round) *tweakable Even-Mansour* (TEM) cipher, constructed from a single  $n$ -bit permutation  $P$  and a uniform and almost XOR-universal family of hash functions  $(H_k)$  from some tweak space to  $\{0, 1\}^n$ , and defined as  $(k, t, x) \mapsto H_k(t) \oplus P(H_k(t) \oplus x)$ , where  $k$  is the key,  $t$  is the tweak, and  $x$  is the  $n$ -bit message, as well as its generalization obtained by cascading  $r$  independently keyed rounds of this construction. Our main result is a security bound up to approximately  $2^{2n/3}$  adversarial queries against adaptive chosen-plaintext and ciphertext distinguishers for the two-round TEM construction, using Patarin’s H-coefficients technique. We also provide an analysis based on the coupling technique showing that asymptotically, as the number of rounds  $r$  grows, the security provided by the  $r$ -round TEM construction approaches the information-theoretic bound of  $2^n$  adversarial queries.

**Keywords:** Tweakable block cipher · CLRW construction · Key-alternating cipher · Even-mansour construction · H-coefficients technique · Coupling technique

## 1 Introduction

TWEAKABLE BLOCK CIPHERS. Tweakable block ciphers (TBCs for short) are a generalization of traditional block ciphers which, in addition to the usual inputs (message and cryptographic key), take an extra (potentially adversarially controlled) input for variability called a *tweak*. Hence, the signature of a tweakable

---

Y. Seurin—This author was partially supported by the French National Agency of Research through the BLOC project (contract ANR-11-INS-011).

block cipher is  $\tilde{E} : \mathcal{K} \times \mathcal{T} \times \mathcal{M} \rightarrow \mathcal{M}$ , where  $\mathcal{K}$  is the key space,  $\mathcal{T}$  the tweak space, and  $\mathcal{M}$  the message space. This primitive has been rigorously formalized by Liskov *et al.* [25], and has proved to be very useful to construct various higher level cryptographic schemes such as (tweakable) length-preserving encryption modes [17, 18], online ciphers [2, 34], message authentication codes [24, 25], and authenticated encryption modes [25, 32, 33].

Tweakable block ciphers can be designed “from scratch” (e.g., the Hasty Pudding cipher [36], Mercy [10], or Threefish, the block cipher on which the Skein hash function [15] is based), however most of the proposed constructions are on top of an existing (traditional) block cipher, in a black-box fashion. In this latter family, constructions where changing the tweak implies to change the key of the underlying block cipher (e.g., Minematsu’s construction [26]) tend to be avoided for efficiency reasons (re-keying a block cipher is often a costly operation). Hence, most of the existing proposals have the property that the key under which the underlying block cipher is called is tweak-independent. Of particular relevance to our work, the original Liskov *et al.*’s paper proposed the so-called LRW construction (sometimes called LRW2 in the literature since this was the second of two constructions suggested in [25]), based on a block cipher  $E$  with key space  $\mathcal{K}_E$  and message space  $\{0, 1\}^n$  and an almost XOR-universal (AXU) family of hash functions  $\mathcal{H} = (H_k)_{k \in \mathcal{K}_H}$  from some set  $\mathcal{T}$  to  $\{0, 1\}^n$ , and defined as

$$\text{LRW}^E((k, k'), t, x) = H_{k'}(t) \oplus E_k(H_{k'}(t) \oplus x), \quad (1)$$

where  $(k, k') \in \mathcal{K}_E \times \mathcal{K}_H$  is the key,  $t \in \mathcal{T}$  is the tweak, and  $x \in \{0, 1\}^n$  is the message. This construction was proved secure in [25] up to the birthday bound, i.e.,  $2^{n/2}$  adversarial queries (assuming the underlying block cipher  $E$  is secure in the traditional sense, i.e., it is a strong pseudorandom permutation). This was later extended by Landecker *et al.* [24] who considered the cascade of two rounds of the LRW construction (with independent block cipher and hash function keys for each round), and proved it secure up to about  $2^{2n/3}$  adversarial queries.<sup>1</sup> This was further generalized to longer cascades by Lampe and Seurin [23] who proved that the  $r$ -round Chained-LRW (CLRW) construction is secure up to roughly  $2^{\frac{rn}{r+2}}$  adversarial queries (they also conjectured that the tight security bound is  $2^{\frac{rn}{r+1}}$  queries).

**THE ITERATED EVEN-MANSOUR CONSTRUCTION.** The iterated Even-Mansour construction abstracts in a generic way the high-level structure of key-alternating ciphers [11]. Concretely, it defines a block cipher from a tuple of  $r$  public  $n$ -bit permutations  $(P_1, \dots, P_r)$ , the ciphertext associated to some message  $x \in \{0, 1\}^n$  being computed as

$$y = k_r \oplus P_r(k_{r-1} \oplus P_{r-1}(\dots P_2(k_1 \oplus P_1(k_0 \oplus x)) \dots)),$$

where the  $n$ -bit round keys  $k_0, \dots, k_r$  are either independent or derived from a master key. This construction was extensively analyzed in the Random Permutation model, where the  $P_i$ ’s are modeled as public random permutation

<sup>1</sup> A flaw was subsequently found in the original proof of [24] and patched by Procter [31].

A different way of fixing the proof was proposed by Landecker *et al.*, see the revised version of [24].

oracles that the adversary can only query (bidirectionally) in a black-box way. This approach was originally taken for  $r = 1$  round in the seminal paper of Even and Mansour [13], who showed that the block cipher encrypting  $x$  into  $k_1 \oplus P(k_0 \oplus x)$  is secure up to  $2^{n/2}$  adversarial queries.<sup>2</sup> Dunkelman *et al.* [12] subsequently remarked that the same security level is retained by the *single-key* one-round Even-Mansour cipher, i.e., when  $k_0 = k_1$ . An important step was later made by Bogdanov *et al.* [5], who showed that for  $r = 2$  rounds, the construction ensures security up to roughly  $2^{2n/3}$  adversarial queries. Bogdanov *et al.*'s paper triggered a spate of results improving the pseudorandomness bound as the number  $r$  of rounds grows [21, 38], culminating with the proof by Chen and Steinberger [7] that the  $r$ -round iterated Even-Mansour construction with  $r$ -wise independent round keys ensures security up to about  $2^{\frac{rn}{r+1}}$  adversarial queries (tightly matching a generic attack described in [5]). Note that a special case of  $r$ -wise independent round keys is obtained by cascading  $r$  single-key one-round Even-Mansour ciphers (with independent keys), viz.

$$E_{k_1, \dots, k_r}(x) = k_r \oplus P_r(k_r \oplus k_{r-1} \oplus P_{r-1}(k_{r-1} \oplus \dots \oplus k_1 \oplus P_1(k_1 \oplus x) \dots)),$$

in which case the high-level similarity with the CLRW construction is obvious.

Besides pseudorandomness, the iterated Even-Mansour construction (with a sufficient number of rounds) has also been shown to achieve resistance to known-key attacks [3], related-key attacks [9, 14], and chosen-key attacks [9], as well as indistinguishability from an ideal cipher [1, 22].

**OUR RESULTS.** We consider the problem of constructing tweakable block ciphers directly from a tuple of public permutations rather than from a full-fledged block cipher. This was partially tackled by Cogliati and Seurin in [9]. They showed how to construct a TBC with  $n$ -bit keys and  $n$ -bit tweaks from three public  $n$ -bit permutations which is secure up to the birthday bound: denoting  $E(k, x)$  the 3-round iterated Even-Mansour cipher with the trivial key schedule (i.e., all round keys are equal to the  $n$ -bit master key  $k$ ), let  $\tilde{E}$  be the TBC defined as

$$\tilde{E}(k, t, x) = E(k \oplus t, x). \quad (2)$$

Hence,  $\tilde{E}$  is simply the 3-round iterated Even-Mansour cipher with round keys replaced by  $k \oplus t$ . Cogliati and Seurin showed<sup>3</sup> that this TBC is provably secure up to  $2^{n/2}$  adversarial queries in the Random Permutation Model (and that two rounds or less are insecure). The drawback of this simple construction is that any TBC of the form (2) with an underlying block cipher  $E$  of key-length  $\kappa$  can deliver at most  $\kappa/2$  bits of security [4], so that there is no hope to improve

<sup>2</sup> When we talk about *adversarial queries* without being more specific in such a context where the attacker, in addition to the construction oracle, also has oracle access to the inner permutation(s), we mean indifferently construction and inner permutation queries.

<sup>3</sup> The focus of [9] is on xor-induced related-key attacks against the traditional iterated Even-Mansour cipher, but their result can be directly transposed to the TBC setting, see the full version of [9].

the number of queries that the construction can securely tolerate by merely increasing the number of rounds to four or more.

In this paper, we aim at getting a tweakable Even-Mansour-like construction with security *beyond the birthday bound*. The naive way of proceeding would be to instantiate the block cipher  $E$  in the CLRW construction with an iterated Even-Mansour cipher based on permutations  $P_1, \dots, P_r$ . However, combining existing results for CLRW on one hand [23, 24], and for the iterated Even-Mansour cipher on the other hand [7], one would need at least  $r^2$  independent permutations to get provable  $\mathcal{O}(2^{\frac{rn}{r+1}})$ -security.<sup>4</sup> A more promising approach, that we take here, is to start with the construction obtained by combining the (one-round) LRW construction and the (one-round) Even-Mansour cipher, yielding what we dub the one-round *tweakable Even-Mansour* construction, defined from a single  $n$ -bit permutation  $P$  and an AXU family of hash functions  $\mathcal{H}' = (H'_{k'})_{k' \in \mathcal{K}'}$  from some tweak space  $\mathcal{T}$  to  $\{0, 1\}^n$  as

$$\text{TEM}^P((k, k'), t, x) = H'_{k'}(t) \oplus k \oplus P(H'_{k'}(t) \oplus k \oplus x), \quad (3)$$

where  $(k, k') \in \{0, 1\}^n \times \mathcal{K}'$  is the key,  $t \in \mathcal{T}$  is the tweak, and  $x \in \{0, 1\}^n$  is the message. Combining the security proofs for LRW [25] and for the one-round single-key Even-Mansour cipher [12, 13] directly yields that this construction ensures security up to  $2^{n/2}$  adversarial queries, in the Random Permutation model for  $P$ . For example, if we use the universal hash function family based on multiplication in the finite field  $\mathbb{F}_{2^n}$ , i.e.,  $H_{k'}(t) = k' \otimes t$ , which is XOR-universal, one obtains a simple tweakable block cipher with  $2n$ -bit keys and  $n$ -bit tweaks which is secure up to the birthday bound.

Our first insight is to consider the slightly more general construction

$$\text{TEM}^P(k, t, x) = H_k(t) \oplus P(H_k(t) \oplus x). \quad (4)$$

It is not too hard to show (as we do in Sect. 3.2) that this more general construction also ensures security up to  $2^{n/2}$  adversarial queries, assuming that the hash function family  $\mathcal{H} = (H_k)_{k \in \mathcal{K}}$ , in addition to being AXU, is also *uniform* (i.e., for any  $t \in \mathcal{T}$  and any  $y \in \{0, 1\}^n$ , the probability over  $k \leftarrow_{\S} \mathcal{K}$  that  $H_k(t) = y$  is equal to  $2^{-n}$ ).<sup>5</sup> This simple observation allows to save  $n$  bits of key material when using multiplication-based hashing, since  $H_k(t) = k \otimes t$  is XOR-universal and uniform if one restricts the tweak space to  $\mathbb{F}_{2^n} \setminus \{0\}$ .

It is naturally tempting to consider cascading  $r > 1$  rounds of construction (4) to obtain a hybrid of the iterated Even-Mansour cipher and the CLRW construction. Our main result is that the two-round construction

$$\text{TEM}^{P_1, P_2}((k_1, k_2), t, x) = H_{k_2}(t) \oplus P_2(H_{k_2}(t) \oplus H_{k_1}(t) \oplus P_1(H_{k_1}(t) \oplus x))$$

<sup>4</sup> For  $r > 2$ , since the analysis of the CLRW construction in [23] is not tight, this is even worse.

<sup>5</sup> Construction (3) is obviously a special case of construction (4), since the hash function family defined by  $H_{k, k'}(t) = H'_{k'}(t) \oplus k$ , where  $(H'_{k'})_{k' \in \mathcal{K}'}$  is AXU and  $k \in \{0, 1\}^n$ , is AXU and uniform.

is secure (against adaptive chosen-plaintext and ciphertext attacks) up to approximately  $2^{2n/3}$  adversarial queries (again, assuming that  $\mathcal{H}$  is uniform and AXU).

To arrive at this result, we could have adapted the game-based proof of [24] for the two-round CLRW construction to accommodate the fact that in the TEM setting, the adversary has additionally oracle access to the inner permutations  $P_1$  and  $P_2$ . Yet we preferred to use the H-coefficients technique [30], which was successfully applied to the analysis of the iterated Even-Mansour cipher [6, 7], and adjust it to take into account the existence of the tweak in the TEM construction. Our choice was motivated by the fact that the H-coefficients-based security proof for the two-round Even-Mansour cipher is (in our opinion) simpler than the game-based proof for the two-round CLRW construction. Actually, our security proof for the two-round TEM construction can easily be simplified (by making the inner permutations secret, or, more formally, letting the number of queries  $q_p$  to the inner permutations be zero in our security bound as given by Theorem 2) to yield a new, H-coefficients-based proof of the security result of [24] for the two-round CLRW construction (our own bound matching Landecker *et al.*'s one [24] up to multiplicative constants).<sup>6</sup> It seems interesting to us that our proof entails a new and conceptually simpler (at least to us) proof of a previous result that turned out quite delicate to get right with game-based techniques [31]. We explain how to “extract” from our work a H-coefficients proof for the two-round CLRW construction in the full version of this paper [8].

We were unable to extend our H-coefficients security proof to  $r > 2$  rounds.<sup>7</sup> Instead, we provide an asymptotic analysis of the TEM construction (as  $r$  grows) based on the coupling technique [19, 28]. This part combines in a rather straightforward way the approach of [21] (which applied the coupling technique to the iterated Even-Mansour cipher) and of [23] (which applied the coupling technique to the CLRW construction). This allows us to prove that the  $r$ -round TEM construction is secure up to roughly  $2^{\frac{rn}{r+2}}$  adversarial queries (against adaptive chosen-plaintext and ciphertext attacks). As with previous work, we conjecture that the “real” security bound is actually  $2^{\frac{rn}{r+1}}$  queries (which we prove to hold for the weaker class of non-adaptive chosen-plaintext adversaries), but that the coupling technique is not adapted to prove this.

<sup>6</sup> In fact, this is not as straightforward as it might seem, since our results assume that the hash function family  $\mathcal{H}$  is uniform in addition to being AXU, whereas the security result of [24] only requires  $\mathcal{H}$  to be AXU. Inspection of our proof indicates however that the uniformity assumption on  $\mathcal{H}$  can be safely lifted when the adversary is not allowed to query the inner permutations.

<sup>7</sup> For readers familiar with [7], which tightly analyzed the security of the traditional iterated EM cipher for any number of rounds, the main obstacle is that in the tweakable EM setting, the paths for two construction queries with distinct tweaks can collide at the input of inner permutations, whereas this can never happen in the traditional EM setting. While this is exactly the difficulty that we are able to handle for  $r = 2$  in Lemma 3, getting a combinatorial lemma similar to [7, Lemma 1] that would allow to analyze good transcripts for any number of rounds in the tweakable setting seems more challenging.

APPLICATION TO RELATED-KEY SECURITY. There are strong connections between tweakable block ciphers and the related-key security of traditional block ciphers [4, 25]. We expand on this in the full version of the paper [8], explaining how our results have immediate implications for the related-key security of the traditional (iterated) Even-Mansour cipher with a nonlinear key-schedule.

RELATED WORK AND PERSPECTIVES. There are very few papers studying generic ways of building tweakable block ciphers from some lower-level primitive than a traditional block cipher. One notable exception is the work of Goldenberg *et al.* [16] who studied how to tweak (generically) Feistel ciphers (in other words, they showed how to construct tweakable block ciphers from pseudorandom functions). This was extended to generalized Feistels by Mitsuda and Iwata [27]. Our own work seems to be the first (besides [9], that capped at the birthday bound) to explore theoretically sound ways to construct “by-design” tweakable block ciphers with an SPN or more generally a key-alternating structure. In a sense, it can be seen as complementary to the recent TWEAKEY framework introduced by Jean *et al.* [20], that tackled a similar goal but adopted a more practical and attack-driven (rather than proof-oriented) angle. We hope that combining these two approaches will pave the way towards efficient and theoretically sound ways of building tweakable key-alternating ciphers, or tweaking existing ones such as AES. We also note that the term *tweakable Even-Mansour* was previously used by the designers of Minalpher [35] (a candidate to the CAESAR competition) to designate a permutation-based variant of Rogaway’s XEX construction [32]. It relates to construction (4) by eliminating the AXU hash function  $H_k(t)$  and replacing it by  $\Delta = (k||t) \oplus P(k||t)$  (thereby halving tweak- and key-length), in about the same way XEX replaces the AXU hash function of the LRW construction (1) by a “gadget” calling the underlying block cipher  $E_k$ . The designers of Minalpher prove that this construction also achieves birthday-bound security.

Finally, we bring up some open problems. First, as already mentioned, it would be very interesting to give a tight analysis of the TEM construction for any number  $r > 2$  of rounds (a first, hopefully simpler step towards this goal would be to give a tight bound for the CLRW construction for  $r > 2$ ). Second, variants with the same permutation and/or non-independent round keys are also worth studying, as was done in [6] for the (traditional) two-round iterated Even-Mansour cipher. Third, since implementing an AXU hash function family might be costly, it would be very valuable to explore whether linear operations for mixing the key and the tweak into the state of an Even-Mansour-like construction might be enough to get security beyond the birthday bound.

## 2 Preliminaries

### 2.1 Notation and General Definitions

GENERAL NOTATION. In all the following, we fix an integer  $n \geq 1$  and denote  $N = 2^n$ . For integers  $1 \leq b \leq a$ , we will write  $(a)_b = a(a-1) \cdots (a-b+1)$  and  $(a)_0 = 1$  by convention. The set of all permutations of  $\{0, 1\}^n$  will be denoted



$\mathbf{P}(n)$ . Given a non-empty set  $X$ , we denote  $x \leftarrow_{\S} X$  the draw of an element  $x$  from  $X$  uniformly at random.

**TWEAKABLE BLOCK CIPHERS.** A *tweakable block cipher* with key space  $\mathcal{K}$ , tweak space  $\mathcal{T}$ , and message space  $\mathcal{M}$  is a mapping  $\tilde{E} : \mathcal{K} \times \mathcal{T} \times \mathcal{M} \rightarrow \mathcal{M}$  such that for any key  $k \in \mathcal{K}$  and any tweak  $t \in \mathcal{T}$ ,  $x \mapsto \tilde{E}(k, t, x)$  is a permutation of  $\mathcal{M}$ . We denote  $\text{TBC}(\mathcal{K}, \mathcal{T}, n)$  the set of all tweakable block ciphers with key space  $\mathcal{K}$ , tweak space  $\mathcal{T}$ , and message space  $\{0, 1\}^n$ . A *tweakable permutation* with tweak space  $\mathcal{T}$  and message space  $\mathcal{M}$  is a mapping  $\tilde{P} : \mathcal{T} \times \mathcal{M} \rightarrow \mathcal{M}$  such that for any tweak  $t \in \mathcal{T}$ ,  $x \mapsto \tilde{P}(t, x)$  is a permutation of  $\mathcal{M}$ . We denote  $\text{TP}(\mathcal{T}, n)$  the set of all tweakable permutations with tweak space  $\mathcal{T}$  and message space  $\{0, 1\}^n$ .

**THE ITERATED TWEAKABLE EVEN-MANSOUR CONSTRUCTION.** Fix integers  $n, r \geq 1$ . Let  $\mathcal{T}$  and  $\mathcal{K}$  be two sets, and  $\mathcal{H} = (H_k)_{k \in \mathcal{K}}$  be a family of functions from  $\mathcal{T}$  to  $\{0, 1\}^n$  indexed by  $\mathcal{K}$ . The  $r$ -round iterated tweakable Even-Mansour construction  $\text{TEM}[n, r, \mathcal{H}]$  specifies, from an  $r$ -tuple  $\mathbf{P} = (P_1, \dots, P_r)$  of permutations of  $\{0, 1\}^n$ , a tweakable block cipher with key space  $\mathcal{K}^r$ , tweak space  $\mathcal{T}$ , and message space  $\{0, 1\}^n$ , simply denoted  $\text{TEM}^{\mathbf{P}}$  in the following (parameters  $[n, r, \mathcal{H}]$  will always be clear from the context) which maps a key  $\mathbf{k} = (k_1, \dots, k_r) \in \mathcal{K}^r$ , a tweak  $t \in \mathcal{T}$ , and a plaintext  $x \in \{0, 1\}^n$  to the ciphertext defined as (see Fig. 1):

$$\text{TEM}^{\mathbf{P}}(\mathbf{k}, t, x) = \Pi_{k_r, t}^{P_r} \circ \dots \circ \Pi_{k_1, t}^{P_1}(x),$$

where  $\Pi_{k, t}^P$  is the permutation of  $\{0, 1\}^n$  (corresponding to one round of the construction) defined as

$$\Pi_{k, t}^P(x) = H_k(t) \oplus P(H_k(t) \oplus x).$$

We will denote  $\text{TEM}_{\mathbf{k}}^{\mathbf{P}}$  the mapping taking as input  $(t, x) \in \mathcal{T} \times \{0, 1\}^n$  and returning  $\text{TEM}^{\mathbf{P}}(\mathbf{k}, t, x)$ .

*Convention 1.* In order to lighten the notation, we will often identify the hash function family  $\mathcal{H}$  and its key space  $\mathcal{K}$ . This way, the key space of the  $r$ -round  $\text{TEM}^{\mathbf{P}}$  tweakable block cipher is simply  $\mathcal{H}^r$ , and we write

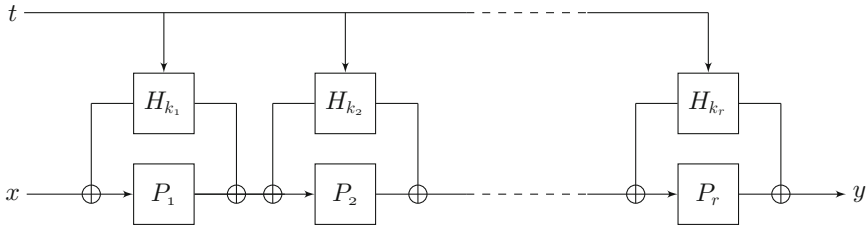
$$\text{TEM}_{\mathbf{h}}^{\mathbf{P}}(t, x) = h_r(t) \oplus P_r(h_r(t) \oplus \dots \oplus h_1(t) \oplus P_1(h_1(t) \oplus x) \dots)$$

where  $\mathbf{h} = (h_1, \dots, h_r) \in \mathcal{H}^r$  is the key of  $\text{TEM}^{\mathbf{P}}$ .

**UNIFORM AXU HASH FUNCTION FAMILY.** We will need the following properties of the hash function family  $\mathcal{H}$ .

**Definition 1.** Let  $\mathcal{H} = (H_k)_{k \in \mathcal{K}}$  be a family of functions from some set  $\mathcal{T}$  to  $\{0, 1\}^n$  indexed by a set of keys  $\mathcal{K}$ .  $\mathcal{H}$  is said to be uniform if for any  $t \in \mathcal{T}$  and  $y \in \{0, 1\}^n$ ,

$$\Pr[k \leftarrow_{\S} \mathcal{K} : H_k(t) = y] = 2^{-n}.$$



**Fig. 1.** The tweakable Even-Mansour construction with  $r$  rounds, based on public permutations  $P_1, \dots, P_r$  and a family of hash functions  $\mathcal{H} = (H_k)_{k \in \mathcal{K}}$ .

$\mathcal{H}$  is said  $\varepsilon$ -almost XOR-universal ( $\varepsilon$ -AXU) if for all distinct  $t, t' \in \mathcal{T}$  and all  $y \in \{0, 1\}^n$ ,

$$\Pr[k \leftarrow_{\S} \mathcal{K} : H_k(t) \oplus H_k(t') = y] \leq \varepsilon.$$

$\mathcal{H}$  is simply said XOR-universal (XU) if it is  $2^{-n}$ -AXU.

*Example 1.* Let  $\mathbb{F}_{2^n}$  be the set  $\{0, 1\}^n$  seen as the field with  $2^n$  elements defined by some irreducible polynomial of degree  $n$  over  $\mathbb{F}_2$ , the field with two elements, and denote  $a \otimes b$  the field multiplication of two elements  $a, b \in \mathbb{F}_{2^n}$ . For any integer  $\ell \geq 1$ , we define the family of functions  $\mathcal{H} = (H_k)_{k \in \mathbb{F}_{2^n}}$  with domain  $(\mathbb{F}_{2^n})^\ell$  and range  $\mathbb{F}_{2^n}$  as

$$H_k(t_1, \dots, t_\ell) = \sum_{i=1}^{\ell} k^i \otimes t_i.$$

Then  $\mathcal{H}$  is  $\ell \cdot 2^{-n}$ -AXU [37]. Note however that  $\mathcal{H}$  is not uniform since  $(0, \dots, 0)$  is always mapped to 0 independently of the key. This can be handled either by adding an independent key (resulting in  $2n$ -bit keys), i.e., defining  $\mathcal{H}' = (H'_{k,k'})_{(k,k') \in (\mathbb{F}_{2^n})^2}$  where  $H'_{k,k'}(t_1, \dots, t_\ell) = H_k(t_1, \dots, t_\ell) \oplus k'$ , or by forbidding the all-zero tweak, in which case the family is not exactly uniform, but rather  $\ell \cdot 2^{-n}$ -almost uniform, i.e., for any  $t \in \mathcal{T} \setminus \{(0, \dots, 0)\}$  and  $y \in \{0, 1\}^n$ ,  $\Pr[k \leftarrow_{\S} \mathcal{K} : H_k(t) = y] \leq \ell \cdot 2^{-n}$ . Our results can be straightforwardly extended to the case of  $\varepsilon$ -almost uniform families of functions.

### 2.2 Security Definitions

Fix some family of functions  $\mathcal{H} = (H_k)_{k \in \mathcal{K}}$  from  $\mathcal{T}$  to  $\{0, 1\}^n$ . To study the security of the construction  $\text{TEM}[n, r, \mathcal{H}]$  in the Random Permutation Model, we consider a distinguisher  $\mathcal{D}$  which interacts with  $r + 1$  oracles that we denote generically  $(\tilde{P}_0, P_1, \dots, P_r)$ , where syntactically  $\tilde{P}_0$  is a tweakable permutation with tweak space  $\mathcal{T}$  and message space  $\{0, 1\}^n$ , and  $P_1, \dots, P_r$  are permutations of  $\{0, 1\}^n$ . The goal of  $\mathcal{D}$  is to distinguish two “worlds”: the so-called *real world*,

where  $\mathcal{D}$  interacts with  $(\text{TEM}_{\mathbf{k}}^{\mathbf{P}}, \mathbf{P})$ , where  $\mathbf{P} = (P_1, \dots, P_r)$  is a tuple of public random permutations and the key  $\mathbf{k} = (k_1, \dots, k_r)$  is drawn uniformly at random from  $\mathcal{K}^r$ , and the so-called *ideal world*  $(\tilde{P}_0, \mathbf{P})$ , where  $\tilde{P}_0$  is a uniformly random tweakable permutation and  $\mathbf{P}$  is a tuple of random permutations of  $\{0, 1\}^n$  independent from  $\tilde{P}_0$ . We will refer to  $\tilde{P}_0$  as the *construction oracle* and to  $P_1, \dots, P_r$  as the *inner permutation oracles*.

Similarly to [21], we consider two classes of distinguishers depending on how they can issue their queries:

- a *non-adaptive chosen-plaintext* (NCPA) distinguisher runs in two phases: during the first phase, it only queries the inner permutations, adaptively and in both directions; in the second phase, it issues a tuple of non-adaptive chosen-plaintext queries to the construction oracle and receives the corresponding answers (this tuple of queries may depend on the answers received in the first phase, but all queries must be chosen non-adaptively before receiving any answer from the construction oracle);
- an *adaptive chosen-plaintext and ciphertext* (CCA) distinguisher is not restricted in how it queries its oracles: it can make adaptive bidirectional queries to all its oracles.

We stress that the NCPA model is not very interesting in itself<sup>8</sup> and will only be useful as an intermediate step for the coupling-based security proof in Sect. 4.

The distinguishing advantage of a distinguisher  $\mathcal{D}$  is defined as

$$\mathbf{Adv}(\mathcal{D}) \stackrel{\text{def}}{=} \left| \Pr \left[ \mathcal{D}^{\text{TEM}_{\mathbf{k}}^{\mathbf{P}}} = 1 \right] - \Pr \left[ \mathcal{D}^{\tilde{P}_0, \mathbf{P}} = 1 \right] \right|,$$

where the first probability is taken over the random choice of  $\mathbf{k}$  and  $\mathbf{P}$ , and the second probability is taken over the random choice of  $\tilde{P}_0$  and  $\mathbf{P}$ . In all the following, we consider computationally unbounded distinguishers, and hence we can assume *wlog* that they are deterministic. We also assume that they never make pointless queries (i.e., queries whose answers can be unambiguously deduced from previous answers).

For non-negative integers  $q_c$ ,  $q_p$  and  $\text{ATK} \in \{\text{NCPA}, \text{CCA}\}$ , we define the insecurity of the  $\text{TEM}[n, r, \mathcal{H}]$  construction against  $\text{ATK}$ -attacks as

$$\mathbf{Adv}_{\text{TEM}[n, r, \mathcal{H}]}^{\text{atk}}(q_c, q_p) = \max_{\mathcal{D}} \mathbf{Adv}(\mathcal{D}),$$

where the maximum is taken over all distinguishers in the class  $\text{ATK}$  making exactly  $q_c$  queries to the construction oracle and exactly  $q_p$  queries to each inner permutation oracle.

<sup>8</sup> Indeed, forbidding the adversary to query the inner permutation oracles at some point of the attack takes us away from the spirit of the Random Permutation model, which is thought as a heuristically sound way of modeling some complex (but otherwise public and fully described) permutation that the adversary can always evaluate at will.

### 3 Tight Bounds for One and Two Rounds

#### 3.1 The H-Coefficients Technique

We start by describing Patarin’s H-coefficients technique [30], which has enjoyed increasing adoption since Chen and Steinberger used it to prove the security of the iterated Even-Mansour cipher for an arbitrary number of rounds [7].

TRANSCRIPT. We summarize the interaction of  $\mathcal{D}$  with its oracles in what we call the *queries transcript*  $(\mathcal{Q}_C, \mathcal{Q}_{P_1}, \dots, \mathcal{Q}_{P_r})$  of the attack, where  $\mathcal{Q}_C$  records the queries to the construction oracle and  $\mathcal{Q}_{P_i}$ ,  $1 \leq i \leq r$ , records the queries to inner permutation  $P_i$ . More precisely,  $\mathcal{Q}_C$  contains all triples  $(t, x, y) \in \mathcal{T} \times \{0, 1\}^n \times \{0, 1\}^n$  such that  $\mathcal{D}$  either made the direct query  $(t, x)$  to the construction oracle and received answer  $y$ , or made the inverse query  $(t, y)$  and received answer  $x$ . Similarly, for  $1 \leq i \leq r$ ,  $\mathcal{Q}_{P_i}$  contains all pairs  $(u, v) \in \{0, 1\}^n \times \{0, 1\}^n$  such that  $\mathcal{D}$  either made the direct query  $u$  to permutation  $P_i$  and received answer  $v$ , or made the inverse query  $v$  and received answer  $u$ . Note that queries are recorded in a directionless and unordered fashion, but by our assumption that the distinguisher is deterministic, there is a one-to-one mapping between this representation and the raw transcript of the interaction of  $\mathcal{D}$  with its oracles (see e.g. [7] for more details). Note also that by our assumption that  $\mathcal{D}$  never makes pointless queries, each query to the construction oracle results in a distinct triple in  $\mathcal{Q}_C$ , and each query to  $P_i$  results in a distinct pair in  $\mathcal{Q}_{P_i}$ , so that  $|\mathcal{Q}_C| = q_c$  and  $|\mathcal{Q}_{P_i}| = q_p$  for  $1 \leq i \leq r$  since we assume that the distinguisher always makes the maximal number of allowed queries to each oracle. In all the following, we also denote  $m$  the number of distinct tweaks appearing in  $\mathcal{Q}_C$ , and  $q_i$  the number of queries for the  $i$ -th tweak,  $1 \leq i \leq m$ , using an arbitrary ordering of the tweaks. Note that  $m$  may depend on the answers received from the oracles, yet one always has  $\sum_{i=1}^m q_i = q_c$ .

We say that a queries transcript is *attainable* (with respect to some fixed distinguisher  $\mathcal{D}$ ) if there exists oracles  $(\tilde{P}_0, \mathbf{P})$  such that the interaction of  $\mathcal{D}$  with  $(\tilde{P}_0, \mathbf{P})$  yields this transcript (said otherwise, the probability to obtain this transcript in the “ideal” world is non-zero). Moreover, in order to have a simple definition of bad transcripts, we reveal to the adversary at the end of the experiment the actual tuple of keys  $\mathbf{k} = (k_1, \dots, k_r)$  if we are in the real world, while in the ideal world, we simply draw dummy keys  $(k_1, \dots, k_r) \leftarrow_{\S} \mathcal{K}^r$  independently from the answers of the oracle  $\tilde{P}_0$ . (This can obviously only increase the advantage of the distinguisher, so that this is without loss of generality). All in all, a transcript  $\tau$  is a tuple  $\tau = (\mathcal{Q}_C, \mathcal{Q}_{P_1}, \dots, \mathcal{Q}_{P_r}, \mathbf{k})$ , and we say that a transcript is attainable if the corresponding queries transcript  $(\mathcal{Q}_C, \mathcal{Q}_{P_1}, \dots, \mathcal{Q}_{P_r})$  is attainable. We denote  $\Theta$  the set of attainable transcripts. In all the following, we denote  $T_{\text{re}}$ , resp.  $T_{\text{id}}$ , the probability distribution of the transcript  $\tau$  induced by the real world, resp. the ideal world (note that these two probability distributions depend on the distinguisher). By extension, we use the same notation to denote a random variable distributed according to each distribution. The main lemma of the H-coefficients technique is the following one (see e.g. [6, 7] for the proof).

**Lemma 1.** Fix a distinguisher  $\mathcal{D}$ . Let  $\Theta = \Theta_{\text{good}} \sqcup \Theta_{\text{bad}}$  be a partition of the set of attainable transcripts. Assume that there exists  $\varepsilon_1$  such that for any  $\tau \in \Theta_{\text{good}}$ , one has<sup>9</sup>

$$\frac{\Pr[T_{\text{re}} = \tau]}{\Pr[T_{\text{id}} = \tau]} \geq 1 - \varepsilon_1,$$

and that there exists  $\varepsilon_2$  such that  $\Pr[T_{\text{id}} \in \Theta_{\text{bad}}] \leq \varepsilon_2$ . Then  $\mathbf{Adv}(\mathcal{D}) \leq \varepsilon_1 + \varepsilon_2$ .

ADDITIONAL NOTATION. Given a permutation queries transcript  $\mathcal{Q}$  and a permutation  $P$ , we say that  $P$  extends  $\mathcal{Q}$ , denoted  $P \vdash \mathcal{Q}$ , if  $P(u) = v$  for all  $(u, v) \in \mathcal{Q}$ . By extension, given a tuple of permutation queries transcript  $\mathcal{Q}_{\mathbf{P}} = (\mathcal{Q}_{P_1}, \dots, \mathcal{Q}_{P_r})$  and a tuple of permutations  $\mathbf{P} = (P_1, \dots, P_r)$ , we say that  $\mathbf{P}$  extends  $\mathcal{Q}_{\mathbf{P}}$ , denoted  $\mathbf{P} \vdash \mathcal{Q}_{\mathbf{P}}$ , if  $P_i \vdash \mathcal{Q}_{P_i}$  for each  $i = 1, \dots, r$ . Note that for a permutation transcript of size  $q_p$ , one has

$$\Pr[P \leftarrow_{\S} \mathbf{P}(n) : P \vdash \mathcal{Q}] = \frac{1}{(N)_{q_p}}. \quad (5)$$

Similarly, given a tweakable permutation transcript  $\tilde{\mathcal{Q}}$  and a tweakable permutation  $\tilde{P}$ , we say that  $\tilde{P}$  extends  $\tilde{\mathcal{Q}}$ , denoted  $\tilde{P} \vdash \tilde{\mathcal{Q}}$ , if  $\tilde{P}(t, x) = y$  for all  $(t, x, y) \in \tilde{\mathcal{Q}}$ . For a tweakable permutation transcript  $\tilde{\mathcal{Q}}$  with  $m$  distinct tweaks and  $q_i$  queries corresponding to the  $i$ -th tweak, one has

$$\Pr[\tilde{P} \leftarrow_{\S} \text{TP}(\mathcal{T}, n) : \tilde{P} \vdash \tilde{\mathcal{Q}}] = \prod_{i=1}^m \frac{1}{(N)_{q_i}}. \quad (6)$$

PRELIMINARY OBSERVATIONS. It is easy to see that the interaction of a distinguisher  $\mathcal{D}$  with oracles  $(\tilde{P}_0, P_1, \dots, P_r)$  yields any attainable queries transcript  $(\mathcal{Q}_C, \mathcal{Q}_{\mathbf{P}})$  with  $\mathcal{Q}_{\mathbf{P}} = (\mathcal{Q}_{P_1}, \dots, \mathcal{Q}_{P_r})$  iff  $\tilde{P}_0 \vdash \mathcal{Q}_C$  and  $P_i \vdash \mathcal{Q}_{P_i}$  for  $1 \leq i \leq r$ . In the ideal world, the key  $\mathbf{k}$ , the permutations  $P_1, \dots, P_r$ , and the tweakable permutation  $\tilde{P}_0$  are all uniformly random and independent, so that, by (5) and (6), the probability of getting any attainable transcript  $\tau = (\mathcal{Q}_C, \mathcal{Q}_{\mathbf{P}}, \mathbf{k})$  in the ideal world is

$$\Pr[T_{\text{id}} = \tau] = \frac{1}{|\mathcal{K}|^r} \times \left( \frac{1}{(N)_{q_p}} \right)^r \times \prod_{i=1}^m \frac{1}{(N)_{q_i}}.$$

In the real world, the probability to obtain  $\tau$  is

$$\Pr[T_{\text{re}} = \tau] = \frac{1}{|\mathcal{K}|^r} \times \left( \frac{1}{(N)_{q_p}} \right)^r \times \Pr \left[ \mathbf{P} \leftarrow_{\S} (\mathbf{P}(n))^r : \text{TEM}_{\mathbf{k}}^{\mathbf{P}} \vdash \mathcal{Q}_C \mid \mathbf{P} \vdash \mathcal{Q}_{\mathbf{P}} \right].$$

Let

$$\rho(\tau) \stackrel{\text{def}}{=} \Pr \left[ \mathbf{P} \leftarrow_{\S} (\mathbf{P}(n))^r : \text{TEM}_{\mathbf{k}}^{\mathbf{P}} \vdash \mathcal{Q}_C \mid \mathbf{P} \vdash \mathcal{Q}_{\mathbf{P}} \right].$$

<sup>9</sup> Recall that for an attainable transcript, one has  $\Pr[T_{\text{id}} = \tau] > 0$ .

Then we have

$$\frac{\Pr[T_{\text{re}} = \tau]}{\Pr[T_{\text{id}} = \tau]} = \mathfrak{p}(\tau) / \prod_{i=1}^m \frac{1}{(N)_{q_i}} = \mathfrak{p}(\tau) \cdot \prod_{i=1}^m (N)_{q_i}. \tag{7}$$

Hence, to apply Lemma 1, we will have to compare  $\mathfrak{p}(\tau)$  and  $\prod_{i=1}^m 1/(N)_{q_i}$ , assuming  $\tau$  is good (for some adequate definition of bad and good transcripts).

### 3.2 Security Proof for One Round

We consider here the one-round construction  $\text{TEM}[n, 1, \mathcal{H}]$ . Using Convention 1, we have

$$\text{TEM}_{h_1}^{P_1}(t, x) = h_1(t) \oplus P_1(h_1(t) \oplus x)$$

where the key is  $h_1 \leftarrow_{\S} \mathcal{H}$ . We prove the following theorem.

**Theorem 1.** *Let  $\mathcal{H}$  be a uniform  $\varepsilon$ -AXU family of functions from  $\mathcal{T}$  to  $\{0, 1\}^n$ . For any integers  $q_c$  and  $q_p$ , one has*

$$\text{Adv}_{\text{TEM}[n, 1, \mathcal{H}]}^{\text{cca}}(q_c, q_p) \leq q_c^2 \varepsilon + \frac{2q_c q_p}{N}.$$

The proof uses the H-coefficients technique that we exposed in Sect. 3.1, and serves as a good warm-up before the more complex two-round case. For reasons of space, it is deferred to the full version of the paper [8].

### 3.3 Security Proof for Two Rounds

STATEMENT OF THE RESULT AND DISCUSSION. Let  $\mathcal{H}$  be an  $\varepsilon$ -AXU and uniform function family. Using Convention 1, the two-round tweakable Even-Mansour construction is written

$$\text{TEM}_{(h_1, h_2)}^{P_1, P_2}(t, x) = h_2(t) \oplus P_2(h_2(t) \oplus h_1(t) \oplus P_1(h_1(t) \oplus x))$$

where  $P_1, P_2$  are two public random permutations,  $(h_1, h_2) \leftarrow_{\S} \mathcal{H}^2$  is the key,  $t$  is the tweak and  $x$  the plaintext. The main result of our paper is the following theorem.

**Theorem 2.** *Let  $\mathcal{H}$  be a uniform  $\varepsilon$ -AXU family of functions from  $\mathcal{T}$  to  $\{0, 1\}^n$ . Assume that  $q_p + 3q_c \leq N/2$  and  $q_c \leq \min\{N^{2/3}, \varepsilon^{-2/3}\}$ . Then*

$$\text{Adv}_{\text{TEM}[n, 2, \mathcal{H}]}^{\text{cca}}(q_c, q_p) \leq \frac{29\sqrt{q_c}q_p}{N} + \varepsilon\sqrt{q_c}q_p + 4\varepsilon q_c^{3/2} + \frac{30q_c^{3/2}}{N}.$$

In particular, assuming  $\mathcal{H}$  is XU for simplicity (i.e.,  $\varepsilon = 2^{-n}$ ), one can see that the two-round TEM construction ensures security up to approximately  $2^{2n/3}$  adversarial queries. In fact, for any number  $q_c \ll 2^{2n/3}$  of construction queries,

the two-round TEM construction remains secure as long as  $q_p$  is small compared with  $2^n/\sqrt{q_c}$ .

The proof uses the H-coefficients technique. As usual, we will first define bad transcripts and upper bound their probability in the ideal world, and then show that the probabilities to obtain any good transcript in the real world and the ideal world are sufficiently close.

**DEFINITION AND PROBABILITY OF BAD TRANSCRIPTS.** Let  $\tau = (\mathcal{Q}_C, \mathcal{Q}_{P_1}, \mathcal{Q}_{P_2}, (h_1, h_2))$  be an attainable transcript, with  $|\mathcal{Q}_C| = q_c$  and  $|\mathcal{Q}_{P_1}| = |\mathcal{Q}_{P_2}| = q_p$ . We let

$$U_1 = \{u_1 \in \{0, 1\}^n : (u_1, v_1) \in \mathcal{Q}_{P_1}\}, \quad V_1 = \{v_1 \in \{0, 1\}^n : (u_1, v_1) \in \mathcal{Q}_{P_1}\},$$

$$U_2 = \{u_2 \in \{0, 1\}^n : (u_2, v_2) \in \mathcal{Q}_{P_2}\}, \quad V_2 = \{v_2 \in \{0, 1\}^n : (u_2, v_2) \in \mathcal{Q}_{P_2}\}$$

denote the domains and ranges of  $\mathcal{Q}_{P_1}$  and  $\mathcal{Q}_{P_2}$  respectively. For each  $u$  and  $v \in \{0, 1\}^n$ , let

$$X_u = \{(t, x, y) \in \mathcal{Q}_C : x \oplus h_1(t) = u\},$$

$$Y_v = \{(t, x, y) \in \mathcal{Q}_C : y \oplus h_2(t) = v\}.$$

We define four quantities characterizing a transcript  $\tau$ , namely

$$\alpha_1 \stackrel{\text{def}}{=} |\{(t, x, y) \in \mathcal{Q}_C : x \oplus h_1(t) \in U_1\}|,$$

$$\alpha_2 \stackrel{\text{def}}{=} |\{(t, x, y) \in \mathcal{Q}_C : y \oplus h_2(t) \in V_2\}|,$$

$$\beta_1 \stackrel{\text{def}}{=} |\{(t, x, y) \in \mathcal{Q}_C : \exists(t', x', y') \neq (t, x, y), x \oplus h_1(t) = x' \oplus h_1(t')\}|,$$

$$\beta_2 \stackrel{\text{def}}{=} |\{(t, x, y) \in \mathcal{Q}_C : \exists(t', x', y') \neq (t, x, y), y \oplus h_2(t) = y' \oplus h_2(t')\}|.$$

In words,  $\alpha_1$  (resp.  $\alpha_2$ ) is the number of queries  $(t, x, y) \in \mathcal{Q}_C$  which “collide” with a query  $(u_1, v_1) \in \mathcal{Q}_{P_1}$  (resp. that collide with a query  $(u_2, v_2) \in \mathcal{Q}_{P_2}$ ), and  $\beta_1$  (resp.  $\beta_2$ ) is the number of queries  $(t, x, y) \in \mathcal{Q}_C$  which “collide” with another query  $(t', x', y')$  at the input of  $P_1$  (resp. at the output of  $P_2$ ). Note that one also has

$$\beta_1 = \sum_{\substack{u \in \{0,1\}^n: \\ |X_u| > 1}} |X_u|, \quad \beta_2 = \sum_{\substack{v \in \{0,1\}^n: \\ |Y_v| > 1}} |Y_v|. \tag{8}$$

**Definition 2.** We say that an attainable transcript  $\tau$  is bad if at least one of the following conditions is fulfilled (see Fig. 2 for a diagram of the first ten conditions):

- (C-1) there exists  $(t, x, y) \in \mathcal{Q}_C$ ,  $u_1 \in U_1$ , and  $v_2 \in V_2$  such that  $x \oplus h_1(t) = u_1$  and  $y \oplus h_2(t) = v_2$ ;
- (C-2) there exists  $(t, x, y) \in \mathcal{Q}_C$ ,  $(u_1, v_1) \in \mathcal{Q}_{P_1}$ , and  $u_2 \in U_2$  such that  $x \oplus h_1(t) = u_1$  and  $v_1 \oplus h_1(t) \oplus h_2(t) = u_2$ ;
- (C-3) there exists  $(t, x, y) \in \mathcal{Q}_C$ ,  $(u_2, v_2) \in \mathcal{Q}_{P_2}$ , and  $v_1 \in V_1$  such that  $y \oplus h_2(t) = v_2$  and  $v_1 \oplus h_1(t) \oplus h_2(t) = u_2$ ;

- (C-4) there exists  $(t, x, y), (t', x', y'), (t'', x'', y'') \in \mathcal{Q}_C$  with  $(t, x, y)$  distinct from  $(t', x', y')$  and from  $(t'', x'', y'')$  such that  $x \oplus h_1(t) = x' \oplus h_1(t')$  and  $y \oplus h_2(t) = y'' \oplus h_2(t'')$ ;
- (C-5) there exists  $(t, x, y) \neq (t', x', y') \in \mathcal{Q}_C$  such that  $x \oplus h_1(t) = x' \oplus h_1(t')$  and  $h_1(t) \oplus h_2(t) = h_1(t') \oplus h_2(t')$ ;
- (C-6) there exists  $(t, x, y) \neq (t', x', y') \in \mathcal{Q}_C$  such that  $y \oplus h_2(t) = y' \oplus h_2(t')$  and  $h_1(t) \oplus h_2(t) = h_1(t') \oplus h_2(t')$ ;
- (C-7) there exists  $(t, x, y) \neq (t', x', y') \in \mathcal{Q}_C$  and  $u_1 \in U_1$  such that  $y \oplus h_2(t) = y' \oplus h_2(t')$  and  $x \oplus h_1(t) = u_1$ ;
- (C-8) there exists  $(t, x, y) \neq (t', x', y') \in \mathcal{Q}_C$  and  $v_2 \in V_2$  such that  $x \oplus h_1(t) = x' \oplus h_1(t')$  and  $y \oplus h_2(t) = v_2$ ;
- (C-9) there exists  $(t, x, y) \neq (t', x', y') \in \mathcal{Q}_C, (u_1, v_1), (u'_1, v'_1) \in \mathcal{Q}_{P_1}$  such that  $x \oplus h_1(t) = u_1, x' \oplus h_1(t') = u'_1$  and  $v_1 \oplus h_1(t) \oplus h_2(t) = v'_1 \oplus h_1(t') \oplus h_2(t')$ ;
- (C-10) there exists  $(t, x, y) \neq (t', x', y') \in \mathcal{Q}_C, (u_2, v_2), (u'_2, v'_2) \in \mathcal{Q}_{P_2}$  such that  $y \oplus h_2(t) = v_2, y' \oplus h_2(t') = v'_2$  and  $u_2 \oplus h_1(t) \oplus h_2(t) = u'_2 \oplus h_1(t') \oplus h_2(t')$ ;
- (C-11)  $\alpha_1 \geq \sqrt{q_c}$ ;
- (C-12)  $\alpha_2 \geq \sqrt{q_c}$ ;
- (C-13)  $\beta_1 \geq \sqrt{q_c}$ ;
- (C-14)  $\beta_2 \geq \sqrt{q_c}$ .

Otherwise we say that  $\tau$  is good. We denote  $\Theta_{\text{good}}$ , resp.  $\Theta_{\text{bad}}$  the set of good, resp. bad transcripts. ◇

We start by upper bounding the probability to get a bad transcript in the ideal world.

**Lemma 2.** *For any integers  $q_c$  and  $q_p$ , one has*

$$\Pr[T_{\text{id}} \in \Theta_{\text{bad}}] \leq \frac{3q_c q_p^2}{N^2} + 2\varepsilon^2 q_c^3 + \frac{\varepsilon q_c^2 q_p}{N} + \frac{2\sqrt{q_c} q_p}{N} + 2\varepsilon q_c^{3/2}.$$

*Proof.* Let  $(\mathcal{Q}_C, \mathcal{Q}_{P_1}, \mathcal{Q}_{P_2})$  be any attainable queries transcript. Recall that in the ideal world,  $(h_1, h_2)$  is drawn independently from the queries transcript. We upper bound the probabilities of the fourteen conditions in turn. We denote  $\Theta_i$  the set of attainable transcripts fulfilling condition (C- $i$ ).

*Conditions (C-1), (C-2), and (C-3).* Consider condition (C-1). For any  $(t, x, y) \in \mathcal{Q}_C, u_1 \in U_1$ , and  $v_2 \in V_2$ , one has, by the uniformity of  $\mathcal{H}$  and since  $h_1$  and  $h_2$  are independently drawn,

$$\Pr [(h_1(t) = x \oplus u_1) \wedge (h_2(t) = y \oplus v_2)] = \frac{1}{N^2}.$$

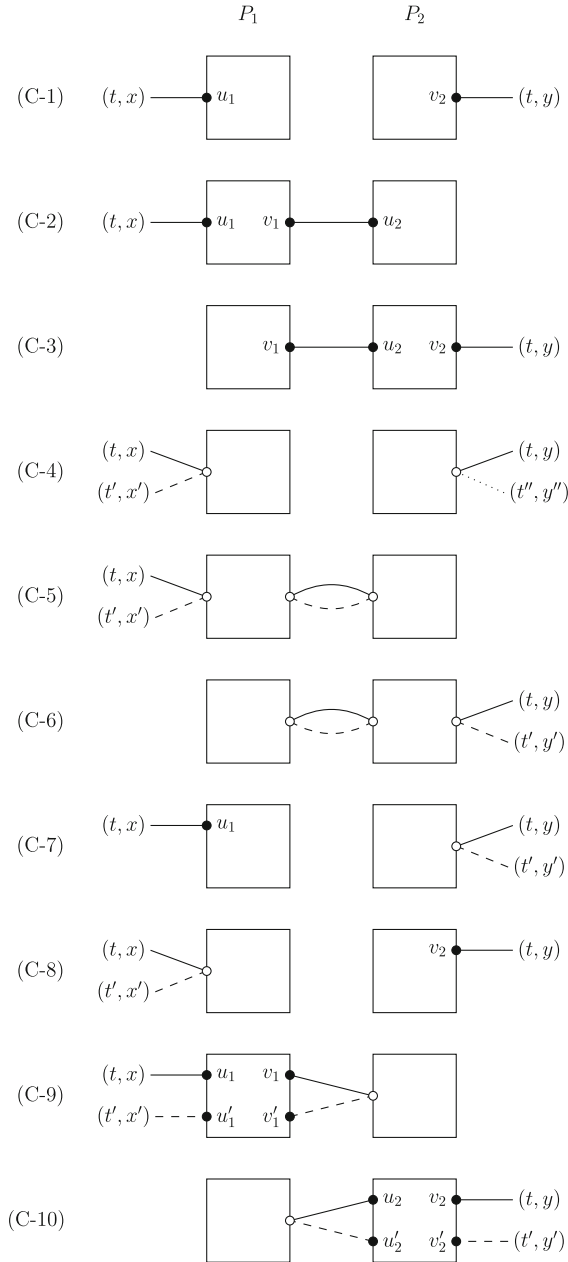
Hence, summing over the  $q_c q_p^2$  possibilities for  $(t, x, y), u_1$ , and  $v_1$  yields

$$\Pr[T_{\text{id}} \in \Theta_1] \leq \frac{q_c q_p^2}{N^2}.$$

Similarly, for (C-2) and (C-3), one obtains

$$\Pr[T_{\text{id}} \in \Theta_2] \leq \frac{q_c q_p^2}{N^2}, \quad \Pr[T_{\text{id}} \in \Theta_3] \leq \frac{q_c q_p^2}{N^2}.$$





**Fig. 2.** The ten “collision” conditions characterizing a bad transcript. Black dots correspond to pairs  $(u_1, v_1) \in \mathcal{Q}_{P_1}$  or  $(u_2, v_2) \in \mathcal{Q}_{P_2}$ . Note that for (C-4) one might have  $(t', x') = (t'', x'')$ , for (C-9) one might have  $(u_1, v_1) = (u'_1, v'_1)$ , and for (C-10) one might have  $(u_2, v_2) = (u'_2, v'_2)$ .

*Condition (C-4).* For any  $(t, x, y), (t', x', y'), (t'', x'', y'') \in \mathcal{Q}_C$  with  $(t, x, y)$  distinct from  $(t', x', y')$  and from  $(t'', x'', y'')$ , one has, by the  $\varepsilon$ -AXU property of  $\mathcal{H}$  and since  $h_1$  and  $h_2$  are drawn independently,

$$\Pr [(h_1(t) \oplus h_1(t') = x \oplus x') \wedge (h_2(t) \oplus h_2(t'') = y \oplus y'')] \leq \varepsilon^2.$$

Note that this also holds when  $t = t'$  (resp.  $t = t''$ ) since in that case necessarily  $x \neq x'$  (resp.  $y \neq y''$ ) by the assumption that  $\mathcal{D}$  never makes pointless queries. Hence, summing over the (at most)  $q_c^3$  possibilities for  $(t, x, y), (t', x', y'), (t'', x'', y'')$ , one obtains

$$\Pr [T_{\text{id}} \in \Theta_4] \leq \varepsilon^2 q_c^3.$$

*Conditions (C-5) and (C-6).* For any two distinct queries  $(t, x, y) \neq (t', x', y') \in \mathcal{Q}_C$ , one has, by the  $\varepsilon$ -AXU property of  $\mathcal{H}$  and since  $h_1$  and  $h_2$  are drawn independently,

$$\Pr [(h_1(t) \oplus h_1(t') = x \oplus x') \wedge (h_2(t) \oplus h_2(t') = h_1(t) \oplus h_1(t'))] \leq \varepsilon^2.$$

Hence, summing over the  $q_c(q_c - 1)/2$  possible pairs of distinct queries, we get

$$\Pr [T_{\text{id}} \in \Theta_5] \leq \frac{\varepsilon^2 q_c^2}{2}, \quad \text{and similarly } \Pr [T_{\text{id}} \in \Theta_6] \leq \frac{\varepsilon^2 q_c^2}{2}.$$

*Conditions (C-7) and (C-8).* For any two distinct queries  $(t, x, y) \neq (t', x', y') \in \mathcal{Q}_C$  and any  $u_1 \in U_1$ , one has, by the  $\varepsilon$ -AXU property and uniformity of  $\mathcal{H}$  and since  $h_1$  and  $h_2$  are drawn independently,

$$\Pr [(h_2(t) \oplus h_2(t') = y \oplus y') \wedge (h_1(t) = x \oplus u_1)] \leq \frac{\varepsilon}{N}.$$

Then, summing over  $(t, x, y) \neq (t', x', y')$  and  $u_1$ ,

$$\Pr [T_{\text{id}} \in \Theta_7] \leq \frac{\varepsilon q_c^2 q_p}{2N}, \quad \text{and similarly } \Pr [T_{\text{id}} \in \Theta_8] \leq \frac{\varepsilon q_c^2 q_p}{2N}.$$

*Conditions (C-9), (C-10), (C-11), and (C-12).* We will deal with conditions (C-9) and (C-11) together, using the fact that

$$\Pr [T_{\text{id}} \in \Theta_9 \cup \Theta_{11}] = \Pr [T_{\text{id}} \in \Theta_{11}] + \Pr [T_{\text{id}} \in \Theta_9 \setminus \Theta_{11}].$$

To upper bound  $\Pr [T_{\text{id}} \in \Theta_{11}]$ , we see  $\alpha_1$  as a random variable over the random choice of  $h_1$  (since  $\alpha_1$  does not depend on  $h_2$ ). First, note that by the uniformity of  $\mathcal{H}$ ,

$$\mathbb{E}[\alpha_1] = \sum_{(t,x,y) \in \mathcal{Q}_C} \sum_{u_1 \in U_1} \Pr [x \oplus h_1(t) = u_1] = \frac{q_c q_p}{N},$$

so that by Markov's inequality,

$$\Pr [T_{\text{id}} \in \Theta_{11}] \leq \frac{\sqrt{q_c q_p}}{N}.$$

Fix any  $h'_1 \in \mathcal{H}$  such that, when  $h_1 = h'_1$ ,  $\alpha_1 < \sqrt{q_c}$ , and fix any queries  $(t, x, y) \neq (t', x', y') \in \mathcal{Q}_C$ ,  $(u_1, v_1), (u'_1, v'_1) \in \mathcal{Q}_{P_1}$  such that  $x \oplus h_1(t) = u_1$  and  $x' \oplus h_1(t') = u'_1$ . Note that since  $\alpha_1 < \sqrt{q_c}$ , there are at most  $\frac{q_c}{2}$  such tuple of queries. Then

$$\Pr \left[ (h_1 = h'_1) \wedge (h_2(t) \oplus h_2(t') = v_1 \oplus h_1(t) \oplus v'_1 \oplus h_1(t')) \right] \leq \frac{\varepsilon}{|\mathcal{H}|},$$

and, by summing over every  $h_1$  such that  $\alpha_1 < \sqrt{q_c}$  and every such tuple of queries, one has

$$\Pr [T_{\text{id}} \in \Theta_9 \setminus \Theta_{11}] \leq \frac{\varepsilon q_c}{2}.$$

Finally,

$$\Pr [T_{\text{id}} \in \Theta_9 \cup \Theta_{11}] \leq \frac{\sqrt{q_c} q_p}{N} + \frac{\varepsilon q_c}{2}.$$

Similarly,

$$\Pr [T_{\text{id}} \in \Theta_{10} \cup \Theta_{12}] \leq \frac{\sqrt{q_c} q_p}{N} + \frac{\varepsilon q_c}{2}.$$

*Conditions (C-13) and (C-14).* For every  $u \in \{0, 1\}^n$ , we see  $|X_u|$  as a random variable over the random choice of  $h_1$ . We also introduce the random variable

$$C = |\{(t, x, y), (t', x', y') \in \mathcal{Q}_C^2, (t, x, y) \neq (t', x', y') : x \oplus h_1(t) = x' \oplus h_1(t')\}|.$$

Then, by definition of  $\beta_1$ ,

$$\beta_1 = |\{(t, x, y) \in \mathcal{Q}_C : \exists (t', x', y') \neq (t, x, y), x \oplus h_1(t) = x' \oplus h_1(t')\}| \leq C.$$

Hence,  $\Pr [T_{\text{id}} \in \Theta_{13}] \leq \Pr [C \geq \sqrt{q_c}]$ . Note that

$$\mathbb{E}[C] = \sum_{(t,x,y) \neq (t',x',y')} \Pr [x \oplus h_1(t) = x' \oplus h_1(t')] \leq \frac{\varepsilon q_c^2}{2}.$$

By Markov's inequality,

$$\Pr [T_{\text{id}} \in \Theta_{13}] \leq \frac{\varepsilon q_c^{3/2}}{2}, \quad \text{and similarly } \Pr [T_{\text{id}} \in \Theta_{14}] \leq \frac{\varepsilon q_c^{3/2}}{2}.$$

The result follows by an union bound over all conditions. □

ANALYSIS OF GOOD TRANSCRIPTS. Next, we have to study good transcripts.

**Lemma 3.** *Let  $q_c$  and  $q_p$  be integers such that  $q_p + 3q_c \leq N/2$ . Then for any good transcript  $\tau$ , one has*

$$\frac{\Pr [T_{\text{re}} = \tau]}{\Pr [T_{\text{id}} = \tau]} \geq 1 - \left( \frac{4q_c(q_p + 2q_c)^2}{N^2} + \frac{14q_c^{3/2} + 4\sqrt{q_c}q_p}{N} \right).$$

*Proof.* Deferred to the full version of the paper [8] for reasons of space. □

CONCLUDING THE PROOF OF THEOREM 2. We are now ready to prove Theorem 2. Combining Lemmas 1, 2, and 3, one has

$$\begin{aligned}
 \text{Adv}_{\text{TEM}[n,2,\mathcal{H}]}^{\text{cca}}(q_c, q_p) &\leq \frac{3q_c q_p^2}{N^2} + 2\varepsilon^2 q_c^3 + \frac{\varepsilon q_c^2 q_p}{N} + \frac{2\sqrt{q_c} q_p}{N} + 2\varepsilon q_c^{3/2} \\
 &\quad + \frac{4q_c(q_p + 2q_c)^2}{N^2} + \frac{14q_c^{3/2} + 4\sqrt{q_c} q_p}{N} \\
 &= \frac{7q_c q_p^2}{N^2} + \frac{16q_c^2 q_p}{N^2} + \frac{6\sqrt{q_c} q_p}{N} + \frac{\varepsilon q_c^2 q_p}{N} + 2\varepsilon^2 q_c^3 + 2\varepsilon q_c^{3/2} \\
 &\quad + \frac{16q_c^3}{N^2} + \frac{14q_c^{3/2}}{N} \\
 &\leq \frac{7q_c q_p^2}{N^2} + \frac{16q_c^2 q_p}{N^2} + \frac{6\sqrt{q_c} q_p}{N} + \frac{\varepsilon q_c^2 q_p}{N} + 4\varepsilon q_c^{3/2} + \frac{30q_c^{3/2}}{N},
 \end{aligned}$$

where for the last inequality we used the assumption that  $q_c \leq \min\{N^{2/3}, \varepsilon^{-2/3}\}$ . Since the result holds trivially when  $q_c q_p^2 > N^2$ , we can assume that  $q_c q_p^2 \leq N^2$ , so that  $q_c q_p^2 / N^2 \leq \sqrt{q_c} q_p / N$ . Moreover, since  $q_c \leq N^{2/3}$ , one has  $q_c^2 / N^2 \leq \sqrt{q_c} / N$  and  $q_c^2 / N \leq \sqrt{q_c}$ , which concludes the proof of Theorem 2.

### 4 Asymptotic Bounds via the Coupling Technique

When the number of rounds  $r$  of the TEM construction grows, one has the following result.

**Theorem 3.** *Let  $r$  be an even integer and  $r' = r/2$ . Let  $q_c, q_p$  be positive integers, and  $\mathcal{H}$  be a uniform  $\varepsilon$ -AXU family of functions from  $\mathcal{T}$  to  $\{0, 1\}^n$ . Then:*

$$\text{Adv}_{\text{TEM}[n,r,\mathcal{H}]}^{\text{cca}}(q_c, q_p) \leq \sqrt{2^{r'+4} \frac{q_c(N\varepsilon q_c + q_p)^{r'}}{N^{r'}}}.$$

For odd  $r$ , we have  $\text{Adv}_{\text{TEM}[n,r,\mathcal{H}]}^{\text{cca}} \leq \text{Adv}_{\text{TEM}[n,r-1,\mathcal{H}]}^{\text{cca}}$ , so that we can use the above bound with  $r - 1$ . Using an  $\varepsilon$ -AXU function family with  $\varepsilon \simeq 2^{-n}$ , we see that the iterated tweakable Even-Mansour cipher with an even number  $r$  of rounds achieves CCA-security up to roughly  $2^{\frac{rn}{r+2}}$  adversarial queries.

The proof relies on the coupling technique. Since it combines in a rather straightforward way the approach of [21, 23], the proof is entirely deferred to the full version of the paper [8].

### References

1. Andreeva, E., Bogdanov, A., Dodis, Y., Mennink, B., Steinberger, J.P.: On the indistinguishability of key-alternating ciphers. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 531–550. Springer, Heidelberg (2013). <http://eprint.iacr.org/2013/061>

2. Andreeva, E., Bogdanov, A., Luykx, A., Mennink, B., Tischhauser, E., Yasuda, K.: Parallelizable and authenticated online ciphers. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part I. LNCS, vol. 8269, pp. 424–443. Springer, Heidelberg (2013)
3. Andreeva, E., Bogdanov, A., Mennink, B.: Towards understanding the known-key security of block ciphers. In: Moriai, S. (ed.) FSE 2013. LNCS, vol. 8424, pp. 348–366. Springer, Heidelberg (2014)
4. 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)
5. Bogdanov, A., Knudsen, L.R., Leander, G., Standaert, F.-X., Steinberger, J., Tischhauser, E.: Key-alternating ciphers in a provable setting: encryption using a small number of public permutations. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 45–62. Springer, Heidelberg (2012)
6. Chen, S., Lampe, R., Lee, J., Seurin, Y., Steinberger, J.: Minimizing the two-round even-mansour cipher. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 39–56. Springer, Heidelberg (2014). <http://eprint.iacr.org/2014/443>
7. Chen, S., Steinberger, J.: Tight security bounds for key-alternating ciphers. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 327–350. Springer, Heidelberg (2014). <http://eprint.iacr.org/2013/222>
8. Cogliati, B., Lampe, R., Seurin, Y.: Tweaking even-mansour ciphers. Full version of this paper. <http://eprint.iacr.org/2015/539>
9. Cogliati, B., Seurin, Y.: On the provable security of the iterated even-mansour cipher against related-key and chosen-key attacks. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 584–613. Springer, Heidelberg (2015). <http://eprint.iacr.org/2015/069>
10. Crowley, P.: Mercy: a fast large block cipher for disk sector encryption. In: Schneier, B. (ed.) FSE 2000. LNCS, vol. 1978, pp. 49–63. Springer, Heidelberg (2001)
11. 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)
12. Dunkelman, O., Keller, N., Shamir, A.: Minimalism in cryptography: the even-mansour scheme revisited. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 336–354. Springer, Heidelberg (2012)
13. Even, S., Mansour, Y.: A construction of a cipher from a single pseudorandom permutation. *J. Crypt.* **10**(3), 151–162 (1997)
14. Farshim, P., Procter, G.: The related-key security of iterated even-mansour ciphers. In: Fast Software Encryption - FSE 2015 (2015, to appear). Full version available at <http://eprint.iacr.org/2014/953>
15. Ferguson, N., Lucks, S., Schneier, B., Whiting, D., Bellare, M., Kohno, T., Callas, J., Walker, J.: The skein hash function family. SHA3 Submission to NIST (Round 3) (2010)
16. Goldenberg, D., Hohenberger, S., Liskov, M., Schwartz, E.C., Seyalioglu, H.: On tweaking luby-rackoff blockciphers. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 342–356. Springer, Heidelberg (2007)
17. Halevi, S., Rogaway, P.: A tweakable enciphering mode. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 482–499. Springer, Heidelberg (2003)
18. Halevi, S., Rogaway, P.: A parallelizable enciphering mode. In: Okamoto, T. (ed.) CT-RSA 2004. LNCS, vol. 2964, pp. 292–304. Springer, Heidelberg (2004)
19. Hoang, V.T., Rogaway, P.: On generalized feistel networks. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 613–630. Springer, Heidelberg (2010)

20. Jean, J., Nikolic, I., Peyrin, T.: Tweaks and keys for block ciphers: the tweakable framework. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8874, pp. 274–288. Springer, Heidelberg (2014)
21. Lampe, R., Patarin, J., Seurin, Y.: An asymptotically tight security analysis of the iterated even-mansour cipher. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 278–295. Springer, Heidelberg (2012)
22. Lampe, R., Seurin, Y.: How to construct an ideal cipher from a small set of public permutations. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part I. LNCS, vol. 8269, pp. 444–463. Springer, Heidelberg (2013). <http://eprint.iacr.org/2013/255>
23. Lampe, R., Seurin, Y.: Tweakable blockciphers with asymptotically optimal security. In: Moriai, S. (ed.) FSE 2013. LNCS, vol. 8424, pp. 133–152. Springer, Heidelberg (2014)
24. Landecker, W., Shrimpton, T., Terashima, R.S.: Tweakable blockciphers with beyond birthday-bound security. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 14–30. Springer, Heidelberg (2012). <http://eprint.iacr.org/2012/450>
25. Liskov, M., Rivest, R.L., Wagner, D.: Tweakable block ciphers. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 31–46. Springer, Heidelberg (2002)
26. Minematsu, K.: Beyond-birthday-bound security based on tweakable block cipher. In: Dunkelman, O. (ed.) FSE 2009. LNCS, vol. 5665, pp. 308–326. Springer, Heidelberg (2009)
27. Mitsuda, A., Iwata, T.: Tweakable pseudorandom permutation from generalized feistel structure. In: Baek, J., Bao, F., Chen, K., Lai, X. (eds.) ProvSec 2008. LNCS, vol. 5324, pp. 22–37. Springer, Heidelberg (2008)
28. Morris, B., Rogaway, P., Stegers, T.: How to encipher messages on a small domain. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 286–302. Springer, Heidelberg (2009)
29. Nyberg, K., Knudsen, L.R.: Provable security against differential cryptanalysis. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 566–574. Springer, Heidelberg (1993)
30. Patarin, J.: The “Coefficients H” technique. In: Avanzi, R.M., Keliher, L., Sica, F. (eds.) SAC 2008. LNCS, vol. 5381, pp. 328–345. Springer, Heidelberg (2009)
31. Procter, G.: A note on the CLRW2 tweakable block cipher construction. IACR Cryptology ePrint Archive, report 2014/111 (2014). <http://eprint.iacr.org/2014/111>
32. Rogaway, P.: Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 16–31. Springer, Heidelberg (2004)
33. Rogaway, P., Bellare, M., Black, J.: OCB: a block-cipher mode of operation for efficient authenticated encryption. *ACM Trans. Inf. Syst. Secur.* **6**(3), 365–403 (2003)
34. Rogaway, P., Zhang, H.: Online ciphers from tweakable blockciphers. In: Kiayias, A. (ed.) CT-RSA 2011. LNCS, vol. 6558, pp. 237–249. Springer, Heidelberg (2011)
35. Sasaki, Y., Todo, Y., Aoki, K., Naito, Y., Sugawara, T., Murakami, Y., Matsui, M., Hirose, S.: Minalpher v1. Submission to the CAESAR competition (2014)
36. Schroepel, R.: The hasty pudding cipher. AES submission to NIST (1998)
37. Shoup, V.: On fast and provably secure message authentication based on universal hashing. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 313–328. Springer, Heidelberg (1996)
38. Steinberger, J.: Improved security bounds for Key-alternating ciphers via Hellinger distance. IACR Cryptology ePrint Archive, report 2012/481 (2012). <http://eprint.iacr.org/2012/481>

# Multi-key Security: The Even-Mansour Construction Revisited

Nicky Mouha<sup>1,2</sup> and Atul Luykx<sup>1</sup> (✉)

<sup>1</sup> Department of Electrical Engineering-ESAT/COSIC,  
KU Leuven, Leuven and iMinds, Ghent, Belgium  
{Nicky.Mouha,Atul.Luykx}@esat.kuleuven.be

<sup>2</sup> Project-team SECRET, Inria, Paris, France

**Abstract.** At ASIACRYPT 1991, Even and Mansour introduced a block cipher construction based on a single permutation. Their construction has since been lauded for its simplicity, yet also criticized for not providing the same security as other block ciphers against generic attacks. In this paper, we prove that if a small number of plaintexts are encrypted under multiple independent keys, the Even-Mansour construction surprisingly offers similar security as an ideal block cipher with the same block and key size. Note that this *multi-key setting* is of high practical relevance, as real-world implementations often allow frequent rekeying. We hope that the results in this paper will further encourage the use of the Even-Mansour construction, especially when a secure and efficient implementation of a key schedule would result in significant overhead.

**Keywords:** Even-Mansour · Multi-key setting · Broadcast attack · Related-key setting

## 1 Introduction

Modern block cipher design is based on the concept of iterating a round function [44]. This round function typically consists of a subkey addition followed by an unkeyed invertible function. All commonly-used block ciphers, including the DES [46] and AES [23] standards, follow this design strategy.

As such, the design of an iterated block cipher consists of two parts: the design of a round function, and a key schedule to generate the subkeys for every round. Although round function design seems to be a relatively well-understood problem, this is much less the case for the key schedule. For example, Rijmen and Daemen already stated in the AES design book that: “*There is no consensus on the criteria that a key schedule must satisfy*” [23, p. 77]. This fact has been repeated many times since, for example in the SHA-3 finalist Grøstl design document [32, p. 5]. In particular, there seems to be no consensus on whether a “strong” or a “simple” key schedule is required, a choice which appears to depend on the application.

An argument for a “simple” key schedule is that keys should be chosen uniformly at random from the entire key space anyway, in order to avoid a speed-up

of brute-force attacks due to low key entropy. As a result, attacks based on weak keys [25] and known keys [42] are no longer applicable. Similarly, when multiple keys are used, they should be chosen independently to prevent a compromise of one key helping the recovery of other keys. This avoids attacks based on related keys [9–11].

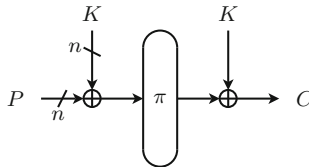
Proponents of a “strong” key schedule point out that in practice, keys may not always be chosen independently from a uniformly random distribution. The cause of this could be a weak protocol, a programming error or an insecure implementation.

Complexity, however, always comes at a cost. It makes cryptosystems more difficult to design, to implement and to analyze. Hence, we argue for a block cipher with a simple key schedule, combined with the use of a secure key derivation function (KDF) [20] for secret-key applications. A KDF can be as simple as using the same block cipher to encrypt a counter, in which case the implementation overhead is minimal. For a theoretical treatment of KDFs, we refer to [1].

Although the use of a KDF avoids attacks on weak keys, known keys and related keys, it cannot prevent multi-key attacks [12, 14, 24, 38]: a plaintext may be encrypted under multiple independent keys. Often overlooked by block cipher designers, multi-key attacks are highly relevant in practice (cf. Sect. 2).

This leads us to the following open problem, formulated by Daemen and Rijmen in 2012 [24]. In their paper, they point out: “A scenario where the adversary can query the block cipher under related keys, or even multiple keys, inevitably leads to security erosion”. Paraphrased, concerning the multi-key setting they ask: “Can we design a secure block cipher with a lighter key schedule and higher key agility if related-key security is not required?”

In this paper, we give a positive answer to their question. Surprisingly, one of the simplest block ciphers, the single-key Even-Mansour construction<sup>1</sup> [26, 29, 30], shown in Fig. 1, offers similar security to an ideal block cipher when a small number of plaintexts can be queried under many keys.



**Fig. 1.** The Even-Mansour construction.

**Outline.** The single-key, related-key and multi-key attack settings are discussed in Sect. 2, from a theoretical as well as a practical perspective. In Sect. 3, we prove tight bounds for the security of an Even-Mansour block cipher and of an ideal

<sup>1</sup> Throughout this paper, we will always refer to the single-key variant of the Even-Mansour construction, that is, using only a single  $n$ -bit key  $K$ .



block cipher in the multi-key setting. The relevance of our observations for the security and efficiency of block cipher implementations is discussed in Sect. 4. We conclude the paper in Sect. 5.

## 2 Attack Settings

### 2.1 Three Attack Settings

**Single-Key Setting.** One key  $K$  is chosen uniformly at random from the key space in the *single-key setting*, also referred to as the *fixed-key setting*. The adversary can then make encryption and decryption queries to block cipher  $E$ , all under the same key  $K$ .

**Related-Key Setting.** In the *related-key setting*, the adversary can perform encryption and decryption queries to block cipher  $E$  under keys  $K_i$ . Each key  $i$  satisfies the relationship  $K_i = \Phi_i(K)$ , where  $K$  is secret, but the functions  $\Phi_i$  are chosen by the adversary. To avoid that not every block cipher  $E$  is vulnerable to a related-key attack, restrictions are necessary on the functions  $\Phi_i$  as explained in [9].

Security against related-key attacks is often considered in the design of a block cipher. For example, it was a stated design goal for the AES block cipher [23], although it was shown that AES is not secure against related-key attacks [15, 16].

Furthermore, it should be noted that certain commonly-used algorithms, including DES and Triple-DES, are trivially insecure against related-key attacks. For DES [46] and Triple-DES [6], this is an immediate result of its complementation property [36]:  $E_K(P) = E_{\bar{K}}(\bar{P})$ , where  $\bar{x}$  represents the bitwise complement of  $x$ .

It is difficult to say whether related-key security should be a requirement, as this depends on the protocol in which the cryptosystem is used. Nevertheless, it seems fair to point out that protocol designers should not assume related-key security, given that several commonly-used designs are (sometimes trivially) insecure in this setting.

**Multi-key Setting.** In the *multi-key setting*, the adversary can query encryption and decryption queries under keys  $K_i$ , where all  $K_i$  are independently chosen, uniformly at random. The multi-key setting can be seen as a generalization of the *multi-user setting* of Chatterjee et al. [19], where encryption queries of only one plaintext  $P$  are allowed under keys  $K_i$ .<sup>2</sup> This multi-user setting is then

---

<sup>2</sup> In the model of Chatterjee et al. [19], an adversary can also *corrupt* any user of its choosing, meaning that their key is given to the adversary. The goal of the adversary is then to win the game for any uncorrupted user. Although it is straightforward to take this refinement into account, we decided not to do this for the clarity of our exposition.

again a further generalization of the *broadcast setting* of Mantin and Shamir [45], where the plaintext  $P$  is unknown to the attacker.

Every attack in the broadcast setting also leads to an attack in the multi-user setting, and every multi-user attack is also a multi-key attack. We will therefore use the multi-key setting throughout this paper, in order to evaluate the security against the most powerful adversaries.

## 2.2 Practical Relevance of the Multi-key Setting

The terms “broadcast” and “multi-user” imply a setting where one message is sent to many users, encrypted under independent keys. Note, however, that this setting does not actually require a large amount of users, and also applies to one user that rekeys frequently.

Frequent rekeying is often a result of the common implementation practice to use *session keys*. As explained in [48, Sect. 12.2.2], session keys limit the available ciphertext under the same key for cryptanalytic attacks, and limit the exposure in case a session key is compromised.

Furthermore, rekeying is necessary in certain scenarios in order to avoid cryptanalytical attacks or to comply with existing standards. It should be noted that several cryptanalytical attacks have a higher success probability when more plaintext-ciphertext pairs under a specific key are available [8].

For example, NIST limits the amount of plaintext that can be processed under the same key to  $2^{32}$  blocks (32 GB) for three-key Triple-DES, and to  $2^{20}$  blocks (16 MB) for two-key Triple-DES [6]. In the case of MAC functions, NIST not only recommends to limit the number of message blocks under the same key, but also to limit the number of MAC failures before rekeying is required [27, 28]. In the case of TLS, rekeying is required after only one MAC failure [34].

AlFardan et al. [3] showed that it is a realistic attack vector in the case of TLS to obtain the encryption of one secret (a cookie or password) under multiple independent keys. They explained that this can be done either by using JavaScript malware to generate multiple sessions, or by causing the session to be terminated, after which some applications automatically reconnect and retransmit the cookie or password. As shown by Paterson et al. [54], the same attack setting also applies to WPA-TKIP because of its use of per-packet keys.

## 2.3 Security in the Multi-key Setting

As noted by Biham [12, 13], there exists a faster generic key-recovery attack on any block cipher in the multi-key setting compared to the single-key setting. This can be seen as follows. To keep our explanation simple, let us assume in this section that key size  $k$  equals the block size  $n$ .

In the single-key setting, an adversary with  $D = 1$  plaintext-ciphertext pair will need on average  $T = 2^{k-1}$  encryptions to recover the key by exhaustive search with a success probability of about 50%. More plaintext-ciphertext pairs will increase the success probability of the attack, as probability decreases that

a random key will be found instead of the correct key, but will not reduce the time complexity of the attack.

Recovering one key can be done with a lower time complexity in the multi-key setting. To see this, let an attacker have  $D$  encryptions

$$E_{K_1}(P), E_{K_2}(P), \dots, E_{K_D}(P) \tag{1}$$

of the same plaintext  $P$  under multiple independent keys  $K_1, K_2, \dots, K_D$ . Then, after on average  $T = 2^{k-1}/D$  encryptions of the plaintext  $P$ , one of the keys  $K_1, K_2, \dots, K_D$  will be recovered with a success probability of about 50%.

Besides this observation, Biham also remarked in [12,13] that key collisions become likely in this multi-key setting after about  $D = 2^{k/2}$  plaintext-ciphertext pairs. Consequently, the key size  $k$  should be chosen to be sufficiently large by design to avoid key collision attacks.

In Sect. 3, we will prove that the Even-Mansour construction has similar security to an ideal block cipher in the multi-key setting, assuming the number of plaintexts queried per key is small. Or put differently, when multi-key attacks with a small amount of plaintext per key are taken into account, there is little advantage in choosing a block cipher with a more complicated key schedule than the Even-Mansour construction.

## 2.4 Related Work

The time-memory tradeoff of Hellman [37] is not a concern for block ciphers with a reasonably long key size, because its precomputation time is the same as that of exhaustive key search. This is different from the time-memory-data tradeoffs for stream ciphers of Babbage-Golić [5,33] and Biryukov-Shamir [17], where the time complexity can be far below exhaustive search.

As shown by Hong and Sarkar [38], and independently by Biryukov [14], the stream cipher time-memory-data tradeoffs can be applied to the block cipher setting as well, assuming that a plaintext is encrypted under multiple keys. Their work generalizes the findings of Biham that we presented in Sect. 2.3.

Chatterjee, Kobitz, Menezes and Sarkar critiqued the security proofs of symmetric-key encryption and authentication modes [19,43,47], pointing out that security is often reduced when a multi-user setting is considered.

Their findings inspired Fouque et al. [31] to look at collision search algorithms in the multi-user setting. One of their results is the first analysis of the Even-Mansour construction in the multi-user setting. We will use an entirely different approach in this paper, by considering information-theoretic adversaries that are only limited by the number of queries to the Even-Mansour block cipher and to the underlying permutation. As we will show, the attack by Fouque et al. reaches the security bound that we will prove for the Even-Mansour construction in the multi-key setting.

A recent paper by Andreeva et al. [4] considers the security of keyed sponge constructions in the single-target and multi-target scenarios, which are similar to our single-key and multi-key settings. The approach that we follow in this

paper is different, as we introduce these concepts in a more general way. Furthermore, we do not express the attacks and security bounds in terms of the “total maximum multiplicity  $\mu$ ”, a parameter that is specific to the analysis of sponge constructions.

Note that the multi-user setting is not only relevant for symmetric-key cryptography, but also for public-key cryptography. For a theoretical treatment of public-key encryption in the multi-user setting, we refer to Bellare et al. [7].

### 3 Security Proofs in the Multi-key Setting

Block cipher security in the multi-key setting is formalized with a distinguisher comparing two worlds, one in which the distinguisher is given access to a block cipher instantiated with  $\ell$  keys, and one in which it is given access to  $\ell$  independent permutations. Our focus is on constructions in the ideal model, meaning they make use of an ideal primitive.

**Definition 1.** *An ideal primitive is a uniformly distributed random variable over a set of functions  $F$ .*

These primitives model basic components from which cryptographic algorithms are constructed. In line with Kerckhoffs’s principle, ideal primitives are public and can be accessed by adversaries in security definitions. The adversaries themselves are information-theoretic and are only bounded in the number of queries they make to each oracle.

Let  $\text{perm}(n)$  denote the set of all permutations on  $n$  bits, and  $\text{block}(k, n)$  denote the set of all block ciphers with  $k$ -bit key and  $n$ -bit block size. Let  $\ell$  denote number of keys  $K_i$  under which the adversary performs queries, that is, there is at least one query for every key  $K_i$  for  $1 \leq i \leq \ell$ .

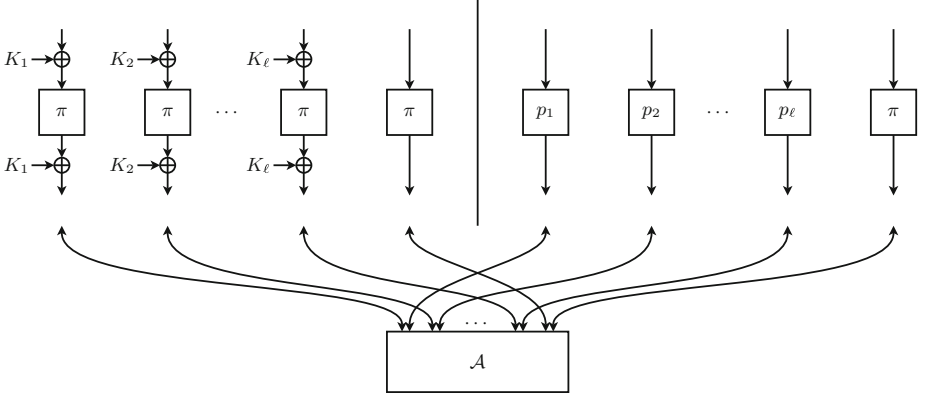
**Definition 2 (Multi-key Security).** *Let  $\Pi$  be a primitive and  $E^\Pi$  a random variable over  $\text{block}(k, n)$ . Given an adversary  $\mathcal{A}$ , its multi-key advantage with respect to  $\ell$  keys is*

$$\text{Adv}_E^{\text{mk}}(\mathcal{A}) = \left| \Pr \left( \mathcal{A}^{E_{K_1}^\Pi, E_{K_2}^\Pi, \dots, E_{K_\ell}^\Pi, \Pi} \rightarrow 1 \right) - \Pr \left( \mathcal{A}^{p_1, p_2, \dots, p_\ell, \Pi} \rightarrow 1 \right) \right|, \quad (2)$$

where the keys  $K_1, \dots, K_\ell$  are independently and uniformly drawn from  $\{0, 1\}^k$ , and  $p_1, \dots, p_\ell$  are independently and uniformly drawn from  $\text{perm}(n)$ . The adversary  $\mathcal{A}$  has access to both forward and inverse oracles.

In the case of the Even-Mansour block cipher, the primitive  $\Pi$  is a permutation, whereas for an ideal block cipher, the primitive  $\Pi$  is the block cipher itself.

Note that our definition is similar to the “3PRP” notion in the security analysis of Chaskey [49], however we consider  $\ell$  independent keys instead of three keys that are related to each other. Our definition also closely follows the “Joint Distinguishing Advantage” of Andreeva et al. [4, Definition 2], except that the total maximum multiplicity  $\mu$  is not a parameter in our security definition.



**Fig. 2.** An Even-Mansour block cipher  $E_K(P) = \pi(P \oplus K) \oplus K$  in the multi-key setting. Although only one direction is shown, inverse oracles can be accessed as well. The number of queries by the adversary  $\mathcal{A}$  to any of the first  $\ell$  oracles is denoted by  $D$ , the number of queries to the last oracle by  $T$ .

**Theorem 1 (Even-Mansour Multi-key Security).** *Let EM be the Even-Mansour block cipher  $E_K(P) = \pi(P \oplus K) \oplus K$ , then for all  $\mathcal{A}$  making at most  $D$  queries to  $E_{K_1}, \dots, E_{K_\ell}$  (resp.  $p_1, \dots, p_\ell$ ) or their inverses and at most  $T$  queries to  $\pi$  or  $\pi^{-1}$ ,*

$$\text{Adv}_{\text{EM}}^{\text{mk}}(\mathcal{A}) \leq \frac{D^2 + 2DT}{2^n}. \quad (3)$$

Our proof is similar to the security proof of the MAC function Chaskey [49], except that we now consider that  $\ell$  keys are drawn independently and uniformly at random. The proof uses Patarin’s H-coefficient technique [53]. For a detailed explanation of this technique, we refer to Chen and Steinberger [21]. The proof can be seen as a generalization of the security analysis of the Even-Mansour block cipher [29, 30].

*Proof.* As shown in Fig. 2, we consider an adversary  $\mathcal{A}$  that has bidirectional access to  $\ell + 1$  oracles  $(\mathcal{O}_1, \dots, \mathcal{O}_{\ell+1})$ . In the real world, these are  $(E_{K_1}, \dots, E_{K_\ell}, \pi)$  (where  $E_K(P) = \pi(P \oplus K) \oplus K$ ) with  $K_i \xleftarrow{\$} \{0, 1\}^n$  for  $i = 1, \dots, \ell$ ,  $\pi \xleftarrow{\$} \text{perm}(n)$ , and in the ideal world these are  $(p_1, \dots, p_\ell, \pi) \xleftarrow{\$} \text{perm}(n)^{\ell+1}$ . Without loss of generality we assume that  $\mathcal{A}$  is deterministic. It makes  $D_i$  queries to oracle  $\mathcal{O}_i$  for  $i = 1, \dots, \ell$ , and  $T$  queries to  $\mathcal{O}_{\ell+1}$ . Let  $D = \sum_{i=1}^{\ell} D_i$ . To be overly generous to the adversary  $\mathcal{A}$ , after it has made all of its  $D + T$  queries, but before it outputs its decision, we will reveal the keys  $K_1, \dots, K_\ell$  (in the real world) or randomly generated dummy keys  $K_1, \dots, K_\ell$  (in the ideal world).

The interaction of  $\mathcal{A}$  with the oracles can be summarized by a transcript  $\tau = (K_1, \dots, K_\ell, \tau_1, \dots, \tau_{\ell+1})$ . Here, the directionless list of queries to  $\mathcal{O}_j$  for  $i = 1, \dots, \ell$  is denoted by  $\tau_i = \{(P_i^{(1)}, C_i^{(1)}), \dots, (P_i^{(D_i)}, C_i^{(D_i)})\}$ , and to  $\mathcal{O}_{\ell+1}$

by  $\tau_{\ell+1} = \{(x^{(1)}, y^{(1)}), \dots, (x^{(T)}, y^{(T)})\}$ . We assume the adversary never makes duplicate queries, so that  $P_i^{(j)} \neq P_i^{(j')}$ ,  $C_i^{(j)} \neq C_i^{(j')}$ ,  $x^{(j)} \neq x^{(j')}$ , and  $y^{(j)} \neq y^{(j')}$  for all  $i, j, j'$  where  $j \neq j'$ .

Given the fixed deterministic adversary  $\mathcal{A}$ , we denote the probability distribution of transcripts in the real world by  $X$ , and in the ideal world by  $Y$ . We say that a transcript  $\tau$  is attainable if it can be obtained from interacting with  $(p_1, \dots, p_\ell, \pi)$ , hence if  $\Pr(Y = \tau) > 0$ . According to the H-coefficient technique, we have (see [21] for a proof):

**Lemma 1 (H-coefficient Technique).** *Let us consider a fixed deterministic adversary  $\mathcal{A}$ , and let  $\mathcal{T} = \mathcal{T}_{\text{good}} \cup \mathcal{T}_{\text{bad}}$  be a partition of the set of attainable transcripts. Let  $\varepsilon$  be such that for all  $\tau \in \mathcal{T}_{\text{good}}$*

$$\frac{\Pr(X = \tau)}{\Pr(Y = \tau)} \geq 1 - \varepsilon. \tag{4}$$

Then,  $\text{Adv}_{\text{EM}}^{\text{mk}}(\mathcal{A}) \leq \varepsilon + \Pr(Y \in \mathcal{T}_{\text{bad}})$ .

We say that a transcript  $\tau$  is *bad* if two different queries would result in the same input or output to  $\pi$ , were  $\mathcal{A}$  interacting with the real world. Put formally,  $\tau$  is bad if one of the following conditions is set:

$$\exists i, i', j, j' : i \neq i' : P_i^{(j)} \oplus P_{i'}^{(j')} = K_i \oplus K_{i'} \vee C_i^{(j)} \oplus C_{i'}^{(j')} = K_i \oplus K_{i'}, \tag{5}$$

$$\exists i, j, j' : P_i^{(j)} \oplus x^{(j')} = K_i \vee C_i^{(j)} \oplus x^{(j')} = K_i. \tag{6}$$

A transcript that is not a *bad* transcript, is referred to as a *good* transcript.

**Upper Bounding  $\Pr(Y \in \mathcal{T}_{\text{bad}})$ .** We want to upper bound the event that a transcript  $\tau$  in the ideal world satisfies (5)–(6). Note that  $K_i \xleftarrow{\$} \{0, 1\}^n$  for  $i = 1, \dots, \ell$  are dummy keys generated independently of  $\tau_1, \dots, \tau_\ell$ . Therefore, there are at most  $2D_i D_{i'}$  possible keys that satisfy (5) for any fixed  $i \neq i'$ . Analogously, there are at most  $2D_i T$  possible keys that satisfy (6) for any fixed  $i$ . Therefore,

$$\Pr(Y \in \mathcal{T}_{\text{bad}}) \leq \frac{\sum_i \sum_{i' < i} 2D_i D_{i'} + \sum_i 2D_i T}{2^n}, \tag{7}$$

$$\leq \frac{D^2 + 2DT}{2^n}. \tag{8}$$

**Lower Bounding Ratio  $\Pr(X = \tau) / \Pr(Y = \tau)$ .** Let us consider a good and attainable transcript  $\tau \in \mathcal{T}_{\text{good}}$ . Then denote by  $\Omega_X = 2^{n\ell} \cdot 2^{n!}$  the set of all possible oracles in the real world and by  $\text{comp}_X(\tau) \subseteq \Omega_X$  the set of oracles in  $\Omega_X$  compatible with transcript  $\tau$ . Define  $\Omega_Y = 2^{n\ell} \cdot (2^{n!})^{\ell+1}$  and  $\text{comp}_Y(\tau)$  similarly. According to the H-coefficient technique:

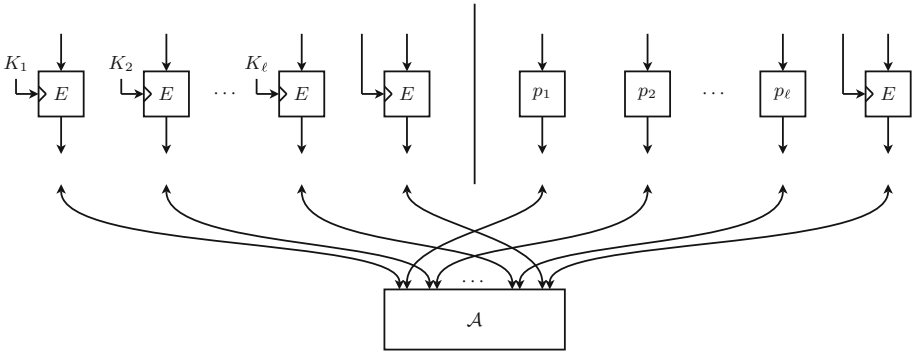
$$\Pr(X = \tau) = \frac{|\text{comp}_X(\tau)|}{|\Omega_X|}, \quad \text{and} \quad \Pr(Y = \tau) = \frac{|\text{comp}_Y(\tau)|}{|\Omega_Y|}. \tag{9}$$

First, we calculate  $|\text{comp}_X(\tau)|$ . As  $\tau \in \mathcal{T}_{\text{good}}$ , there are no two queries in  $\tau$  with the same input to or output of the underlying permutation. Any query tuple in  $\tau$  therefore fixes exactly one input-output pair of the underlying oracle. Because  $\tau$  consists of  $D+T$  query tuples, the number of possible oracles in the real world equals  $(2^n - D - T)!$ . By a similar reasoning, the number of possible oracles in the ideal world equals  $\prod_{i=1}^{\ell} (2^n - D_i)! \cdot (2^n - T)!$ . Therefore,

$$\Pr(X = \tau) = \frac{(2^n - D - T)!}{2^{n\ell} \cdot 2^n!}, \tag{10}$$

$$\Pr(Y = \tau) = \frac{\prod_{i=1}^{\ell} (2^n - D_i)! \cdot (2^n - T)!}{2^{n\ell} \cdot (2^n!)^{\ell+1}} \leq \frac{(2^n - D - T)! \cdot (2^n!)^{\ell}}{2^{n\ell} \cdot (2^n!)^{\ell+1}}. \tag{11}$$

It then follows that  $\Pr(X = \tau) / \Pr(Y = \tau) \geq 1$ . □



**Fig. 3.** An ideal block cipher  $E_K$  in the multi-key setting. Although only one direction is shown, all oracles are assumed to be bidirectional.

**Theorem 2 (Ideal Block Cipher Multi-key Security).** *Let the ideal block cipher IBC be uniformly distributed random variable over  $\text{block}(k, n)$ , then for all  $\mathcal{A}$  making at most  $D$  queries to  $E_{K_1}, \dots, E_{K_\ell}$  (resp.  $p_1, \dots, p_\ell$ ) or their inverses and at most  $T$  queries to  $E_K$  or its inverse under adversary-chosen keys,*

$$\text{Adv}_{\text{IBC}}^{\text{mk}}(\mathcal{A}) \leq \frac{\ell^2 + 2\ell T}{2^{k+1}}. \tag{12}$$

*Proof.* We consider the adversary  $\mathcal{A}$  shown in Fig. 3. Define  $\mathbf{E}$  to be the event where either

1. there exists  $i \neq j$  such that  $K_i = K_j$  or
2. there exists a query  $E(K, X)$  or  $E^{-1}(K, X)$  such that  $K = K_i$  for some  $i$ .

Given that  $\mathbf{E}$  does not happen,  $E_{K_1}, \dots, E_{K_\ell}$  are drawn independently and uniformly at random from  $\text{perm}(n)$ , and all queries made to  $E$  are independent of

$E_{K_1}, \dots, E_{K_\ell}$ . Therefore  $(E_{K_1}, \dots, E_{K_\ell}, E)$  and  $(p_1, \dots, p_\ell, E)$  are indistinguishable given the negation of  $\mathbf{E}$ , and by the fundamental lemma of game playing,

$$\mathbf{Adv}_{\text{IBC}}^{\text{mk}}(\mathcal{A}) \leq \Pr(\mathbf{E}). \quad (13)$$

The probability that there exists an adversary query  $E(K, X)$  or  $E^{-1}(K, X)$  such that  $K = K_i$  is at most  $\frac{T\ell}{2^k}$ . The probability that two keys collide is bounded above by  $\frac{\ell^2}{2^{k+1}}$ , hence

$$\Pr(\mathbf{E}) \leq \frac{\ell^2 + 2\ell T}{2^{k+1}}. \quad (14)$$

□

By our definition, there must be at least one query for every key. Therefore  $D \geq \ell$ , so that the following corollary can be derived from Theorem 2:

**Corollary 1 (Corollary of Theorem 2).**

$$\mathbf{Adv}_{\text{IBC}}^{\text{mk}}(\mathcal{A}) \leq \frac{D^2 + 2DT}{2^{k+1}}. \quad (15)$$

Observe that when the amount of plaintext per key is small, this bound is close to that of Theorem 2.

### 3.1 Tightness of the Security Bounds

Several attacks have been published that match the security bound of the Even-Mansour block cipher in the single-key setting. The first attacks were published by Daemen [22]: a known-plaintext attack for  $D = 2$  and a chosen-plaintext attack for any value of  $D$ . Biruykov and Wagner [18] presented a known-plaintext attack for  $D \geq 2^{n/2}$ . A known-plaintext attack for any value of  $D$  was given by Dunkelman et al. [26].

The single-key setting is a special case of the multi-key setting where  $\ell = 1$ . We proved in Sect. 3 (see Theorem 1) that the security bound of Even-Mansour in multi-key setting is a straightforward extension of the single-key setting. Therefore, the bound that we derived for Even-Mansour in the multi-key setting is also tight.

An attack matching our Even-Mansour security bound in the multi-key setting was recently given by Fouque et al. [31] for  $\ell = 2^{n/3}$ ,  $D_i = 2^{n/3}$  for  $i = 1, \dots, \ell$  and  $T = 2^{n/3}$ .

In the case of an ideal block cipher in the multi-key setting with  $D = \ell$ , the key collision and time-memory trade-off attacks of Biham [12, 13] show that the security bound of Corollary 1 is also tight. For a discussion of these attacks and their subsequent improvements, we refer to Sect. 2.3. Evidently, these attacks are also applicable to the Even-Mansour block cipher in the multi-key setting.



## 4 Discussion

As we proved in Sect. 3, the Even-Mansour block cipher has similar security in the multi-key setting as an ideal block cipher with the same block and key size, assuming  $D \approx \ell$ . In Sect. 2.2, we pointed out the relevance of this multi-key setting in practice.

The Even-Mansour block cipher is interesting from a design point of view because of its simplicity. As Dunkelman et al. [26] argued, it also achieves minimalism, in the sense that removing any component (one of the two key additions or the permutation) results in an insecure construction. But the Even-Mansour construction also has many implementation advantages.

From an efficiency point of view, the Even-Mansour block cipher avoids that round keys need to be precalculated and stored in memory, or that they need to be calculated on-the-fly. Avoiding precalculation of the key schedule results in a higher key agility, because of the lower cost to rekey. If round keys do not need to be calculated on-the-fly, the efficiency of every block cipher call increases. For software implementations, avoiding a key schedule reduces register pressure and decreases RAM requirements. The amount of RAM is very critical on certain microcontrollers, as shown for example in [39].

From a security point of view, the Even-Mansour block cipher avoids the need to store round keys securely. Note that in the case of AES-128, recovery of any round key leads to recovery of the encryption key. In the case of AES with 192-bit or 256-bit keys, any two consecutive round keys can be used to recover the encryption key.

Secure key storage is not only a problem for smart cards and RFID tags, but is also difficult to ensure on general purpose CPUs. The virtual memory system may move cryptographic keys into swap storage, which necessitates error-prone techniques to avoid swapping, or to use swap encryption [57].

But even in the presence of these countermeasures, cold boot attacks [35] may be used to exploit the fact that DRAM retains a large part of its memory for several seconds after removing power. Cooling techniques may be used to increase this time to several hours or even days.

If the round keys that are recovered by a cold boot attack contain errors (due to memory bit decay), it may still be possible to recover the encryption key. For AES, Halderman et al. [35] describe a simple algorithm to recover the encryption key in this case. Improved attacks were later given by Albrecht et al. [2] using integer programming, and by Tsow [58], and Kamal and Youssef [41] using a SAT solver.

The Even-Mansour block cipher avoids all attacks that recover the encryption key from multiple noisy round keys, as it avoids the calculation of round keys altogether. The leakage of encryption keys or round keys cannot be avoided without additional security measures, for example by ensuring they are only stored in the processor registers and not in RAM [50, 51]. However, this problem becomes much more manageable for the Even-Mansour construction, as only one  $n$ -bit key needs to be protected, instead of multiple round keys.

## 5 Conclusion and Future Work

Rekeying occurs frequently in real-world implementations, meaning that a plaintext may be encrypted under different keys. This setting is used, for example, in the attacks by AlFardan et al. [3] on TLS, and by Paterson et al. [54] on WPA-TKIP.

This setting is often referred to as the *broadcast setting* or the *multi-user setting*. In this paper, we introduced the *multi-key setting* to generalize the aforementioned settings. In the multi-key setting, the adversary can perform chosen-plaintext and chosen-ciphertext attacks under a set of unknown keys.

In the multi-key setting, we proved that the Even-Mansour block cipher is secure up to  $(D^2 + 2DT)/2^n$  queries. We proved a similar bound for an ideal block cipher with  $k = n$ , and showed that both bounds are tight. We used our proofs to argue in favor of the Even-Mansour construction: not only because of the simplicity of its design, but also because the lack of a key schedule makes it easier to generate fast and secure implementations.

Modes of operation for encryption and/or authentication may be designed more efficiently, if it is known that the underlying block cipher follows the Even-Mansour construction. It also seems that an Even-Mansour block cipher may still be secure if the underlying permutation is far from ideal, an idea that was pioneered by the design of the Chaskey MAC function [49]. We leave the further exploration of these research questions to future work.

**Acknowledgments.** The authors would like to thank the anonymous reviewers and Bart Mennink for their useful comments and suggestions. This work was supported in part by the Research Council KU Leuven: GOA TENSE (GOA/11/007), by Research Fund KU Leuven, OT/13/071, and by the French Agence Nationale de la Recherche through the BLOC project under Contract ANR-11-INS-011. Nicky Mouha is supported by a Postdoctoral Fellowship from the Flemish Research Foundation (FWO-Vlaanderen). Atul Luykx is supported by a Ph.D. Fellowship from the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen).

## References

1. Abdalla, M., Bellare, M.: Increasing the lifetime of a key: a comparative analysis of the security of re-keying techniques. In: Okamoto [52], pp. 546–559
2. Albrecht, M., Cid, C.: Cold boot key recovery by solving polynomial systems with noise. In: Lopez, J., Tsudik, G. (eds.) ACNS 2011. LNCS, vol. 6715, pp. 57–72. Springer, Heidelberg (2011)
3. AlFardan, N.J., Bernstein, D.J., Paterson, K.G., Poettering, B., Schuldt, J.C.: On the security of RC4 in TLS and WPA. In: USENIX Security Symposium (2013)
4. Andreeva, E., Daemen, J., Mennink, B., Assche, G.V.: Security of keyed sponge constructions using a modular proof approach. In: Demirci, H., Leander, G. (eds.) FSE 2015. LNCS, Springer (2015, to appear). <https://www.cosic.esat.kuleuven.be/publications/article-2502.pdf>

5. Babbage, S.H.: Improved “exhaustive search” attacks on stream ciphers. In: ECOS 95 (European Convention on Security and Detection), Conference publication No. 408, pp. 161–166, May 1995
6. Barker, W.C., Barker, E.: SP 800–67 Revision 1: Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher, January 2012. <http://csrc.nist.gov/publications/nistpubs/800-67-Rev1/SP-800-67-Rev1.pdf>
7. Bellare, M., Boldyreva, A., Micali, S.: Public-key encryption in a multi-user setting: security proofs and improvements. In: Preneel [56], pp. 259–274
8. Bellare, M., Desai, A., Jokipii, E., Rogaway, P.: A concrete security treatment of symmetric encryption. In: 38th Annual Symposium on Foundations of Computer Science, FOCS 1997, Miami Beach, Florida, USA, October 19–22, 1997, pp. 394–403. IEEE Computer Society (1997)
9. 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)
10. Biham, E.: New types of cryptanalytic attacks using related keys (extended abstract). In: Helleseht, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 398–409. Springer, Heidelberg (1994)
11. Biham, E.: New types of cryptanalytic attacks using related keys. *J. Cryptology* 7(4), 229–246 (1994)
12. Biham, E.: How to Forge DES-Encrypted Messages in  $2^{28}$  Steps. Technical report CS0884, Technion Computer Science Department, Israel (1996)
13. Biham, E.: How to decrypt or even substitute DES-encrypted messages in  $2^{28}$  steps. *Inf. Process. Lett.* 84(3), 117–124 (2002)
14. Biryukov, A.: Some Thoughts on Time-Memory-Data Tradeoffs. Cryptology ePrint Archive, Report 2005/207 (2005). <http://eprint.iacr.org/>
15. Biryukov, A., Khovratovich, D.: Related-key cryptanalysis of the full AES-192 and AES-256. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 1–18. Springer, Heidelberg (2009)
16. 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)
17. Biryukov, A., Shamir, A.: Cryptanalytic time/memory/data tradeoffs for stream ciphers. In: Okamoto [52], pp. 1–13
18. Biryukov, A., Wagner, D.: Advanced slide attacks. In: Preneel [56], pp. 589–606
19. Chatterjee, S., Menezes, A., Sarkar, P.: Another look at tightness. In: Miri, A., Vaudenay, S. (eds.) SAC 2011. LNCS, vol. 7118, pp. 293–319. Springer, Heidelberg (2012)
20. Chen, L.: Recommendation for Key Derivation Using Pseudorandom Functions (Revised). NIST special publication 800–108, National Institute of Standards and Technology (NIST), October 2009. <http://csrc.nist.gov/publications/nistpubs/800-108/sp800-108.pdf>
21. Chen, S., Steinberger, J.: Tight security bounds for key-alternating ciphers. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 327–350. Springer, Heidelberg (2014)
22. Daemen, J.: Limitations of the even-mansour construction. In: Imai et al. [40], pp. 495–498
23. Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. Springer, Heidelberg (2002)
24. Daemen, J., Rijmen, V.: On the related-key attacks against AES. In: Proceedings of the Romanian Academy, Series A, vol. 13(4), pp. 395–400 (2012)

25. Davies, D.W.: Some regular properties of the ‘data encryption standard’ algorithm. In: Chaum, D., Rivest, R.L., Sherman, A.T. (eds.) CRYPTO, pp. 89–96. Plenum Press, New York (1982)
26. Dunkelman, O., Keller, N., Shamir, A.: Minimalism in cryptography: the even-mansour scheme revisited. In: Pointcheval and Johansson [55], pp. 336–354
27. Dworkin, M.: Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication. NIST special publication 800–38b, National Institute of Standards and Technology (NIST), May 2005. [http://csrc.nist.gov/publications/nistpubs/800-38B/SP\\_800-38B.pdf](http://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf)
28. Dworkin, M.: Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC, November 2007. <http://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf>
29. Even, S., Mansour, Y.: A construction of a cipher from a single pseudorandom permutation. In: Imai et al. [40], pp. 210–224
30. Even, S., Mansour, Y.: A construction of a cipher from a single pseudorandom permutation. *J. Cryptology* **10**(3), 151–162 (1997)
31. Fouque, P.-A., Joux, A., Mavromati, C.: Multi-user collisions: applications to discrete logarithm, even-mansour and PRINCE. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8873, pp. 420–438. Springer, Heidelberg (2014)
32. Gauravaram, P., Knudsen, L.R., Matusiewicz, K., Mendel, F., Rechberger, C., Schl affer, M., Thomsen, S.S.: Gr ostl - a SHA-3 candidate. Submission to the NIST SHA-3 Competition (Round 3) (2011). <http://www.groestl.info/Groestl.pdf>
33. Goli c, J.D.: Cryptanalysis of alleged A5 stream cipher. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 239–255. Springer, Heidelberg (1997)
34. Group, I.N.W.: The Transport Layer Security (TLS) Protocol (2006). <http://tools.ietf.org/html/rfc4346>
35. Halderman, J.A., Schoen, S.D., Heninger, N., Clarkson, W., Paul, W., Calandrino, J.A., Feldman, A.J., Appelbaum, J., Felten, E.W.: Lest we remember: cold boot attacks on encryption keys. In: van Oorschot, P.C. (ed.) Proceedings of the 17th USENIX Security Symposium, July 28–August 1, 2008, San Jose, CA, USA, pp. 45–60. USENIX Association (2008)
36. Hellman, M.E., Merkle, R., Schroepfel, R., Diffie, W., Pohlig, S., Schweitzer, P.: Results of an Initial Attempt to Cryptanalyze the NBS Data Encryption Standard. Technical report, Stanford University, USA (1976)
37. Hellman, M.E.: A cryptanalytic time-memory trade-off. *IEEE Trans. Inf. Theory* **26**(4), 401–406 (1980)
38. Hong, J., Sarkar, P.: New applications of time memory data tradeoffs. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 353–372. Springer, Heidelberg (2005)
39. Ideguchi, K., Owada, T., Yoshida, H.: A Study on RAM Requirements of Various SHA-3 Candidates on Low-cost 8-bit CPUs. *Cryptology ePrint Archive*, Report 2009/260 (2009). <http://eprint.iacr.org/>
40. Matsumoto, T., Imai, H., Rivest, R.L. (eds.): ASIACRYPT 1991. LNCS, vol. 739. Springer, Heidelberg (1993)
41. Kamal, A.A., Youssef, A.M.: Applications of SAT solvers to AES key recovery from decayed key schedule images. In: 2010 Fourth International Conference on Emerging Security Information Systems and Technologies (SECURWARE), pp. 216–220. IEEE (2010)
42. 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)

43. Koblitz, N., Menezes, A.: Another look at HMAC. *J. Math. Cryptology* **7**(3), 225–251 (2013)
44. Lai, X., Massey, J.L.: Markov ciphers and differential cryptanalysis. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 17–38. Springer, Heidelberg (1991)
45. Mantin, I., Shamir, A.: A practical attack on broadcast RC4. In: Matsui, M. (ed.) FSE 2001. LNCS, vol. 2355, pp. 152–164. Springer, Heidelberg (2002)
46. Mehuron, W.: Data Encryption Standard (DES), October 1999. <http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf>
47. Menezes, A.: Another look at provable security. In: Pointcheval and Johansson [55], p. 8
48. Menezes, A., van Oorschot, P.C., Vanstone, S.A.: *Handbook of Applied Cryptography*. CRC Press, USA (1997)
49. Mouha, N., Mennink, B., Van Herrewege, A., Watanabe, D., Preneel, B., Verbauwhede, I.: Chaskey: an efficient MAC algorithm for 32-bit microcontrollers. In: Joux, A., Youssef, A. (eds.) SAC 2014. LNCS, vol. 8781, pp. 306–323. Springer, Heidelberg (2014)
50. Müller, T., Dewald, A., Freiling, F.C.: AESSE: a cold-boot resistant implementation of AES. In: Costa, M., Kirda, E. (eds.) Proceedings of the Third European Workshop on System Security, EUROSEC 2010, Paris, France, April 13, 2010, pp. 42–47. ACM (2010)
51. Müller, T., Freiling, F.C., Dewald, A.: TRESOR runs encryption securely outside RAM. In: Proceedings of 20th USENIX Security Symposium, San Francisco, CA, USA, August 8–12, 2011. USENIX Association (2011)
52. Okamoto, T. (ed.): ASIACRYPT 2000. LNCS, vol. 1976. Springer, Heidelberg (2000)
53. Patarin, J.: The “coefficients H” technique. In: Avanzi, R.M., Keliher, L., Sica, F. (eds.) SAC 2008. LNCS, vol. 5381, pp. 328–345. Springer, Heidelberg (2009)
54. Paterson, K.G., Poettering, B., Schuldt, J.C.N.: Plaintext recovery attacks against WPA/TKIP. In: Cid, C., Rechberger, C. (eds.) FSE 2014. LNCS, vol. 8540, pp. 325–349. Springer, Heidelberg (2015)
55. Pointcheval, D., Johansson, T. (eds.): EUROCRYPT 2012. LNCS, vol. 7237. Springer, Heidelberg (2012)
56. Preneel, B. (ed.): EUROCRYPT 2000. LNCS, vol. 1807. Springer, Heidelberg (2000)
57. Provos, N.: Encrypting virtual memory. In: Bellare, S.M., Rose, G. (eds.) 9th USENIX Security Symposium, Denver, Colorado, USA, August 14–17, 2000. USENIX Association (2000)
58. Tsow, A.: An improved recovery algorithm for decayed AES key schedule images. In: Jacobson Jr, M.J., Rijmen, V., Safavi-Naini, R. (eds.) SAC 2009. LNCS, vol. 5867, pp. 215–230. Springer, Heidelberg (2009)

# Reproducible Circularly-Secure Bit Encryption: Applications and Realizations

Mohammad Hajiabadi and Bruce M. Kapron<sup>(✉)</sup>

Department of Computer Science, University of Victoria, Victoria V8W 3P6, Canada  
{mhaji, bmkapron}@cs.uvic.ca

**Abstract.** We give generic constructions of several fundamental cryptographic primitives based on a new encryption primitive that combines *circular security* for bit encryption with the so-called *reproducibility property* (Bellare et al. PKC 2003). At the heart of our constructions is a novel technique which gives a way of de-randomizing reproducible public-key bit-encryption schemes and also a way of reducing one-wayness conditions of a constructed trapdoor-function family (TDF) to circular security of the base scheme. The main primitives that we build from our encryption primitive include *k-wise one-way* TDFs (Rosen and Segev TCC 2009), CCA2-secure encryption and deterministic encryption. Our results demonstrate a new set of applications of circularly-secure encryption beyond fully-homomorphic encryption and symbolic soundness. Finally, we show the plausibility of our assumptions by showing that the DDH-based circularly-secure scheme of Boneh et al. (Crypto 2008) and the subgroup indistinguishability based scheme of Brakerski and Goldwasser (Crypto 2010) are both reproducible.

**Keywords:** Circular security · Correlated-input security · Trapdoor functions · (non-)shielding CCA construction · Deterministic encryption

## 1 Introduction

A central problem in cryptography is delineating the assumptions required for the existence of cryptographic primitives. One way to differentiate assumptions is by whether they refer to the hardness of a *specific* computational problem (e.g., factoring), or refer *generically* to a class of problems (e.g., inverting efficiently computable functions). Assumptions of the former sort often lead to primitives which are more practical, e.g., in terms of efficiency or levels of security achieved. Those of the latter sort are useful for gaining deeper insights into the security requirements of a primitive, and also as a means of unifying specific assumptions. However, these approaches are not mutually exclusive. In particular, in cases where we have not been able to obtain constructions based on generic assumptions, we may consider strengthening an assumption with some more specific properties. This is the approach we take in this paper. By adding a syntactic property to *circularly-secure* bit encryption, we are able to obtain constructions of several powerful cryptographic primitives.

More precisely, we give constructions of various cryptographic primitives based on a general encryption primitive, which combines *circular security* with a property called *reproducibility* [5], which, the latter, gives a way of reusing randomness across independent public keys. We show the following results.

- (1) We give a novel generic construction of TDFs from reproducible bit encryption, and under this construction we show that successively stronger circular-security conditions result in successively stronger one-wayness conditions: we give a hierarchy of circular security notions, called *k-rec circular security*, all of which are weaker than those of [2, 11, 12], and show if the base scheme is *k-rec* circularly secure, the constructed TDF is *k-wise* one-way, in the sense of [28].
- (2) We show how to extract many hardcore bits for our constructed TDFs, and by applying the results of [28] we obtain a blackbox (BB) construction of CCA2-secure encryption from our assumptions. Our CCA2 construction is *non-shielding* in the sense of [18]. We partially justify this fact by showing wrt a weaker encryption primitive than ours, a non-shielding BB CCA2 construction is possible, while a shielding CCA2 construction is BB impossible.
- (3) By slightly extending our base primitive, we show how to obtain deterministic encryption schemes secure under *block-source* inputs, as defined by [9].
- (4) We realize our base encryption primitive by showing the circularly-secure schemes of [11, 12] are reproducible.

In what follows, we provide some background, give a more detailed exposition of our results and describe our constructions and proof techniques. First of all, we assume the following notation and conventions throughout the introduction. Unless otherwise stated, an encryption scheme is bit encryption with randomness space  $\{0, 1\}^\rho$  and secret-key space  $\{0, 1\}^l$ , where  $l = l(n)$  and  $\rho = \rho(n)$ ; by  $E_{pk}(m)$ , for  $m \in \{0, 1\}^*$ , we mean bitwise encryption of  $m$ .

**Trapdoor Functions.** Central to public-key cryptography is the notion of *injective trapdoor one-way function*, which refers to a family of functions, where each function in the family is easy to compute, but a randomly chosen function is hard to invert without a *trapdoor key*. A related notion is *witness-recovering CPA-secure encryption*: CPA-secure public-key encryption (PKE) where the decryption algorithm also recovers the randomness used for encryption. It is well-known that these two primitives are equivalent. However, as shown by Gertner et al. [19], there is a BB separation between CPA-secure PKE and TDFs. An interpretation of this result is that a construction of a TDF from PKE should either be non-blackbox, or should rely on specific properties of the PKE. Indeed, under specific assumptions, TDFs may be constructed “directly” (e.g., under the factoring assumption), or may be constructed by using the specifics of a particular PKE scheme (e.g., the strong homomorphisms, among other properties, of ElGamal encryption [26]).

A folklore attempt to build a TDF from PKE is to encrypt a message  $x$  under a randomness string derived deterministically from  $x$ . However, by [19], such a methodology is in general not sound. A naturally arising question is what properties of PKE enable sound realizations of this approach. The starting

point of our work is a related question, namely: when does a PKE scheme allow “secure” encryption of  $r$ , using  $r$  itself as randomness? By security we mean it be hard to recover  $r$  from  $(E_{pk_1}(r_1; r), \dots, E_{pk_\rho}(r_\rho; r))$ . Note that this immediately yields a TDF.

To address this question we first review a property of PKE schemes, called *reproducibility* [5]:  $\mathcal{E} = (Gen, E, D)$  is reproducible if there exists an efficient deterministic function  $R$ , which given a ciphertext  $c = E_{pk}(m; r)$ , a message  $m_1$ , and public/secret keys  $(pk_1, sk_1)$ , computes  $E_{pk_1}(m_1; r)$ , which we denote by  $R(c, m_1, sk_1)$ . Namely, there is an efficient way to transfer the randomness underlying a given encryption to another, provided the secret key for the second encryption is known. Although this notion may seem overly strong, natural cryptosystems (e.g., ElGamal, hash-proof-system-based cryptosystems) do satisfy this property. Indeed, under ElGamal a group element  $q$  is encrypted as  $(g^r, g^{r \cdot sk} \cdot q)$ , allowing the (encoded) randomness  $g^r$  be reused under a new secret key. Let  $\mathcal{E} = (Gen, E, D, R)$  be a reproducible PKE scheme. Define  $\mathcal{E}' = (Gen', E', D')$  as follows:  $(pk', sk') \leftarrow Gen'$ , where  $sk' = r$  and  $pk' = c = E_{pk}(0; r)$  (i.e., the secret key is a randomness string  $r$  and the public key is a dummy ciphertext formed under  $r$ );  $E'_c(b)$  samples  $(pk_1, sk_1) \leftarrow Gen$ , computes  $c' = R(c, b, sk_1)$  and returns  $(pk_1, c')$  (i.e.,  $E'_c$  encrypts  $b$  by reusing the randomness underlying  $c$ ); and  $D'_r(pk_1, c')$  returns the bit  $b$  that  $E_{pk_1}(b; r) = c'$ . Intuitively, CPA security of  $\mathcal{E}'$  follows from reproducibility and CPA security of  $\mathcal{E}$ . Moreover, the construction swaps the key and randomness spaces of  $\mathcal{E}$ , and so the task of securely encrypting randomness in  $\mathcal{E}'$  reduces to that of securely self-encrypting the secret key in  $\mathcal{E}$ ; this latter is the problem of *circular security*, a special case of the well-studied problem of *key-dependent-message* security [1–3, 8, 11–13, 23]. The discussion above suggests a general technique for de-randomizing reproducible bit-encryption schemes, sketched below, which is the basis for all our subsequent constructions.

For  $\mathcal{E} = (Gen, E, D, R)$  define  $\mathcal{F} = C(\mathcal{E}) = (G, F, F^{-1})$  as follows. The domain space of  $F$  is the set of all pairs of public/secret keys generated under  $Gen(1^n)$ .

- $G$ : To produce index/trapdoor keys  $(ik, tk)$ , let  $(pk, sk) \leftarrow Gen(1^n)$ , set  $ik = (pk, E_{pk}(0; r_1), \dots, E_{pk}(0; r_l))$ , for random  $r_i$ 's, and set  $tk = (r_1, \dots, r_l)$ .
- $F(\cdot, \cdot)$ : On key  $ik = (pk, c_1, \dots, c_l)$  and domain input  $(pk', sk')$ , return  $(pk', c'_1, \dots, c'_l)$ , where  $c'_i = R(c_i, sk'_i, sk')$ . (Here,  $sk'_i$  denotes the  $i$ th bit of  $sk'$ .)
- $F^{-1}(\cdot, \cdot)$ : given trapdoor key  $tk = (r_1, \dots, r_l)$  and image point  $(pk', c'_1, \dots, c'_l)$ , return  $(pk', b_1 \dots b_l)$ , where  $b_i$  is the bit which satisfies  $c'_i = E_{pk'}(b_i; r_i)$ .

Correctness of  $\mathcal{F}$  follows by the reproduction property of  $R$ . Also, since  $R$  is deterministic, so is the evaluation algorithm  $F$ . Finally, we take advantage of the fact that  $\mathcal{E}$  is bit encryption to ensure efficient inversion for  $\mathcal{F}$ .

To discuss one-wayness we need the following definitions. For  $(pk, sk)$  output by  $Gen$  we refer to  $E_{pk}(sk)$  as an *sk-self-encryption*. We call  $\mathcal{E}$  *k-rec circularly secure* if no adversary can recover (with a nonnegligible chance) a random



$sk$  from  $k$  independent  $sk$ -self-encryptions, and call  $\mathcal{E}$  *k-ind circularly secure* if no adversary can distinguish between  $k$  independent  $sk$ -self-encryptions and encryptions of, say, zero. The notion of circular security in the literature is that of  $k$ -ind circular security, for unbounded  $k$ . For the construction above we show the following *tight* reduction.

**Theorem 1.** *If  $\mathcal{E}$  is reproducible and 1-rec circularly secure then  $C(\mathcal{E})$  is one-way.*

The reduction above is “security preserving” in the following sense: assuming  $\mathcal{E}$  is reproducible, then  $\mathcal{E}$  is 1-rec circularly secure iff  $C(\mathcal{E})$  is one-way. Indeed, as we show next, by strengthening the condition of 1-rec circular security we achieve stronger forms of one-wayness.

A family of TDFs is called *k-wise one-way* [28] if one-wayness holds even if the given input is evaluated under  $k$  independently chosen functions.<sup>1</sup> More formally,  $\mathcal{F} = (G, F, F^{-1})$  is called *k-wise one-way*, if  $\mathcal{F}$ 's *k-wise product*, defined as  $F_{ik_1, \dots, ik_k}(x) = (F_{ik_1}(x), \dots, F_{ik_k}(x))$  is one-way. Rosen and Segev [28] showed the utility of this notion by giving a blackbox construction of CCA2-secure encryption based on  $k$ -wise one-way TDFs, for a sufficiently large  $k$ , simplifying a prior construction [26] based on lossy TDFs (LTDFs). Despite their utility,  $k$ -wise one-way TDFs (even for  $k = 2$ ) are very strong primitives, whose only generic constructions so far have been based on LTDFs. Indeed, as shown by Vahlis [30], even 2-wise one-way TDFs cannot be constructed in a blackbox way from trapdoor permutations (TDPs).

Our TDF construction provides an easy means for obtaining  $k$ -wise one-way TDFs: we can generalize Theorem 1 to show the following

*If  $\mathcal{E}$  is reproducible and k-rec circularly secure then  $C(\mathcal{E})$  is k-wise one-way.*

To put our construction of  $k$ -wise one-way TDFs in context, we compare it to the LTDF-based construction [28]: the security reduction of [28] involves both statistical and computational arguments, allowing one to obtain only  $k$ -wise one-way TDFs for a priori fixed but arbitrarily large values of  $k$  (which does suffice for CCA2 encryption) from sufficiently lossy TDFs. Our reduction argument, on the other hand, is entirely computational, allowing us to obtain unbounded  $k$ -wise one-way TDFs (i.e., a TDF that is  $k$ -wise one-way for any value of  $k$ ) from the full circular security assumption.

As for the base assumptions, the relationships among the circular-security notions we described is not well-understood (beyond the trivial ones). Under certain assumptions these notions become equivalent. For example, any *re-randomizable* 1-rec circularly-secure scheme is poly-ind circularly secure: this follows by considering that a 1-rec circularly-secure scheme is already poly-rec circularly secure (because of re-randomizability), and that any poly-rec circularly-secure scheme is also poly-ind circularly secure [29, Theorem 8]. For the rest of the introduction, however, for simplicity, we describe the results wrt full circular security.

<sup>1</sup> Actually, [28] chose another name for this particular notion, but we refer to it as  $k$ -wise one-wayness for simplicity.

We extend Construction  $C$  for the case in which the base scheme is  $t$ -circularly secure (i.e., circularly-secure wrt  $t$  keys): the input of each TDF is  $t$  pairs of public/secret keys, the index key contains  $l \cdot t$  dummy ciphertexts, and the evaluation algorithm on  $(pk_0, sk_0, \dots, pk_{t-1}, sk_{t-1})$  returns  $(pk_0, \dots, pk_{t-1})$  along with  $t \cdot l$  ciphertexts formed by encrypting each bit of  $sk_i$  under  $pk_{(i+1) \bmod t}$  (deterministically) by reusing the randomness of the corresponding ciphertext of the index key.

**Extracting Hardcore Bits.** Given the TDFs built above, we may apply the general Goldreich-Levin (GL) theorem [20] to extract a hardcore bit. We would like to, however, avoid the use of the GL theorem for several reasons. First, the GL reduction, due to its generality, is not tight, while we would like to achieve CCA security with tight reductions. Second, for our deterministic encryption results we need to be able to extract many hardcore bits. Finally, since our base assumptions are strictly BB-stronger (by Vahlis’s result) than one-way TDFs, we should look for more specialized methods. We sketch below two deterministic methods for extracting many hardcore bits with tight security reductions for our constructed TDFs. The first method applies to  $t$ -circular security and allows us to extract  $\log((t-1)!)$  bits, with the advantage that it only increases the domain size. The second method allows us to extract any, a priori fixed, number of bits, but it enlarges other spaces as well.

**First Method: A Cycle Hides its Ordering.** For simplicity, we describe the idea for 3-circular security, showing how to extract a single hardcore bit. The idea is 3-circularly security implies no adversary can distinguish between the sequence  $(E_{pk_1}(sk_2), E_{pk_2}(sk_3), E_{pk_3}(sk_1))$  and  $(E_{pk_1}(sk_3), E_{pk_2}(sk_1), E_{pk_3}(sk_2))$ . Now we augment our TDF construction described above (for  $t$ -circular security) so that the evaluation algorithm, besides  $(pk_1, sk_1), (pk_2, sk_2), (pk_3, sk_3)$ , also receives an additional bit  $b$ , used to dictate the ordering used to form the cycle. The inversion algorithm can open the ciphertexts, as before, and recover the bit  $b$ , by checking, say, whether the key encrypted under  $pk_1$  is a secret key for  $pk_2$  or for  $pk_3$ .<sup>2</sup> This technique extends to the  $t$ -circular security case for any  $t > 3$ , allowing us to “hide” a random ordering, providing  $\log((t-1)!)$  hardcore bits.

**Second Method.** We describe the idea for 1-circular security. We extend construction  $C$  above to be parameterized over an integer  $m = m(n)$  and to result in a TDF whose input now consists of triples  $(pk, sk, x)$ , where  $x \in \{0, 1\}^m$ . Moreover, we augment the index key to contain  $m$  added ciphertexts and let the trapdoor key contain their underlying randomness strings. Now  $F(ik, (pk, sk, x))$  proceeds as before, but it also “encrypts”  $x$  in the process by again reusing randomness. For this TDF, we show that  $x$  remain pseudorandom even knowing  $F(ik, (pk, sk, x))$ . Finally, assuming the property that public keys under the base scheme are computed deterministically from their secret keys (plus perhaps some public parameters), we show how to obtain TDFs that hide a  $(1 - o(1))$  fraction of their input bits.

<sup>2</sup> This, however, imposes a negligible inversion error.

**CCA-secure Encryption.** Using results on  $k$ -wise one-way TDFs with many hardcore bits,<sup>3</sup> we may now use the BB construction of Rosen and Segev [28] to build a many-bit CCA2-secure PKE from a reproducible, circularly secure bit-encryption scheme. Specifically, [28] gives a BB construction of CCA2-secure encryption from  $k$ -wise one-way TDFs, for  $k \in \Omega(n)$ ; they also show that  $k \in \omega(\log n)$  suffices for CCA1 encryption. Our CCA constructions, by relying on that of [28], result in schemes whose decryption functions query the encryption function of the base scheme. Gertner et al. [18] refer to such constructions as *non-shielding*, and show that there exist no *shielding* BB construction of CCA-secure from CPA-secure encryption. Since our base assumptions are BB-stronger than CPA security, it is natural to ask whether the non-shielding nature of our CCA2 construction is just an artifact of the construction of [28] or whether it is inherent. We were not able to answer this question for our encryption primitive, mainly because of the presence of the reproduction function. However, we are able to answer this wrt a weaker primitive than ours, which is a special case of *randomness-dependent-message-secure (RDMS)* encryption [7], which allows multiple bitwise-encryptions of a randomness string  $r$  under  $r$  itself as randomness (Formalized in Definition 2). Calling this new primitive RDMS encryption, we show that RDMS encryption is implied by our base assumptions, and also that it enables a non-shielding construction of CCA-secure encryption. We prove this by directly instantiating  $k$ -wise one-way TDFs under RDMS encryption. Next we observe that the shielding-BB impossibility result of [18] extends if the base scheme is an RDMS encryption primitive (Theorem 5). Indeed, it seems that this latter statement is true for most encryption primitives whose security requirements are defined wrt passive indistinguishability (i.e., no decryption oracles); see Sect. 4 for more details. Thus, we obtain an encryption primitive, wrt which a non-shielding BB CCA-secure construction is possible, but under which a shielding CCA-secure construction is BB impossible.

**Deterministic Encryption (DE).** Following [9], a deterministic  $l$ -bit-encryption scheme is called  $(\lambda, l)$ -IND secure if encryptions of any two (efficient)  $\lambda$ -sources (i.e., distributions with min-entropy  $\lambda$ ) result in computationally indistinguishable ciphertexts. We formulate two extended notions of circular security, called  $(\lambda, l)$ -entropy circular security and *strong*- $(\lambda, l)$ -entropy circular security, both of which require circular security hold even if the secret key  $sk \in \{0, 1\}^l$  is sampled from a  $\lambda$ -source distribution, while the strong-entropy version requires one more assumption, related to the public-key distribution.<sup>4</sup>

We show our TDF construction immediately gives us a  $(\lambda, l)$ -IND-secure DE scheme if the base scheme satisfies strong  $(\lambda, l)$ -entropy circular security. We also show, by appropriately choosing the parameters, the schemes of [11, 12] provide strong-entropy circular security, meaning our generic transformation applies to these two schemes to obtain secure DE schemes, which explains the striking similarities between (especially) the DDH-based DE scheme of [9] and the scheme

<sup>3</sup> We note that our hardcore-security results hold not only for  $\mathcal{F} = C(\mathcal{E})$ , but also for  $\mathcal{F}$ 's  $k$ -wise products (under the respective assumptions). See Sect. 3.

<sup>4</sup> The notion of weak-entropy circular security was also considered by [13] in the context of KDM amplification.

of Boneh et al. [11]. We also note that the extra condition of strong-entropy circular security may be satisfied if, informally, the key-generation algorithm acts as a *strong extractor*, producing the public key from the secret key, taken as the source, based on a public parameter, taken as the seed. Similar structural assumptions are made in other settings, e.g., [32], to obtain DE schemes.

For weak-entropy circular security we also show how to obtain a secure DE scheme but with looser parameters, i.e., the  $(\lambda, l)$ -parameters of the base scheme are not maintained. We follow the so-called *encrypt-with-hardcore* technique, implicitly used in [4, 6, 9], and formalized in [17]. A high-level description of the idea is as follows. Assume  $\mathcal{F} = (G, F, F^{-1})$  is a TDF with an associated hardcore function  $h$  producing  $\Omega(n)$  hardcore bits, and we want to make  $\mathcal{F}$  a secure DE scheme. Suppose we have the bonus that  $h$  preserves hardcore security even if  $x$  is sampled from a biased, high-min entropy distribution. Now we can build a DE scheme by encrypting the output of  $F$  using the hardcore bitstring under a randomized-encryption scheme  $\mathcal{E}'$ : namely,  $E_{ik, pk}(x) = E'_{pk}(F(ik, x), h(x))$ ; decryption can be done using  $ik$ 's trapdoor key and  $pk$ 's secret key. Security of  $E$  comes from the fact that  $(F(ik, x), h(x))$  is computationally indistinguishable from  $(F(ik, x), r)$ , so  $h(x)$  is as good as a fresh randomness string. The only remaining issue is that  $E$  may require a longer randomness string, which, however, can be handled by applying a pseudorandom generator to  $h(x)$ .

**Further Discussion.** Since LTDFs [26] are the only generic assumption (to the best of our knowledge) that imply  $k$ -wise one-way TDFs, it is natural to ask about the relationship between LTDFs and our base primitive. We believe these notions are incomparable. First, under our encryption primitive, we are able to obtain a TDF that is  $k$ -wise one-way for unbounded  $k$ 's; LTDFs are known to achieve bounded  $k$ -wise one-way TDFs, but this does not seem to generalize to the unbounded case, mainly due to the nature of LTDF-based proof techniques that also rely on statistical arguments. On the other hand, LTDFs have powerful statistical properties (i.e., losing information in lossy mode) which do not seem to be realizable under our assumptions. Choi and Wee [14], by abstracting the DDH-based TDF construction of [26], show how to obtain LTDFs from reproducible encryption that is homomorphic wrt both messages and randomness. For similar reasons our assumptions seem incomparable to those of [26].

We note that almost all BB CCA2-constructions, based on encryption or TDFs, are non-shielding [24, 26, 28], except for a few cases which rely on very powerful primitives, e.g., [10]. Intuitively, the non-shielding property of those constructions is used to do consistency checks on ciphertexts. It would be interesting to explore if there exist weaker encryption primitives (than those we consider) for which the BB separation of [18] is the best possible.

Our results show an alternative way (to those presented in [16, 26]) of constructing DDH-based TDFs. Right now, by instantiating our TDF construction under the DDH-based circularly-secure scheme [11], we obtain no improvement in efficiency over existing constructions. This motivates the search for more efficient DDH-based circularly-secure schemes.

Finally, we discuss adaptations of Construction  $C(\mathcal{E})$  to the case in which the secret-key space of  $\mathcal{E}$  is a subset of the plaintext space  $\mathcal{M}$  (which allows the secret

key to be encrypted as a whole) and reproducibility holds wrt  $\mathcal{M}$ . For this case we may substantially improve efficiency by having each index key contain only one ciphertext, whose randomness will be reused to self-encrypt the secret key (as a whole) given as input to the evaluation algorithm. To perform inversion, however, we would need to rely on one more assumption: it is efficiently possible to recover  $m$  from  $E_{pk}(m; r)$  and  $r$ , for all  $pk, m$  and  $r$ . This last property by itself is satisfied by natural cryptosystems, e.g., ElGamal. Moreover, there is a standard way to make any CPA-secure scheme (for which  $\{0, 1\}^l \subseteq \mathcal{M}$ ) “one-shot” circularly secure; this transformation, however, does not (necessarily) maintain this last property. Thus, our results suggest the CPA-to-one-shot-circular transformation may be non-trivial (and interesting) if it is to maintain the last property.

## 2 Basic Notation and Definitions

**Remark.** Since we gave outlines of the proofs of most theorems in the introduction we defer the full proofs to the full version of the paper.

**Notation.** For a finite set  $S$  we use  $x \leftarrow S$  to denote sampling  $x$  uniformly at random from  $S$  and denote by  $U_S$  the uniform distribution on  $S$ . If  $D$  is a distribution then  $x \leftarrow D$  denotes choosing  $x$  according to  $D$ . We use the word PPT in this paper in the standard sense. We use  $A(\dots; r)$  to denote the deterministic output of PPT function  $A$  when the randomness is fixed to  $r$ , and use  $x \leftarrow A(a_1, a_2, \dots)$  to denote the distribution formed by outputting  $A(a_1, a_2, \dots; r)$  for a uniformly-random  $r$ . If  $A(x_1, \dots, x_m; r)$  outputs a tuple of strings, we let  $A_i(x_1, \dots, x_m)$  be the distribution formed by outputting the  $i$ th component of  $A(x_1, \dots, x_m)$ . We denote the support set of a distribution  $D$  by  $Sup(D)$ , and write  $x \in D$  to indicate  $x \in Sup(D)$ . We call  $f : \mathbb{N} \rightarrow \mathbb{R}$  negligible if  $f(n) < 1/P(n)$ , for any polynomial  $P$  and sufficiently large  $n$ . We write  $negl$  to denote unspecified negligible functions. We denote by  $f^{-1}$  the inverse of an injective function  $f$ . For two ensembles  $X = \{X_i\}_{i \in \mathbb{N}}$  and  $\{Y_i\}_{i \in \mathbb{N}}$  of random variables we say  $X$  is computationally indistinguishable from  $Y$ , denoted  $X \equiv^c Y$ , if for any bit-valued, PPT function  $D$ , we have  $|\Pr[D(X_n) = 1] - \Pr[D(Y_n) = 1]| = negl(n)$ . We write  $X \equiv Y$  to mean  $X$  and  $Y$  are identically distributed. All functions, adversaries, distributions, etc., that appear in this paper, if not otherwise stated, are assumed to be efficiently computable/samplable. For  $x, y \in \{0, 1\}^*$  we use  $|x|$  to denote the bit length of  $x$ , use  $x_i$ , for  $1 \leq i \leq |x|$ , to denote the  $i$ th bit of  $x$ , and use  $x||y$  to denote the concatenation of  $x$  and  $y$ .

**Trapdoor Functions.** We first start by giving the standard definitions related to trapdoor functions and hardcore bits.

A collection,  $\mathcal{F} = (G, F)$ , of functions is defined as follows. The algorithm  $G(1^n)$  returns a function index  $s$ , and the deterministic algorithm  $F(s, \cdot)$  computes a function  $f_s : D_n \rightarrow R_n$ . We stress both the domain and range of  $f_s$  only depend on the security parameters,  $1^n$ . We call  $\{D_n\}$  the domain space of  $\mathcal{F}$ .

Assuming that  $\mathcal{D} = \{\mathcal{D}_n\}$  is a distribution over  $\{D_n\}$  and  $h : D_n \rightarrow \{0, 1\}^{p(n)}$  is a deterministic function, we define the following notions. We say  $\mathcal{F}$  is  $\mathcal{D}$ -one-way if for any adversary  $\mathcal{A}$ ,  $\Pr[f_s(\mathcal{A}(s, f_s(x))) = f_s(x)] = \text{negl}(n)$ , where the probability is computed over  $s \leftarrow G(1^n)$ ,  $x \leftarrow \mathcal{D}_n$  and  $\mathcal{A}$ 's coins.

We say that  $h$  is a  $\mathcal{D}$ -hardcore function for  $\mathcal{F}$  if for any adversary  $\mathcal{A}$ ,

$$|\Pr[\mathcal{A}(s, f_s(x), h(x)) = 1] - \Pr[\mathcal{A}(s, f_s(x), U_{\{0,1\}^{p(n)}}) = 1]| = \text{negl}(n),$$

where  $s \leftarrow G(1^n)$  and  $x \leftarrow \mathcal{D}_n$ . We may omit  $\mathcal{D}$ , from  $\mathcal{D}$ -hardcore, etc., when it is clear from context. Next, we define TDFs and their  $k$ -wise products [28].

A collection of injective trapdoor functions (TDFs)<sup>5</sup> is given by three algorithms  $\mathcal{F} = (G, F, F^{-1})$ , where  $G(1^n)$  randomly produces a pair  $(ik, tk)$  of index/trapdoor keys, the deterministic algorithm  $F(ik, \cdot)$  computes an injective function  $f_{ik} : D_n \rightarrow R_n$ , and  $F^{-1}(tk, \cdot)$  computes  $f_{ik}^{-1}(\cdot)$ . We stress that the input domain of  $f_{ik}$  only depends on the security parameter  $1^n$ . We may sometimes relax the definition by allowing a negligible inversion error. The  $k$ -wise product of  $\mathcal{F}$ , denoted  $\mathcal{F}^{(k)} = (G^{(k)}, F^{(k)})$ , is defined as follows. The algorithm  $G^{(k)}(1^n)$  runs  $G(1^n)$  independently  $k$  times to output  $k$  index keys,  $(ik_1, \dots, ik_k)$ ; on input  $x$ ,  $F^{(k)}((ik_1, \dots, ik_k), \cdot)$  returns  $(F(ik_1, x), \dots, F(ik_k, x))$ .

Assume  $\mathcal{F}$  is a TDF with domain  $D = \{D_n\}$  and  $\mathcal{D} = \{\mathcal{D}_n\}$  is a distribution on  $D$ . We say  $\mathcal{F}$  is  $k$ -wise  $\mathcal{D}$ -one-way if  $\mathcal{F}^{(k)}$  is  $\mathcal{D}$ -one-way.

**Bit Encryption Schemes.** All encryption schemes that appear throughout, if not explicitly stated, are *bit-encryption* schemes. In our applications we need to work with a more general notion of encryption schemes involving *public parameters*. A bit-encryption scheme  $\mathcal{E} = (Param, Gen, E, Dec)$  is defined as follows. *Param* on input  $1^n$  outputs a random parameter, *par*. The *key-generation algorithm*, *Gen* on inputs  $1^n$  and *par* generates a public/secret key  $(pk, sk) \leftarrow Gen(1^n, par)$ ; we assume *pk* includes *par*, so we do not include *par* as input to other algorithms. The *encryption algorithm*, *E*, on inputs  $1^n$ , *pk*, bit  $b$  and randomness  $r \in \mathcal{R}_n$ , outputs ciphertext  $c = E_{pk}(b; r)$ . The *decryption algorithm*, *Dec*, takes a secret key *sk* and ciphertext  $c$ , and deterministically outputs a bit  $b = Dec_{sk}(c)$ . For correctness, we require  $\Pr[Dec_{sk}(E_{pk}(b)) = b] = 1$ , for  $par \leftarrow Param(1^n)$ ,  $(pk, sk) \leftarrow Gen(1^n, par)$  and  $b \leftarrow \{0, 1\}$ . We assume the following: for any fixed *par*, all secret keys output by  $Gen(1^n)$  are bitstrings of the same length, and, whenever we are generating many public keys, all keys are generated wrt a single initial *par*. Thus, we make *Param* implicit henceforth.

We say  $\mathcal{E} = (Gen, E, Dec)$  is *CPA secure* if  $(pk, E_{pk}(0)) \equiv^c (pk, E_{pk}(1))$ , where *pk* is chosen according to  $Gen(1^n)$ . For  $m \in \{0, 1\}^*$ , we extend *E* to define  $E_{pk}(m) = (E_{pk}(m_1), \dots, E_{pk}(m_{|m|}))$ . If  $\mathbf{r} = (r_1, \dots, r_t)$  and  $m \in \{0, 1\}^t$  we write  $E_{pk}(m; \mathbf{r}) = (E_{pk}(m_1; r_1), \dots, E_{pk}(m_t; r_t))$ .

We now give definitions for circular security. We say  $\mathcal{E} = (Gen, E, Dec)$  is *k-rec t-circularly secure* if  $\Pr[\mathcal{A}(pk_1, \dots, pk_t, \mathbf{c}_1, \dots, \mathbf{c}_k) = sk_1] = \text{negl}(n)$  for every adversary  $\mathcal{A}$ , where  $(pk_1, sk_1), \dots, (pk_t, sk_t) \leftarrow Gen(1^n)$  and for every  $1 \leq i \leq k$

$$\mathbf{c}_i \leftarrow (E_{pk_2}(sk_1), E_{pk_3}(sk_2), \dots, E_{pk_1}(sk_t));$$

<sup>5</sup> We use TDF to refer to a collection of injective trapdoor functions henceforth.

We say  $\mathcal{E}$  is  $k$ -ind  $t$ -circularly secure if  $\mathcal{E}$  is CPA secure and also it holds that  $(\mathbf{c}_1, \dots, \mathbf{c}_k) \equiv^c (\mathbf{c}'_1, \dots, \mathbf{c}'_k)$ , where

$$\mathbf{c}'_i \leftarrow (E_{pk_2}(0^l), E_{pk_3}(0^l), \dots, E_{pk_1}(0^l)),$$

for every  $1 \leq i \leq k$ , and  $l = |sk_1|$ . Note that we add CPA security as a separate condition because otherwise the definition may be satisfied trivially, e.g., consider the encryption scheme under which the secret key is always the all-zero string and the encryption function is the identity function.

Henceforth, when we say  $k$ -rec circular security (or  $k$ -ind circular security) we are referring to the definition wrt a single pair of public/secret keys.

**Definition 1.** We call  $\mathcal{E} = (Gen, E, Dec)$  reproducible if there exists a deterministic function  $R$ , called the reproduction function, s.t. for any  $(pk_1, sk_1), (pk_2, sk_2) \in Gen(1^n)$ ,  $r \in \mathcal{R}_n$  and  $b_1, b_2 \in \{0, 1\}$ ,

$$R(pk_1, E_{pk_1}(b_1; r), b_2, pk_2, sk_2) = E_{pk_2}(b_2; r).$$

For simplicity we omit the inclusion of  $pk_1$  and  $pk_2$  as inputs to  $R$ .

### 3 Constructing TDFs and Hardcore Bits

**TDFs From Reproducible Encryption.** We begin by giving a construction that takes as input a reproducible bit-encryption scheme and produces a TDF. We then show how to achieve increasingly stronger guarantees of one-wayness for the constructed TDF from corresponding assumptions on the base encryption primitive. We present the construction adapted to the  $t$ -circular security case (i.e., circular security wrt  $t$  keys), meaning that we will obtain guarantees of one-wayness for the constructed TDF from  $t$ -circular security assumptions.

We use  $D^t$  to denote the  $t$ 'th Cartesian power of a set  $D$ . If  $\mathcal{D}$  is a distribution,  $D^t$  denotes the  $t$ -tuple formed by sampling  $t$  times independently from  $\mathcal{D}$ .

**Construction 1.** Construction  $C_1$  takes as input a reproducible bit-encryption scheme  $\mathcal{E} = (Gen, E, Dec, R)$  and  $t = t(n)$  and it outputs a TDF,  $\mathcal{F} = (G, F, F^{-1})$ , with domain space  $D^t$ , where  $D = Sup(Gen(1^n))$ . Let  $l = l(n)$  be the length of a secret keys output by  $Gen(1^n)$ .

- $G(1^n)$ : Let  $(pk, sk) \leftarrow Gen(1^n)$ , and form  $tk = (r_{1,1}, \dots, r_{1,l}, \dots, r_{t,1}, \dots, r_{t,l})$ , for independent  $r_{i,j}$ 's, and  $ik = (pk, c_{1,1}, \dots, c_{1,l}, \dots, c_{t,1}, \dots, c_{t,l})$ , where for  $1 \leq i \leq t$  and  $1 \leq j \leq l$ ,  $c_{i,j} = E_{pk}(0; r_{i,j})$ . Return  $(ik, tk)$
- $F((pk, c_{1,1}, \dots, c_{t,l}), (pk_1, sk_1, \dots, pk_t, sk_t))$  returns  $(pk_1, \dots, pk_t, c'_{1,1}, \dots, c'_{t,l})$ , where for  $1 \leq i \leq t - 1$  and  $1 \leq j \leq l$  we set  $c'_{i,j} = R(c_{i,j}, b_{i,j}, sk_{i+1})$ , and  $c'_{t,j} = R(c_{t,j}, b_{t,j}, sk_1)$ , with  $b_{i,j}$  being the  $j$ th bit of  $sk_i$ .
- $F^{-1}((r_{1,1}, \dots, r_{t,l}), (pk_1, \dots, pk_t, c'_{1,1}, \dots, c'_{t,l}))$ : Retrieve each  $sk_i$ , for  $1 \leq i \leq t$ , bit-by-bit by encrypting back both 0 and 1 with the provided randomness (and under the appropriate public key) and finding the matching bit.

The TDF’s completeness follows by reproducibility. We point out a few remarks. First, the efficiency of the search performed by the inversion algorithm relies on the fact that each ciphertext is hiding a single bit, encrypted under the randomness known to the inverter. Second, our construction is entirely blackbox, also accessing (during evaluation) the reproduction function. Third, our construction extends to the non-bit-encryption case, by still continuing to encrypt the secret key bit-by-bit, but by fixing a mapping from bits to two fixed plaintext messages; for this case, the one-wayness of the constructed TDF reduces to bit-wise circular security of the base scheme (wrt the fixed mapping).

**Theorem 1.** *Assume  $\mathcal{E}$  is a reproducible bit-encryption scheme and  $\mathcal{F}$  is the TDF built from  $\mathcal{E}$  in Construction 1 based on integer  $t$ . Then,  $\mathcal{E}$  is  $k$ -rec  $t$ -circularly secure if and only if  $\mathcal{F}$  is  $k$ -wise  $\mathcal{D}$ -one-way, where  $\mathcal{D} = (\text{Gen}(1^n))^t$ . Moreover, the reductions are tight.*

**Extracting Many Hardcore Bits.** We present two deterministic methods for extracting many hardcore bits from the TDF presented in Construction 1, with tight and efficient reductions to the indistinguishability variants of circular security assumptions. The first method applies to  $t$ -circular security for  $t \geq 2$ , allowing us to directly extract  $\log((t - 1)!)$  bits, by expanding only the domain space by the same number of bits (but without affecting the sizes of the other system’s parameters). The second method is less restrictive, allowing us to extract (from  $t$ -circular security, for any  $t \geq 1$ .)  $m(n)$  hardcore bits, where  $m$  is an arbitrary but a priori fixed poly function, by increasing the domain space by  $m(n)$  bits and the image, index-key and trapdoor-key spaces by poly factors of  $m(n)$ . In particular, by choosing the parameter  $m$  appropriately we obtain TDFs which hide a  $1 - o(1)$  fraction of their input bits.

**First Hardcore Extraction Method.** We begin with some notation. Define  $[t] = \{1, \dots, t\}$ . Let

$$S = \{f: [t] \rightarrow [t] \mid f \text{ is injective} \ \& \ \forall X, \text{ s.t. } \emptyset \subsetneq X \subsetneq [t], \{f(y) \mid y \in X\} \neq X\},$$

for which we have  $|S| = (t - 1)!$ . Intuitively, each  $f \in S$  defines a possible circular ordering of encrypting a sequence of  $t$  pairs of keys, by having  $pk_i$  encrypt  $sk_{f(i)}$ . The condition  $\forall X \subsetneq [t], \{f(y) \mid y \in X\} \neq X$  guarantees that we have a single, full cycle. For example, it is not the case that  $pk_1$  encrypts  $sk_2$ ,  $pk_2$  encrypts  $sk_1$  and the remaining keys encrypt each other in a circular manner. Fix  $\mathcal{O} : \mathbb{Z}_{(t-1)!} \rightarrow S$  to be an efficient index function defined using a canonical ordering of the elements of  $S$ . We will also write  $\mathcal{O}(i, x)$  to denote  $f_i(x)$ , where  $f_i$  is the  $i$ th function according to the ordering. We also require that, for any  $f \in S$ , given  $sq = \{(x, f(x)) \mid x \in [t]\}$ , it is possible to efficiently compute the index of  $f$  according to the ordering<sup>6</sup>, which we (by slightly abusing the notation) denote by  $\mathcal{O}^{-1}(sq)$ . We now proceed to describe the modified TDF construction and the associated hardcore function.

<sup>6</sup> Such an ordering for which we have such a function  $\mathcal{O}$  can be defined by fixing an efficient way of enumeration.



**Construction 2.** Let  $\mathcal{E} = (\text{Gen}, E, \text{Dec}, R)$ ,  $t$  and  $D^t$  be as in Construction 1. The domain space of the TDF,  $\mathcal{F} = (G, F, F^{-1})$ , we build is now  $(D^t, \mathbb{Z}_{(t-1)!})$ .

- $G(1^n)$ : As in Construction 1.
- $F((pk, c_{1,1}, \dots, c_{t,l}), (pk_1, sk_1, \dots, pk_t, sk_t, u))$  is computed as follows. Define  $(ind_1, \dots, ind_t) = (\mathcal{O}(u, 1), \dots, \mathcal{O}(u, t))$ . Informally, the output will be  $pk_1, \dots, pk_t$  together with a cycle of encrypted keys, where  $pk_i$  encrypts the bits of  $sk_{ind_i}$ . Return  $(pk_1, \dots, pk_t, c'_{1,1}, \dots, c'_{t,l})$ , where, for  $1 \leq i \leq t$  and  $1 \leq j \leq l$ ,  $c'_{i,j} = R(c_{i,j}, b_{i,j}, sk_i)$ , with  $b_{i,j}$  being the  $j$ th bit of  $sk_{ind_i}$ .
- $F^{-1}((r_{1,1}, \dots, r_{t,l}), (pk_1, \dots, pk_t, c'_{1,1}, \dots, c'_{t,l}))$ : do the following steps:
  - for each  $1 \leq i \leq t$ , recover the bitstring,  $x_i$ , encrypted under  $pk_i$  bit-by-bit as follows: to retrieve the  $j$ th bit of  $x_i$ , encrypt both 0 and 1 under  $pk_i$  using randomness  $r_{i,j}$  and check the result against  $c'_{i,j}$ ;
  - for each  $1 \leq i \leq t$ , let  $ind_i$ , where  $1 \leq ind_i \leq t$ , be the index for which it holds that  $pk_{ind_i}$  is the matching public key of  $x_i$ ,<sup>7</sup> and let  $sk_{ind_i} = x_i$ . Form  $sq = \{(1, ind_1), \dots, (t, ind_t)\}$ ; return  $(pk_1, sk_1, \dots, pk_t, sk_t, \mathcal{O}^{-1}(sq))$ .

**Hardcore Function:** For  $\mathcal{F}$  given above we define  $h: (D^t, \mathbb{Z}_{(t-1)!}) \rightarrow \mathbb{Z}_{(t-1)!}$  as  $h(pk_1, sk_1, \dots, pk_t, sk_t, u) = u$ .

Correctness of the new TDF follows immediately. Note that Construction 1 is a special case of Construction 2, by forming the encrypted cycle wrt the fixed function  $f: f(1) = t; f(2) = 1; \dots, f(t) = t - 1$ . In contrast, Construction 2 forms the encrypted cycle according to a random  $f$  (provided as input to the TDF), where, as we show below, the random choice of  $f$  is what is computationally hidden by the output. We now have

**Theorem 2.** Assuming  $\mathcal{E} = (\text{Gen}, E, \text{Dec}, \text{Rep})$  is  $k$ -ind  $t$ -circularly-secure, it holds that  $\mathcal{F}$  is  $k$ -wise one-way and  $h$  is a hardcore function for  $\mathcal{F}^k$ .

**Second Hardcore Extraction Method.** The second construction allows us to extract any (a priori fixed) number of pseudorandom bits, where these bits are the last input block of the TDF.

**Construction 3.** Let  $\mathcal{E} = (\text{Gen}, E, \text{Dec}, R)$ ,  $t$  and  $D^t$  be as in Construction 1, and let  $m = m(n)$  be an integer. The domain space of the TDF we build is  $(D^t, \{0, 1\}^m)$ . We define  $\mathcal{F} = (G, F, F^{-1})$  as follows.

- $G(1^n)$ : Let  $(pk, sk) \leftarrow \text{Gen}(1^n)$ , and form  $tk = (r_{1,1}, \dots, r_{t,l}, r_1, \dots, r_m)$ , where  $r_{i,j}$ 's and  $r_h$ 's are independent randomness values, and form  $ik = (pk, \mathbf{c})$ , where  $\mathbf{c}$  consists of  $t \cdot l + m$  encryptions of zero under  $pk$  using  $r_{i,j}$ 's and  $r_h$ 's as randomness. Return  $(ik, tk)$ .
- Define  $F((pk, c_{1,1}, \dots, c_{t,l}, c_1, \dots, c_m), (pk_1, sk_1, \dots, pk_t, sk_t, x))$  to be equal to  $(pk_1, \dots, pk_t, c'_{1,1}, \dots, c'_{t,l}, c'_1, \dots, c'_m)$ , where  $c'_{i,j} = R(c_{i,j}, b_{i,j}, sk_{i+1})$  for  $1 \leq i \leq t - 1$ ,  $c'_{t,j} = R(c_{t,j}, b_{t,j}, sk_1)$  and  $c'_h = R(c_h, x_h, sk_1)$ , where  $1 \leq h \leq m$ ,  $1 \leq j \leq l$  and  $b_{w,j}$  is the  $j$ th bit of  $sk_w$ , for  $1 \leq w \leq t$ .

<sup>7</sup> This can be done by encrypting many bits under the public key and decrypting them under a candidate secret key. This, however, results in a negligible inversion error.

–  $F^{-1}((r_{1,1}, \dots, r_{t,l}, r_1, \dots, r_m), (pk_1, \dots, pk_t, c'_{1,1}, \dots, c'_{t,l}, c'_1, \dots, c'_m))$ : as in the previous constructions.

**Hardcore Function:** For  $\mathcal{F}$  given above, we let  $h: (D^t, \{0, 1\}^m) \rightarrow \{0, 1\}^m$  be defined as  $h(pk_1, sk_1, \dots, pk_t, sk_t, x) = x$ .

Correctness of inversion is again evident, and we have security as follows.

**Theorem 3.** Assuming  $\mathcal{E} = (Gen, E, D, Rep)$  is  $k$ -ind  $t$ -circularly-secure, it holds that  $\mathcal{F}$  is  $k$ -wise one-way and  $h$  is a hardcore function for  $\mathcal{F}^k$ .

*Remark 1.* In many concrete settings, for a PKE  $(Param, Gen, E, Dec)$ , we have  $Gen(1^n, par) \equiv (Pub_{par}(sk), sk)$ , for a deterministic function  $Pub$  (recall  $par$  is output by  $Param$ ): namely, the public key is obtained deterministically from the secret key and public parameters. We may now easily modify Construction 3, so that the index key also includes  $par$  and that the evaluation function no longer takes  $pk$  as input (so its entire input is a bitstring), by computing  $pk = Pub_{par}(sk)$  by itself. Now letting  $m \in \omega(t \cdot l)$  we obtain a TDF (from the assumptions stated in Theorem 3) hiding a  $(1 - o(1))$ -fraction of its input bits.

### 4 Construction of CCA Secure Encryption

Rosen and Segev [28, Theorem 1] give a BB construction of CCA1-secure encryption from any  $\omega(\log n)$ -wise TDF and a BB CCA2-secure encryption from any  $\Omega(n)$ -wise TDFs. We may use our results and those of [28] to build CCA-secure encryption. For concreteness, we give the CCA1 construction here, which simplifies that obtained by directly instantiating [28] under our base encryption primitive. The construction for the CCA2 case is obtained similarly.

We fix the following notation. For  $\mathbf{c} = (c_1, \dots, c_m)$ ,  $\mathbf{b} = (b_1, \dots, b_m)$  we extend the reproduction function  $R$  so that  $R(\mathbf{c}, \mathbf{b}, sk)$  denotes the sequence  $(R(c_1, b_1, sk), \dots, R(c_m, b_m, sk))$ . We give the CCA1 construction below.

Suppose  $\mathcal{E} = (Gen, E, Dec, R)$  has randomness space  $\mathcal{R}_n$  and secret-key space  $\{0, 1\}^l$ . We build a many-bit scheme  $\hat{\mathcal{E}} = (\hat{Gen}, \hat{E}, \hat{Dec})$  as follows.

- $\hat{Gen}(1^n)$  samples  $\mathbf{r}_0^1, \mathbf{r}_1^1, \dots, \mathbf{r}_0^t, \mathbf{r}_1^t \leftarrow \mathcal{R}_n^l$ ,  $(pk, sk) \leftarrow Gen(1^n)$  and returns  $(\mathbf{pk}, \mathbf{sk})$ , where  $\mathbf{pk} = (pk, E_{pk}(0^l; \mathbf{r}_0^1), E_{pk}(0^l; \mathbf{r}_1^1), \dots, E_{pk}(0^l; \mathbf{r}_0^t), E_{pk}(0^l; \mathbf{r}_1^t))$  and  $\mathbf{sk} = (\mathbf{r}_0^1, \mathbf{r}_1^1, \dots, \mathbf{r}_0^t, \mathbf{r}_1^t)$ ;
- $\hat{E}_{\mathbf{pk}}(m)$  parses  $\mathbf{pk} = (pk, \mathbf{c}_0^1, \mathbf{c}_1^1, \dots, \mathbf{c}_0^t, \mathbf{c}_1^t)$ , samples  $(pk', sk') \leftarrow Gen(1^n)$ ,  $u \leftarrow \{0, 1\}^t$  and returns  $(u, pk', E_{pk'}(m), \mathbf{c}'_{u_1}, \dots, \mathbf{c}'_{u_t})$ , where, for  $1 \leq i \leq t$ ,  $\mathbf{c}'_{u_i} = R(\mathbf{c}_{u_i}^i, sk', sk')$  (Note that each  $\mathbf{c}'_{u_i}^i$  is a self-encryption of  $sk'$ );
- $\hat{Dec}_{\mathbf{sk}}(u, pk', c, \mathbf{c}'_{u_1}, \dots, \mathbf{c}'_{u_t})$  parses  $\mathbf{sk} = (\mathbf{r}_0^1, \mathbf{r}_1^1, \dots, \mathbf{r}_0^t, \mathbf{r}_1^t)$ , lets  $sk_i$ , for each  $1 \leq i \leq t$ , be the plaintext obtained bit-by-bit by “opening”  $\mathbf{c}'_{u_i}^i$  relative to public key  $pk'$  and randomness vector  $\mathbf{r}_{u_i}^i$ , checks whether  $sk_1 = \dots = sk_t$  (if this check fails it returns  $\perp$ ), and returns  $Dec_{sk_1}(c)$ . Here by opening we mean finding the corresponding bit that encrypts to the given ciphertext under the specified randomness and public key.

In words,  $\hat{E}$  samples  $(pk', sk')$  and a string  $u$ , and returns  $u$ , an encryption of  $m$  under  $pk'$  as well as  $t$  self-encrypted versions of  $sk'$ , where the  $i$ th version reuses the randomness embedded in  $\mathbf{c}_{u_i}^i$ . We have the following theorem.

**Theorem 4.** *If  $t \in \omega(\log n)$  and  $\mathcal{E}$  is a reproducible,  $t$ -ind circularly-secure bit-encryption scheme, then  $\hat{\mathcal{E}}$ , constructed above, is CCA1 secure.*

The construction above is *non-shielding* [18], since the constructed decryption function queries the base encryption function.<sup>8</sup> By [18], there are no BB shielding constructions of CCA1-secure encryption from CPA-secure encryption. Since our base assumptions are strictly stronger than CPA security (at least in a BB sense), a natural question is whether or not it is possible to give a shielding construction based on our assumptions. At this point, we do not know the answer to this question, but as we show below, there exists an encryption primitive implied by our assumptions, based on which a non-shielding CCA1-construction is possible, but from which no *fully-blackbox*<sup>9</sup> shielding CCA1-construction is possible. Our new encryption primitive is an extension of CPA-secure encryption, asking that security holds even when encrypting certain *randomness-dependent messages*.

**Definition 2.** *A bit-encryption scheme  $\mathcal{E} = (Gen, E, Dec)$  with randomness space  $\{0, 1\}^\rho$  is  $q$ -randomness-dependent-message (RDM) secure if*

$$\begin{aligned} & \{E_{pk_1^1}(r_1; r), \dots, E_{pk_\rho^1}(r_\rho; r)\}, \dots, \{E_{pk_1^q}(r_1; r), \dots, E_{pk_\rho^q}(r_\rho; r)\} \\ & \equiv^c \{E_{pk_1^1}(0; r), \dots, E_{pk_\rho^1}(0; r)\}, \dots, \{E_{pk_1^q}(0; r), \dots, E_{pk_\rho^q}(0; r)\}, \end{aligned}$$

where  $r \leftarrow \{0, 1\}^\rho$  and all public keys are chosen at random according to  $Gen$ . For better readability, we made the inclusion of the public keys implicit.

In the definition above, since we are encrypting the randomness string bitwise, we should use independent public keys for each encryption. Otherwise, an adversary can easily distinguish between the two distributions. Our definition is basically an adaptation of those of [7] to the bit-encryption case. We show below that this primitive is implied by our assumptions.

Given  $\mathcal{E} = (Gen, E, Dec, R)$ , define  $\mathcal{E}' = (Gen', E', Dec')$ , whose randomness space is the key space of  $\mathcal{E}$ , as follows:  $Gen'(1^n)$  samples  $(pk, sk) \leftarrow Gen(1^n)$  and  $r \leftarrow \mathcal{R}_n$  and returns  $pk = E_{pk}(0; r)$  and  $sk = r$ . The encryption  $E'_c(b; (pk', sk'))$  returns  $(pk', R(c, b, sk'))$ ; and, finally,  $Dec'_r(pk', c')$  returns the bit  $b$  for which  $E_{pk'}(b; r) = c'$ . Using ideas described in Sect. 3 we can show, for any poly  $q$ , if  $\mathcal{E}$  is  $q$ -ind circularly secure, then  $\mathcal{E}'$  is  $q$ -RDM secure.

Next, we show  $q$ -RDM-secure encryption easily implies  $q$ -wise one-way TDFs. Let  $\mathcal{E}$ 's randomness space be  $\{0, 1\}^\rho$ , and define TDF  $\mathcal{TF} = (G, F, F^{-1})$  as follows.  $G$  runs  $Gen(1^n)$   $\rho$  times and returns  $ik = (pk_1, \dots, pk_\rho)$  and  $tk = (sk_1, \dots, sk_\rho)$ ; let  $F$ 's domain space be  $\mathcal{R}_n$  and define  $F_{pk_1, \dots, pk_\rho}(r)$  to equal  $(F_{pk_1}(r_1; r), \dots, F_{pk_\rho}(r_\rho, r))$ . The inversion algorithm  $F^{-1}$  is defined in an obvious way. Now it is not hard to show if  $\mathcal{E}$  is  $q$ -RDM secure,  $\mathcal{TF}$  is  $q$ -wise one-way. A summary of the discussion above is the following.

<sup>8</sup> Due to lack of space, we refer the reader directly to [18] for a formal definition of shielding constructions.

<sup>9</sup> We are using the notion of fully-blackbox reductions as defined in [27].

**Corollary 1.** *For any  $q \in \omega(\log n)$  there exists a shielding BB construction of CCA1-secure encryption from  $q$ -RDM-secure bit-encryption.*

We now show the BB separation of [18], stating that there are no shielding BB constructions of CCA1-secure encryption from CPA-secure encryption, extends even if the base scheme is RDM-secure, for any poly-bounded  $q$ . Combined with the corollary above, this gives us an encryption primitive which permits a non-shielding BB CCA1-secure construction, but from which no shielding BB CCA1-secure construction is possible. Specifically, [18] introduces a tuple of oracles  $\mathcal{O} = (\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathbf{w}, \mathbf{u})$ , where  $\mathcal{O}_1 = (\mathbf{g}, \mathbf{e}, \mathbf{d})$  model an idealized encryption scheme (when the oracle is chosen at random), and  $\mathcal{O}_2 = (\mathbf{d}, \mathbf{w})$  are two security-weakening components, defined based on  $\mathcal{O}_1$ . They show that (\*) for any candidate oracle-construction  $\mathcal{E} = (Gen^{\mathcal{O}_1}, Enc^{\mathcal{O}_1}, Dec^{\mathbf{g}, \mathbf{d}})$  there exists an oracle-adversary  $\mathcal{A}^{\mathcal{O}}$ , which is unbounded in time but poly-bounded in the number of oracle calls, that breaks the CCA1 security of  $\mathcal{E}$  *almost always* (i.e., except for measure-zero of oracles). Thus, to rule-out shielding fully-BB constructions, it suffices to show that (\*\*) for *almost any* selection of  $\mathcal{O}$  (i.e., measure-one of oracles),  $(\mathbf{g}, \mathbf{e}, \mathbf{d})$  is CPA-secure against any oracle-adversary  $\mathcal{A}^{\mathcal{O}}$  with constraints mentioned above.<sup>10</sup> Therefore, to rule out shielding BB constructions of CCA2 secure encryption from a new encryption primitive, it suffices to prove (\*\*) with respect to the new primitive. This is what we do below wrt RDM secure encryption. We first give the formal description of the oracles as in [18].

**Definition 3.** ([18]) *Define  $\psi$ , a distribution on oracles  $(\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathbf{w}, \mathbf{u})$ , defined for each  $n \in \mathbb{N}$ , as follows.*

- $\mathbf{g}: \{0, 1\}^n \mapsto \{0, 1\}^{3n}$  is a random one-to-one function. Function  $\mathbf{g}$  is considered as a key generator, with  $sk$  being the secret key and  $pk = \mathbf{g}(sk)$  as the public key.
- $\mathbf{e}: \{0, 1\}^{3n} \times \{0, 1\} \times \{0, 1\}^n \mapsto \{0, 1\}^{3n}$  is a random one-to-one function.
- $\mathbf{d}: \{0, 1\}^n \times \{0, 1\}^{3n} \mapsto \{0, 1, \perp\}$  is the unique function specified based on  $(\mathbf{g}, \mathbf{e})$ , where  $\mathbf{d}(sk, c) = b$  if there exists  $r \in \{0, 1\}^n$  such that  $\mathbf{e}(\mathbf{g}(sk), b, r) = c$ ; otherwise,  $\mathbf{d}(sk, c) = \perp$ .
- $\mathbf{w}: \{0, 1\}^{3n} \mapsto \{0, 1\}^{3n \times n} \cup \{\perp\}$  is a random function sampled as follows. For  $\mathbf{w}(pk)$ , if  $\mathbf{g}^{-1}(pk) = \emptyset$  then  $\mathbf{w}(pk) = \perp$ ; otherwise, sample  $r_1, \dots, r_n \leftarrow \{0, 1\}^n$  and return  $(\mathbf{e}(pk, sk_1, r_1), \dots, \mathbf{e}(pk, sk_n, r_n))$ , where  $sk = \mathbf{g}^{-1}(pk)$ .
- $\mathbf{u}: \{0, 1\}^{3n} \times \{0, 1\}^{3n} \mapsto \{\top, \perp\}$  is a deterministic function which returns  $\top$  if there exists  $sk, b$  and  $r$  such that  $\mathbf{g}(sk) = pk$  and  $\mathbf{e}(pk, b, r) = c$ , and returns  $\perp$ , otherwise.

For consistency, we may sometimes write  $\mathbf{e}(pk, b, r)$  and  $\mathbf{d}(sk, c)$ , respectively, as  $\mathbf{e}_{pk}(b; r)$  and  $\mathbf{d}_{sk}(c)$ .

We give the following theorem, a CPA version of which was proved in [18].

<sup>10</sup> We abuse notation somewhat here. By scheme  $(\mathbf{g}, \mathbf{e}, \mathbf{d})$  we mean the oracle-aided scheme  $(G^{\mathbf{g}}, E^{\mathbf{e}}, D^{\mathbf{d}})$  which just copies its oracle, e.g.,  $Gen(s)$  simply returns  $\mathbf{g}(s)$ .

**Theorem 5.** *For any adversary  $\mathcal{A}$  and poly-bounded  $q$ , there exists a negligible function  $\text{negl}$  such that*

$$\Pr_{\mathcal{O}=(\mathbf{g},\mathbf{e},\mathbf{d},\mathbf{w},\mathbf{u})\leftarrow\psi} [\Pr [\mathcal{A}^\mathcal{O}(ds_b) = b] \leq \frac{1}{2} + \text{negl}(n)] \geq 1 - \frac{1}{2^{n/2}}, \quad (1)$$

where the inner probability is over  $b$ , the randomness of  $\mathcal{A}$  and  $ds_b \leftarrow \mathcal{DS}_b$ , for

$$\mathcal{DS}_0 \equiv \{\mathbf{e}_{pk_1^1}(r_1; r), \dots, \mathbf{e}_{pk_n^1}(r_n; r)\}, \dots, \{\mathbf{e}_{pk_1^q}(r_1; r), \dots, \mathbf{e}_{pk_n^q}(r_n; r)\} \quad (2)$$

$$\mathcal{DS}_1 \equiv \{\mathbf{e}_{pk_1^1}(0; r), \dots, \mathbf{e}_{pk_n^1}(0; r)\}, \dots, \{\mathbf{e}_{pk_1^q}(0; r), \dots, \mathbf{e}_{pk_n^q}(0; r)\}, \quad (3)$$

in which  $r \leftarrow \{0, 1\}^n$  and the tuples  $(pk_1^1, \dots, pk_n^1) \dots (pk_1^q, \dots, pk_n^q)$  are formed, for every  $1 \leq i \leq n$  and  $1 \leq j \leq q$ , by sampling  $sk_i^j \leftarrow \{0, 1\}^n$  and setting  $pk_i^j = \mathbf{g}(sk_i^j)$ .

We point out a few comments. First, the choice of  $1 - \frac{1}{2^{n/2}}$  for the quantity above is not strict; we made that choice just to be consistent with that of [18]. It can in fact be, for any constant  $c < 1$ , as large as  $1 - \frac{1}{2^{n/c}}$  by choosing appropriately the negligible function used to bound the inner probability in Eq. 1. Using standard techniques (especially applying the Borel-Cantelli lemma) [21], the inequality above may then be used to conclude that for measure-one of oracles  $\mathcal{O} = (\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathbf{w}, \mathbf{u})$ , the scheme  $(\mathbf{g}, \mathbf{e}, \mathbf{d})$  is  $q$ -RDM secure against all oracle-adversaries  $\mathcal{A}^\mathcal{O}$ .

By Theorem 5 and the results of [18], as discussed above, we have

**Corollary 2.** *For any  $q \in \omega(\log n)$  there exists a non-shielding blackbox construction of CCA1 encryption from  $q$ -RDM-secure encryption. Moreover, for any poly-bounded  $q$ , there exists no shielding blackbox construction of CCA1 encryption from  $q$ -RDM-secure encryption.*

We note that it seems that one can generalize Corollary 2 to rule out the existence of shielding BB CCA1 constructions from a large class of encryption primitives whose security is defined in terms of indistinguishability against passive attacks (i.e., no decryption oracles). In other words, the BB separation generalizes to any (base) security requirement that is realized by an ideal encryption scheme  $(\mathbf{g}, \mathbf{e}, \mathbf{d})$  in the presence of  $(\mathbf{w}, \mathbf{u})$ ; for example, Corollary 2 still holds true if RDM security is replaced with circular security.

## 5 Deterministic Encryption (DE) and Instantiations

We start by reviewing some basic facts related to entropy. The *min-entropy* of a distribution  $\mathcal{D}$  is defined as  $H_\infty(\mathcal{D}) = \min_{d \in \mathcal{D}} \log(1/\Pr[\mathcal{D} = d])$ . If  $l = H_\infty(\mathcal{D})$  we call  $\mathcal{D}$  an  $l$ -source. We also recall the notion of *average min entropy*, formalized by Dodis et al. [15], defined as  $\tilde{H}_\infty(\mathcal{X}|\mathcal{Y}) = -\log(E_{y \leftarrow \mathcal{Y}}(2^{-H_\infty(\mathcal{X}|\mathcal{Y}=y)}))$ .

**DE Schemes.** Since a DE scheme is syntactically the same as a TDF, we denote a DE scheme as  $\mathcal{DE} = (G, F, F^{-1})$ . We make a few assumptions in this section. We assume the conditions stated in Remark 1 hold for any randomized

encryption (RE) scheme used in this section:  $Gen_1(1^n) \equiv Pub_{par}(sk)$ , where  $Pub$  is a deterministic function; we often drop  $par$ . We use  $l = l(n)$  to denote the length of a secret key of a RE scheme, and also the message length of a DE scheme.

We start by defining an extended notion of circular security, requiring circular security hold even if the secret key is sampled from a non-full-entropy distribution. For technical reasons, we need to allow some information about the secret key to be leaked, assuming the average min entropy of the secret key conditioned on the leaked information is high. The following definition generalizes a similar definition of [13] to the average case. We note it is possible to prove our results wrt the weaker definition of [13], but the proofs become more complex.

**Definition 4.** We say a bit-encryption scheme  $\mathcal{E} = (Gen, E, Dec)$  is  $(\lambda, l)$ -entropy circularly secure if for any joint distribution  $(\mathcal{SK}, \mathcal{X})$ , with  $\tilde{H}_\infty(\mathcal{SK}|\mathcal{X}) \geq \lambda$ , we have  $(pk, E_{pk}(sk), E_{pk}(1), x) \equiv^c (pk, E_{pk}(0^l), E_{pk}(0), x)$ , where  $(sk, x) \leftarrow (\mathcal{SK}, \mathcal{X})$  and  $pk = Pub(sk)$ .

Next we define a strengthening of the notion of [13], which adds the requirement that the public key distributions formed under high-entropy secret keys be computationally indistinguishable. This may be guaranteed if, e.g.,  $Pub$  is a *strong extractor* [25], as is the case with known circularly-secure schemes [11, 12].

**Definition 5.** We say a bit-encryption scheme  $\mathcal{E} = (Gen, E, Dec)$  is *strongly*- $(\lambda, l)$ -entropy circularly secure if (a) for any  $\lambda$ -source  $\mathcal{SK}$ ,

$$(pk, E_{pk}(sk), E_{pk}(1)) \equiv^c (pk, E_{pk}(0^l), E_{pk}(0)),$$

where  $sk \leftarrow \mathcal{SK}$  and  $pk = Pub(sk)$ ; and (b) for any  $\lambda$ -sources  $\mathcal{SK}_1$  and  $\mathcal{SK}_2$ , it holds that  $Pub(\mathcal{SK}_1) \equiv^c Pub(\mathcal{SK}_2)$ .

We now define our DE security notion, which is essentially the single-message, indistinguishability-based notion of [9]. See [9] for definitional equivalences.

**Definition 6.** We say  $\mathcal{DE} = (G, F, F^{-1})$  is secure wrt indistinguishability of  $\lambda$ -source inputs (shortly,  $(\lambda, l)$ -IND secure) if for any  $\lambda$ -sources  $\mathcal{M}_0$  and  $\mathcal{M}_1$ , it holds  $(ik, F_{ik}(\mathcal{M}_0)) \equiv^c (ik, F_{ik}(\mathcal{M}_1))$  where  $(ik, tk) \leftarrow G(1^n)$ .

Now we show that by starting from a reproducible encryption scheme which provides strong  $(\lambda, l)$ -entropy circular security, Construction 1 immediately gives us a  $(\lambda, l)$ -IND secure deterministic scheme—i.e., it preserves the parameters.

**Theorem 6.** Let  $\mathcal{E} = (Gen, E, Dec, R)$  be a reproducible bit-encryption scheme and  $\mathcal{DE} = C_1(\mathcal{E}, 1)$  be the DE scheme built in Construction 1 based on  $\mathcal{E}$  and  $t = 1$ .<sup>11</sup> If  $\mathcal{E}$  is strongly- $(\lambda, l)$ -entropy circularly secure  $\mathcal{F}$  is  $(\lambda, l)$ -IND secure.

Next we show the “weaker” entropy circular security assumption also gives rise to DE schemes, but with looser security bounds. Our construction employs the encrypt-with-hardcore (EWH) technique, described in the introduction. To this end, we assume that the ciphertext space of our (base) encryption scheme is also a bitstring space, since our construction (by employing the EWH technique) results in double encryption. We give the main theorem below.

<sup>11</sup> Here we are working with a modified version of Construction 1 stated in Remark 1.

**Theorem 7.** *Let  $\mathcal{E} = (\text{Gen}, E, \text{Dec}, R)$  be a reproducible  $(\lambda, l)$ -entropy circularly secure encryption scheme, with randomness space  $\mathcal{R}_n = \{0, 1\}^{p_r}$ . There exists an  $(l + p_r + u, 2l + p_r - \lambda)$ -IND-secure deterministic encryption scheme, where  $u \in \omega(\log n)$  is an arbitrary function.*

An outline of the proof follows, using notation given in the theorem above. The first step is to show we can use reproducibility of  $\mathcal{E}$  to encrypt any arbitrarily-long message using a  $p_r$ -long randomness string, by reusing randomness across different public keys. Next, consider the TDF given by Construction 3, based on  $t = 1$  and  $m = l + p_r - \lambda$ , and define  $hc(sk, x) = (h, h(x))$ , where  $h: \{0, 1\}^m \mapsto \{0, 1\}^{p_r}$  is chosen from a family of universal hash functions, and show  $hc$  is a hardcore function for the TDF. Now to be able to apply the EWH method, we need to show, for  $\mathcal{DS}_1 \equiv (h, h(x), E_{pk}(sk; \mathbf{r}_1), E_{pk}(x; \mathbf{r}_2), E_{pk}(0^l; \mathbf{r}_1), E_{pk}(0^{|x|}; \mathbf{r}_1))$  and  $\mathcal{DS}_2 \equiv (h, y, E_{pk}(sk; \mathbf{r}_1), E_{pk}(x; \mathbf{r}_2), E_{pk}(0^l; \mathbf{r}_1), E_{pk}(0^{|x|}; \mathbf{r}_1))$ , that  $\mathcal{DS}_1 \equiv^c \mathcal{DS}_2$ , where  $y \leftarrow \{0, 1\}^{p_r}$ ,  $(sk, x) \leftarrow (\mathcal{SK}, \mathcal{X})$ ,  $pk = \text{Pub}(sk)$  and  $H_\infty(\mathcal{SK}, \mathcal{X}) \geq l + p_r + u$ . (Also,  $\mathbf{r}_1$  and  $\mathbf{r}_2$  are chosen independently.) Now since  $\tilde{H}_\infty(\mathcal{SK}|\mathcal{X}) \geq \lambda + u$  (which follows from standard average min-entropy facts) we may appeal to  $(\lambda, l)$ -entropy circular security of  $\mathcal{E}$  to replace  $E_{pk}(sk; \mathbf{r}_1)$ , in both  $\mathcal{DS}_1$  and  $\mathcal{DS}_2$ , with an all-zero encryption; in the next step we do the same for  $E_{pk}(x; \mathbf{r}_1)$  (i.e., getting rid of the occurrences of  $x$  as a plaintext); and finally, using the facts that  $\tilde{H}_\infty(\mathcal{X}|\mathcal{SK}) \geq p_r + u$ ,  $h$  is an average-case extractor and  $u \in \omega(\log n)$ , we replace  $h(x)$  with a random string.

**Instantiations.** In the remainder of this section we briefly and informally review the scheme of Boneh et al. [11] (BHHO) and show it is reproducible. We defer the proof for the scheme of [12] as well as the proofs of entropy circular security to the full version.

Letting  $\mathcal{G}$  be a group scheme, generate  $\mathbb{G} \leftarrow \mathcal{G}$ ,  $\mathbf{g} \leftarrow \mathbb{G}^l$  and set  $par = (\mathbb{G}, \mathbf{g})$  and  $o = |\mathbb{G}|$ . Define  $(\text{Gen}, E, \text{Dec})$  as follows.  $\text{Gen}(1^n)$ : samples  $sk \leftarrow \{0, 1\}^l$  and sets  $pk = sk \cdot \mathbf{g}$  (where  $\cdot$  denotes the inner product);  $E_{pk}(g_1; r)$ : samples  $r \leftarrow \mathbb{Z}_o$  and returns  $(\mathbf{g}^r, pk^r \cdot g_1)$ , where  $\mathbf{g}^r$  denotes element-wise exponentiation; and  $D_{sk}(\mathbf{g}^r, g')$ : clear from the encryption algorithm. To show reproducibility, we need to show given  $pk_1 = sk_1 \cdot \mathbf{g}$ ,  $c_1 = (\mathbf{g}^r, pk_1^r \cdot g_1)$ ,  $sk_2$  and  $g_2$ , we can compute  $(\mathbf{g}^r, pk_2^r \cdot g_2)$ , where  $pk_2 = sk_2 \cdot \mathbf{g}$ , which is clear from the group properties. As for (strong)- $(\lambda, l)$ -entropy circular security, we note that for the schemes [11, 12] the fraction  $l/\lambda$  can be set arbitrarily large.

## 6 Conclusions and Open Problems

We gave generic constructions of several cryptographic primitives based on a general technique for de-randomizing reproducible bit-encryption schemes. For all the primitives we built it is already known that a BB construction from CPA-secure encryption alone is either impossible, or very difficult to find. We mention two main open problems that arise from our work. First, it would be interesting to see to if the BB result of [19] already separates TDFs from circularly-secure

encryption; showing this would imply that our reliance on an additional property, i.e., reproducibility, is unavoidable. Second, we would like to see whether the LWE-based circularly-secure scheme of Applebaum et al. [2] can be used to instantiate our base assumptions.

## References

1. Applebaum, B.: Key-dependent message security: generic amplification and completeness. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 527–546. Springer, Heidelberg (2011)
2. Applebaum, B., Cash, D., Peikert, C., Sahai, A.: Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 595–618. Springer, Heidelberg (2009)
3. Barak, B., Haitner, I., Hofheinz, D., Ishai, Y.: Bounded key-dependent message security. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 423–444. Springer, Heidelberg (2010)
4. 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)
5. Bellare, M., Boldyreva, A., Staddon, J.: Randomness re-use in multi-recipient encryption schemes. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 85–99. Springer, Heidelberg (2003)
6. Bellare, M., Fischlin, M., O’Neill, A., Ristenpart, T.: Deterministic encryption: definitional equivalences and constructions without random oracles. In: Wagner [31], pp. 360–378
7. Birrell, E., Chung, K.-M., Pass, R., Telang, S.: Randomness-dependent message security. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 700–720. Springer, Heidelberg (2013)
8. Black, J., Rogaway, P., Shrimpton, T.: Encryption-scheme security in the presence of key-dependent messages. In: Nyberg, K., Heys, H. (eds.) SAC 2002. LNCS, vol. 2595, pp. 62–75. Springer, Heidelberg (2003)
9. Boldyreva, A., Fehr, S., O’Neill, A.: On notions of security for deterministic encryption, and efficient constructions without random oracles. In: Wagner [31], pp. 335–359
10. Boneh, D., Canetti, R., Halevi, S., Katz, J.: Chosen-ciphertext security from identity-based encryption. *SIAM J. Comput.* **36**(5), 1301–1328 (2007)
11. Boneh, D., Halevi, S., Hamburg, M., Ostrovsky, R.: Circular-secure encryption from decision Diffie-Hellman. In: Wagner [31], pp. 108–125
12. Brakerski, Z., Goldwasser, S.: Circular and leakage resilient public-key encryption under subgroup indistinguishability. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 1–20. Springer, Heidelberg (2010)
13. Brakerski, Z., Goldwasser, S., Kalai, Y.T.: Black-box circular-secure encryption beyond affine functions. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 201–218. Springer, Heidelberg (2011)
14. Choi, S.G., Wee, H.: Lossy trapdoor functions from homomorphic reproducible encryption. *Inf. Process. Lett.* **112**(20), 794–798 (2012)
15. 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)



16. Freeman, D.M., Goldreich, O., Kiltz, E., Rosen, A., Segev, G.: More constructions of lossy and correlation-secure trapdoor functions. *J. Cryptology* **26**(1), 39–74 (2013)
17. Fuller, B., O’Neill, A., Reyzin, L.: A unified approach to deterministic encryption: new constructions and a connection to computational entropy. In: Cramer, R. (ed.) *TCC 2012*. LNCS, vol. 7194, pp. 582–599. Springer, Heidelberg (2012)
18. Gertner, Y., Malkin, T., Myers, S.: Towards a separation of semantic and CCA security for public key encryption. In: Vadhan, S.P. (ed.) *TCC 2007*. LNCS, vol. 4392, pp. 434–455. Springer, Heidelberg (2007)
19. Gertner, Y., Malkin, T., Reingold, O.: On the impossibility of basing trapdoor functions on trapdoor predicates. In: *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001*, 14–17 October 2001, Las Vegas, Nevada, USA, pp. 126–135. IEEE Computer Society (2001)
20. Goldreich, O., Levin, L.A.: A hard-core predicate for all one-way functions. In: Johnson [22], pp. 25–32
21. Impagliazzo, R., Rudich, S.: Limits on the provable consequences of one-way permutations. In: Johnson [22], pp. 44–61
22. Johnson, D.S. (ed.): *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*. ACM, New York (1989)
23. Malkin, T., Teranishi, I., Yung, M.: Efficient circuit-size independent public key encryption with KDM security. In: Paterson, K.G. (ed.) *EUROCRYPT 2011*. LNCS, vol. 6632, pp. 507–526. Springer, Heidelberg (2011)
24. Myers, S., Shelat, A.: Bit encryption is complete. In: *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009*, 25–27 October 2009, Atlanta, Georgia, USA, pp. 607–616. IEEE Computer Society (2009)
25. Nisan, N., Zuckerman, D.: Randomness is linear in space. *J. Comput. Syst. Sci.* **52**(1), 43–52 (1996)
26. Peikert, C., Waters, B.: Lossy trapdoor functions and their applications. In: Dwork, C. (ed.) *STOC*, pp. 187–196. ACM (2008)
27. 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)
28. Rosen, A., Segev, G.: Chosen-ciphertext security via correlated products. *SIAM J. Comput.* **39**(7), 3058–3088 (2010)
29. Rothblum, R.D.: On the circular security of bit-encryption. In: Sahai, A. (ed.) *TCC 2013*. LNCS, vol. 7785, pp. 579–598. Springer, Heidelberg (2013)
30. Vahlis, Y.: Two is a crowd? A black-box separation of one-wayness and security under correlated inputs. In: Micciancio, D. (ed.) *TCC 2010*. LNCS, vol. 5978, pp. 165–182. Springer, Heidelberg (2010)
31. Wagner, D.: *Advances in Cryptology - CRYPTO 2008*. LNCS, vol. 5157. Springer, Heidelberg (2008)
32. Wee, H.: Dual projective hashing and its applications — lossy trapdoor functions and more. In: Pointcheval, D., Johansson, T. (eds.) *EUROCRYPT 2012*. LNCS, vol. 7237, pp. 246–262. Springer, Heidelberg (2012)

# **Multilinear Maps and IO**

# Zeroizing Without Low-Level Zeroes: New MMAP Attacks and their Limitations

Jean-Sébastien Coron<sup>1</sup>, Craig Gentry<sup>2</sup>, Shai Halevi<sup>2</sup>, Tancrede Lepoint<sup>3</sup>,  
Hemanta K. Maji<sup>4,5</sup>, Eric Miles<sup>4</sup>, Mariana Raykova<sup>6</sup> (✉),  
Amit Sahai<sup>4</sup>, and Mehdi Tibouchi<sup>7</sup>

<sup>1</sup> University of Luxembourg, Luxembourg, Luxembourg

`jean-sebastien.coron@uni.lu`

<sup>2</sup> IBM Research, New York, USA

<sup>3</sup> CryptoExperts, Paris, France

`tancrede.lepoint@cryptoexperts.com`

<sup>4</sup> Center for Encrypted Functionalities, University of California,  
Los Angeles, USA

`{enmiles,hmaji,sahai}@cs.ucla.edu`

<sup>5</sup> Purdue University, West Lafayette, USA

<sup>6</sup> SRI International, Menlo Park, USA

`mariana@cs.columbia.edu`

<sup>7</sup> NTT Secure Platform Laboratories, Tokyo, Japan

`tibouchi.mehdi@lab.ntt.co.jp`

**Abstract.** We extend the recent zeroizing attacks of Cheon, Han, Lee, Ryu and Stehlé (Eurocrypt’15) on multilinear maps to settings where no encodings of zero below the maximal level are available. Some of the new attacks apply to the CLT13 scheme (resulting in a total break) while others apply to (a variant of) the GGH13 scheme (resulting in a weak-DL attack). We also note the limits of these zeroizing attacks.

**Keywords:** Cryptanalysis · Hardness assumptions · Multilinear maps

---

T. Lepoint—This work has been supported in part by the European Union’s H2020 Programme under grant agreement number ICT-644209.

H. K. Maji, E. Miles and A. Sahai—Research supported in part from a DARPA/ONR PROCEED award, a DARPA/ARL SAFEWARE award, NSF Frontier Award 1413955, NSF grants 1228984, 1136174, 1118096, and 1065276, a Xerox Faculty Research Award, a Google Faculty Research Award, an equipment grant from Intel, and an Okawa Foundation Research Grant. This material is based upon work supported by the Defense Advanced Research Projects Agency through the U.S. Office of Naval Research under Contract N00014-11-1-0389. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense, the National Science Foundation, or the U.S. Government.

M. Raykova—This work has been supported in part from NSF Award 1421102.

© International Association for Cryptologic Research 2015

R. Gennaro and M. Robshaw (Eds.): CRYPTO 2015, Part I, LNCS 9215, pp. 247–266, 2015.

DOI: 10.1007/978-3-662-47989-6.12

# 1 Introduction

The GGH13 [7] and CLT13 [6] “approximate multilinear maps” candidates suffer from *zeroizing attacks*, where encodings of zero at levels below the top (zero-test) level can be exploited to recover information that should have been hidden by the encoding scheme. The essence of these attacks is using successful zero tests to obtain equations over the base ring ( $\mathbb{Z}$  or  $\mathbb{Z}[X]/F(X)$ ), then solving these equations to get the desired information. First presented in the context of the GGH13 candidate [7], such attacks were recently extended by Cheon et al. [5] also to the CLT13 candidate, where they were shown to be particularly devastating, leading to a total break (when they can be mounted).

As explicitly discussed in [5], however, these attacks seem to depend on the availability of low-level encoding of zeros. This limits the applicability of these attacks, especially since several high-profile applications of multilinear maps (such as for obfuscation [8]) do not reveal such low-level zero encodings.

In this work we show that it is possible to “zeroize without low-level zeroes”: that is, we extend the attacks from [5] and apply them against both CLT13 encodings and a matrix variant of GGH13 encodings, even in settings where no low-level encodings of zero are available to the adversary. We further systematize the new attacks and show that they can overcome recent proposals to “immunize” against them [3,9]. Our extensions to the attacks from [5] include replacing low-level zero encodings by “orthogonal encodings” (this extension was observed independently also by Boneh et al. [3]), dealing with cases where more than one monomial is needed to get a zero, and dealing with modifications of the CLT13 and GGH13 schemes that use matrix-based encodings with the encoded values embedded in the eigenvalues of the matrix. Before describing our zeroizing attacks, we discuss the impact and limitations of these attacks.

## 1.1 Impact of Our Attacks

**Broken Assumptions and Constructions.** The most direct consequence of our work is that more hardness assumptions and constructions from the literature are broken. Prior to our work, the attacks of [5] already broke several assumptions and constructions using CLT13 encodings because they provided low-level encodings of zero. Our work extends to new assumptions and constructions, even where no low-level encodings of zero are available. For example, our extensions can be used to break instances of the meta-assumption of Pass et al. [16] (using either GGH13 or CLT13 encodings), even when used without low-level encodings of zero. Furthermore, we show that natural attempts to “immunize” CLT13 or GGH13 encodings by removing low-level encodings of zero [3,9] fail. In particular, the assumptions used by Gentry et al. [10,11] are broken, even when “immunized” using the technique from [3]. Perhaps more surprisingly, we also show that simplified variants of certain obfuscation schemes can be broken:

- We show that the GGHSW branching-program obfuscation procedure from [8], implemented over the CLT13 scheme [6], can be broken when it is applied to branching programs with a very specific “decomposable” structure. See Sect. 3.3.
- In the full version of this report, we also show that the simplified circuit obfuscation scheme of Zimmerman [17, Appendix A] and Applebaum-Brakerski [1] can be broken when applied to very simple circuits (e.g., point functions).

## 1.2 Limitations of Zeroizing Attacks

Potent as they are, zeroizing attacks have their limitations. For example, so far we do not have attacks on *any of the NC<sup>1</sup> obfuscation candidates in the literature*. Moreover the “dual-input straddling sets” technique that is used in several obfuscation schemes [2,4,17] appears to be effective in thwarting these attacks. See more details in Sect. 2.4.

**Successful Zero Tests are Necessary.** Our work demonstrate that some attacks are possible even if we only have top-level encoded zeros, but crucially all of these attacks depend on *successful zero tests* to get equations over the base ring. Some constructions or assumptions may not provide these zeros, and in that case it is plausible that the GGH13 and CLT13 candidates could even provide semantic security [12] of the encoded values. Even more, as far as we know the standard generic multilinear-map model could provide a good approximation of GGH13 and CLT13 in settings where top-level encoding of zeros are not available.

**The Equations Must be Simple.** In zeroizing attacks, each successful zero-test provides the adversary one equation over the base ring, and the attack relies on the attacker’s ability to solve the resulting system of equations. The successful attacks detailed in our paper (as well as those from [5,7]) arise in situations where the adversary has *substantial freedom* in creating top-level encodings of zero, and can exploit this freedom to obtain “a simple system of equations” over the base ring that can be solved using linear algebraic techniques.

There are many cases, however, in which the available encodings are constructed such that only very particular combinations of them yield a top-level encoding of zero, and those combinations do not seem to yield efficiently solvable system of equations. Two such examples, illustrated in Sect. 2.4, are obfuscation schemes that rely on Barrington’s theorem, and schemes that use the “dual-input straddling sets” technique.

We believe that long-term understanding of the security offered by current multilinear map candidates will require tackling long-standing questions about which kinds of systems of nonlinear equations are feasible to solve efficiently, and which are not.

## 2 Background and Overview

### 2.1 A Brief Description of the GGH13 and CLT13 Schemes

We begin with a brief description of the GGH13 and CLT13 schemes, omitting many details that are irrelevant for the attacks in question. Both these schemes implement graded encoding schemes where “plaintext elements” are encoded in a way that hides their value but allows to add and multiply them, and also allows to test if a degree- $k$  expression in these values is equal to zero (where  $k$  is the “multi-linearity parameter”).

**The GGH13 Scheme.** For GGH13 [7], the plaintext space is a quotient ring  $R_g = R/gR$  where  $R$  is the ring of integers in a number field and  $g \in R$  is a “small element” in that ring. The space of encodings is  $R_q = R/qR$  where  $q$  is a “big integer”. An instance of the scheme relies on two secret elements, the generator  $g$  itself and a uniformly random denominator  $z \in R_q$ . A plaintext element (which is a coset  $a = \alpha + gR$ ) is encoded “at level one” as  $u = [e/z]_q$  where  $e$  is a “small element” in the coset  $a$  (i.e.,  $e = \alpha + gr$  for some  $r \in R$ ). More generally, a level- $i$  encoding of the coset  $a$  has the form  $u = [e/z^i]_q$  for a small  $e \in \alpha + gR$ .

Addition/subtraction of encodings at the same level is just addition in  $R_q$ , and it results in an encoding of the sum at the same level, so long as the numerators do not wrap around modulo  $q$ . Similarly multiplication of elements at levels  $i, i'$  is a multiplication in  $R_q$ , and as long as the numerators do not wrap around modulo  $q$  the result is an encoding of the product at level  $i + i'$ .

The scheme also includes a “zero-test parameter” in order to enable testing for zero at level  $k$ . Noting that a level- $k$  encoding of zero is of the form  $u = [gr/z^k]_q$ , the zero-test parameter is an element of the form  $\mathbf{p}_{zt} = [hz^k/g]_q$  for a “somewhat small element”  $h \in R$ . This lets us eliminate the  $z^k$  in the denominator and the  $g$  in the numerator by computing  $[\mathbf{p}_{zt} \cdot u]_q = h \cdot r$ , which is much smaller than  $q$  because both  $h, r$  are small. If  $u$  is an encoding of a non-zero  $\alpha$ , however, then multiplying by  $\mathbf{p}_{zt}$  leaves a term of  $[h\alpha/g]_q$  which is not small. Testing for zero therefore consists of multiplying by the zero-test parameter modulo  $q$  and checking if the result is much smaller than  $q$ .

*Matrix-GGH13.* An unpublished variant of GGH13 (that was meant to protect against zeroizing attacks) uses matrices of native GGH13 encodings, where the encoded value is an eigenvalue of the matrix and the zero-test parameter includes also the corresponding eigenvector. This is essentially the same as the GGHZ countermeasure construction from [9, Sect. 7] (which is described in Sect. 3.2), except that it uses GGH13 encodings rather than CLT13 encodings.<sup>1</sup>

<sup>1</sup> Our attack from Sect. 3.2 applies for the most part to this GGH13 variant too, except that in this case we only get a weak-DL attack rather than a complete break; see the full version for details.

**The CLT13 Scheme.** The CLT13 scheme [6] is similar to above, but it relies on CRT representation modulo a composite integer  $x_0 = \prod_{j=1}^n p_j$ , where the  $p_j$ 's are “large primes”, all of about the same size. We let  $\text{CRT}(a_1, \dots, a_t)$  denote the unique element  $a \in \mathbb{Z}_{x_0}$  that is congruent to  $a_j$  modulo  $p_j$  for all  $j$ . Also we often use the shorthand  $\text{CRT}(a_j)_j$  to denote the same.<sup>2</sup>

The plaintext space in CLT13 consists of vectors  $\mathbf{a} \in \mathbb{Z}_{g_1} \times \dots \times \mathbb{Z}_{g_n}$ , where all the  $g_j$ 's are much smaller than their corresponding  $p_j$ 's. An instance of the scheme relies on the secrets  $g_j$  and  $p_j$  (with  $x_0$  public), and on a secret uniformly random denominator  $z \in \mathbb{Z}_{x_0}$ . Such a vector  $\mathbf{a} = (\alpha_1, \dots, \alpha_n)$  is encoded at level one as  $[\text{CRT}(\alpha_1 + g_1 r_1, \dots, \alpha_n + g_n r_n) / z]_{x_0}$ , where the  $r_j$ 's are all small. More generally a level- $i$  encoding of this vector is of the form  $[\text{CRT}(\alpha_j + g_j r_j)_j / z^i]_{x_0}$ .

Addition/subtraction of encodings at the same level is just addition in  $\mathbb{Z}_{x_0}$ , and it results in an encoding of the sum at the same level, so long as the numerators in the different CRT components do not wrap around modulo their respective  $p_j$ 's. Similarly multiplication of elements at levels  $i, i'$  is a multiplication in  $\mathbb{Z}_{x_0}$ , and as long as the numerators in the different CRT components do not wrap around modulo their respective  $p_j$ 's, the result is an encoding at level  $i + i'$  of the entry-wise product of the two vectors.

For zero-testing, let us denote  $p_j^* = x_0/p_j = \prod_{i \neq j} p_i$ , and note the following easy corollary of the Chinese Remainder Theorem:

**Proposition 1.** For all  $a_1, \dots, a_n \in \mathbb{Z}$ ,  $\text{CRT}(p_j^* a_j)_j = \sum_{j=1}^n p_j^* a_j \pmod{x_0}$ .

Namely when each CRT component  $j$  is divisible by  $p_j^*$ , then the CRT composition can be computed just by adding all the CRT components modulo  $x_0$ .

The zero-test parameter in CLT13 is  $\mathbf{p}_{\text{zt}} = [z^k \cdot \text{CRT}(p_j^* h_j g_j^{-1})_j]_{x_0}$  for small elements  $h_j \ll p_j$ , where  $g_j^{-1}$  is computed modulo  $p_j$ . Multiplying this zero-test parameter by a level- $k$  encoding of zero, that has the form  $u = [\text{CRT}(g_j r_j)_j / z^k]_{x_0}$ , yields

$$[\mathbf{p}_{\text{zt}} \cdot u]_{x_0} = \text{CRT}(p_j^* h_j r_j)_j = \sum_j p_j^* h_j r_j.$$

Since  $h_j r_j \ll p_j$  for all  $j$ , then  $p_j^* h_j r_j = (x_0/p_j) h_j r_j \ll x_0$ , and also the sum is much smaller than  $x_0$ . Testing for zero therefore consists of multiplying by the zero-test parameter modulo  $x_0$  and checking if the result is much smaller than  $x_0$ .

**Common Properties.** The GGH13 and CLT13 schemes share a very similar structure; here we summarize the common features that are used in the attacks:

- Each encoding is “associated” with the vector of small integers in the numerator. For GGH13 this is a 1-vector consisting of a single algebraic integer,<sup>3</sup>

<sup>2</sup> We do not assume that the  $a_j$ 's are smaller than their corresponding  $p_j$ 's.

<sup>3</sup> The matrix-GGH13 variant has vectors in the numerator rather than a single algebraic integer.

and for CLT13 this is a vector of  $n$  integers in  $\mathbb{Z}$ . Below we write informally  $u \sim (a_1, \dots, a_n)$  to denote the fact that the encoding  $u$  is associated with the vector of  $a_i$ 's. Roughly speaking, the goal of the attacks is to recover the vector  $(a_j)_j$  from the encoding  $u$ . Recovering this vector (even if not in full) is usually considered a break of the scheme.

- An encoding of zero is associated with a vector divisible by the  $g_j$ 's, namely  $u \sim (g_j r_j)_j$  for some  $r_j$ 's.
- Addition and multiplication of encodings acts entry-wise on the vector of integers in the numerator. Importantly, the addition and multiplication of these vectors is done *over the integers, with no modular reduction*. This is because a wrap-around in these operations is an error condition, and so the parameters are always set to ensure that it does not happen.
- If  $u \sim (g_j r_j)_j$  is an encoding of zero at the top level, then applying the zero-test to  $u$  returns the integer  $w = \sum_j r_j \rho_j$ , where the  $r_j$ 's are the multipliers from the numerator vector and the  $\rho_j$ 's are system parameters independent of  $u$ .

In other words, applying the zero-test to an encoding of zero yields the inner-product of the associated vector (sans the  $g_j$ 's) with a fixed secret vector. (In GGH13 this is the 1-vector  $(h)$ , in CLT13 the vector is  $(p_j^* h_j)_j$ .) Importantly, here too the inner product is over the integers, with no modular reduction.

## 2.2 Overview of Existing Attacks

*The GGH13 Zeroizing Attack.* The following “zeroizing” attack on the GGH13 scheme was described in [7]. It gets as input a level- $t$  encoding of zero  $u_0 \sim (gr)$  and many other level- $(k - t)$  encodings  $u_m \sim (a_m)$ . Multiplying  $u_0$  by any of the  $u_m$ 's yields a top-level encoding of zero  $u_0 u_m \sim (gra_m)$ , and applying the zero-test yields the algebraic integer  $w_m = hra_m$ . Note that this almost recovers the numerators  $a_m$ 's; indeed we have them up to the common factor  $h' = hr$ .

If we also knew the ideal  $I_g = gR$  that defines the plaintext space, then being able to recover the numerator up to a constant is enough to break many hardness assumptions. For example, given an encoded matrix we could compute its determinant (mod  $I_g$ ) up to a constant, which would tell us whether or not the encoded matrix has full rank.

Even when  $I_g$  is not explicitly given, Garg et al. described in [7] how it can be recovered in certain cases using GCD computations. Roughly, we can use GCD to identify and remove the common factor  $h'$ , thereby getting the  $a_m$ 's themselves, except that these are all algebraic integers so we only have GCD in terms of their ideals. Recovering the ideal  $I_a = aR$  is not always useful, e.g., if  $I_a$  and  $I_g$  are co-prime then knowing  $I_a$  does not tell us anything about our plaintext coset  $a + I_g$ . However if some of the  $u_i$ 's are themselves encoding of zero, namely  $a_i = gr_i$ , then given enough ideals  $I_{a_i} = gr_i R$  we could again use GCD calculations to recover the ideal  $I_g$  itself, and then use that knowledge to attack the non-zero encodings among the  $u_i$ 's. This attack was called in [7] a



“weak discrete-log attack”. Recently, this attack was used by Hu and Jia [14] as a component in a new attack that breaks the key-exchange protocol from [7].

We note that the GGH13 zeroizing attack does not work against CLT13 encodings, since rather than a simple product we now have an inner product  $w_m = \sum_j a_{m,j} \rho_j$ , and we cannot use this to compute GCDs. (For the same reason, this attack does not work against the matrix-GGH13 variant.)

*The CHLRS Zeroizing Attack.* Cheon, Han, Lee, Ryu and Stehlé recently described in [5] a major upgrade of the GGH13 zeroizing attack, which can be used to completely break CLT13-based schemes in some cases, recovering the factorization of  $x_0$  and all secret information. To mount the CHLRS zeroizing attack we need three sets of encoded inputs, which we denote by  $\mathcal{A} = \{A_i : i = 1, \dots, n\}$ ,  $\mathcal{B} = \{B_0, B_1\}$ , and  $\mathcal{C} = \{C_j : j = 1, \dots, n\}$  (with  $n$  the dimension of the numerator vectors). The  $A$ ’s are all random encoding of zeros, the  $B$ ’s are the target of the attack, and the  $C$ ’s are just helper encodings of random vectors. The levels of these encodings are such that multiplying  $A_i \cdot B_\sigma \cdot C_j$  yields a top-level encoding of zero for any  $i, \sigma, j$ . Below we denote the numerator vectors associated with these encodings by

$$A_i \sim (g_1 r_{i,1}, \dots, g_n r_{i,n}), \quad B_\sigma \sim (b_{\sigma,1}, \dots, b_{\sigma,n}), \quad \text{and} \quad C_j \sim (c_{j,1}, \dots, c_{j,n}).$$

Multiplying  $A_i \cdot B_\sigma \cdot C_j$  yields a top-level encoding of zero, associated with the vector  $A_i \cdot B_\sigma \cdot C_j \sim (g_1 r_{i,1} b_{\sigma,1} c_{j,1}, \dots, g_n r_{i,n} b_{\sigma,n} c_{j,n})$ . Applying the zero-test we get a four-wise inner product, yielding the integer  $w_\sigma[i, j] = \sum_{k=1}^n \rho_k r_{i,k} b_{\sigma,k} c_{j,k}$ . We can write this four-wise inner product in matrix form as

$$w_\sigma[i, j] = (r_{i,1} \ \dots \ r_{i,n}) \times \begin{pmatrix} \rho_1 b_{\sigma,1} & & \\ & \ddots & \\ & & \rho_n b_{\sigma,n} \end{pmatrix} \times \begin{bmatrix} c_{j,1} \\ \vdots \\ c_{j,n} \end{bmatrix},$$

and denote the vector on the left by  $\mathbf{a}_i$ , the matrix in the middle by  $B'_\sigma$ , and the vector on the right by  $\mathbf{c}_j$ . For a fixed  $\sigma$ , let  $i, j$  range over  $1, \dots, n$ . This yields an  $n \times n$  matrix of integers  $W_\sigma = [w_\sigma[i, j]]_{i,j} = A' \times B'_\sigma \times C'$ , where  $A'$  has the  $\mathbf{a}_i$ ’s for rows and  $C'$  has the  $\mathbf{c}_j$ ’s for columns. Since the  $r_{i,k}$ ’s,  $b_{\sigma,k}$ ’s,  $c_{j,k}$ ’s and  $\rho_k$ ’s are all random (small) quantities, then with high probability the matrices are all invertible (over the rationals). Having computed the matrices  $W_\sigma$ , the attacker now sets

$$W = W_0 \times W_1^{-1} = (A' B'_0 C') \times (A' B'_1 C')^{-1} = A' \times (B'_0 \times B_1^{-1}) \times A'^{-1}.$$

Observe now that  $B^* = B'_0 \times B_1^{-1}$  is a diagonal matrix with  $b_{0,j}/b_{1,j}$  on the diagonal, and thus the eigenvalues of  $B^*$  are all the ratios  $b_{0,j}/b_{1,j}$ . And since  $W$  and  $B^*$  are similar matrices, then also the eigenvalues of  $W$  are the  $b_{0,j}/b_{1,j}$ ’s. Hence once it computes  $W$ , the attacker can find its eigenvalues (over the rationals) and obtain all the ratios  $b_{0,j}/b_{1,j}$ .

These ratios may be enough by themselves to break some hardness assumptions, but for CLT13 it is possible to use them to factor  $x_0$ , thereby getting

a complete break. Specifically, since each ratio is rational it can be written as  $u/v = b_{0,j}/b_{1,j}$  with  $u, v$  co-prime integers. Recalling now that  $B_0, B_1$  are two encodings at the same level (say, level  $t$ ) with numerator vectors  $(b_{0,1}, \dots, b_{0,n})$  and  $(b_{1,1}, \dots, b_{1,n})$ , respectively, we get that

$$uB_1 - vB_0 = [\text{CRT}(ub_{1,1} - vb_{0,1}, \dots, ub_{1,n} - vb_{0,n})/z^t]_{x_0}.$$

This means that the  $j$ 'th CRT component is  $ub_{1,j} - vb_{0,j} = 0$ , and with high probability the others are not, so we get  $GCD(x_0, uB_1 - vB_0) = p_j$ .

### 2.3 Extending the CHLRS Attack

In the current work we describe several extensions to attacks of Cheon et al. from [5]; below we describe these extensions briefly.

**GGH13 vs. CLT13.** We can also apply these zeroizing attacks to a matrix variant of GGH13, not just to CLT13 encodings, resulting in a “weak discrete-log” attack. This is described in the full version.

**Orthogonal Encodings.** We also note that these attacks do not actually require low-level encoding of zeros. Indeed all we need is that for every  $i, \sigma, j$ , the product  $A_i B_\sigma C_j$  is a top-level encoding of zero, so we could have the  $A$ 's with zeros in a few CRT components, the  $B$ 's with zeros in some other components, and the  $C$ 's with zeros in all the CRT components not covered by the  $A$ 's and  $B$ 's. This observation was also made concurrently by Boneh et al. [3].

**More than One Monomial.** The attack also extends to a setting where more than a single monomial is needed to get a zero. For example, consider the case where we have not three but six sets of encodings. Similar to before we have  $\mathcal{A} = \{A_i : i = 1, \dots, 2n\}$ ,  $\mathcal{B} = \{B_0, B_1\}$ , and  $\mathcal{C} = \{C_j : j = 1, \dots, 2n\}$ , but now we also have  $\tilde{\mathcal{A}} = \{\tilde{A}_i : i = 1, \dots, 2n\}$ ,  $\tilde{\mathcal{B}} = \{\tilde{B}_0, \tilde{B}_1\}$ , and  $\tilde{\mathcal{C}} = \{\tilde{C}_j : j = 1, \dots, 2n\}$ . (Note that the indices  $i, j$  now range over  $[1, 2n]$ , not  $[1, n]$ ). The new attack requires that  $A_i B_\sigma C_j + \tilde{A}_i \tilde{B}_\sigma \tilde{C}_j$  is a top-level encoding of zero for every  $i, \sigma, j$ . We denote the numerator vectors associated with these encodings by

$$\begin{aligned} A_i &\sim (a_{i,1}, \dots, a_{i,n}), & B_\sigma &\sim (b_{\sigma,1}, \dots, b_{\sigma,n}), & C_j &\sim (c_{j,1}, \dots, c_{j,n}), \\ \tilde{A}_i &\sim (\tilde{a}_{i,1}, \dots, \tilde{a}_{i,n}), & \tilde{B}_\sigma &\sim (\tilde{b}_{\sigma,1}, \dots, \tilde{b}_{\sigma,n}), & \tilde{C}_j &\sim (\tilde{c}_{j,1}, \dots, \tilde{c}_{j,n}). \end{aligned}$$

We can think of the pairs  $(A_i, \tilde{A}_i)$ ,  $(B_\sigma, \tilde{B}_\sigma)$ ,  $(C_j, \tilde{C}_j)$  as encodings that are associated with numerator vectors of twice the dimension, and the CHLRS attack can be applied to these new “double encodings”. The only difference (other than the larger dimension) is that we can no longer associate the division-by- $g_i$  with any single vector. Instead, applying the zero-test to  $A_i B_\sigma C_j + \tilde{A}_i \tilde{B}_\sigma \tilde{C}_j$  yields a four-wise inner product *divided by the  $g_i$ 's*, which we can write in matrix form:



## 2.4 Attack Limitations

As sketched in the introduction, zeroizing attacks have their limitations, in that they require zeros and moreover need the equations that yield these zeros to be “simple.” Two scenarios that seem outside the scope of these attacks due to “non-simple” equations are discussed next.

*Obfuscation Using Barrington’s Theorem.* Consider the obfuscation schemes in the literature that obfuscate matrix-based branching programs (BP) resulting from Barrington’s theorem [2,4,8,16]. These schemes are designed so that the only way to get a top-level zero encoding is using the prescribed routines for evaluating the obfuscated circuit on various inputs, so we only need to examine the type of expressions that arise from such evaluation.

Recall that a matrix-based BP has a sequence of steps, each specified by two matrices and controlled by an input bit. On a given input, we choose one of the two matrices in each step (based on the corresponding input bit), then multiply all of the selected matrices in order to get the result. In the BPs that are generated by Barrington’s theorem, each input bit controls several steps that are spaced far apart, and so changing the value of that bit changes the selection of all these matrices. This makes it hard to apply our attacks in this setting, since these attacks require a multilinear setting where we can get many different zeros by changing just a single variable in every monomial. Therefore, even though we do get equations over the base ring from top-level zeros in this scheme, these equations appear to be correlated in a highly non-linear manner, foiling our attempts to glean useful information from them.

We contrast this situation with the attack that we describe in Sect. 3.3, that breaks obfuscation of very simple branching programs which are “separable” in the sense that different subsets of the input bits control different consecutive intervals of steps, thus giving us the simple system of equations that we need.

*Binding Variables.* The CHLRS attacks and our extensions rely on the ability to partition the variables into groups ( $\mathcal{A}, \mathcal{B}, \mathcal{C}$  above), so that we can independently choose variables from the different groups and every such choice yields a top-level zero. Several schemes in the literature use explicit binding variables to make it hard to partition the encodings into independent sets. For example, the obfuscation schemes of Barak et al. [2] and Zimmerman [17] use “dual-input straddling sets” to create a “high connectivity” interlocking set of encodings.

These schemes contain, for each pair  $i, j$  of input bits, four encoded variables  $U_{i,j,0,0}, U_{i,j,0,1}, U_{i,j,1,0}$ , and  $U_{i,j,1,1}$ , such that obtaining a top-level encoding of zero requires multiplying  $U_{i,j,*,*}$ ’s that are consistent with some  $n$ -bit input  $x$  (i.e., it requires computing some expression  $\cdot \prod_{i,j} U_{i,j,x_i,x_j}$ ). This structure seems to foil attempts of separating the variables into independent sets, since changing any input bit creates a cascading effect. To illustrate the difficulty of applying the attack in this setting, we describe in the full version a relatively simple source-group hardness assumption involving such binding variables, which we do not know how to break even though we are given many low-level CLT13 encodings of zero.

### 3 A Unified Attack Against CLT13-Based Schemes

Below we present a general attack on CLT13-based schemes that combines all the ideas from Sect. 2.3, and show how this attack can be used against:

- The proposed CLT13 modification by Garg et al. [9, Sect. 7] (that was suggested in response to the CHLRS attacks);
- Obfuscations of branching programs with specific structure using the iO procedure of Garg et al. [8].

Central to our general attack is the notion of a “good attack set,” which roughly plays the role of the sets  $\mathcal{A}, \mathcal{B}, \mathcal{C}$  from Sect. 2 (together with the zero-test parameter). To define this notion formally, fix an instance of CLT13 with  $n$  secret primes  $p_1, \dots, p_n$  and modulus  $x_0 := \prod_i p_i$ . An *attack set* (of dimension  $d$ ) consists of three sets of matrices  $\mathcal{A}, \mathcal{B}, \mathcal{C} \subset \mathbb{Z}_{x_0}^{d \times d}$ , of sizes  $|\mathcal{A}| = |\mathcal{C}| = nd$  and  $|\mathcal{B}| = 2$ , and two vectors  $s \in \mathbb{Z}_{x_0}^{1 \times d}$  and  $t \in \mathbb{Z}_{x_0}^{d \times 1}$ . These sets are constructed from the available public parameters and encodings of a given scheme, in such a way that for every choice of  $(A_i, B_\sigma, C_j) \in \mathcal{A} \times \mathcal{B} \times \mathcal{C}$ , the value

$$W_\sigma[i, j] := s \times A_i \times B_\sigma \times C_j \times t \in \mathbb{Z}_{x_0}$$

is a zero-tested top-level encoding of 0. (The CHLRS attack can be thought as a special case where all the “matrices” are of dimension  $d = 1$ , and we have  $s = 1$  and  $t = \mathbf{p}_{zt}$ .) Given such an attack set, the attack proceeds as in Fig. 1, where we denote by  $[z]_p$  the reduction of  $z$  modulo  $p$  into the interval  $[-p/2, p/2)$ , and this notation extends entry-wise to vectors and matrices.

**Input:**  $\mathcal{A} = \{A_i\}_i$ ,  $\mathcal{B} = \{B_\sigma\}_\sigma$ ,  $\mathcal{C} = \{C_j\}_j$ ,  $s$ ,  $t$

1. Compute  $(nd) \times (nd)$  matrices  $W_0, W_1$  as  $W_\sigma[i, j] := [s \times A_i \times B_\sigma \times C_j \times t]_{x_0}$ .
2. Compute  $W := W_0 \times W_1^{-1}$  over  $\mathbb{Q}$ , and  $M := B_0 \times B_1^{-1} \pmod{x_0}$ .
3. Compute  $W$ 's characteristic polynomial  $f := \text{charPoly}(W)$  over  $\mathbb{Q}$ , and factor it into monic irreducible factors over  $\mathbb{Q}$  as  $f = f_1 f_2 \cdots f_m$ .
4. For all  $k \in \{1, \dots, m\}$  define  $F_k := f/f_k = \prod_{i \neq k} f_i \in \mathbb{Q}[X]$ , let  $d_k$  be the common denominator of the coefficients of  $F_k$ , and set  $G_k := F_k \cdot d_k$ .
5. Evaluate the  $G_k$ 's at the matrix  $M \pmod{x_0}$ ,  $M_k := [G_k(M)]_{x_0} \forall k \leq m$ .
6. Compute  $S := \{\text{GCD}(M_k[i, j], x_0) \mid i, j \in [nd]; k \in [m]\}$ , and return  $\{x_0/q \mid q \in S\}$ .

**Fig. 1.** Our general attack on CLT13-based schemes

#### 3.1 Sufficient Conditions for the Attack to Succeed

Next we state and prove sufficient conditions on the attack set that ensures that the attack in Fig. 1 succeeds. Specifically, we would like to show that each  $M_k$  in

step 5 must be zero modulo all the primes except one, and hence any non-zero entry in it yields a nontrivial factor of  $x_0$  (i.e. the product of those primes).

Referring to the intuition from Sect. 2.3, the matrix  $W = A \times B^* \times A^{-1}$  is similar to a block-diagonal matrix  $B^*$  that has one block for each CRT component. Specifically, the  $j$ th block of  $B^*$  is  $B_j^* = [B_0]_{p_j} \times ([B_1]_{p_j})^{-1}$  (inverse over  $\mathbb{Q}$ ). The characteristic polynomial of  $W$  is then the product of the characteristic polynomials of all the blocks. For simplicity, assume the block polynomials are the irreducible factors  $f_i$  from Fig. 1. Then each  $F_k$  is thus the product of all block polynomials except the  $k$ th, and by the Cayley-Hamilton theorem we have that  $F_k(B_j^*) = 0$  (and therefore also  $G_k(B_j^*) = 0$ ) for all blocks  $j \neq k$ . But  $G_k(B_j^*) = 0$  over  $\mathbb{Q}$  implies that also  $G_k(B_0 \times B_1^{-1}) = 0 \pmod{p_j}$ , so  $G_k(M)$  is zero modulo all primes  $j \neq k$ . The only thing left to ensure is that for the last prime  $p_k$  we get  $G_k(M) \neq 0 \pmod{p_k}$ , which is the essence of our sufficient condition. The actual condition in Definition 1 below is slightly more complex, to account for the case when the block polynomials are reducible over  $\mathbb{Q}$ .

**Definition 1.** Fix an attack set  $(\mathcal{A}, \mathcal{B}, \mathcal{C}, s, t)$ . Let  $B_0, B_1, M, W$  be the matrices from Fig. 1, and let  $g_j := \text{charPoly}([B_0]_{p_j} \times [B_1]_{p_j}^{-1})$  over  $\mathbb{Q}$ . We say that  $(\mathcal{A}, \mathcal{B}, \mathcal{C}, s, t)$  is good if:

1.  $f := \text{charPoly}(W) = \prod_{j \leq n} g_j$ ;
2.  $B_1$  is non-singular modulo  $x_0$ ;
3. The common denominators  $d_k$  from step 4 are all co-prime with  $x_0$ ;
4. For any  $j \leq n$  and any divisor  $f_k$  of  $g_j$  of degree  $\geq 1$  (possibly  $f_k = g_j$ ), denoting  $G_k = d_k \cdot f / f_k$  as in step 4, we have  $G_k(M) \neq 0 \pmod{p_j}$ .

**Theorem 1.** For any good attack set  $(\mathcal{A}, \mathcal{B}, \mathcal{C}, s, t)$ , the algorithm in Fig. 1 recovers the secret primes  $p_1, \dots, p_n$ .

To prove Theorem 1 we use the following lemma:

**Lemma 1.** Let  $p > 1$  and  $u_1, \dots, u_t, v_1, \dots, v_t$  be integers, s.t. the  $v_i$ 's are invertible mod  $p$ , and denote  $w_i = [u_i \cdot v_i^{-1}]_p$ . If  $g$  is a multivariate integer polynomial such that  $g(\frac{u_1}{v_1}, \dots, \frac{u_t}{v_t}) = 0$  over  $\mathbb{Q}$ , then  $g(w_1, \dots, w_t) = 0 \pmod{p}$ .

*Proof.* It is enough to prove it for a linear  $g$ , since we can replace any non-linear term  $\prod_{i \in I} (\frac{u_i}{v_i})^{e_i}$  (for some  $I \subset [t]$  and  $e_i$ 's) by new variables  $u' = \prod_{i \in I} u_i^{e_i}$ ,  $v' = \prod_{i \in I} v_i^{e_i}$ , and  $w' = [\prod_{i \in I} w_i^{e_i}]_p = [u' \cdot v'^{-1}]_p$ , and then prove the same statement on the resulting new polynomial.

Now denote  $V = \prod_i v_i$  and for each  $i$  denote  $v_i^* = V/v_i = \prod_{j \neq i} v_j$ . For a linear  $g$  we can write  $\sum_i g_i \cdot \frac{u_i}{v_i} = 0$  over  $\mathbb{Q}$ , so also  $\sum g_i u_i v_i^* = V \cdot \sum_i g_i \cdot \frac{u_i}{v_i} = 0$ , and in particular  $\sum g_i u_i v_i^* = 0 \pmod{p}$ . Finally, since  $V$  is invertible modulo  $p$  we get

$$\sum_i g_i w_i = \sum_i g_i u_i v_i^{-1} = V^{-1} \cdot \sum_i g_i u_i v_i^* = 0 \pmod{p}. \quad \square$$

*Proof (of Theorem 1).* For all  $i$  denote  $B_i^* = [B_0]_{p_i} \times [B_1]_{p_i}^{-1}$  over  $\mathbb{Q}$  and  $\hat{B}_i = [B_0]_{p_i} \times [B_1]_{p_i}^{-1}$  over  $\mathbb{Z}_{p_i}$ . Let  $t_i := \det([B_1]_{p_i})$  (over  $\mathbb{Q}$ ), and since  $B_1$  is non-singular modulo  $x_0$  then in particular  $t_i \neq 0 \pmod{p_i}$ . We can therefore write  $B_i^* = \hat{B}_i/t_i$  for an integer matrix  $\hat{B}_i$ , and clearly we also have  $\hat{B}_i = \hat{B}_i \cdot t^{-1} \pmod{p_i}$ .

Denote the characteristic polynomial of  $B_i^*$  over  $\mathbb{Q}$  by  $g_i := \text{charPoly}(B_i^*)$ . By the first condition in Definition 1 we have  $f := \text{charPoly}(W) = \prod_{j \leq n} g_j$ . Note, however, that the  $g_j$ 's are not necessarily irreducible, so there isn't necessarily a 1-1 correspondence between the  $g_j$ 's and the irreducible factors  $f_k$  of  $f$ .

Fix an index  $j \leq n$  and we show that for some  $k$  it holds that  $G_k(M) \neq 0 \pmod{p_j}$  but  $G_k(M) = 0 \pmod{p_i}$  for all  $i \neq j$ . Clearly this  $g_j$  is divisible by at least one  $f_k$  (which has degree  $\geq 1$ ), so the last condition of Definition 1 implies that  $G_k(M) = d_k \cdot F_k(M) \neq 0 \pmod{p_j}$ . It remains to show that for all the other primes  $p_i, i \neq j$ , we have  $G_k(M) = 0 \pmod{p_i}$ .

Clearly  $F_k$  is divisible by  $g_i$  for every  $i \neq j$ , so the Cayley-Hamilton theorem implies that  $F_k(B_i^*) = 0$  (over  $\mathbb{Q}$ ) for all  $i \neq j$ , and therefore also  $G_k(B_i^*) = 0$ . Viewing  $G_k(B_i^*)$  as a collection of multivariate polynomials over the elements of  $B_i^*$ , and using the facts that  $B_i^* = \tilde{B}_i/t_i$  and  $\hat{B}_i = \tilde{B}_i \cdot t^{-1} \pmod{p_i}$ , we can apply Lemma 1 to conclude that also  $G_k(\hat{B}_i) = 0 \pmod{p_i}$ . And since  $M = \hat{B}_i \pmod{p_i}$  then also  $G_k(M) = 0 \pmod{p_i}$ , as needed.

We have shown that  $M_k := G_k(M)$  satisfies  $M_k \neq 0 \pmod{p_j}$  but  $M_k = 0 \pmod{p_i}$  for all  $i \neq j$ , so there exists an entry  $z = M_k[a, b]$  such that  $z \neq 0 \pmod{p_j}$  but  $z = 0 \pmod{p_i}$  for all  $i \neq j$ . Thus  $GCD(z, x_0) = \prod_{i \neq j} p_i$ , and  $x_0/GCD(z, x_0) = p_j$ .  $\square$

Below we construct good attack sets for some schemes in the literature. More examples can be found in the full version. We will repeatedly use the fact that for a CLT13 encoding  $u$  associated with numerator vector  $u \sim (r_i g_i + m_i)_i$ , the randomization vector  $(r_i)_{i \in [n]}$  is nearly uniform for each encoding. Specifically we have the following, which is proved in [3, Lemma 5.7].

**Lemma 2 ([3]).** *There exists a prime  $q = 2^{\Omega(n)}$  which is determined by the CLT13 system parameters such that, for each encoding, the distribution on  $(r_i \pmod q)_{i \in [n]}$  is  $\text{negl}(n)$ -close to the uniform distribution on  $\mathbb{Z}_q^n$ .*

### 3.2 Attacking the Garg-Gentry-Halevi-Zhandry Countermeasure

Garg, Gentry, Halevi, and Zhandry proposed in [9, Sect. 7] a variant of the CLT13 scheme, that was designed to resist the CHLRS attack. This variant uses matrices of native CLT13 encodings, where the encoded value is an eigenvalue of the matrix and the zero-test parameter includes also the corresponding eigenvector. The CHLRS attack from [5] indeed does not apply to this variant, but below we show that this variant still gives rise to a good attack set, and thus our new attack from Fig. 1 recovers the secret primes.

The GGHZ variant relies on the same parameters as CLT13, namely we choose  $(\{g_i\}_i, \{p_i\}_i, \mathbf{p}_{zt}, \{z_i\})$  (with  $x_0 := \prod_i p_i$  and top level corresponding to

denominator  $z^* = \prod z_i$ ). Let  $d := 2\kappa + 1$ , and choose a secret matrix  $T \in \mathbb{Z}_{x_0}^{d \times d}$  uniformly. An encoding of a plaintext value  $c$  at some level is given by  $C \in \mathbb{Z}_{x_0}^{d \times d}$ , where<sup>4</sup>

$$C := T \times \underbrace{\begin{bmatrix} \hat{\$} & \hat{0} & \dots & \hat{0} \\ \hat{0} & \hat{\$} & \dots & \hat{0} \\ \vdots & & & \vdots \\ \hat{0} & \hat{0} & \dots & \hat{c} \end{bmatrix}}_{C^*} \times T^{-1} \pmod{x_0}.$$

Each  $\hat{\$}$  in  $C^*$  is a “native CLT13 encoding” of an independent random value at the given level, each  $\hat{0}$  is an independent native encoding of 0, and  $\hat{c}$  is a native encoding of  $c$ . For zero-testing, two dimension- $d$  vectors  $s, t$  are provided:

$$\begin{aligned} s &:= [\hat{\$} \dots \hat{\$} \hat{0} \dots \hat{0} \hat{\$}] \times T^{-1} \pmod{x_0} \\ t &:= \mathbf{p}_{zt} \cdot T \times [\hat{0} \dots \hat{0} \hat{\$} \dots \hat{\$}]^T \pmod{x_0} \end{aligned}$$

where  $\hat{0}$  and  $\hat{\$}$  are CLT13 native “level-zero” encodings (i.e., corresponding to denominator 1). Then a GGHZ-encoding  $C$  as above at the top level level can be zero tested by computing  $s \times C \times t = (\hat{\$} \cdot \hat{c} + \hat{0}) \cdot \mathbf{p}_{zt} \pmod{x_0}$  and checking for smallness.

*Attack Set.* The matrix sets  $\mathcal{A}, \mathcal{B}, \mathcal{C}$  consist directly of GGHZ-encodings, since these are already in matrix form. Specifically, we assume that  $[1, \kappa]$  is partitioned into three intervals  $I_A = [1, k_A], I_B = [k_A + 1, k_B], I_C = [k_B + 1, \kappa]$ , such that we have GGHZ-encodings

- $\mathcal{A} = \left\{ A_i = T \times A_i^* \times T^{-1} : A_i \text{ encoded at level } I_A \right\}_{i \in [nd]}$
- $\mathcal{B} = \left\{ B_\sigma = T \times B_\sigma^* \times T^{-1} : B_\sigma \text{ encoded at level } I_B \right\}_{\sigma \in \{0,1\}}$
- $\mathcal{C} = \left\{ C_k = T \times C_k^* \times T^{-1} : C_k \text{ encoded at level } I_C \right\}_{k \in [nd]}$

where  $A_i \times B_\sigma \times C_k$  is a GGHZ-encoding of 0 for all  $i, k \in [nd]$  and  $\sigma \in \{0, 1\}$ . The vectors  $s$  and  $t$  are the zero testing vectors from the GGHZ scheme.

*Attack Set Properties.* We prove that  $(\mathcal{A}, \mathcal{B}, \mathcal{C}, s, t)$  form a good attack set according to Definition 1. We write

$$\begin{aligned} W_\sigma[i, k] &= s \times A_i \times B_\sigma \times C_k \times t \\ &= s \times T \times A_i^* \times B_\sigma^* \times C_k^* \times T^{-1} \times t = \mathbf{a}^i \times B_\sigma^* \times \mathbf{c}^k \end{aligned}$$

---

<sup>4</sup> The attack applies also when one uses many matrices  $T_0, T_0^{-1}, \dots, T_\kappa, T_\kappa^{-1}$  (rather than just  $T, T^{-1}$ ), so multiplication can only be performed in a specific order, as described in [9].



where  $\mathbf{a}^i := s' \times A_i^*$  and  $\mathbf{c}^k := C_k^* \times t'$  are dimension- $d$  vectors. The above equality holds over the integers, not only modulo  $x_0$ , since all the variables in the final right-hand-side are small compared to  $x_0$ .

We denote  $\mathbf{a}_\ell^i := \mathbf{a}^i \bmod p_\ell$  and  $\mathbf{c}_\ell^k := \mathbf{c}^k \bmod p_\ell$  for  $i \in [nd], \ell \in [n]$ . Now we can write  $W_\sigma = \tilde{A} \times \tilde{B}_\sigma \times \tilde{C}$ , where  $\tilde{A}$  is an  $nd \times n^2d$  matrix,  $\tilde{C}$  is an  $n^2d \times nd$  matrix, and  $\tilde{B}_\sigma$  is a  $n^2d \times n^2d$  block-diagonal matrix, defined as follows.

$$\tilde{A} = \begin{bmatrix} \mathbf{a}_1^1 & \mathbf{a}_2^1 & \cdots & \mathbf{a}_n^1 \\ \mathbf{a}_1^2 & \mathbf{a}_2^2 & \cdots & \mathbf{a}_n^2 \\ \vdots & \vdots & & \vdots \\ \mathbf{a}_1^{nd} & \mathbf{a}_2^{nd} & \cdots & \mathbf{a}_n^{nd} \end{bmatrix} \quad \tilde{C} = \begin{bmatrix} (\mathbf{c}_1^1)^T & (\mathbf{c}_1^2)^T & \cdots & (\mathbf{c}_1^{nd})^T \\ (\mathbf{c}_2^1)^T & (\mathbf{c}_2^2)^T & \cdots & (\mathbf{c}_2^{nd})^T \\ \vdots & \vdots & & \vdots \\ (\mathbf{c}_n^1)^T & (\mathbf{c}_n^2)^T & \cdots & (\mathbf{c}_n^{nd})^T \end{bmatrix}$$

$$\tilde{B}_\sigma = \begin{bmatrix} B_\sigma^* \bmod p_1 & 0 & & 0 \\ 0 & B_\sigma^* \bmod p_2 & & 0 \\ & & \ddots & \\ 0 & 0 & & B_\sigma^* \bmod p_n \end{bmatrix}$$

Using Lemma 2 and the Schwartz-Zippel lemma, it can be shown that with high probability over the randomness in the CLT13 encodings,  $\tilde{A}$ ,  $\tilde{C}$ , and each  $B_\sigma^*$  have full rank  $nd$ . Under this condition each  $W_\sigma$  has rank  $nd$  and is thus invertible, so we can write  $W = W_0 \times W_1^{-1} = \tilde{A} \times \tilde{B}_0 \times \tilde{B}_1^{-1} \times \tilde{A}^{-1}$ , where  $\tilde{A}^{-1}$  denotes the right inverse of the (non-square, full-rank) matrix  $\tilde{A}$ . Then we have

$$\begin{aligned} \text{charPoly}(W) &= \text{charPoly}(\tilde{B}_0 \times \tilde{B}_1^{-1}) = \prod_{i=1}^n \text{charPoly}([B_0^*]_{p_i} \times [B_1^*]_{p_i}^{-1}) \\ &= \prod_{i=1}^n \text{charPoly}([B_0]_{p_i} \times [B_1]_{p_i}^{-1}) \end{aligned}$$

so the first property of Definition 1 holds. The second property of Definition 1 holds with high probability over the choice of randomness in the CLT13 encodings. We were not able to prove that the last two properties in Definition 1 hold, but we verified them experimentally by running the attack on several random instances and checking that they indeed hold in all of them. For the fourth property, we can prove that it holds under the following natural conjecture:

*Conjecture 1.* For each  $i \in [n]$ , with high probability over the randomness in the CLT13 encodings,  $\text{charPoly}([B_0^*]_{p_i} \times [B_1^*]_{p_i}^{-1})$  is irreducible over  $\mathbb{Q}$ .

We make two remarks about this conjecture. First, we have verified it experimentally. Second, a work of Kuba [15] shows that among the degree- $n$  univariate integer polynomials whose coefficients are bounded in absolute value by an integer  $t$ , the polynomials that are reducible over  $\mathbb{Q}$  make up a roughly  $1/t$  fraction. In particular, a random polynomial with  $r$ -bit coefficients is irreducible over  $\mathbb{Q}$  with probability roughly  $1 - 2^{-r}$ . Thus provided that  $\text{charPoly}([B_0^*]_{p_i} \times [B_1^*]_{p_i}^{-1})$

is well-distributed among polynomials with an appropriate coefficient bound, Conjecture 1 should hold. We note that the relationship between a random polynomial and the characteristic polynomial of a random matrix has been explored by Hansen and Schmutz [13]. However, their results do not seem directly applicable here because they study polynomials over a finite field  $\mathbb{F}$ , and a uniform degree- $n$  polynomial is irreducible over  $\mathbb{F}$  only with probability  $\approx 1/n$ .

Assuming Conjecture 1, the fourth property of Definition 1 reduces to showing that for every prime factor  $p_j$  of  $x_0$ ,  $\left(\prod_{i \neq j} d_i f_i\right)(M) \neq 0 \pmod{p_j}$  where  $d_i$ ,  $f_i$ , and  $M$  are as in Fig. 1. Choose all values in the CLT13 encodings except for the random values in the  $j$ th slot of the encodings in  $B_0$ , and call the unchosen values  $R$ . With high probability over this choice, each entry of  $M$  is a non-trivial linear polynomial in  $R$ , and  $\left(\prod_{i \neq j} d_i f_i\right)$  is a non-trivial degree- $(n - 1)$  polynomial in  $M$ . Thus each entry of  $\left(\prod_{i \neq j} d_i f_i\right)(M)$  is a non-trivial degree- $(n - 1)$  polynomial in  $R$ , and is non-zero modulo  $p_i$  with high probability by Lemma 2 and the Schwartz-Zippel lemma.

### 3.3 Attacking GGHRSW Obfuscation for Simple Branching Programs

We observe that our unified attack can be applied also to the candidate obfuscation construction of Garg et al. [8] when instantiated with the CLT13 multilinear maps and applied to branching programs with specific “partitionable” structure that we define below. We stress that applying Barrington’s theorem to a circuits *does not have the required structure*, so as far as we know, the iO candidate from [8] for  $NC^1$  circuits remains plausible.

**The GGHRSW Obfuscation Candidate for Branching Programs.** Recall that the obfuscator of Garg et al. [8] consists of encoded, randomized versions of two BPs; one is the BP that we want to obfuscate and the other is a “dummy BP” consisting of only identity matrices (and hence computing the all-one function). Even though neither program computes a zero, they are constructed such that their difference on accepting computations yields an encoding of zero, which can be recognized by zero testing. The core construction from [8] works with oblivious branching programs. An oblivious branching program of length  $L$  over  $\ell$  input variables is defined as follows

$$BP = \{(\text{inp}(i), A_{i,0}, A_{i,1}) : i \in [L], \text{inp}(i) \in [\ell], A_{i,b} \in \{0, 1\}^{w \times w}\},$$

where the  $A_{i\sigma}$ ’s are invertible matrices and  $\text{inp}(i)$  is the input bit position examined in step  $i$ . The function computed by this branching program is defined (using some fixed matrix  $A_0 \neq I$ ) as

$$f_{BP,A,I} = \begin{cases} 0 & \text{if } \prod_{i=1}^L A_{i,x_{\text{inp}i}} = A_0 \\ 1 & \text{if } \prod_{i=1}^L A_{i,x_{\text{inp}i}} = I \\ \text{undef} & \text{otherwise.} \end{cases}$$

Let  $\mathbb{Z}_p$  be a ring that we use for randomization, and for each input bit  $j$  denote by  $I_j := \{i \in [L] : \text{inp}(i) = j\}$  the set of steps where the branching program examine the  $j$ 'th input bit. The GGHRSW construction, on input an  $L$ -step branching program  $BP$  over  $\ell$  input bits, proceeds as follows:

1. Sample random and independent scalars  $\{\alpha_{i,0}, \alpha_{i,1}, \alpha'_{i,0}, \alpha'_{i,1} \in \mathbb{Z}_p : i \in [L]\}$ , subject to the constraint that for any input bit  $j \in [\ell]$ , we have  $\prod_{i \in I_j} \alpha_{i,0} = \prod_{i \in I_j} \alpha'_{i,0}$  and  $\prod_{i \in I_j} \alpha_{i,1} = \prod_{i \in I_j} \alpha'_{i,1}$ .
2. Let  $m = 2L + w$ . For every  $i \in [n]$ , choose two block-diagonal  $m \times m$  matrices  $D_{i,0}, D_{i,1}$  where the diagonal entries  $1, \dots, 2L$  are chosen at random ( $\$$ ) and the bottom-right  $w \times w$  are the scaled  $A_{j,b}$ 's. Also choose two more  $m \times m$  matrices  $D'_{i,0}, D'_{i,1}$  where the diagonal entries  $1, \dots, 2L$  are random and the bottom-right  $w \times w$  are the scaled identity:

$$D_{i,b} \sim \begin{pmatrix} \$ & & & \\ & \ddots & & \\ & & \$ & \\ & & & \alpha_{i,b} A_{i,b} \end{pmatrix}, \quad D'_{i,b} \sim \begin{pmatrix} \$ & & & \\ & \ddots & & \\ & & \$ & \\ & & & \alpha'_{i,b} I \end{pmatrix}, \quad b \in \{0, 1\}.$$

3. Choose vectors  $\mathbf{s}$  and  $\mathbf{t}$ , and  $\mathbf{s}'$  and  $\mathbf{t}'$  of dimension  $m = 2L + w$  as follows:

$$\begin{aligned} \mathbf{s} &\sim (0 \dots 0 \$ \dots \$ \mathbf{s}^*) & \mathbf{t} &\sim (\$ \dots \$ 0 \dots 0 \mathbf{t}^*)^T \\ \mathbf{s}' &\sim (0 \dots 0 \$ \dots \$ \mathbf{s}'^*) & \mathbf{t}' &\sim (\$ \dots \$ 0 \dots 0 \mathbf{t}'^*)^T \end{aligned}$$

Here  $\mathbf{s}^*, \mathbf{t}^*, \mathbf{s}'^*, \mathbf{t}'^* \in \mathbb{Z}_p^w$  are uniform up to  $\langle \mathbf{s}^*, \mathbf{t}^* \rangle = \langle \mathbf{s}'^*, \mathbf{t}'^* \rangle$ , and  $0 \dots 0$  and  $\$ \dots \$$  are length- $L$  vectors of zeros and uniform elements of  $\mathbb{Z}_p$ , respectively.

4. Sample  $2(L + 1)$  uniform full-rank matrices  $R_0, \dots, R_L, R'_0, \dots, R'_L \in \mathbb{Z}_p^{m \times m}$ .
5. The randomized branching program over  $\mathbb{Z}_p$  is the following:

$$\mathcal{RN}\mathcal{D}_p(BP) = \left\{ \begin{aligned} &\tilde{\mathbf{s}} = \mathbf{s}R_0^{-1}, \quad \tilde{\mathbf{t}} = R_n \mathbf{t}, & \tilde{\mathbf{s}}' &= \mathbf{s}'(R'_0)^{-1}, \quad \tilde{\mathbf{t}}' = R'_n \mathbf{t}' \\ &\{\tilde{D}_{i,b} = R_{i-1} D_{i,b} R_i^{-1}\}_{i \in [L], b \in \{0,1\}}, & \{\tilde{D}'_{i,b} &= R'_{i-1} D'_{i,b} (R'_i)^{-1}\}_{i \in [L], b \in \{0,1\}} \end{aligned} \right\}$$

6. Finally, encode the randomized program using an  $(L + 2)$ -level asymmetric multilinear map scheme. Here we use the CLT13 scheme, choosing  $x_0 = \prod_{i=1}^n p_i$ , for equal-size primes  $p_i$ ,  $g = \text{CRT}(g_i)$  for small  $g_i \ll p_i$ 's, random denominators  $z_0, z_1, \dots, z_{L+1} \in \mathbb{Z}_{x_0}$  with  $z^* = [\prod_i z_i]_{x_0}$ , and an element  $h$  with mid-size CRT components, used for the zero-testing parameter  $\mathbf{p}_{zt} = [hz^*g^{-1}]_{x_0}$ .

Choose random small vectors  $\mathbf{r}_s \mathbf{r}'_s \mathbf{r}_t \mathbf{r}'_t$ , and random small matrices  $U_{i,b}$  and  $U'_{i,b}$ , and publish the zero-testing parameter  $\mathbf{p}_{zt}$  and the obfuscation

$$\mathcal{O}(BP) = \left\{ \begin{aligned} &\hat{\mathbf{s}} = [z_0^{-1}(\tilde{\mathbf{s}} + g\mathbf{r}_s)]_{x_0}, & \hat{\mathbf{t}} &= [z_{L+1}^{-1}(\tilde{\mathbf{t}} + g\mathbf{r}_t)]_{x_0}, \\ &\{\hat{D}_{i,b} = [z_i^{-1}(\tilde{D}_{i,b} + gU_{i,b})]_{x_0}\}_{i \in [L], b \in \{0,1\}}, & & \\ &\hat{\mathbf{s}}' = [z_0^{-1}(\tilde{\mathbf{s}}' + g\mathbf{r}'_s)]_{x_0}, & \hat{\mathbf{t}}' &= [z_{L+1}^{-1}(\tilde{\mathbf{t}}' + g\mathbf{r}'_t)]_{x_0}, \\ &\{\hat{D}'_{i,b} = [z_i^{-1}(\tilde{D}'_{i,b} + gU'_{i,b})]_{x_0}\}_{i \in [L], b \in \{0,1\}} \end{aligned} \right\}.$$

To evaluate  $\mathcal{O}(BP)(x)$ , compute  $y = \tilde{s} \left( \prod_{i=1}^L \tilde{D}_{i, \text{inp}(i)} \right) \tilde{t} - \tilde{s}' \left( \prod_{i=1}^L \tilde{D}'_{i, \text{inp}(i)} \right) \tilde{t}'$ , and output 1 if  $y$  encodes 0 (as determined by  $\mathbf{p}_{\text{zt}}$ ).

**Attack.** Our attack is applicable to branching programs with the following structure: there exists a partition of the input bits  $[\ell] = X_1 \cup X_2 \cup X_3$  and the branching program steps  $[L] = A \cup B \cup C$  such that  $A, B$  and  $C$  consist of consecutive steps in the branching program and  $\text{inp}(i) \in X_1 \forall i \in A, \text{inp}(i) \in X_2 \forall i \in B$  and  $\text{inp}(i) \in X_3 \forall i \in C$ . We consider a branching program  $BP$  of length  $L$  and input length  $\ell$ , computing the constant-1 function, that can be written as  $BP(x) = A(x_1) \circ B(x_2) \circ C(x_3)$ , where  $A(x_1), B(x_2)$ , and  $C(x_3)$  are branching programs over positions in the sets  $A, B$ , and  $C$  depending on inputs  $x_1, x_2$ , and  $x_3$ , respectively. We are given the obfuscation:

$$\mathcal{O}(BP) = \left( \mathbf{p}_{\text{zt}}, \hat{s}, \hat{t}, \hat{s}', \hat{t}', \{\hat{D}_{i,b}, \hat{D}'_{i,b}\}_{i \in [L], b \in \{0,1\}} \right).$$

*Attack Sets.* We construct the sets  $\mathcal{A}, \mathcal{B}$  and  $\mathcal{C}$  as follows. Let  $A(x) = \prod_{i \in A} D_{i, \text{inp}(i)}, A'(x) = \prod_{i \in A} D'_{i, \text{inp}(i)}$ . We define similarly  $B(x), B'(x)$  and  $C(x), C'(x)$ . We note that using  $\mathcal{O}$  we can compute  $R_0 A(x) R_{|A|}^{-1} = \prod_{i \in A} \tilde{D}_{i, \text{inp}(i)}$  and  $R_0 A'(x) R_{|A|}^{-1} = \prod_{i \in A} \tilde{D}'_{i, \text{inp}(i)}$ , and so on. Let  $\alpha_1, \dots, \alpha_{mn} \in \{0, 1\}^{|X_1|}$  be any set of distinct strings, and similarly for  $\beta_0, \beta_1 \in \{0, 1\}^{|X_2|}$  and  $\gamma_1, \dots, \gamma_{mn} \in \{0, 1\}^{|X_3|}$ . We set  $s = (\tilde{s}, \tilde{s}')$  and  $t = (\tilde{t}, -\tilde{t}') \mathbf{p}_{\text{zt}}$ , and define

$$\begin{aligned} \mathcal{A} &= \left\{ \tilde{A}_i = \begin{bmatrix} R_0 A(\alpha_i) R_{|A|}^{-1} & 0 \\ 0 & R_0 A'(\alpha_i) R_{|A|}^{-1} \end{bmatrix} \right\}_{i \in [(2L+w)n]} \\ \mathcal{B} &= \left\{ \tilde{B}_\sigma = \begin{bmatrix} R_{|A|} B(\beta_\sigma) R_{|A \cup B|}^{-1} & 0 \\ 0 & R_{|A|} B'(\beta_\sigma) R_{|A \cup B|}^{-1} \end{bmatrix} \right\}_{\sigma \in \{0,1\}} \\ \mathcal{C} &= \left\{ \tilde{C}_k = \begin{bmatrix} R_{|A \cup B|} C(\gamma_k) R_L^{-1} & 0 \\ 0 & R_{|A \cup B|} C'(\gamma_k) R_L^{-1} \end{bmatrix} \right\}_{k \in [(2L+w)n]} \end{aligned}$$

*Set Properties.* We consider the values

$$W_0[i, k] = s \times \tilde{A}_i \times \tilde{B}_0 \times \tilde{C}_k \times t = (s \times A_i \times B_0 \times C_k \times t - s' \times A'_i \times B'_0 \times C'_k \times t') \mathbf{p}_{\text{zt}}.$$

Since  $W_0[i, k]$  is a zero-tested encoding of zero by the definition of the obfuscated branching programs, the above equality holds not only mod  $x_0$  but also over the integers.  $W_1$  is constructed analogously.

The rest of the attack proceeds in the same manner as the attack on GGHZ encodings from Sect. 3.2. Let  $\mathbf{a}_i = (s \times A_i, s' \times A'_i)$  for  $i \in [(2m+w)n]$ ,  $\mathbf{c}_k = (C_k \times t \times \mathbf{p}_{\text{zt}}, -C'_k \times t' \times \mathbf{p}_{\text{zt}})$  for  $k \in [(2m+w)n]$  and  $X_0 = \begin{bmatrix} B_0 & 0 \\ 0 & B'_0 \end{bmatrix}$ . We set the matrix  $\hat{A}$  to have  $i$ -th row that is concatenations of the vectors  $\mathbf{a}_i \bmod p_j$  for

$j \in [n]$ , the matrix  $\hat{C}$  to have  $i$ -th column that is concatenation of  $\mathbf{c}_i^T \bmod p_j$  for  $j \in [n]$ , and the matrix  $\hat{B}_0$  to be a diagonal matrix with diagonal consisting of  $X_0 \bmod p_j$  for  $j \in [n]$ . Then we have that  $W_0 = \hat{A} \times \hat{B}_0 \times \hat{C}$ . We compute analogously  $W_1 = \hat{A} \times \hat{B}_1 \times \hat{C}$ . We use these matrices as in the attack on GGHZ encodings to break the underlying CLT13 encodings.

## 4 Conclusion

In this work we extended the recent CHLRS zeroizing attacks to many new settings, and also illustrated some of the limitations of this attack technique. The underlying message of recent attacks is that for current multilinear-map candidates, successful zero-tests give the adversary equations over the base ring (i.e. the integers or the ring of integers in a number field). Understanding the security of these candidates therefore hinges on a better understanding of which types of systems of nonlinear equations can be solved efficiently.

## References

1. Applebaum, B., Brakerski, Z.: Obfuscating circuits via composite-order graded encoding. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015, Part II. LNCS, vol. 9015, pp. 528–556. Springer, Heidelberg (2015). <http://eprint.iacr.org/2015/025>
2. Barak, B., Garg, S., Kalai, Y.T., Paneth, O., Sahai, A.: Protecting obfuscation against algebraic attacks. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 221–238. Springer, Heidelberg (2014). [http://dx.doi.org/10.1007/978-3-642-55220-5\\_13](http://dx.doi.org/10.1007/978-3-642-55220-5_13)
3. Boneh, D., Wu, D.J., Zimmerman, J.: Immunizing multilinear maps against zeroizing attacks. Cryptology ePrint Archive, Report 2014/930 (2014). <http://eprint.iacr.org/>
4. Brakerski, Z., Rothblum, G.N.: Virtual black-box obfuscation for all circuits via generic graded encoding. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 1–25. Springer, Heidelberg (2014)
5. Cheon, J.H., Han, K., Lee, C., Ryu, H., Stehlé, D.: Cryptanalysis of the multilinear map over the integers. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 3–12. Springer, Heidelberg (2015). <http://eprint.iacr.org/2014/906>
6. Coron, J.-S., Lepoint, T., Tibouchi, M.: Practical multilinear maps over the integers. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 476–493. Springer, Heidelberg (2013). [http://dx.doi.org/10.1007/978-3-642-40041-4\\_26](http://dx.doi.org/10.1007/978-3-642-40041-4_26)
7. Garg, S., Gentry, C., Halevi, S.: Candidate multilinear maps from ideal lattices. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 1–17. Springer, Heidelberg (2013). [http://dx.doi.org/10.1007/978-3-642-38348-9\\_1](http://dx.doi.org/10.1007/978-3-642-38348-9_1)
8. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: FOCS 2013, pp. 40–49. IEEE Computer Society (2013). <http://doi.ieeecomputersociety.org/10.1109/FOCS.2013.13>

9. Garg, S., Gentry, C., Halevi, S., Zhandry, M.: Fully secure functional encryption without obfuscation. Cryptology ePrint Archive, Report 2014/666 (2014). <http://eprint.iacr.org/>
10. Gentry, C., Lewko, A.B., Sahai, A., Waters, B.: Indistinguishability obfuscation from the multilinear subgroup elimination assumption. IACR Cryptology ePrint Archive 2014, 309 (2014). <http://eprint.iacr.org/2014/309>
11. Gentry, C., Lewko, A., Waters, B.: Witness encryption from instance independent assumptions. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 426–443. Springer, Heidelberg (2014). [http://dx.doi.org/10.1007/978-3-662-44371-2\\_24](http://dx.doi.org/10.1007/978-3-662-44371-2_24)
12. Goldwasser, S., Micali, S.: Probabilistic encryption. J. Comput. Syst. Sci. **28**(2), 270–299 (1984). [http://dx.doi.org/10.1016/0022-0000\(84\)90070-9](http://dx.doi.org/10.1016/0022-0000(84)90070-9)
13. Hansen, J.C., Schmutz, E.: How random is the characteristic polynomial of a random matrix? Math. Proc. Camb. Phi. Soc. **114**, 507–515 (1993)
14. Hu, Y., Jia, H.: Cryptanalysis of GGH map. Cryptology ePrint Archive, Report 2015/301 (2015). <http://eprint.iacr.org/>
15. Kuba, G.: On the distribution of reducible polynomials. Math. Slovaca **59**(3), 349–356 (2009)
16. Pass, R., Seth, K., Telang, S.: Indistinguishability obfuscation from semantically-secure multilinear encodings. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 500–517. Springer, Heidelberg (2014). [http://dx.doi.org/10.1007/978-3-662-44371-2\\_28](http://dx.doi.org/10.1007/978-3-662-44371-2_28)
17. Zimmerman, J.: How to obfuscate programs directly. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 439–467. Springer, Heidelberg (2015). <http://eprint.iacr.org/2014/776>

# New Multilinear Maps Over the Integers

Jean-Sébastien Coron<sup>1</sup>(✉), Tancrede Lepoint<sup>2</sup>, and Mehdi Tibouchi<sup>3</sup>

<sup>1</sup> University of Luxembourg, Luxembourg

`jean-sebastien.coron@uni.lu`

<sup>2</sup> CryptoExperts, Paris Cedex, France

`tancrede.lepoint@cryptoexperts.com`

<sup>3</sup> NTT Secure Platform Laboratories, Tokyo, Japan

`tibouchi.mehdi@lab.ntt.co.jp`

**Abstract.** In the last few years, cryptographic multilinear maps have proved their tremendous potential as building blocks for new constructions, in particular the first viable approach to general program obfuscation. After the first candidate construction by Garg, Gentry and Halevi (GGH) based on ideal lattices, a second construction over the integers was described by Coron, Lepoint and Tibouchi (CLT). However the CLT scheme was recently broken by Cheon et al.; the attack works by computing the eigenvalues of a diagonalizable matrix over  $\mathbb{Q}$  derived from the multilinear map.

In this paper we describe a new candidate multilinear map over the integers. Our construction is based on CLT but with a new arithmetic technique that makes the zero-testing element non-linear in the encoding, which prevents the Cheon et al. attack. Our new construction is relatively practical as its efficiency is comparable to the original CLT scheme. Moreover the subgroup membership and decisional linear assumptions appear to hold in the new setting.

## 1 Introduction

**Multilinear Maps.** Since the breakthrough construction of Garg, Gentry and Halevi [GGH13a], there has been a growing interest in *cryptographic multilinear maps*. They have spurred scores of new cryptographic applications. Chiefly among them is possibly the first proposed approach to general program obfuscation [GGH+13b]. Currently only three candidate constructions are known. Shortly after the first candidate construction of multilinear maps based on ideal lattices [GGH13a] (which we will refer to as GGH), Coron, Lepoint and Tibouchi proposed a second construction over the integers (CLT) using the same general paradigm [CLT13]. Recently, Gentry, Gorbunov and Halevi proposed another multilinear maps in which the map is defined with respect to a directed acyclic graph [GGH15].

A straightforward application of multilinear maps is multipartite Diffie-Hellman key exchange with  $\kappa + 1$  users, where  $\kappa$  is the maximum level of the multilinear map scheme. Initially each user publishes a level-1 encoding of a random element while keeping a level-0 encoding of the same element private.

Then each user can compute the product its level-0 by the product of the level-1 encodings of the other users. With  $\kappa + 1$  users this gives a level- $\kappa$  encoding from which the same secret value can be extracted by all users. The security of the protocol relies on a new hardness assumption which is a natural extension of the Decisional Diffie-Hellman assumption.

**The CLT Multilinear Map Over the Integers.** We recall the multilinear maps scheme over the integers from [CLT13]. One generates  $n$  secret primes  $p_i$  and publishes  $x_0 = \prod_{i=1}^n p_i$  (where  $n$  is large enough to ensure security); one also generates  $n$  small secret primes  $g_i$  and a random secret integer  $z$  modulo  $x_0$ . The message space is  $R = \mathbb{Z}_{g_1} \times \cdots \times \mathbb{Z}_{g_n}$ . A level- $k$  encoding of a vector  $m = (m_i) \in R$  is then an integer  $c$  such that for all  $1 \leq i \leq n$ :

$$c \equiv \frac{r_i \cdot g_i + m_i}{z^k} \pmod{p_i} \tag{1}$$

for some small random integers  $r_i$ ; the integer  $c$  is therefore defined modulo  $x_0$  by CRT. Encodings can then be added and multiplied modulo  $x_0$ , as long as the noise  $r_i$  is such that  $r_i \cdot g_i + m_i < p$  for each  $i$ . The multiplication of a level- $i$  encoding by a level- $j$  encoding gives an encoding at level  $i + j$ .

For level- $\kappa$  encodings one defines a zero-testing parameter  $p_{zt}$  with:

$$p_{zt} = \sum_{i=1}^n h_i \cdot (z^\kappa \cdot g_i^{-1} \pmod{p_i}) \cdot \frac{x_0}{p_i} \pmod{x_0}$$

for some small integers  $h_i$ . Given a level- $\kappa$  encoding  $c$  as in (1), as a zero-testing procedure one computes  $\omega = p_{zt} \cdot c \pmod{x_0}$  which gives:

$$\omega = \sum_{i=1}^n h_i \cdot (r_i + m_i \cdot (g_i^{-1} \pmod{p_i})) \cdot \frac{x_0}{p_i} \pmod{x_0}. \tag{2}$$

If  $m_i = 0$  for all  $i$ , since the  $r_i$ 's and  $h_i$ 's are small, we obtain that  $\omega$  is small compared to  $x_0$ ; this enables to test whether  $c$  is an encoding of  $\mathbf{0}$  or not. Moreover for non-zero encodings the leading bits of  $\omega$  only depend on the  $m_i$ 's and not on the noise  $r_i$ ; for level- $\kappa$  encodings this enables to extract a function of the  $m_i$ 's only, which eventually defines a degree- $\kappa$  multilinear map.

**Cheon et al. Attack.** The CLT scheme above was completely broken by a recent attack from Cheon, Han Lee, Ryu and Stehlé [CHL+15]; the attack runs in polynomial time, and recovers all secret parameters. The attack works by computing the eigenvalues of a diagonalizable matrix over  $\mathbb{Q}$  derived from the multilinear map. More precisely, when applying the zero-testing procedure to the product of two encodings  $x$  and  $x'$ , where  $x$  is an encoding of 0, the resulting  $\omega$  in (2) can be seen as a diagonal quadratic form over  $\mathbb{Z}$  in the CRT components  $x \pmod{p_i}$  and  $x' \pmod{p_i}$ . By computing the values  $\omega_{jk}$  of the quadratic form for  $n^2$  product pairs of encodings  $x_j \cdot x'_k$ , one can then recover the coefficients of the quadratic form using eigendecomposition, which reveals all the secret  $p_i$ 's and completely breaks the scheme. We recall the attack in more details in Sect. 3.



**Tentative Fixes.** Shortly after Cheon et al. attack, two independent approaches to fix the CLT scheme have been proposed on the Cryptology ePrint Archive, due to Garg, Gentry, Halevi and Zhandry on the one hand [GGHZ14, Sect. 7]<sup>1</sup>, and Boneh, Wu and Zimmerman on the other [BWZ14]. However, both countermeasures were shown to be insecure in [CLT14, CGH+15]. Indeed, although these countermeasures do not expose encodings of zero, the value  $\omega$  from the zero-testing procedure can still be expressed as a quadratic form in the CRT components of encodings. As a result, they can both be broken by a variant of the original Cheon et al. attack. Further extensions of the Cheon et al. attack along those lines are presented in [GHMS14, CGH+15].

**Our New Construction.** Our new construction keeps the same CLT encodings but departs from the two previous countermeasures by modifying the zero-testing procedure itself. Namely, we modify the definition of the zero-testing element  $p_{zt}$  so that  $\omega$  cannot be expressed as a quadratic form anymore. For this we use a new arithmetic technique that maps the  $n$  CRT components  $c \bmod p_i$  to some value modulo an independent integer  $N$ , so that the resulting  $\omega$  in the zero-testing procedure depends on the CRT components in a non-linear way, rather than linearly as in (2).

The technique works as follows. Consider a level- $\kappa$  encoding  $c$  as in (1); by the Chinese Remainder Theorem, we can write a relation of the form:

$$c = \sum_{i=1}^n (r_i + m_i \cdot (g_i^{-1} \bmod p_i)) \cdot u_i - a \cdot x_0 \tag{3}$$

over  $\mathbb{Z}$  for some  $a \in \mathbb{Z}$ , where the  $u_i$ 's are the CRT coefficients corresponding to the primes  $p_i$ 's, and scaled by  $g_i \cdot z^{-\kappa}$  for each  $i$ . Let  $N$  be a large integer and let  $p_{zt} \in \mathbb{Z}_N$ . For the zero-testing procedure we compute  $\omega = p_{zt} \cdot c \bmod N$  which gives from (3):

$$\omega \equiv \sum_{i=1}^n (r_i + m_i \cdot (g_i^{-1} \bmod p_i)) \cdot v_i - a \cdot v_0 \pmod{N} \tag{4}$$

where  $v_i := p_{zt} \cdot u_i \bmod N$  and  $v_0 := p_{zt} \cdot x_0 \bmod N$ . Assume now that we can generate  $p_{zt}$  and  $N$  such that all the  $v_i$ 's are small compared to  $N$ , including  $v_0$ . Now if  $m_i = 0$  for all  $i$ , since the  $r_i$ 's are small, the integer  $a$  in (3) is also small, which implies that  $\omega$  in (4) will also be small compared to  $N$ . This enables to test whether  $c$  is an encoding of 0 or not. As previously for level- $\kappa$  encodings one can then extract a function of the  $m_i$ 's only, which gives a degree- $\kappa$  multilinear map. We show that such an element  $p_{zt}$  can be efficiently generated for any large enough  $N$ , owing to the particular structure of the CRT coefficients  $u_i$ .

---

<sup>1</sup> We refer to the revised version of [GGHZ14] of November 12 2014, accessible on the Cryptology ePrint Archive.

**Security Analysis.** By comparing Eqs. (2) and (4), we see that the original CLT scheme is actually a particular case, with  $N = x_0$  and  $v_0 = 0$ . Therefore the main difference in the new scheme is that  $v_0 \neq 0$ , which causes the value  $\omega$  in (4) to depend on the integer  $a$  in (3). But that integer  $a$  depends on the CRT components  $r_i$  in a non-linear way. As a result, it is no longer true that the value  $\omega$  computed from encoding products  $x_j \cdot x'_k$  can be expressed as a quadratic form in the CRT components of  $x_j$  and  $x'_k$ , and the Cheon et al. attack is thus thwarted.

Another difference with the original CLT scheme is that we cannot publish  $x_0 = \prod_{i=1}^n p_i$  anymore. Namely for encodings of 0 we get a small  $\omega$  and therefore (4) holds over  $\mathbb{Z}$ . Therefore from  $x_0$  one could compute  $v_0 = p_{zt} \cdot x_0 \bmod N$  and apply the Cheon et al. attack modulo  $v_0$  instead of over  $\mathbb{Z}$ . It is not a problem to keep  $x_0$  private, however, as we can mimic the technique introduced by van Dijk et al. for their fully homomorphic encryption scheme over the integers [DGHV10] and approximate modular reduction by  $x_0$  with a ladder of encodings of zero of increasing sizes.

We provide a detailed security analysis of our new construction in Sect. 3 (for the Cheon et al. attack and its variants) and Sect. 4 (for lattice attacks). We also explain why the subgroup membership (SubM) and decisional linear (DLIN) problems, which are known to be easy in the GGH scheme [GGH13a], seem to be hard in our new setting.

**Implementation.** We describe an implementation of our scheme, with a few optimizations. Instead of using a ladder of encodings of 0 at every level, we publish a small multiple  $x'_0$  of  $x_0$  so that intermediate encodings can be reduced modulo  $x'_0$ ; only at the last level do we use a ladder of a few level- $\kappa$  encodings of 0. Additionally, to reduce the size of public parameters, we store only a small subset of the public elements needed for re-randomization and combine them pairwise to generate the full public parameters, as in [CLT13]; such an optimization was originally described in [GH11]. With these optimizations our scheme is relatively practical; for reasonable security parameters a multipartite Diffie-Hellman computation with 7 users requires about 30 seconds, with a public parameter size of roughly 6 GBytes; a proof-of-concept implementation is available at [Lep15].

## 2 New Multilinear Map Over the Integers

In this section we define our new multilinear scheme. Our scheme is actually a *graded encoding scheme* (GES) as in previous works [GGH13a, CLT13]; we recall the notion of GES in the full version of this paper [CLT15]. As explained in introduction, our new multilinear map scheme keeps the same CLT encodings as given by (1), with two main differences:

1. The zero-testing parameter  $\mathbf{p}_{zt}$  is computed differently, so that the CRT components modulo  $p_i$  of a level- $\kappa$  encoding  $c$  are mapped to some value

modulo an independent integer  $N$ , instead of modulo  $x_0$ . The resulting  $\omega$  in the zero-testing procedure then depends on those CRT components in a non-linear way, rather than linearly in the original CLT scheme, which prevents the Cheon et al. attack.

2. The integer  $x_0 = \prod_{i=1}^n p_i$  is kept private. For re-randomization, this implies that we must slightly modify the proof of statistical indistinguishability. To reduce the size of intermediate encodings back to the size of  $x_0$ , we publish a ladder of encodings of 0. In Sect. 5 we describe a simple optimization with a public multiple  $x'_0$  of  $x_0$ .

### 2.1 Scheme Description

**System Parameters.** The system parameters are similar to the original CLT scheme. One first defines the security parameter  $\lambda$  and the required multilinearity level  $\kappa \leq \text{poly}(\lambda)$ . Based on  $\lambda$  and  $\kappa$ , we choose:

- $n$ : the vector dimension
- $\eta$ : the bit-size of the primes  $p_i$
- $\alpha$ : the bit-size of the primes  $g_i$
- $\rho$ : the bit-size of the randomness used in encodings

and various other parameters that will be specified later. The constraints that these parameters must satisfy are described in Sect. 2.2. For integers  $z, p$  we denote the reduction of  $z$  modulo  $p$  by  $(z \bmod p)$  or  $[z]_p$  with  $-p/2 < [z]_p \leq p/2$ . For integers  $x_1, \dots, x_n$  we denote  $\text{CRT}_{p_1, \dots, p_n}(x_1, \dots, x_n)$  the unique integer  $x$  such that  $x \equiv x_i \pmod{p_i}$  for all  $1 \leq i \leq n$  and  $0 \leq x < \prod_{i=1}^n p_i$ .

As in the original CLT scheme a level- $k$  encoding of a vector  $\mathbf{m} = (m_i)$  is an integer  $c$  such that for all  $1 \leq i \leq n$ :

$$c \equiv \frac{r_i \cdot g_i + m_i}{z^k} \pmod{p_i} \tag{5}$$

where the  $r_i$ 's are  $\rho$ -bit random integers (specific to the encoding  $c$ ), with the following secret parameters: the  $p_i$ 's are random  $\eta$ -bit prime integers, the  $g_i$ 's are random  $\alpha$ -bit primes, and the denominator  $z$  is a random (invertible) integer modulo  $x_0 = \prod_{i=1}^n p_i$ . The integer  $c$  is therefore defined by CRT modulo  $x_0$ , but as opposed to the original CLT scheme,  $x_0$  is kept secret. We denote by  $\gamma$  the size of  $x_0$  in bits. As in the CLT scheme the domain is the ring  $R = \mathbb{Z}_{g_1} \times \dots \times \mathbb{Z}_{g_n}$ , so that for  $\mathbf{m} = (m_i) \in R$  the components  $m_i$  are defined modulo  $g_i$  for all  $1 \leq i \leq n$ .

**Instance Generation:**  $(\text{pp}, \mathbf{p}_{zt}) \leftarrow \text{instGen}(1^\lambda, 1^\kappa)$ . Instance generation is similar to [CLT13], except for the generation of  $\mathbf{p}_{zt}$ ; moreover  $x_0$  is kept private. We generate  $n$  secret random  $\eta$ -bit primes  $p_i$  and compute  $x_0 = \prod_{i=1}^n p_i$ . We generate a random invertible integer  $z$  modulo  $x_0$ . We generate  $n$  random  $\alpha$ -bit prime integers  $g_i$ , and various other parameters that will be specified later.

We publish the parameters  $(\text{pp}, \mathbf{p}_{zt})$  with

$$\text{pp} = \left( n, \eta, \alpha, \rho, \beta, \tau, \ell, \mu, y, \{x'_j\}_{j=1}^\ell, \{X_j^{(k)}\}, \{x_j\}_{j=1}^\tau, \{\Pi_j\}_{j=1}^{n+1}, s \right).$$

**Sampling Level-Zero Encodings:**  $c \leftarrow \text{samp}(\text{pp})$ . Since the primes  $p_i$ 's in (5) must remain secret, the user cannot encode a vector  $\mathbf{m} \in R$  by CRT directly from (5). Instead, as in [CLT13], a level-0 encoding  $c$  is generated as a random subset sum of random level-0 encodings  $x'_j$  from the public parameters. The only difference with [CLT13] is that the random subset-sum is computed over  $\mathbb{Z}$  instead of modulo  $x_0$ , since  $x_0$  is not public.

Therefore we publish as part as our instance generation a set of  $\ell$  integers  $x'_j$ , where each  $x'_j$  encodes at level-0 the column vector  $\mathbf{a}_j \in \mathbb{Z}^n$  of a secret matrix  $\mathbf{A} = (a_{ij}) \in \mathbb{Z}^{n \times \ell}$ , where each component  $a_{ij}$  is randomly generated in  $[0, g_i) \cap \mathbb{Z}$ . More precisely, using the CRT modulo  $x_0$  we generate integers  $x'_j$  such that:

$$1 \leq j \leq \ell, \quad x'_j \equiv r'_{ij} \cdot g_i + a_{ij} \pmod{p_i} \tag{6}$$

where the  $r'_{ij}$ 's are randomly generated in  $(-2^\rho, 2^\rho) \cap \mathbb{Z}$ .

To generate a level-0 encoding  $c$ , we first generate a random binary vector  $\mathbf{b} = (b_j) \in \{0, 1\}^\ell$  and output the level-0 encoding

$$c = \sum_{j=1}^{\ell} b_j \cdot x'_j.$$

From (6), this gives  $c \equiv (\sum_{j=1}^{\ell} r'_{ij} b_j) \cdot g_i + \sum_{j=1}^{\ell} a_{ij} b_j \pmod{p_i}$ ; as required the output  $c$  is a level-0 encoding:

$$c \equiv r_i \cdot g_i + m_i \pmod{p_i} \tag{7}$$

of some vector  $\mathbf{m} = \mathbf{A} \cdot \mathbf{b} \in R$  which is a random subset-sum of the column vectors  $\mathbf{a}_j$ . We note that for such level-0 encodings we get  $|r_i \cdot g_i + m_i| \leq \ell \cdot 2^{\rho + \alpha}$  for all  $i$ . As in [CLT13] by applying the leftover hash lemma over  $R = \mathbb{Z}_{g_1} \times \dots \times \mathbb{Z}_{g_n}$  the distribution of  $\mathbf{m}$  can be made statistically close to uniform over  $R$ .

**Lemma 1.** ([CLT13]). *Let  $c \leftarrow \text{samp}(\text{pp})$  and write  $c \equiv r_i \cdot g_i + m_i \pmod{p_i}$ . Assume  $\ell \geq n \cdot \alpha + 2\lambda$ . The distribution of  $(\text{pp}, \mathbf{m})$  is statistically close to the distribution of  $(\text{pp}, \mathbf{m}')$  where  $\mathbf{m}' \leftarrow R$ .*

As opposed to [CLT13] we cannot reduce  $c$  modulo  $x_0$ ; we only have the upper-bound  $|c| \leq \ell \cdot 2^\gamma$ , where  $\gamma$  is the size of  $x_0$  in bits. In the full version of this paper [CLT15], we show that instead of random sampling one can also publicly encode elements from the domain  $R$ , using a technique described in [BWZ14].

**Encoding at Higher Levels:**  $c_k \leftarrow \text{enc}(\text{pp}, k, c)$ . As in [CLT13], to allow encoding at higher levels, we publish as part of our instance-generation a level-one random encoding of  $\mathbf{1}$ , namely an integer  $y$  such that:

$$y \equiv \frac{r_i \cdot g_i + 1}{z} \pmod{p_i}$$

for random  $r_i \in (-2^\rho, 2^\rho) \cap \mathbb{Z}$ ; as previously the integer  $y$  is computed by CRT modulo  $x_0$ . Given a level-0 encoding  $c$  of  $\mathbf{m} \in R$  as given by (7), we can then compute a level-1 encoding of the same  $\mathbf{m}$  by computing over  $\mathbb{Z}$ :

$$c_1 = c \cdot y.$$

Namely we obtain as required:

$$c_1 \equiv \frac{r'_i \cdot g_i + m_i}{z} \pmod{p_i}$$

for some integers  $r'_i$ . From  $|c| \leq \ell \cdot 2^\gamma$ , we obtain  $|c_1| \leq \ell \cdot 2^{2\gamma}$ .

The difference with [CLT13] is that we cannot reduce  $c_1$  modulo  $x_0$ . Instead we provide a ladder of level-1 encodings of zero  $X_j^{(1)}$  of increasing size, so that the size of a level-1 encoding can be progressively reduced down to the size of  $x_0$ , as in the DGHV scheme [DGHV10, Sect. 3.3.1]. Specifically, for  $j = 0, \dots, \gamma + \lceil \log_2 \ell \rceil$ , we set:

$$X_j^{(1)} = \text{CRT}_{p_1, \dots, p_n} ([r_{1j} \cdot g_1 / z]_{p_1}, \dots, [r_{nj} \cdot g_n / z]_{p_n}) + q_j \cdot x_0$$

where  $r_{ij} \leftarrow (-2^\rho, 2^\rho) \cap \mathbb{Z}$  and  $q_j \leftarrow [2^{\gamma+j-1}/x_0, 2^{\gamma+j}/x_0) \cap \mathbb{Z}$ .

We can then iteratively reduce the size of  $c_1$  down to the size of  $x_0$ , first by  $X_{\gamma + \lceil \log_2 \ell \rceil}^{(1)}$  and eventually by  $X_0^{(1)}$ . Since the size reduction is done bit-by-bit, at each step some integer  $b_j \cdot X_j^{(1)}$  is subtracted from  $c_1$ , for  $b_j \in \{0, 1\}$ . Therefore the noise increases additively by at most  $(\gamma + \lceil \log_2 \ell \rceil + 1) \cdot 2^\rho$  in absolute value. After reduction, the resulting encoding  $\hat{c}_1$  will be such that

$$\hat{c}_1 \equiv (\hat{r}_i \cdot g_i + m_i) / z \pmod{p_i}, \tag{8}$$

with  $|\hat{r}_i \cdot g_i + m_i| \leq \ell \cdot 2^{\rho+\alpha} \cdot 2^{\rho+\alpha} + (\gamma + \lceil \log_2 \ell \rceil + 1) \cdot 2^\rho \leq 2\ell \cdot 2^{2\rho+2\alpha}$  for all  $i$ .

More generally to generate a level- $k$  encoding we compute  $c_k = c_0 \cdot y^k$ , and the size of  $c_k$  can be iteratively reduced after each multiplication by  $y$  using ladders of similarly designed level- $k$  encodings  $\{X_j^{(k')}\}_{j=0}^{\gamma + \lceil \log_2 \ell \rceil}$  for levels  $k' = 1, \dots, k$ .

**Re-randomization:**  $c' \leftarrow \text{reRand}(\text{pp}, k, \hat{c}_k)$ . Our re-randomization procedure is similar to [CLT13] except that again we cannot reduce the encodings modulo  $x_0$ . We describe the re-randomization of encodings at level  $k = 1$ ; the procedure can be easily adapted to randomize at level  $k > 1$ . We publish as part of our instance-generation a set of  $n + 1$  integers  $\Pi_j$ :

$$1 \leq j \leq n + 1, \quad \Pi_j = \sum_{i=1}^n \varpi_{ij} \cdot g_i \cdot u_i + \varpi_{n+1,j} \cdot x_0$$

where the  $u_i$ 's are appropriate CRT coefficients so that the  $\Pi_j$ 's are all level-1 random encodings of zero:

$$1 \leq j \leq n + 1, \quad \Pi_j \equiv \frac{\varpi_{ij} \cdot g_i}{z} \pmod{p_i}.$$

Namely, we let for all  $1 \leq i \leq n$ :

$$u_i := \left( z^{-1} \cdot \left( \frac{x_0}{p_i} \right)^{-1} \pmod{p_i} \right) \cdot \frac{x_0}{p_i} \tag{9}$$

The matrix  $\mathbf{\Pi} = (\varpi_{ij}) \in \mathbb{Z}^{(n+1) \times (n+1)}$  is a diagonally dominant matrix generated as follows: the non-diagonal entries are randomly and independently generated in  $(-2^\rho, 2^\rho) \cap \mathbb{Z}$ , while the diagonal entries are randomly generated in  $((n+1)2^\rho, (n+2)2^\rho) \cap \mathbb{Z}$ .

We also publish as part of our instance-generation a set of  $\tau$  integers  $x_j$ :

$$1 \leq j \leq \tau, \quad x_j = \sum_{i=1}^n r_{ij} \cdot g_i \cdot u_i + r_{n+1,j} \cdot x_0$$

so that each  $x_j$  is a level-1 random encoding of zero:

$$1 \leq j \leq \tau, \quad x_j \equiv \frac{r_{ij} \cdot g_i}{z} \pmod{p_i}$$

and where the column vectors of the matrix  $\mathbf{X} = (r_{ij}) \in \mathbb{Z}^{(n+1) \times \tau}$  are randomly and independently generated in the half-open parallelepiped spanned by the columns of the previous matrix  $\mathbf{\Pi}$ ; an algorithm to generate such  $r_i$ 's is described in [CLT13, Appendix E]; we obtain  $|r_{ij} \cdot g_i| \leq 3n2^{\rho+\alpha}$  for all  $i, j$ .

Given as input a (reduced) level-1 encoding  $\hat{c}_1$  as given by Eq. (8), we randomize  $\hat{c}_1$  with a random subset-sum of the  $x_j$ 's and a linear combination of the  $\Pi_j$ 's, over  $\mathbb{Z}$ :

$$c'_1 = \hat{c}_1 + \sum_{j=1}^{\tau} b_j \cdot x_j + \sum_{j=1}^{n+1} b'_j \cdot \Pi_j \tag{10}$$

where  $b_j \leftarrow \{0, 1\}$ , and  $b'_j \leftarrow [0, 2^\mu) \cap \mathbb{Z}$ , where  $\mu := \rho + \alpha + \lambda$ . The following Lemma shows that as required the distribution of  $c'_1$  is nearly independent of the input (as long as it encodes the same  $\mathbf{m}$ ). This essentially follows from the ‘‘leftover hash lemma over lattices’’ of [CLT13, Sect. 4.2]; the proof is given in the full version of this paper [CLT15].

**Lemma 2.** *Let the encodings  $c \leftarrow \text{samp}(\text{pp})$ ,  $\hat{c}_1 \leftarrow \text{enc}(\text{pp}, 1, c)$ , and  $c'_1$  as given by (10). Write  $c'_1 \equiv (r_i \cdot g_i + m_i)/z \pmod{p_i}$  for all  $1 \leq i \leq n$  and  $r_{n+1} = (c'_1 - \sum r_i \cdot g_i \cdot u_i)/x_0$ , and define  $\mathbf{r} = (r_1, \dots, r_n, r_{n+1})^T$ . If  $2(\rho + \alpha + \lambda) \leq \eta$  and  $\tau \geq (n+2) \cdot \rho + 2\lambda$ , then the distribution of  $(\text{pp}, \mathbf{r})$  is statistically close to that of  $(\text{pp}, \mathbf{r}')$ , where  $\mathbf{r}' \in \mathbb{Z}^{n+1}$  is randomly generated in the half-open parallelepiped spanned by the column vectors of  $2^\mu \mathbf{\Pi}$ . Moreover we have  $|r_i \cdot g_i + m_i| \leq 4n^2 \cdot 2^{2\rho+2\alpha+\lambda}$  for all  $1 \leq i \leq n$ .*

Finally, we can reduce the size of  $c'_1$  down to the size of  $x_0$  using the ladder  $\{X_j^{(1)}\}$ , and we obtain an encoding  $\tilde{c}'_1$ . Writing  $\tilde{c}'_1 \equiv (\hat{r}'_i \cdot g_i + m_i)/z \pmod{p_i}$ , we obtain

$$|\hat{r}'_i \cdot g_i + m_i| \leq 4n^2 \cdot 2^{2\rho+2\alpha+\lambda} + (\gamma + \lceil \log_2 \ell \rceil + 1) \cdot 2^\rho \leq 5n^2 \cdot 2^{2\rho+2\alpha+\lambda}.$$

**Adding, Negating and Multiplying Encodings.** As in [CLT13] we can add, negate and multiply encodings. The difference is that we do those operations over  $\mathbb{Z}$  instead of modulo  $x_0$ . More precisely, given level-one encodings  $v_j$  of vectors  $\mathbf{m}_j \in \mathbb{Z}^n$  for  $1 \leq j \leq \kappa$ , with  $v_j \equiv (r_{ij} \cdot g_i + m_{ij})/z \pmod{p_i}$ , we compute over  $\mathbb{Z}$ :

$$v = \prod_{j=1}^{\kappa} v_j.$$

This gives:

$$v \equiv \frac{\prod_{j=1}^{\kappa} (r_{ij} \cdot g_i + m_{ij})}{z^{\kappa}} \equiv \frac{r_i \cdot g_i + \left(\prod_{j=1}^{\kappa} m_{ij}\right) \pmod{g_i}}{z^{\kappa}} \pmod{p_i}$$

for some integers  $r_i \in \mathbb{Z}$ . Hence we obtain a level- $\kappa$  encoding of the vector  $\mathbf{m}$  obtained by componentwise product of the vectors  $\mathbf{m}_j$ , as long as the components do not wrap modulo  $p_i$ , that is  $\prod_{j=1}^{\kappa} (r_{ij} \cdot g_i + m_{ij}) < p_i$  for all  $i$ . Then, using the ladder  $X_j^{(\kappa)}$  one can reduce its size down to the size of  $x_0$ , at the cost of an additive increase in absolute value of the noise.

In multipartite Diffie-Hellman key exchange we compute the product of  $\kappa$  level-1 encodings from `reRand` and one level-0 encoding from `samp`, which gives from previous bounds for all  $i$ :

$$|r_i| \leq (6n^2 2^{2\rho+2\alpha+\lambda})^{\kappa} \cdot \ell \cdot 2^{\rho+1}$$

In Sect. 5 we describe an optimization in which we publish a multiple  $x'_0$  of  $x_0$ ; then all intermediate encodings can be reduced modulo  $x'_0$ , instead of using a ladder of encodings of zero; only at the last stage do we need a ladder of a few level- $\kappa$  encodings of zero.

**Zero Testing.** `isZero(pp,  $p_{zt}$ ,  $c$ )`  $\stackrel{?}{=} 0/1$ . To prevent the Cheon et al. attack, we keep the same encoding as in (1) but we compute the  $p_{zt}$  differently; this is the most important difference. Let  $c$  be a level- $\kappa$  encoding. We assume  $0 \leq c < x_0$ , as a result of approximate modular reduction using a ladder of level- $\kappa$  encodings of 0. From (5) we can write by CRT:

$$\begin{aligned} c &\equiv \sum_{i=1}^n \left( \frac{r_i \cdot g_i + m_i}{z^{\kappa}} \pmod{p_i} \right) \cdot \left( \left( \frac{x_0}{p_i} \right)^{-1} \pmod{p_i} \right) \cdot \frac{x_0}{p_i} \pmod{x_0} \\ c &\equiv \sum_{i=1}^n (r_i + m_i \cdot g_i^{-1} \pmod{p_i}) \cdot \left( g_i \cdot z^{-\kappa} \cdot \left( \frac{x_0}{p_i} \right)^{-1} \pmod{p_i} \right) \cdot \frac{x_0}{p_i} \pmod{x_0} \end{aligned}$$

Therefore we can write over the integers:

$$c = \sum_{i=1}^n (r_i + m_i \cdot g_i^{-1} \pmod{p_i}) \cdot u'_i - a \cdot x_0 \tag{11}$$

for some integer  $a$ , where the  $u'_i$ 's are the scaled CRT coefficients:

$$u'_i = \left( g_i \cdot z^{-\kappa} \cdot \left( \frac{x_0}{p_i} \right)^{-1} \pmod{p_i} \right) \cdot \frac{x_0}{p_i} \tag{12}$$

We generate a random prime integer  $N$  of size  $\gamma + 2\eta + 1$  bits. Using LLL in dimension 2, we obtain<sup>2</sup> pairs of nonzero integers  $(\alpha_i, \beta_i)$  satisfying:

$$|\alpha_i| < 2^{\eta-1} \quad |\beta_i| \leq \frac{4}{3} \cdot \frac{N}{2^{\eta-1}} < 2^{2-\eta} \cdot N \quad \beta_i \equiv \alpha_i \cdot (u'_i/p_i) \pmod{N}.$$

We also generate as in [CLT13] an integer matrix  $\mathbf{H} = (h_{ij}) \in \mathbb{Z}^{n \times n}$  such that  $\mathbf{H}$  is invertible in  $\mathbb{Z}$  and both  $\|\mathbf{H}^T\|_\infty \leq 2^\beta$  and  $\|(\mathbf{H}^{-1})^T\|_\infty \leq 2^\beta$ , for some parameter  $\beta$  specified later; here  $\|\cdot\|_\infty$  is the operator norm on  $n \times n$  matrices with respect to the  $\ell^\infty$  norm on  $\mathbb{R}^n$ . A technique for generating such  $\mathbf{H}$  is discussed in the full version of this paper [CLT15]. We then publish as part of our instance generation the following zero-testing vector  $\mathbf{p}_{zt} \in \mathbb{Z}^n$ :

$$(\mathbf{p}_{zt})_j = \sum_{i=1}^n h_{ij} \cdot \alpha_i \cdot p_i^{-1} \pmod{N} \tag{13}$$

To determine whether a level- $\kappa$  encoding  $c$  is an encoding of zero or not, we compute the vector  $\boldsymbol{\omega} = c \cdot \mathbf{p}_{zt} \pmod{N}$  and test whether  $\|\boldsymbol{\omega}\|_\infty$  is small:

$$\text{isZero}(\text{pp}, \mathbf{p}_{zt}, c) = \begin{cases} 1 & \text{if } \|c \cdot \mathbf{p}_{zt} \pmod{N}\|_\infty < N \cdot 2^{-\nu} \\ 0 & \text{otherwise} \end{cases}$$

for some parameter  $\nu$  specified later.

Namely for a level- $\kappa$  ciphertext  $c$  we obtain from (11):

$$\begin{aligned} (\boldsymbol{\omega})_j &= (c \cdot \mathbf{p}_{zt} \pmod{N})_j = \sum_{i=1}^n h_{ij} \cdot \alpha_i \cdot p_i^{-1} \cdot c \pmod{N} \\ &= \sum_{i=1}^n h_{ij} \cdot \alpha_i \cdot p_i^{-1} \cdot \left( \sum_{k=1}^n (r_k + m_k \cdot g_k^{-1} \pmod{p_k}) \cdot u'_k - a \cdot x_0 \right) \pmod{N} \end{aligned}$$

which gives:

$$\begin{aligned} (\boldsymbol{\omega})_j &= \sum_{i=1}^n h_{ij} \cdot \left( (r_i + m_i \cdot g_i^{-1} \pmod{p_i}) \cdot \beta_i \right. \\ &\quad \left. + \alpha_i \cdot \sum_{k=1, k \neq i}^n (r_k + m_k \cdot g_k^{-1} \pmod{p_k}) \cdot \frac{u'_k}{p_i} - a \cdot \alpha_i \cdot \frac{x_0}{p_i} \right) \pmod{N} \tag{14} \end{aligned}$$

<sup>2</sup> More precisely, we apply Legendre reduction to the 2-dimensional lattice generated by the rows of  $\begin{pmatrix} \lceil N/B^2 \rceil & u'_i/p_i \pmod{N} \\ 0 & N \end{pmatrix}$ , where  $B = (3/4)^{1/4} 2^{\eta-1}$ . The shortest vector is of the form  $(\alpha_i \lceil N/B^2 \rceil, \beta_i)$ .



Recall that  $\alpha_i$  is at most  $\eta - 1$  bits, therefore  $\alpha_i \cdot u'_k/p_i$  has size at most  $\eta - 1 + \gamma - (\eta - 1) = \gamma$  bits; the integer  $\alpha_i \cdot x_0/p_i$  has size also at most  $\gamma$  bits; moreover  $\beta_i$  is at most  $|N| - \eta + 1$  bits. Therefore in Eq. (14) the integers  $\beta_i$ ,  $\alpha_i \cdot u_k/p_i$  and  $\alpha_i \cdot x_0/p_i$  are all small compared to  $N$ . This implies that if  $m_i = 0$  for all  $1 \leq i \leq n$ , then  $\omega_j$  will be small compared to  $N$ , when the  $r_i$ 's are small enough, i.e. a limited number of additions/multiplications on encodings has been performed. Conversely if  $m_i \neq 0$  for some  $i$  we show that  $\|\omega\|_\infty$  must be large. This shows the correctness of our zero-testing procedure. More precisely we prove the following lemma in the full version of this paper [CLT15].

**Lemma 3.** *Let  $n, \eta, \alpha$  and  $\beta$  be as in our parameter setting. Let  $\rho_f$  be such that  $\alpha + \log_2 n < \rho_f \leq \eta - 2\beta - 2\alpha - \lambda - 8$ , and let  $\nu = \eta - \rho_f - \beta - \lambda - 3 \geq 2\alpha + \beta + 5$ . Let  $c$  be such that  $c \equiv (r_i \cdot g_i + m_i)/z^\kappa \pmod{p_i}$  for all  $1 \leq i \leq n$ , where  $0 \leq m_i < g_i$  for all  $i$ . Let  $\mathbf{r} = (r_i)_{1 \leq i \leq n}$  and assume that  $\|\mathbf{r}\|_\infty < 2^{\rho_f}$ . If  $\mathbf{m} = 0$  then  $\|\omega\|_\infty < 2^{-\nu-\lambda} \cdot N$ . Conversely if  $\mathbf{m} \neq 0$  then  $\|\omega\|_\infty > 2^{-\nu+2} \cdot N$ .*

**Extraction.**  $sk \leftarrow \text{ext}(\text{pp}, \mathbf{p}_{zt}, u_\kappa)$ . This part is essentially the same as in [GGH13a]. To extract a random function of the vector  $\mathbf{m}$  encoded in a level- $\kappa$  encoding  $c$ , we multiply  $c$  by the zero-testing parameter  $\mathbf{p}_{zt}$  modulo  $N$ , collect the  $\nu$  most significant bits of each of the  $n$  components of the resulting vector, and apply a strong randomness extractor (using the seed  $s$  from  $\text{pp}$ ):

$$\text{ext}(\text{pp}, \mathbf{p}_{zt}, c) = \text{Extract}_s(\text{msbs}_\nu(c \cdot \mathbf{p}_{zt} \bmod N))$$

where  $\text{msbs}_\nu$  extracts the  $\nu$  most significant bits of the result.

Namely if two encodings  $c$  and  $c'$  encode the same  $\mathbf{m} \in \mathbb{Z}^n$  then from Lemma 3 we have  $\|(c - c') \cdot \mathbf{p}_{zt} \bmod N\|_\infty < N \cdot 2^{-\nu-\lambda}$ , and therefore we expect that  $\omega = c \cdot \mathbf{p}_{zt} \bmod N$  and  $\omega' = c' \cdot \mathbf{p}_{zt} \bmod N$  agree on their  $\nu$  most significant bits, and therefore extract to the same value.

Conversely if  $c$  and  $c'$  encode different vectors then by Lemma 3 we must have  $\|(c - c') \cdot \mathbf{p}_{zt} \bmod N\|_\infty > N \cdot 2^{-\nu+2}$ , and therefore the  $\nu$  most significant bits of the corresponding  $\omega$  and  $\omega'$  must be different. This implies that for random  $\mathbf{m} \in R = \mathbb{Z}_{g_1} \times \cdots \times \mathbb{Z}_{g_n}$  the min-entropy of  $\text{msbs}_\nu(c \cdot \mathbf{p}_{zt} \bmod N)$  when  $c$  encodes  $\mathbf{m}$  is at least  $\log_2 |R| \geq n(\alpha - 1)$ . Therefore we can use a strong randomness extractor to extract a nearly uniform bit-string of length  $\lfloor \log_2 |R| \rfloor - \lambda$ .

This concludes the description of our new multilinear encoding scheme.

*Remark 1.* By comparing Eqs. (2) and (4) we see that the original CLT scheme is a particular case with  $N = x_0$  and  $\alpha_i = 0$  for all  $1 \leq i \leq n$ . Therefore the main difference of our construction is that it incorporates the additional term  $a$ , which depends on the  $r_i$ 's in a non-linear way; this is to prevent the Cheon et al. attack (see Sect. 3).

## 2.2 Setting the Parameters

The constraints on the system parameters are similar to [CLT13].

- The bit-size  $\rho$  of the randomness used for encodings must satisfy  $\rho = \Omega(\lambda)$  to avoid brute force attack on the noise. The improved attacks from [CN12] and [LS14] both have complexity  $\tilde{O}(2^{\rho/2})$ , but with a large overhead, so in practice we can take  $\rho = \lambda$ .
- The bit-size  $\alpha$  of the primes  $g_i$  must be large enough so that the order of the group  $R = \mathbb{Z}_{g_1} \times \cdots \times \mathbb{Z}_{g_n}$  does not contain small prime factors (see the full version of this paper [CLT15]). One can take  $\alpha = \lambda$ .
- The parameter  $n$  must be large enough to thwart lattice-based attacks on the encodings, namely  $n = \omega(\eta \log \lambda)$ ; see Sect. 4.
- The number  $\ell$  of level-0 encodings  $x'_j$  for **samp** must satisfy  $\ell \geq n \cdot \alpha + 2\lambda$  in order to apply the leftover hash lemma; see Lemma 1.
- The number  $\tau$  of level-1 encodings  $x_j$  must satisfy  $\tau \geq (n+2) \cdot \rho + 2\lambda$  in order to apply the leftover hash lemma over lattices; see Lemma 2.
- As a conservative security precaution, we take  $\beta = 3\lambda$  (see the full version of this paper [CLT15]).
- The bit-size  $\eta$  of the primes  $p_i$  must satisfy  $\eta \geq \rho_f + 2\alpha + 2\beta + \lambda + 8$ , where  $\rho_f$  is the maximum bit size of the randoms  $r_i$  a level- $\kappa$  encoding (see Lemma 3). When computing the product of  $\kappa$  level-1 encodings and an additional level-0 encoding (as in a multipartite Diffie-Hellman key exchange with  $\kappa + 1$  users), one obtains  $\rho_f = \kappa \cdot (2\rho + 2\alpha + \lambda + 2 \log_2 n + 3) + \rho + \log_2 \ell + 1$  (see previous Section).
- We set  $\nu = \eta - \rho_f - \lambda - \beta - 3$  for the number of most significant bits to extract (see Lemma 3).

### 2.3 Security of Our Construction

As in the original CLT scheme [CLT13] and in the GGH scheme [GGH13a] the security of our construction does not seem to be reducible to more classical assumptions, such as for example the Approximate-GCD problem. To prove the security of the one-round  $(\kappa + 1)$ -way Diffie-Hellman key exchange protocol, as in [GGH13a] one must therefore make the assumption that solving the Graded DDH problem (GDDH) is hard in our scheme; see the full version of this paper [CLT15].

## 3 Cheon et al. Attack

The goal of this section is to argue that the Cheon et al. attack [CHL+15] is prevented in our new construction.

### 3.1 Attack Description

We first recall the Cheon et al. attack against the original CLT scheme. This attack makes use of low-level encodings of 0: if such encodings are made public, one can recover in polynomial time all secret parameters. In the CLT scheme

such encodings of 0 are used for the rerandomization procedure, therefore the Cheon et al. attack leads to a complete break of CLT.

In the following we describe a slight simplification of [CHL+15] in which only a single ciphertext  $c$  is used instead of two ciphertexts  $c_0$  and  $c_1$ ; this enables to obtain as eigenvalues directly the CRT components of  $c$ , instead of the ratios of the CRT components of  $c_0$  and  $c_1$ . For simplicity we assume  $\kappa = 2$ ; the attack is easily extended to any  $\kappa > 2$ . Let  $c$  be a level-0 encoding with  $c \equiv c_i \pmod{p_i}$ . Let  $x$  be a level-1 encoding with  $x \equiv x_i/z \pmod{p_i}$ , and let  $x'$  be a level-1 encoding of 0 with  $x' \equiv r'_i \cdot g_i/z \pmod{p_i}$ . Let  $c'$  be the level- $\kappa$  product encoding

$$c' = x \cdot c \cdot x' \pmod{x_0}$$

From  $c' \equiv x_i \cdot c_i \cdot r'_i \cdot g_i \cdot z^{-2} \pmod{p_i}$ , we obtain by CRT:

$$c' \equiv \sum_{i=1}^n x_i \cdot c_i \cdot r'_i \cdot u_i \pmod{x_0} \tag{15}$$

with the CRT coefficients:

$$u_i = \left( g_i \cdot z^{-2} \cdot \left( \frac{x_0}{p_i} \right)^{-1} \pmod{p_i} \right) \cdot \frac{x_0}{p_i}$$

In the original CLT scheme, the zero-testing parameter  $p_{zt}$  is given by

$$p_{zt} = \sum_{i=1}^n h_i \cdot (z^2 \cdot g_i^{-1} \pmod{p_i}) \cdot \frac{x_0}{p_i} \pmod{x_0}$$

Using  $p_{zt} \cdot u_i \equiv h_i \cdot x_0/p_i \pmod{x_0}$  for all  $1 \leq i \leq n$ , we obtain from (15):

$$\omega = [p_{zt} \cdot c']_{x_0} = \sum_{i=1}^n x_i \cdot c_i \cdot r'_i \cdot h_i \cdot x_0/p_i \tag{16}$$

where the last equality holds over  $\mathbb{Z}$  because  $c'$  is an encoding of 0.

More generally, let  $x_j$  be level-1 encodings with  $x_j \equiv x_{ij}/z \pmod{p_i}$ , and let  $x'_k$  be a level-1 encodings of 0 with  $x'_k \equiv r'_{ik} \cdot g_i/z \pmod{p_i}$ . One can therefore compute for  $1 \leq j, k \leq n$ :

$$\omega_{jk} = [(x_j \cdot c \cdot x'_k) \cdot p_{zt}]_{x_0} \tag{17}$$

which gives as previously:

$$\omega_{jk} = \sum_{i=1}^n x_{ij} \cdot c_i \cdot r'_{ik} \cdot h_i \cdot x_0/p_i \tag{18}$$

over the integers. We note that  $\omega_{jk}$  is a diagonal quadratic form over  $\mathbb{Z}$  in the  $x_{ij}$ 's and the  $r'_{ik}$ 's. By spanning  $1 \leq j, k \leq n$ , one can construct a matrix  $\mathbf{W}_c = (\omega_{jk})_{1 \leq j, k \leq n}$  such that

$$\mathbf{W}_c = \mathbf{X} \times \mathbf{C} \times \mathbf{R}, \tag{19}$$

where  $\mathbf{C} = \text{diag}(c_1, c_2, \dots, c_n)$ ,  $\mathbf{X} = (x_{ij} \cdot h_i \cdot x_0 / p_i)_{1 \leq j, i \leq n}$  and  $\mathbf{R} = (r'_{ik})_{1 \leq i, k \leq n}$ .

We perform the same computation with  $c = 1$  in (17); one can therefore compute a matrix  $\mathbf{W}_1$  such that  $\mathbf{W}_1 = \mathbf{X} \times \mathbf{I} \times \mathbf{R}$ , where  $\mathbf{I}$  is the  $n \times n$  identity matrix. Finally, one can publicly compute:

$$\mathbf{W} = \mathbf{W}_c \cdot \mathbf{W}_1^{-1} = \mathbf{X} \times \mathbf{C} \times \mathbf{X}^{-1}.$$

Since  $\mathbf{C}$  is a diagonal matrix, by computing the eigenvalues of  $\mathbf{W}$  one can recover the  $c_i$ 's, and then the  $p_i$ 's. Finally, Cheon et al. describe how to recover all the other secret values in [CHL+15].

**Extension.** A similar attack applies against two independent approaches to fix the CLT scheme, [GGHZ14, Sect. 7] and [BWZ14], proposed shortly after the Cheon et al. attack. Namely, although the two countermeasures do not expose encodings of zero, the value  $\omega$  from the zero-testing procedure can still be expressed as a diagonal quadratic form in the CRT components of encodings, as in Eq. (18), hence the two countermeasure can be broken by the same technique; we refer to [CLT14] for a description of the modified attacks.

### 3.2 Non-applicability of Cheon et al. Attack

In this section we explain why the above attack does not apply against our new scheme. As previously we let  $x$  be a level-1 encoding with  $x \equiv x_i / z \pmod{p_i}$ , and let  $x'$  be a level-1 encoding of 0 with  $x' \equiv r'_i \cdot g_i / z \pmod{p_i}$ . We consider as previously the level- $\kappa$  product encoding, with  $\kappa = 2$ :

$$c' = x \cdot c \cdot x'$$

Here we cannot reduce  $c'$  modulo  $x_0$  since  $x_0$  is kept private; instead we must use a ladder of level-2 encodings of zero. Let  $c''$  be the resulting encoding, with  $0 \leq c'' < x_0$ ; we obtain:

$$c'' \equiv c' + \frac{s_i \cdot g_i}{z^2} \pmod{p_i}$$

for some integers  $s_i$  of size roughly  $\rho$  bits. Therefore instead of (15) we obtain over the integers:

$$c'' = \sum_{i=1}^n (x_i \cdot c_i \cdot r'_i + s_i) \cdot u_i - a \cdot x_0 \tag{20}$$

for some integer  $a$ . Using the new definition of  $p_{zt} \in \mathbb{Z}_N$ , and letting  $v_i = p_{zt} \cdot u_i \pmod N$  for all  $1 \leq i \leq n$  and  $v_0 = p_{zt} \cdot x_0 \pmod N$ , we obtain from (20):

$$\omega = [p_{zt} \cdot c'']_N = \sum_{i=1}^n (x_i \cdot c_i \cdot r'_i + s_i) \cdot v_i - a \cdot v_0 \tag{21}$$

where as previously the last equality holds over  $\mathbb{Z}$ .

Now comparing equalities (16) and (21), we see that we obtain two additional terms: the  $s_i$ 's and the integer  $a$ . The  $s_i$ 's come from reducing  $c'$  with the ladder of level- $\kappa$  encodings of 0, so that eventually  $0 \leq c'' < x_0$ ; therefore the  $s_i$ 's depend on  $x \cdot c \cdot x'$  in a non-linear way. Similarly the integer  $a$  in (21), which is the quotient of the division of  $\sum_{i=1}^n (x_i \cdot c_i \cdot r'_i + s_i) \cdot u_i$  by  $x_0$ , depends on the  $x_i \cdot c_i \cdot x'_i$  in a non-linear way. Therefore, if we apply Cheon et al. attack, we do not obtain a quadratic form as in (18) anymore.

More precisely, we can let as previously  $x_j$  be level-1 encodings with  $x_j \equiv x_{ij}/z \pmod{p_i}$ , and let  $x'_k$  be a level-1 encodings of 0 with  $x'_k \equiv r'_{ik} \cdot g_i/z \pmod{p_i}$ . As previously for all  $1 \leq j, k \leq n$ , we can compute the product encodings  $c'_{jk} = x_j \cdot c \cdot x'_k$  and we let  $c''_{jk}$  be the encodings obtained after reducing  $c'_{jk}$  such that  $0 \leq c''_{jk} < x_0$ , using the ladder of level- $\kappa$  encodings of zero. This gives:

$$\omega_{jk} = [p_{zt} \cdot c''_{jk}]_N = \sum_{i=1}^n (x_{ij} \cdot c_i \cdot r'_{ik} + s_{ijk}) \cdot v_i - a_{jk} \cdot v_0 \tag{22}$$

for integers  $s_{ijk}$  and  $a_{jk}$ . Compared to (18), we see that the previous equation has two additional terms  $s_{ijk}$  and  $a_{jk}$ . As previously we can write:

$$\mathbf{W}_c = \mathbf{X} \times \mathbf{C} \times \mathbf{R} + \mathbf{S} - \mathbf{A} \cdot v_0 \tag{23}$$

for some matrices  $\mathbf{S}$  and  $\mathbf{A}$ . However we see that the previous attack does not apply, because of the additional terms  $\mathbf{S}$  and  $\mathbf{A} \cdot v_0$ . Namely if as previously we perform the same computation with  $c = 1$ , we obtain:

$$\mathbf{W}_1 = \mathbf{X} \times \mathbf{I} \times \mathbf{R} + \mathbf{S}' - \mathbf{A}' \cdot v_0 \tag{24}$$

but as opposed to the CLT scheme we cannot get a simple expression for  $\mathbf{W} = \mathbf{W}_c \times \mathbf{W}_1^{-1}$ . More generally, as opposed to the CLT case, it seems difficult to extract useful information about  $\mathbf{C}$  from the matrices  $\mathbf{W}_c$  and  $\mathbf{W}_1$ , since in Eqs. (23) and (24) all terms  $\mathbf{X}$ ,  $\mathbf{R}$ ,  $\mathbf{S}$ ,  $\mathbf{S}'$ ,  $\mathbf{A}$ ,  $\mathbf{A}'$  and  $v_0$  are unknown.

*Remark 2.* If we do not reduce  $c'_{jk}$  with the ladder of encodings, the  $s_{ijk}$  terms disappear but the integers  $a_{jk}$  becomes too large and (22) does not hold over  $\mathbb{Z}$  anymore. The equation still holds modulo  $N$ , however there is still the additional term  $a_{jk}$  that prevents the Cheon et al. attack.

### 3.3 Attack with Known $x_0$

In this section we describe an extension of the Cheon et al. attack against our scheme when  $x_0$  is known; this explains why  $x_0$  must be kept secret in our scheme.

When  $x_0$  is known, we can reduce the previous ciphertexts  $c'_{jk}$  modulo  $x_0$ , and therefore the  $s_{ijk}$  terms in (22) disappear. Moreover  $v_0 = [p_{zt} \cdot x_0]_N$  is known. Therefore we can compute the  $\mathbf{W}_c$  matrix as previously, and we obtain from (23) with  $\mathbf{S} = 0$ :

$$\mathbf{W}_c = \mathbf{X} \times \mathbf{C} \times \mathbf{R} \pmod{v_0}$$

which is the same equation as (19) in the original attack except that it holds modulo  $v_0$  instead of over  $\mathbb{Z}$ .

Therefore we can apply the Cheon et al. attack modulo  $v_0$  instead of over  $\mathbb{Z}$ . If  $v_0$  is prime, one can recover the eigenvalues of  $\mathbf{W} = \mathbf{W}_c \cdot \mathbf{W}_1^{-1} \bmod v_0$  by factoring the characteristic polynomial modulo  $v_0$ , which reveals the  $c_i$ 's as previously. If a prime  $p$  can be extracted from  $v_0$ , one can still apply the attack modulo  $p$  and recover the  $c_i$ 's modulo  $p$ ; for large enough  $p$  this reveals the  $c_i$ 's; alternatively for sufficiently many such primes  $p$ , the  $c_i$ 's could be recovered by CRT.

Actually the attack also works even if  $v_0$  is hard to factor and no prime can be extracted. Namely the eigenvalues  $c_i$ 's are small, so to recover the roots of the characteristic polynomial one can use Coppersmith's first theorem for finding small roots of polynomial equations modulo an integer of unknown factorization [Cop97]. Namely Coppersmith's bound applies: with a modulus  $v_0$  of size roughly  $\gamma$  bits and a characteristic polynomial of degree  $n$ , the roots have size only roughly  $\rho$  bits, with  $\rho \ll \eta \simeq \gamma/n$ .

### 3.4 Attack for Small Multiple of $x_0$

In Sect. 5 we describe an optimization with a known multiple  $x'_0 = q \cdot x_0$ , in order to avoid the ladder of encodings of 0. Here we show that we cannot take a too small multiple  $x'_0$ , otherwise the attacker can compute:

$$v'_0 = [p_{zt} \cdot x'_0]_N = [p_{zt} \cdot q \cdot x_0]_N = q \cdot v_0 \bmod N$$

where, as in Sect. 3.3, we let  $v_0 := p_{zt} \cdot x_0 \bmod N$ . If the prime  $q$  is small enough then the previous equation holds over the integers, and the attacker obtains  $v'_0 = q \cdot v_0$ . Therefore the attacker can possibly extract a few primes from  $v'_0$  and therefore from  $v_0$ . Letting  $b$  be a divisor of  $v_0$ , one could then apply the Cheon et al. attack modulo  $b$  instead of modulo  $v_0$  and recover all secret parameters. Therefore one should make sure that  $q \cdot v_0$  is greater than  $N$ . Letting  $\eta_q$  be the bitsize of  $q$ , this gives the condition  $\eta_q + \gamma \geq \gamma + 2\eta + 1$ . Therefore we can take  $\eta_q = 2\eta + \lambda$ .

### 3.5 The Subgroup Membership and Decision Linear Problems

In the full version of this paper [CLT15] we also explain why the subgroup membership (SubM) and decisional linear (DLIN) problems, which are known to be easy in the GGH scheme [GGH13a], seem to be hard in our new setting.

## 4 Lattice Attacks

### 4.1 Lattice Attack on the Encodings

The first attack considered in [CLT13] against the original CLT scheme was based on computing a short basis for the lattice of vectors orthogonal modulo  $x_0$

to  $\mathbf{x} = (x_j)_{1 \leq j \leq t}$ , where the  $x_j$ 's are level-0 encodings of zero [CLT13, Sect. 5.1]. If the reduced basis vectors are short enough, they can reveal the noise values of the  $x_j$ 's and hence break the scheme.

The attack does not apply directly to our modified scheme, because  $x_0$  is now secret, and it is therefore no longer possible to compute a basis for the lattice of vectors orthogonal to  $\mathbf{x}$  modulo  $x_0$ . However, we can also mount the attack using the lattice  $\mathbf{x}^\perp$  of vectors orthogonal to  $\mathbf{x}$  over  $\mathbb{Z}$ , or the lattice of vectors orthogonal to  $\mathbf{x}$  modulo some multiple  $x'_0$  of  $x_0$  when using the optimization suggested in Sect. 5 below.

Just as in [CLT13, Sect. 5.1], though, the complexity of these extended attacks remains exponential in  $n$ ; it is in fact slightly worse, because the new lattice has slightly longer vectors for a given choice of the lattice dimension  $t$ . In particular, the complexity lower bound of  $2^{\Omega(\gamma/\eta^2)}$  applies *a fortiori*. The attack is therefore defeated by letting  $n = \omega(\eta \log \lambda)$ .

### 4.2 Lattice Attack Against $p_{zt}$

From  $x_0 = \prod_{i=1}^n p_i$  and  $(\mathbf{p}_{zt})_j = \sum_{i=1}^n h_{ij} \cdot \alpha_i \cdot p_i^{-1} \pmod N$ , we obtain:

$$x_0 \cdot (\mathbf{p}_{zt})_j = \sum_{i=1}^n h_{ij} \cdot \alpha_i \cdot \frac{x_0}{p_i} \pmod N.$$

Now  $x_0$  is of size  $\gamma$  bits, and the right-hand side of this congruence, which we denote by  $w_j$ , is bounded above by  $n2^{\beta+\gamma}$ : they are both small compared to  $N$ . Therefore, if we consider a vector  $\mathbf{p}$  formed by a subset of the  $(\mathbf{p}_{zt})_j$ 's, say  $\mathbf{p} = ((\mathbf{p}_{zt})_j)_{1 \leq j \leq t} \in \mathbb{Z}^t$ , it may be possible to recover  $\mathbf{w} = (w_j)_{1 \leq j \leq t}$  as a short vector in the lattice generated by  $\mathbf{p}$  and  $N\mathbb{Z}^t$ , and obtain  $x_0$  accordingly.

We describe the attack in more details in the full version of this paper [CLT15]. We show that the lattice attack has a complexity lower bound of  $2^{\Omega(n/\eta)} = 2^{\Omega(\gamma/\eta^2)}$ , just as in Sect. 4.1. Thus, this attack is thwarted by our choice of parameters.

In the full version of this paper [CLT15], we consider three other lattice attacks on the zero-testing parameter  $p_{zt}$ , which are variants of the lattice attacks considered in [CLT13, Sects. 5.2, 5.3 and 5.4]. We show that they are also thwarted by our choice of parameters.

## 5 Optimizations and Implementation

In this section we describe an implementation of our new multilinear map scheme in the one-round  $(\kappa + 1)$ -way Diffie-Hellman key exchange protocol; we recall the protocol in the full version of this paper [CLT15], following [BS03,GGH13a]. We use the following optimizations, described in details in the full version of this paper [CLT15]:

1. Integer  $p_{zt}$ : as in [CLT13] we use a single integer  $p_{zt}$  instead of a vector  $\mathbf{p}_{zt}$  with  $n$  components, as this is enough for Diffie-Hellman key exchange. Moreover the integer  $N$  can be generated as the product of large enough prime integers, instead of being prime.
2. Known multiple of  $x_0$ : we publish a multiple  $x'_0 = q \cdot x_0$  of  $x_0$ , so that all intermediate encodings can be reduced modulo  $x'_0$ , instead of using a ladder of encodings of 0 at each level.
3. Quadratic re-randomization: as in [CLT13] we only store a small subset of encodings which are later combined pairwise to generate the full set of encodings. This implies that the randomization of encodings becomes heuristic only. We describe a slightly more efficient variant.

**Parameters and Timings.** We have implemented a one-round  $(\kappa + 1)$ -way Diffie-Hellman key exchange protocol with  $\kappa + 1 = 7$  users, in C++ using the GMP library [Gt14] to perform operations on large integers and fpLLL [ACPS] for LLL. We provide our concrete parameters and the resulting timings in Table 1, for security parameters ranging from 52 to 80 bits. As in [CLT13], for a security level  $\lambda$  we expect that the best attack requires at least  $2^\lambda$  clock cycles. The timings of Table 1 show that the implementation of our scheme improves upon the implementation in [CLT13], especially for the Setup phase.

**Table 1.** Parameters and timings to instantiate a *one-round 7-way Diffie-Hellman key exchange protocol* with  $\kappa = 6$ ,  $\ell = 2\lambda$  and  $\alpha, \beta, \nu = \lambda$  on a 16-core computer (Intel Xeon E7-8837 at 2.67 GHz). Setup was run in parallel on the 16 cores, while the other steps ran on a single core. Publish and KeyGen timings are per party.

Instantiation	$\lambda$	$\kappa$	$n$	$\eta$	$\Delta$	$\rho$	$\gamma = n \cdot \eta$	pk size	Setup	Publish	KeyGen
Small	52	6	540	1679	23	52	$0.9 \cdot 10^6$	27 MB	5.9 s	0.10 s	0.17 s
Medium	62	6	2085	1989	45	62	$4.14 \cdot 10^6$	175 MB	36 s	0.33 s	1.06 s
Large	72	6	8250	2306	90	72	$19.0 \cdot 10^6$	1.2 GB	583 s	2.05 s	6.17 s
Extra	80	6	25305	2619	159	85	$66.3 \cdot 10^6$	6.1 GB	4528 s	7.8 s	23.9 s

## References

[ACPS] Albrecht, M., Cadé, D., Pujol, X., Stehlé, D.: fpLLL-4.0, a floating-point LLL implementation. <http://perso.ens-lyon.fr/damien.stehle>

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



- [BWZ14] Boneh, D., Wu, D.J., Zimmerman, J.: Immunizing multilinear maps against zeroizing attacks. Cryptology ePrint Archive, report 2014/930 (2014). <http://eprint.iacr.org/>
- [CGH+15] Coron, J.-S., Gentry, C., Halevi, S., Lepoint, T., Maji, H.K., Miles, E., Raykova, M., Sahai, A., Tibouchi, M.: Zeroizing without low-level zeroes: new attacks on multilinear maps and their limitations. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO, LNCS. Springer (2015, to appear)
- [CHL+15] Cheon, J.H., Han, K., Lee, C., Ryu, H., Stehlé, D.: Cryptanalysis of the multilinear map over the integers. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 3–12. Springer, Heidelberg (2015)
- [CLT13] Coron, J.-S., Lepoint, T., Tibouchi, M.: Practical multilinear maps over the integers. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 476–493. Springer, Heidelberg (2013)
- [CLT14] Coron, J.-S., Lepoint, T., Tibouchi, M.: Cryptanalysis of two candidate fixes of multilinear maps over the integers. Cryptology ePrint Archive, report 2014/975 (2014). <http://eprint.iacr.org/>
- [CLT15] Coron, J.-S., Lepoint, T., Tibouchi, M.: New multilinear maps over the integers. Cryptology ePrint Archive, report 2015/162 (2015). <http://eprint.iacr.org/>. Full version of this paper
- [CN12] Chen, Y., Nguyen, P.Q.: Faster algorithms for approximate common divisors: breaking fully-homomorphic-encryption challenges over the integers. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 502–519. Springer, Heidelberg (2012)
- [Cop97] Coppersmith, D.: Small solutions to polynomial equations, and low exponent RSA vulnerabilities. *J. Crypt.* **10**(4), 233–260 (1997)
- [DGHV10] van Dijk, M., Gentry, C., Halevi, S., Vaikuntanathan, V.: Fully homomorphic encryption over the integers. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 24–43. Springer, Heidelberg (2010)
- [GGH13a] Garg, S., Gentry, C., Halevi, S.: Candidate multilinear maps from ideal lattices. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 1–17. Springer, Heidelberg (2013)
- [GGH+13b] Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: FOCS, pp. 40–49. IEEE Computer Society (2013)
- [GGH15] Gentry, C., Gorbunov, S., Halevi, S.: Graph-induced multilinear maps from lattices. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015, Part II. LNCS, vol. 9015, pp. 498–527. Springer, Heidelberg (2015)
- [GGHZ14] Garg, S., Gentry, C., Halevi, S., Zhandry, M.: Fully secure functional encryption without obfuscation. Cryptology ePrint Archive, report 2014/666 (2014). <http://eprint.iacr.org/>
- [GH11] Gentry, C., Halevi, S.: Implementing Gentry’s fully-homomorphic encryption scheme. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 129–148. Springer, Heidelberg (2011)
- [GHMS14] Gentry, C., Halevi, S., Maji, H.K., Sahai, A.: Zeroizing without zeroes: cryptanalyzing multilinear maps without encodings of zero. Cryptology ePrint Archive, report 2014/929 (2014). <http://eprint.iacr.org/>

- [Gt14] Granlund, T. and the GMP development team. GNU MP: the GNU multiple precision arithmetic library, 6.0.0 edn. (2014). <http://gmplib.org/>
- [Lep15] Lepoint, T.: Proof-of-concept implementation of the “new” multilinear maps over the integers (2015). <https://github.com/tlepoint/new-multilinear-maps>
- [LS14] Lee, H.T., Seo, J.H.: Security analysis of multilinear maps over the integers. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 224–240. Springer, Heidelberg (2014)

# Constant-Round Concurrent Zero-Knowledge from Indistinguishability Obfuscation

Kai-Min Chung<sup>1</sup>, Huijia Lin<sup>2</sup>, and Rafael Pass<sup>3</sup>

<sup>1</sup> Academia Sinica, Taipei, Taiwan  
kmchung@iis.sinica.edu.tw

<sup>2</sup> University of California, Santa Barbara, USA  
rachel.lin@cs.ucsb.edu

<sup>3</sup> Cornell University, Ithaca, USA  
rafael@cs.cornell.edu

**Abstract.** We present a constant-round concurrent zero-knowledge protocol for NP. Our protocol relies on the existence of families of collision-resistant hash functions, one-way permutations, and indistinguishability obfuscators for  $\mathbf{P}/poly$  (with slightly super-polynomial security).

## 1 Introduction

Zero-knowledge ( $\mathcal{ZK}$ ) interactive proofs [30] are paradoxical constructs that allow one player (called the Prover) to convince another player (called the Verifier) of the validity of a mathematical statement  $x \in L$ , while providing *zero additional knowledge* to the Verifier. Beyond being fascinating in their own right,  $\mathcal{ZK}$  proofs have numerous cryptographic applications and are one of the most fundamental cryptographic building blocks.

The notion of concurrent zero knowledge, first introduced and achieved in the paper by Dwork, Naor and Sahai [24], considers the execution of zero-knowledge proofs in an asynchronous and concurrent setting. More precisely, we consider a single adversary mounting a coordinated attack by acting as a verifier in many concurrent executions (called sessions). Concurrent  $\mathcal{ZK}$  proofs are significantly harder to construct and analyze. Since the original protocol by DNS Dwork, Naor and Sahai (which relied on “timing assumptions”), various other concurrent  $\mathcal{ZK}$  protocols have been obtained based on different set-up assumptions (e.g., [22, 25]), or in alternative models (e.g., super-polynomial-time simulation [44, 53]).

In the standard model, without set-up assumptions (the focus of our work), Canetti, Kilian, Petrank and Rosen [14] (building on earlier works by [38, 58]) show that concurrent  $\mathcal{ZK}$  proofs for non-trivial languages, with “black-box” simulators, require at least  $\tilde{\Omega}(\log n)$  number of communication rounds. Richardson and Kilian [56] constructed the first concurrent  $\mathcal{ZK}$  argument in the standard model without any extra set-up assumptions. Their protocol, which uses a black-box simulator, requires  $O(n^\epsilon)$  number of rounds. The round-complexity was later improved in the work of Kilian and Petrank (KP) [37] to  $\tilde{O}(\log^2 n)$  round. More

recent work by Prabhakaran, Rosen and Sahai [55] improves the analysis of the KP simulator, achieving an essentially optimal, w.r.t. black-box simulation, round-complexity of  $\tilde{O}(\log n)$ ; see also [52] for an (arguably) simplified and generalized analysis.

The central open problem in the area is whether a *constant-round* concurrent  $\mathcal{ZK}$  protocol (for a non-trivial language) can be obtained. Note that it could very well be the case that all “classic” zero-knowledge protocols already are concurrent zero-knowledge; thus, simply assuming that those protocols are concurrent zero-knowledge yields an assumption under which constant-round concurrent zero-knowledge (trivially) exists—in essence, we are assuming that for every attacker a simulator exists. Furthermore, as shown in [33] (and informally discussed in [16]) under various “extractability” assumptions of the knowledge-of-exponent type [8, 21, 34], constant-round concurrent zero-knowledge is easy to construct. But such extractability assumptions also simply assume that for every attacker, a simulator (in essence, “the extractor” guaranteed by the extractability assumption) exists. In particular, an explicit construction of the concurrent zero-knowledge simulator is not provided—it is simply assumed that one exists. For some applications of zero-knowledge such as *deniability* (see e.g., [24, 44]), having an explicit simulator is crucial. Rather, we are here concerned with the question of whether constant-round concurrent zero-knowledge, *with an explicit simulator*, exists.

### 1.1 Towards Constant-Round Concurrent Zero-Knowledge

Recently, the authors [16] provided a first construction a constant-round concurrent zero-knowledge protocol with an explicit simulator, based on a new cryptographic hardness assumption—the existence of so-called  $\mathbf{P}$ -certificates, roughly speaking, succinct non-interactive arguments for languages in  $\mathbf{P}$ . An issue with their approach, however, is we only have candidate constructions of  $\mathbf{P}$ -certificates that are sound against *uniform* polynomial-time attackers (as opposed to non-uniform ones), and the protocol of [16] inherits the soundness property of the underlying  $\mathbf{P}$ -certificate. Additionally, whereas the assumption that a particular proof system is a  $\mathbf{P}$ -certificates is a falsifiable assumption [42, 54], it is unclear whether the existence of  $\mathbf{P}$ -certificates itself can be based on some more natural hardness assumptions.

A very recent elegant work by Pandey, Prabhakaran and Sahai [43] takes a different approach and instead demonstrates the existence of constant-round concurrent zero-knowledge protocol with an explicit simulator based on the existence of *differing-input obfuscation* ( $\mathbf{diO}$ ) for (restricted classes of)  $\mathbf{P}/\text{poly}$  [1, 6, 11]. Whereas the assumption that a particular scheme is a  $\mathbf{diO}$  is an “extractability” assumption (similar in flavor to knowledge-of-exponent type [8, 21, 34] assumptions), the intriguing part of the scheme of Pandey et al. [43] is that the extractability assumption is only used to prove *soundness* of the protocol; concurrent zero-knowledge is proved in the “standard” model, through providing an explicit simulator. Nevertheless,  $\mathbf{diO}$  is a strong and subtle assumption—as shown by recent work [12, 27, 36]; unless we restrict the class of programs

for which  $\mathbf{diO}$  should hold, we may end up with a notion that is unsatisfiable. Additionally, there are currently no known approaches for basing  $\mathbf{diO}$  on more “natural” (or in fact *any*) hardness (as opposed to extractability) assumption.

## 1.2 Our Results

In this paper, we combine the above-mentioned two approaches. Very roughly speaking, we will use obfuscation to obtain a variant of the notion of a  $\mathbf{P}$ -certificate, and we next show that this variant still suffices to obtain constant-round concurrent zero-knowledge (where the soundness conditions holds also against non-uniform PPT attackers). More importantly, rather than using  $\mathbf{diO}$ , we are able to use *indistinguishability obfuscation* ( $\mathbf{iO}$ ) [6, 26]. Following the groundbreaking work of Garg et al. [26], there are now several candidate constructions of  $\mathbf{iO}$  that can be based on hardness assumptions on (approximate) multilinear maps [29, 51].

**Theorem 1.** *Assume the existence of indistinguishability obfuscation for  $\mathbf{P}/\text{poly}$  (with slightly super-polynomial security), one-way permutations (with slightly super-polynomial security) and collision-resistant hash function. Then there exists a constant-round concurrent zero-knowledge argument for NP.*

In more details, our approach proceeds in the following steps:

1. We first observe that a warm-up case considered in [16]—which shows the existence of constant-round concurrent zero-knowledge based on, so-called, *unique  $\mathbf{P}$ -certificates* (that is,  $\mathbf{P}$ -certificates for which there exists at most one accepting certificate for each statement) directly generalizes also to unique  $\mathbf{P}$ -certificates in the Common *Random* String model (a.k.a. the Uniform Random String model (URS)) satisfying an *adaptive soundness* property (where the statement to be proved can be selected after the URS).
2. We next show that by appropriately modifying the protocol, we can handle also unique  $\mathbf{P}$ -certificates in the URS model satisfying even just a “static” soundness condition (where the statement needs to be selected before the URS is picked), and additionally also unique  $\mathbf{P}$ -certificates (with static soundness) in the Common *Reference* String (CRS) model, where the reference string no longer is required to be uniform. Unique  $\mathbf{P}$ -certificates in the CRS model (also with non-uniform soundness) can be constructed based on the existence of  $\mathbf{diO}$  for (a restricted class of)  $\mathbf{P}/\text{poly}$  [12], and as such this preliminary step already implies the result of [43] in a modular way (but with worse concrete round complexity).
3. We next show how to use fully homomorphic encryption (FHE) [28, 57] and  $\mathbf{iO}$  to modify the protocol to handle also *two-round* unique  $\mathbf{P}$  certificates. Two-round  $\mathbf{P}$ -certificates are a generalization of  $\mathbf{P}$ -certificates in the CRS model, where we allow the CRS (i.e., the “first message” from the verifier to the prover) to depend on the statement to be proven.
4. We finally leverage recent results on delegation of computation based on  $\mathbf{iO}$  from [9, 13, 39] and show that the beautiful scheme of Koppula, Lewko and Waters [39] can be modified into a two-message unique  $\mathbf{P}$ -certificate (also with non-uniform soundness).

More precisely, we show that any “succinct” *message hiding encoding* [39], which is a relaxed version of a “succinct” randomized encoding [9, 35], together with injective one-way functions yields a two-round unique  $\mathbf{P}$ -certificate. [39] shows how to construct succinct message-hiding encodings based on  $\mathbf{iO}$  and injective PRGs.

The above steps show how to obtain constant-round concurrent  $\mathcal{ZK}$  based on collision-resistant hashfunctions,  $\mathbf{iO}$  for  $\mathbf{P}/poly$ , one-way permutations, and FHE. We finally observe that the message-hiding encoding of [39] has a particular nice structure that enables us to refrain from using FHE in the final protocol, thus reaching our final theorem.

### 1.3 Outline of Our Techniques

We here provide a detailed outline of our techniques. As mentioned, our construction heavily relies on a “warm-up” case of the construction of [16], which we start by recalling (closely following the description in [16]). The starting point of the construction of [16] is the construction is Barak’s [2] non-black-box zero-knowledge argument for NP. Below, we briefly recall the ideas behind his protocol (following a slight variant of this protocol due to [47]).

**Barak’s Protocol.** Roughly speaking, on common input  $1^n$  and  $x \in \{0, 1\}^{\text{poly}(n)}$ , the Prover  $\mathbf{P}$  and Verifier  $V$ , proceed in two stages. In Stage 1,  $P$  starts by sending a computationally-binding commitment  $c \in \{0, 1\}^n$  to  $0^n$ ;  $V$  next sends a “challenge”  $r \in \{0, 1\}^{2n}$ . In Stage 2,  $P$  shows (using a witness indistinguishable argument of knowledge) that either  $x$  is true, or there exists a “short” string  $\sigma \in \{0, 1\}^n$  such that  $c$  is a commitment to a program  $M$  such that  $M(\sigma) = r$ .<sup>1</sup>

Soundness follows from the fact that even if a malicious prover  $P^*$  tries to commit to some program  $M$  (instead of committing to  $0^n$ ), with high probability, the string  $r$  sent by  $V$  will be different from  $M(\sigma)$  for every string  $\sigma \in \{0, 1\}^n$ . To prove  $\mathcal{ZK}$ , consider the non-black-box simulator  $S$  that commits to the code of the malicious verifier  $V^*$ ; note that by definition it thus holds that  $M(c) = r$ , and the simulator can use  $\sigma = c$  as a “fake” witness in the final proof. To formalize this approach, the witness indistinguishable argument in Stage 2 must actually be a witness indistinguishable *universal argument* (WIUA) [4, 41] since the statement that  $c$  is a commitment to a program  $M$  of *arbitrary* polynomial-size, and that  $M(c) = r$  within some *arbitrary* polynomial time, is not in NP.

Now, let us consider concurrent composition. That is, we need to simulate the view of a verifier that starts *poly*( $n$ ) concurrent executions of the protocol. The above simulator no longer works in this setting: the problem is that the

---

<sup>1</sup> We require that  $C$  is a commitment scheme allowing the committer to commit to an arbitrarily long string  $m \in \{0, 1\}^*$ . Any commitment scheme for fixed-length messages can easily be modified to handle arbitrarily long messages by asking the committer to first hash down  $m$  using a collision-resistant hash function  $h$  chosen by the receiver, and next commit to  $h(m)$ .

verifier’s code is now a function of *all* the prover messages sent in different executions. (Note that if we increase the length of  $r$  we can handle a bounded number of concurrent executions, by simply letting  $\sigma$  include all these messages).

So, if the simulator could commit not only to the code of  $V^*$ , but also to a program  $M$  that generates all other prover messages, then we would seemingly be done. And at first sight, this doesn’t seem impossible: since the simulator  $S$  is actually the one generating all the prover messages, why don’t we just let  $M$  be an appropriate combination of  $S$  and  $V^*$ ? This idea can indeed be implemented [47, 50], but there is a serious issue: if the verifier “nests” its concurrent executions, the running-time of the simulation quickly blows up exponentially—for instance, if we have three nested sessions, to simulate session 3 the simulator needs to generate a WIUA regarding the computation needed to generate a WIUA for session 2 which in turn is regarding the generation of the WIUA of session 1 (so even if there is just a constant overhead in generating a WIUA, we can handle at most  $\log n$  nested sessions).

**Unique P-certificates to The Rescue: The “Warm-Up” Case [16].** As shown in [16], the blow-up in the running-time can be prevented using Unique P-certificates. Roughly speaking, we say that  $(P, V)$  is a **P-certificate system** if  $(P, V)$  is a non-interactive proof system (i.e., the prover send a single message to the verifier, who either accepts or rejects) allowing an efficient prover to convince the verifier of the validity of any *deterministic polynomial-time computation*  $M(x) = y$  using a “certificate” of some *fixed* polynomial length (independent of the size and the running-time of  $M$ ) whose validity the verifier can check in some fixed polynomial time (independent of the running-time of  $M$ ). The P-certificate system is *unique* if there exists at most one accepted proof for any statement.

The protocol proceeds just as Barak’s protocol except that Stage 2 is modified as follows: instead of having  $P$  prove (using a WIUA) that either  $x$  is true, or there exists a “short” string  $\sigma \in \{0, 1\}^{2n}$  such that  $c$  is a commitment to a program  $M$  such that  $M(\sigma) = r$ , we now ask  $P$  to prove (using a WIUA again) that either  $x$  is true, or

- **Commitment Consistency:**  $c$  is a commitment to a program  $M_1$ , and
  - **Input Certification:** there exists a vector  $\lambda = ((1, \pi_1), (2, \pi_2), \dots)$  and a vector of messages  $\mathbf{m}$  such that  $\pi_j$  certifies that  $M_1(\lambda_{<j})$  outputs  $m_j$  in its  $j$ ’th communication round, where  $\lambda_{<j} = ((1, \pi_1), \dots, (j-1, \pi_{j-1}))$ , and
  - **Prediction Correctness:** there exists a P-certificate  $\pi$  of length  $n$  demonstrating that  $M_1(\lambda) = r$ .

Soundness of the modified protocol, roughly speaking, follows since by the unique certificate property, for every program  $M_1$  it inductively follows that for every  $j$ ,  $m_j$  is uniquely defined, and thus also the *unique* (accepting) certificate  $\pi_j$  certifying  $M_1(\lambda_{<j}) = m_j$ ; it follows that  $M_1$  determines a unique vector  $\lambda$  that passes the input certification conditions, and thus there exists a single  $r$  that make  $M_1$  also pass the prediction correctness conditions. Note that we here inherently rely on the fact that the P-certificate is unique to argue that the sequence  $\lambda$

is uniquely defined. (Technically, we here need to rely on a  $\mathbf{P}$ -certificate that is sound for slightly super-polynomial-time as there is no a-priori polynomial bound on the running-time of  $M_1$ , nor the length of  $\lambda$ .)

To prove zero-knowledge, roughly speaking, our simulator will attempt to commit to its own code in a way that prevents a blow-up in the running-time. Recall that the main reason that we had a blow-up in the running-time of the simulator was that the generation of the WIUA is expensive. Observe that in the new protocol, the only expensive part of the generation of the WIUA is the generation of the  $\mathbf{P}$ -certificates  $\pi$ ; the rest of the computation has *a-priori* bounded complexity (depending only on the size and running-time of  $V^*$ ). To take advantage of this observation, we thus have the simulator only commit to a program that generates prover messages (in identically the same way as the actual simulator), but getting certificates  $\pi$  as input.

In more detail, to describe the actual simulator  $S$ , let us first describe two “helper” simulators  $S_1, S_2$ .  $S_1$  is an interactive machine that simulates prover messages in a “right” interaction with  $V^*$ . Additionally,  $S_1$  is expecting some “external” messages on the “left”—looking forward, these “left” messages will later be certificates provided by  $S_2$ .

$S_1$  proceeds as follows in the right interaction. In Stage 1 of every session  $i$ ,  $S_1$  first commits to a machine  $\tilde{S}_1(j', \tau)$  that emulates an interaction between  $S_1$  and  $V^*$ , feeding  $S_1$  input  $\tau$  as messages on the left, and finally  $\tilde{S}_1$  outputs the verifier message in the  $j'$ 'th communication round in the right interaction with  $V^*$ . (Formalizing what it means for  $S_1$  to commit to  $\tilde{S}_1$  is not entirely trivial since the definition of  $\tilde{S}_1$  depends on  $S_1$ ; we refer the reader to the formal proof for a description of how this circularity is broken.<sup>2</sup>)  $S_1$  next simulates Stage 2 by checking if it has received a message  $(j, \pi_j)$  in the left interaction, where  $j$  is the communication round (in the right interaction with  $V^*$ ) where the verifier sends its random challenge and expects to receive the first message of Stage 2; if so, it uses  $M_1 = \tilde{S}_1$  (and the randomness it used to commit to it),  $j$  and  $\sigma$  being the list of messages received by  $S_1$  in the left interaction, as a “fake” witness to complete Stage 2.

The job of  $S_2$  is to provide  $\mathbf{P}$ -certificates  $\pi_j$  for  $S_1$  allowing  $S_1$  to complete its simulation.  $S_2$  emulates the interaction between  $S_1$  and  $V^*$ , and additionally, at each communication round  $j$ ,  $S_2$  feeds  $S_1$  a message  $(j, \pi_j)$  where  $\pi_j$  is a  $\mathbf{P}$ -certificate showing that  $\tilde{S}_1(j, \sigma_{<j}) = r_j$ , where  $\sigma_{<j}$  is the list of messages already generated by  $S_2$ , and  $r_j$  is the verifier message in the  $j$ 'th communication round. Finally,  $S_2$  outputs its view of the full interaction.

The actual simulator  $S$  just runs  $S_2$  and recovers from the view of  $S_2$  the view of  $V^*$  and outputs it. Note that since  $S_1$  has polynomial running-time, generating each certificate about  $\tilde{S}_1$  (which is just about an interaction between  $S_1$  and  $V^*$ ) also takes polynomial time. As such  $S_2$  can also be implemented in polynomial time and thus also  $S$ .

---

<sup>2</sup> Roughly speaking, we let  $S_1$  take the description of a machine  $M$  as input, and we then run  $S_1$  on input  $M = S_1$ .



Finally, indistinguishability of this simulation, roughly speaking, follow from the hiding property of the commitment in Stage 1, and the WI property of the WIUA in Stage 2. (There is another circularity issue that arises in formalizing this—as  $S_1$  in essence needs to commit to its own randomness—but it can be dealt with as shown in [15, 16]; in this overview, we omit the details as they are not important for our modifications to the protocol, but they can be found in the formal proof.)

**Generalizing to Unique  $\mathbf{P}$ -certificates in CRS model.** The key technical contribution in [16] was to generalize the above approach to deal also with “non-unique”  $\mathbf{P}$ -certificates. Here we instead aim to generalize the above approach to work with  $\mathbf{P}$ -certificates in the CRS model, but still relying on the uniqueness property.

Let us first note that if we had access to unique  $\mathbf{P}$ -certificate in the URS (i.e., the uniform reference string) model satisfying an *adaptive soundness* property (where the statement to be proved can be selected after the URS, then above-mentioned protocol can be almost directly generalized to work with them (as opposed to using unique  $\mathbf{P}$ -certificates in the “plain” model) by simply having the Verifier send the URS  $\rho$  along with its first message of the protocol.<sup>3</sup> The only issue that needs to be addressed in implementing this change is to specify what it means that “ $\pi_j$  certifies that  $M_1(\lambda_{<j})$  outputs  $m_j$ ” in the input certification step in Stage 2, since this certification needs to be done with respect to some URS. We modify Stage two to require that  $M_1$  outputs not only messages  $m_i$ , but also reference strings  $\rho_i$ . Let us remark that to ensure that soundness still holds, we require the  $\mathbf{P}$ -certificate system to satisfy a strong uniqueness property: uniqueness of accepting proofs needs to hold for *all* reference strings  $\rho$ .

We next note that the protocol can be further generalized to handle also unique  $\mathbf{P}$ -certificates in the URS model satisfying even just a *static soundness* condition (where the statement needs to be selected before the URS is picked) by proceeding as follows:

- We add a Stage 1.5 to the protocol where the Prover is asked to provide a commitment  $c_2$  to  $0^n$  and then asked to provide a WIUARG that either  $x \in L$  or  $c_2$  is a commitment to a “well-formed” statement (but not that the statement is true) for the  $\mathbf{P}$ -certificate in use in Stage 2.
- Stage 2 of the protocol is then modified to first have the Verifier send the URS for the  $\mathbf{P}$ -certificate, and then requiring that the prover uses a  $\mathbf{P}$ -certificate for the statement committed to in  $c_2$ . In other words, we require the Prover to commit in advance, and prove knowledge of, the statement to be used in the  $\mathbf{P}$ -certificate and thus static soundness suffices.

Additionally, this approach generalizes also to deal with unique  $\mathbf{P}$ -certificates in the *Common Reference String* (CRS) model (where the reference string no

<sup>3</sup> To make this work, we need to rely on  $\mathbf{P}$ -certificates in the URS model with perfect completeness. This requirement can be removed by additionally performing a coin-tossing to determine the URS. For simplicity of exposition, we here simply assume perfect completeness.

longer needs to be uniform), by having the Verifier provide a zero-knowledge proof that the CRS was well-formed.<sup>4</sup> Let us again remark that to ensure that soundness still holds, we require the uniqueness property of the **P**-certificate system to hold for all reference strings  $\rho$ , *even invalid ones*.

**Generalizing to Two-round Unique P-certificates.** The notion of a **P**-certificate in the CRS model requires that the same CRS can be used to prove *any* statement  $q$  of any (polynomially-related) length. We will now consider a weaker notion of a **P**-certificate in the CRS model, where the CRS is “statement-dependent”—that is, the CRS is generated as a function of the statement  $q$  to be proved. (On the other hand, while we allow the CRS to depend on the statement  $q$ , we require the length of the CRS to be independent of the *length* of  $q$ .) In essence, we are considering *two-round* publicly-verifiable delegation protocols. We refer to such schemes as *two-round P-certificates*. We now generalize the above approach to work with unique two-round **P**-certificates.

- Instead of having the Verifier send the CRS in the clear (which it cannot compute as it does not know the statement  $q$  on which it will be run), it simply send an FHE encryption  $\hat{\alpha}$  of random coins  $\alpha$  needed to run the CRS generation. (Using PRGs, we may assume wlog that the length of  $\alpha$  is  $n$ .)
- The Prover is then asked to provide a third commitment  $c_3$  to  $0^n$  and provide a WIARG that either  $x \in L$  or  $c_3$  is a commitment to an FHE encryption  $\hat{\rho}$  obtained by running the CRS-generation procedure (using the appropriate FHE operations) on the ciphertext  $\hat{\alpha}$ . (That is,  $\hat{\rho}$  is an encryption of the CRS  $\rho$  obtained by running the CRS generation algorithm with random coins  $\alpha$ .)
- Next, the Verifier sends an indistinguishability obfuscation  $\tilde{H} = \mathbf{iO}(H)$  of a program  $H$  that on input a decommitment  $(\hat{\rho}, r')$  to  $c_3$  decrypts  $\hat{\rho}$  (using the FHE secret key) into a CRS  $\rho$  and outputs it. (The reason that the Verifier cannot simply decrypt  $\hat{\rho}$  for the Prover is that  $\hat{\rho}$  cannot be sent to the Verifier in the clear; recall that the honest prover will never compute any such ciphertext, it is meant to commit to  $0^n$  and prove that  $x \in L$ .) Additionally, the verifier gives a zero-knowledge proof that the obfuscation is correctly computed.
- Then, the Prover provides a commitment  $c_4$  to  $0^n$  and provides a WI proof of knowledge that  $x \in L$  or  $c_4$  is a commitment to a CRS  $\rho$  computed by applying the obfuscated code  $\tilde{H}$  to a proper decommitment of  $c_3$ .
- Finally, in Stage 2 of the protocol, we require the Prover to provide **P**-certificates w.r.t to the CRS  $\rho$  committed to in  $c_4$ .

Note that if  $c_3$  is perfectly binding, then by **iO** security of the obfuscation, we can replace  $H$  with a program that has the CRS  $\rho$  hardcoded (without any knowledge of the random coins  $\alpha$  used to generate  $\rho$ ), and this suffices for arguing

---

<sup>4</sup> Again, we here rely on **P**-certificates in the CRS model with perfect completeness. This requirement can also be avoided by having the prover and the verifier perform coin-tossing-in-the-well to determine the secret coins the verifier should use for generating the CRS. As our instantiations of **P**-certificates will satisfy perfect completeness, we do not further formalize this approach.

that soundness of the protocol still holds. On the other hand, the simulation can proceed just as before except that the simulator now uses the obfuscated code  $\tilde{I}$  to generate the CRS  $\rho$  and commits to it in  $c_4$ .

**Realizing Unique Two-Round  $\mathbf{P}$ -Certificates.** We finally leverage recent results on delegation of computation based on  $\mathbf{iO}$  from [9, 13, 39] and show that the beautiful scheme of Koppula, Lewko and Waters [39] can be massaged (and slightly modified) into a two-message unique  $\mathbf{P}$ -certificate. More precisely, we show how to use the notion of a “succinct” *message hiding encoding* [39]—a relaxed version of a “succinct” randomized encoding [9, 35]—together with injective one-way functions to construct a two-round unique  $\mathbf{P}$ -certificate. [39] shows how to construct succinct message-hiding encodings (in fact, even succinct randomized encodings) based on  $\mathbf{iO}$  for  $\mathbf{P}/poly$  and injective PRGs.

Let us point out that, just as [16], our protocol requires the use of  $\mathbf{P}$ -certificates that satisfy a slightly strong soundness condition—namely, we require soundness to hold against circuits of size  $T(\cdot)$  where  $T(\cdot)$  is some “nice” (slightly) super-polynomial function (e.g.,  $T(n) = n^{\log \log \log n}$ ). To achieve such (delegatable)  $\mathbf{P}$ -certificates, we thus rely on  $\mathbf{iO}$  for  $\mathbf{P}/poly$  secure against  $T(\cdot)$ -size circuits.

**Removing the Use of FHE.** In a final step, we note that by relying on specific nice properties of the message-hiding encoding of [39], we obtain a two-round  $\mathbf{P}$ -certificate satisfying a desirable property: Only a “small” part of the CRS generation procedure relies on secret coins. More precisely, the CRS generation procedure proceeds in three steps: (1) first, secret coins are used to generate a public parameter  $PP$  and a secret parameter  $K$  (this is done independently of the statement  $q$ ), (2) next, only  $PP$  is used to *deterministically* process the statement  $q$  into a “short” digest  $d$  (independent of the length of  $q$ ), and (3) the digest  $d$  and the secret parameter  $K$  is efficiently processed to finally generate the CRS (independent of the length of  $q$ ). To emphasize, only step 2 requires work proportional to the length of  $q$ , but this work only requires public information.

We refer to such schemes as *delegatable  $\mathbf{P}$ -certificates in the CRS model* and note that if we rely on such a scheme, then we can dispense the need for FHE in our final protocol, as the computation of the digest can be directly delegated to the prover without the need of FHE.

This completes the informal description of our protocol and its proof of security. In our formal description of the final protocol, for simplicity, we directly present a solution using such delegatable  $\mathbf{P}$ -certificates (without going through the construction using FHE). As mentioned above, the above description ignores certain subtleties required to prevent circularities in the simulation and the proof of security. To deal with these issues (already considered in [16]) as well as to streamline the description of the final protocol (to enable a better concrete round-complexity) the formal description slightly differs from what is outlined above.

**Other Related Works.** Since the work of Barak [2], non-black-box simulation techniques have been used in several other contexts: Non-malleability [3, 45, 48, 49],

concurrent secure computation [7, 40, 45, 46], resettable-soundness [5, 10, 17, 18, 20, 23], covert secure computation [32] and more. We believe our techniques may yield improved constructions also in these settings.

We also mention recent work of [15, 31] that constructs *public-coin* concurrent zero-knowledge protocols using non-black-box simulation; these protocols are not constant-round but instead rely on “standard” assumptions. Let us finally mention that the constant-round concurrent zero-knowledge protocol of [16] (which relies on non-interactive  $\mathbf{P}$ -certificates) actually also is public-coin, whereas our protocol is not. We leave open the question of basing public-coin concurrent zero-knowledge on  $\mathbf{iO}$ .

**Organization.** In Sect. 2, we define unique two-message  $\mathbf{P}$ -certificates, and the property of delegatable CRS generation. We show how to instantiate them using message hiding encoding of [39] in the full version of the paper [19]. In Sect. 3, we present our constant-round concurrent  $\mathcal{ZK}$  protocol, and its simulator. We refer the reader to the full version [19] for preliminaries and full proof of our protocol.

## 2 Two-Message $\mathbf{P}$ -certificates

We consider the following canonical languages for  $\mathbf{P}$ : for every constant  $c \in \mathbb{N}$ , let  $L_c = \{(M, x, y) : M(x) = y \text{ within } |x|^c \text{ steps}\}$ . Let  $T_M(x)$  denotes the running time of  $M$  on input  $x$ .

**Definition 1 (Two-Message  $\mathbf{P}$ -certificate).** *A tuple of probabilistic interactive Turing machines,  $(\text{Gen}, \text{P}_{\text{cert}}, \text{V}_{\text{cert}})$ , is a (Two-Message)  $\mathbf{P}$ -certificate system if there exist polynomials  $l_{\text{CRS}}, l_{\pi}$ , and the following holds:*

**Syntax and Efficiency:** *For every  $c \in \mathbb{N}$ , every  $q = (M, x, y) \in L_c$ , and every  $k \in \mathbb{N}$ , the verification of the statement proceed as follows:*

**CRS GENERATION:**  $\text{CRS} \stackrel{\$}{\leftarrow} \text{Gen}(1^k, c, q)$ , where  $\text{Gen}$  runs in time  $\text{poly}(k, |q|)$ .  
The length of  $\text{CRS}$  is bounded by  $l_{\text{CRS}}(k)$ .

**PROOF GENERATION:**  $\pi \stackrel{\$}{\leftarrow} \text{P}_{\text{cert}}(1^k, c, q, \text{CRS})$ , where  $\text{P}_{\text{cert}}$  runs in time  $\text{poly}(k, |x|, \min(T_M(x), |x|^c))$  with  $T_M(x) \leq |x|^c$  the running time of  $M$  on input  $x$ . The length of the proof  $\pi$  is bounded by  $l_{\pi}(k)$ .

**PROOF VERIFICATION:**  $b = \text{V}_{\text{cert}}(1^k, c, \text{CRS}, q, \pi)$ , where  $\text{V}_{\text{cert}}$  runs in time  $\text{poly}(k, |q|)$ .

**(Perfect) Completeness:** *For every  $c, d \in \mathbb{N}$ , there exists a negligible function  $\mu$  such that for every  $k \in \mathbb{N}$  and every  $q = (M, x, y) \in L_c$  such that  $|q| \leq k^d$ , the probability that in the above execution  $\text{V}_{\text{cert}}$  outputs 1 is 1.*

**Definition 2 (Selective Strong Soundness).** *We say that a  $\mathbf{P}$ -certificate system  $(\text{Gen}, \text{P}_{\text{cert}}, \text{V}_{\text{cert}})$  is (selectively) strong sound if the following holds:*

**Strong Soundness:** *There exists some “nice” super-polynomial function (e.g.,  $T(n) = n^{\log \log \log n}$ )  $T(k) \in k^{\omega(1)}$  and some “nice” super-constant function (e.g.,  $C(k) = \log \log \log n$ )  $C(\cdot) \in \omega(1)$  such that for every probabilistic algorithm  $P^*$*

with running-time bounded by  $T(\cdot)$ , there exists a negligible function  $\mu$ , such that, for every  $k \in N$ ,  $c \leq C(k)$ ,

$$\Pr \left[ \begin{array}{l} (q, \mathbf{st}) \stackrel{\$}{\leftarrow} P^*(1^k, c) \\ \text{CRS} \stackrel{\$}{\leftarrow} \text{Gen}(1^k, c, q) \\ \pi \stackrel{\$}{\leftarrow} P^*(\mathbf{st}, \text{CRS}) \end{array} : \text{V}_{\text{cert}}(1^k, c, \text{CRS}, q, \pi) = 1 \wedge q \notin L_c \right] \leq \mu(k)$$

**Definition 3 (Uniqueness).** We say that a  $\mathbf{P}$ -certificate system  $(\text{Gen}, \text{P}_{\text{cert}}, \text{V}_{\text{cert}})$  is unique if for every  $k \in N$ , every constant  $c \in N$ , string  $\text{CRS} \in \{0, 1\}^*$  and string  $q \in \{0, 1\}^*$ , there exists at most one string  $\pi \in \{0, 1\}^*$ , such that  $\text{V}_{\text{cert}}(1^k, c, \text{CRS}, q, \pi) = 1$ .

**Definition 4 (Delegatable CRS Generation).** We say that a (two-message)  $\mathbf{P}$ -certificate  $(\text{Gen}, \text{P}_{\text{cert}}, \text{V}_{\text{cert}})$  has delegatable CRS generation if the CRS generation algorithm  $\text{Gen}$  consists of three subroutines ( $\text{Setup}, \text{PreGen}, \text{CRSGen}$ ), and there are polynomials  $l_d$  and  $l_\kappa$ , such that, the following holds:

**Delegatable CRS Generation:**  $\text{Gen}(1^k, c, q)$  proceeds in the following steps:

1. GENERATE PARAMETERS:  $(PP, K) \stackrel{\$}{\leftarrow} \text{Setup}(1^k, c)$ , where  $\text{Setup}$  is probabilistic and runs in time  $\text{poly}(k)$ . We call  $PP$  the public parameter and  $K$  the key.
2. (PUBLIC) STATEMENT PROCESSING:  $d = \text{PreGen}(PP, q)$ , where  $\text{PreGen}$  is deterministic and runs in time  $\text{poly}(k, |q|)$ , and the length of  $d$  is bounded by  $l_d(k)$ . We call  $d$  the digest of the statement.
3. (PRIVATE) CRS GENERATION:  $\kappa \stackrel{\$}{\leftarrow} \text{CRSGen}(PP, K, d)$ , where  $\text{CRSGen}$  is probabilistic and runs in time  $\text{poly}(k)$ , and the length of  $\kappa$  is bounded by  $l_\kappa(k)$ .

Finally,  $\text{Gen}$  outputs  $\text{CRS} = (PP, \kappa)$ .

The reason that we say such a CRS generation procedure is delegatable is because the only part of computation that depends on the statement is the statement processing step; all other steps runs in time a fixed polynomial in the security parameter. However, the statement processing step depends only on the public parameter and the statement; hence to ensure soundness, one only needs to ensure the correctness of this computation, without ensuring the “secrecy” of the computation. Therefore, we also call this step “public” statement processing.

**Simple Verification Procedure.** Finally, we define an additional property of  $\mathbf{P}$ -certificates: We say that the verification algorithm of a  $\mathbf{P}$ -certificate system is *simple* if  $\text{V}_{\text{cert}}$  only depends on the security parameter  $1^k$ , the CRS  $\text{CRS}$  and the proof  $\pi$  (independent of the statement  $q$  and the language index  $c$ ). Naturally, the uniqueness property of this instantiation is that for any  $1^k$  and CRS string  $\text{CRS}$ , there is at most one unique accepting proof.

**Instantiation of  $\mathbf{P}$ -certificates.** In the full version of the paper [19], we show that unique two-message  $\mathbf{P}$  certificates can be constructed from any “message hiding encoding scheme” [39] and injective one-way functions. Furthermore, we show that the  $\mathbf{P}$  certificates instantiated using the specific message hiding encoding of [39] has delegatable CRS generation and a simple verification procedure.

### 3 Our Protocol

Our constant-round concurrent  $\mathcal{ZK}$  protocol relies on the following primitives:

1. A non-interactive *perfectly binding* commitment scheme  $\text{com}$ . We assume without loss of generality that  $\text{com}$  only needs  $n$  bits of randomness to commit to any  $n$ -bit string, (as it can always expand these  $n$  bits into a longer sequence using a PRG).

The requirement for a *perfectly binding* commitment scheme can be weakened to rely only on a *statistically binding* commitment scheme. See Remark 2 in the full version of the paper [19] for more details.

2. A strong (two-message)  $\mathbf{P}$ -certificate system  $(\text{Gen}, \text{P}_{\text{cert}}, \text{V}_{\text{cert}})$  with delegatable CRS generation  $\text{Gen} = (\text{Setup}, \text{PreGen}, \text{CRSGen})$  (and simple verification). The strong soundness property is associated with parameter  $T(\cdot)$  and  $C(\cdot)$ , where  $T(\cdot)$  is a “nice” super-polynomial function and  $C(\cdot)$  is a “nice” super-constant function. The uniqueness property ensures that for every string  $\text{CRS}$ , there exists at most one proof  $\pi$  that is accepted by  $\text{V}_{\text{cert}}(1^n, \text{CRS}, \pi) = 1$ . This allows us to define the following deterministic oracle  $\mathcal{O}_{\text{V}_{\text{cert}}}^n$ , which will be used in the CZK protocol later.

$$\mathcal{O}_{\text{V}_{\text{cert}}}^n(\text{CRS}) = \begin{cases} \pi & \text{If there exists unique } \pi \text{ s.t. } \text{V}_{\text{cert}}(1^n, \text{CRS}, \pi) = 1 \\ \perp & \text{otherwise} \end{cases}$$

We call  $\mathcal{O}_{\text{V}_{\text{cert}}}^n$  the  $\mathbf{P}$ -certificate oracle. Additionally, we consider a universal emulator  $\text{Emulator}^n$  that on input  $(P, x, O)$  emulates the execution of a *deterministic oracle machine*  $P$  on input  $x$  with oracle  $\mathcal{O}_{\text{V}_{\text{cert}}}^n$  as follows: It parses  $O$  as a vector; to answer the  $i^{\text{th}}$  query  $\text{CRS}_i$  from  $P$ , it checks whether  $O_i$  is the right answer from this CRS (i.e.,  $\text{V}_{\text{cert}}(1^n, \text{CRS}_i, O_i) = 1$ ); if so, it returns  $O_i$  to  $P$ ; otherwise, it aborts and outputs  $\perp$ . Finally, the emulator outputs the output of  $P$ .

For simplicity, we assume that the lengths of the CRS, the proof  $\pi$ , and the digest of statement  $d$  are all bounded by  $n$ , the security parameter. This is without loss of generality, and can be achieved by scaling down the security parameter.

We assume by default that the two message  $\mathbf{P}$ -certificate system has a simple verification procedure (i.e.,  $\text{V}_{\text{cert}}$  depends only on  $1^k, \text{CRS}, \pi$ , but not the statement); this is w.l.o.g., since our instantiation based on the message hiding encoding of [39] satisfies this property. But this is not necessary. See Remark 3 in the full version of the paper [19] on how to avoid using this property.

3. A family of hash functions  $\{\mathcal{H}_n\}_n$ : to simplify the exposition, we here assume that both  $\text{com}$  and  $\{\mathcal{H}_n\}_n$  are collision resistant against circuits of size  $T'(\cdot)$ , where  $T'(\cdot)$  is “nice” super-polynomial function.

As in [4], this assumption can be weakened to just collision resistance against polynomial-size circuits by modifying the protocol to use a “good” error-correcting code ECC (i.e., with constant distance and with polynomial-time encoding and decoding), and replace commitments  $\text{com}(h(\cdot))$  with

$\text{com}(h(\text{ECC}(\cdot)))$ . See Remark 1 in the full version of the paper [19] for more discussion.

4. An indistinguishability obfuscator  $i\mathcal{O}$  for circuits.
5. A constant-round WIUA argument system, a constant-round  $WISSP$  proof system, and a constant-round  $\mathcal{ZK}$  argument system.

Let us now turn to specifying the protocol  $(P, V)$ . The protocol makes use of three parameters:  $m(\cdot)$  is a polynomial that upper bounds the number of concurrent sessions;  $\Gamma(\cdot)$  is a “nice” super-polynomial function such that  $T(n), T'(n) \in \Gamma(n)^{\omega(1)}$ , and  $D(\cdot)$  is a “nice” super-constant function such that  $D(n) \leq C(n)$ . Let  $m = m(n)$ ,  $\Gamma = \Gamma(n)$  and  $D = D(n)$ . In the description below, when discussing  $\mathbf{P}$ -certificates, we always consider the language  $L_D$ . For simplicity, below we do not explicitly discuss about the length of the random strings used by various algorithms. The prover  $P$  and the verifier  $V$ , on common input  $1^n$  and  $x$  and private input a witness  $w$  to  $P$ , proceed as follow:

**Phase 1—Program Slot:**  $P$  and  $V$  exchanges the following three messages.

- (a)  $V$  chooses a randomly sampled hash function  $h \leftarrow \mathcal{H}_n$ .
- (b)  $P$  sends a commitment  $c$  to  $0^n$  using  $\text{com}$ , and random coins  $\rho_1$ .
- (c)  $V$  replies with a random “challenge”  $r$  of length  $4n$ .

We call  $(c, r)$  the program-slot.

NOTE: In simulation, the simulator commits to a program  $\tilde{S}_1$ .

**Phase 2—Commit to Statement:**  $P$  and  $V$  exchanges the following messages.

- (a)  $P$  sends a commitment  $c_2$  to  $0^n$  using  $\text{com}$ , and random coins  $\rho_2$ .
- (b)  $P$  gives a WIUA argument of the statement that either  $x \in L$  OR there exists  $\tilde{S}_1 \in \{0, 1\}^{\Gamma(n)}$ ,  $j \in [m]$ ,  $s \in \{0, 1\}^n$ ,  $\pi \in \{0, 1\}^n$ ,  $\sigma \in \{0, 1\}^{\Gamma(n)}$ ,  $\rho$ ,  $\rho_2$  such that,

**Knowledge of Statement:**  $c_2 = \text{com}(h(q); \rho_2)$ , where  $q \in \{0, 1\}^{3\Gamma}$ .

**Correctness of Statement:** The statement  $q$  satisfies

- USE OF EMULATOR:  $q$  is parsed into  $(\text{Emulator}^n, (\tilde{S}_1, (1^n, j, s), \sigma), r)$ .
- PROGRAM CONSISTENCY:  $c = \text{com}(h(\tilde{S}_1); \rho)$ .

If the argument is not accepting,  $V$  aborts.

NOTE: By definition of the emulator  $\text{Emulator}^n$ , on input  $(\tilde{S}_1, (1^n, j, s), \sigma)$ , it will emulate the execution of the deterministic oracle machine  $\tilde{S}_1(1^n, j, s)$  with oracle  $\mathcal{O}_{V_{\text{cert}}}^n$  using answers stored in vector  $\sigma$ .

The purpose of this phase is twofold: First, it enforces a cheating prover to commit to the “trapdoor” statement before the CRS of the  $\mathbf{P}$ -certificate is generated, and hence the soundness of the protocol only relies on the selective soundness of the  $\mathbf{P}$ -certificate. Second, it checks whether the “trapdoor” statement has the right structure, in particular, the statement is about whether  $\tilde{S}_1^{\mathcal{O}_{V_{\text{cert}}}}(1^n, j, s) = r$ , when the oracle is emulated by  $\text{Emulator}^n$  using  $\sigma$ , who checks the correctness of the proofs in  $\sigma$ .

Note that the soundness of the protocol will crucially rely on the fact that the input to  $\tilde{S}_1$  has length at most  $3n$ , much smaller than the length,  $4n$ , of the output  $r$  (and the deterministic oracle  $\mathcal{O}_{V_{\text{cert}}}$  is emulated correctly by  $\text{Emulator}^n$ ). On the other hand, in the simulation, the simulator will commit

to the “trapdoor” statement,  $q = (\text{Emulator}^n, (\tilde{S}_1, (1^n, j, s), \sigma), r)$  in order to “cheat”.

**Phase 3—Delegate Public Statement Processing:**  $V$  delegates the public statement processing to  $P$ :

- (a)  $V$  generates  $(PP, K) = \text{Setup}(1^n, D; \rho_{\text{Setup}})$  using random coins  $\rho_{\text{Setup}}$ , and sends  $PP$ .
- (b)  $P$  sends a commitment  $c_3$  to  $0^n$  using  $\text{com}$ , and random coins  $\rho_3$ .
- (c)  $P$  gives a WIUA argument of the statement that either  $x \in L$  OR there exists,  $d \in \{0, 1\}^n$ ,  $q \in \{0, 1\}^{3T}$ ,  $\rho_2, \rho_3$ , such that,

**Statement Consistency:**  $c_2 = \text{com}(h(q); \rho_2)$ .

**Digest Consistency:**  $c_3 = \text{com}(d; \rho_3)$ .

**Correctness of Digest:**  $d = \text{PreGen}(PP, q)$ .

If the argument is not accepting,  $V$  aborts.

NOTE: The purpose of this Phase is to allow the verifier to delegate the computation of the digest of the statement to  $P$ . In simulation, the simulator will compute, commit to and prove correctness of  $d = \text{PreGen}(PP, q)$ .  $V$  cannot compute  $d$  itself, since (1) it does not know the “trapdoor” statement  $q$  and (2) the computation takes  $\text{poly}(n, |q|)$ , which is too expensive for the verifier.

**Phase 4—Delegate Private CRS Generation:**  $V$  delegates the private CRS generation to  $P$ :

- (a)  $V$  sends the indistinguishability obfuscation  $\Lambda \stackrel{\$}{\leftarrow} i\mathcal{O}(\mathbf{P})$  of program  $\mathbf{P} = \mathbf{P}^{n, c_3, PP, K, \rho_{\text{CRSGen}}}$  with  $c_4, K$ , and a random string  $\rho_{\text{CRSGen}}$  hard-wired in.  $\mathbf{P}$  on input  $(d', \rho')$  checks whether  $c_3 = \text{com}(d', \rho')$  and outputs  $\kappa = \text{CRSGen}(PP, K, d; \rho_{\text{CRSGen}})$  if it is the case, and  $\perp$  otherwise. The functionality of  $\mathbf{P}$  is described formally in Fig. 1.

- (b)  $V$  gives a  $\mathcal{ZK}$  argument of the statement that there exists  $K \in \{0, 1\}^n$ ,  $\rho_{\text{Setup}}, \rho_{\text{CRSGen}}, \rho_{i\mathcal{O}}$ , such that,

**Correctness of Public Parameter:**  $(PP, K) = \text{Setup}(1^n, D; \rho_{\text{Setup}})$ .

**Correctness of Obfuscation:**  $\Lambda = i\mathcal{O}(\mathbf{P}^{c_3, PP, K, \rho_{\text{CRSGen}}}; \rho_{i\mathcal{O}})$

If the argument is not accepting,  $P$  aborts.

- (c)  $P$  sends commitment  $c_4$  of  $0^n$  using  $\text{com}$  and random coins  $\rho_4$ .
- (d)  $P$  gives a  $WISSP$  proof of the statement that either  $x \in L$  OR there exists  $\text{CRS} \in \{0, 1\}^n$ ,  $d' \in \{0, 1\}^n$ ,  $\rho', \rho_4$ , such that,

**CRS Consistency:**  $c_4 = \text{com}(\text{CRS}; \rho_4)$ .

**Correctness of CRS:**  $\text{CRS} = (PP, \kappa)$  and  $\kappa = \bar{\mathbf{P}}(d', \rho')$ .

If the proof is not accepting,  $V$  aborts.

NOTE: The purpose of this Phase is to allow the verifier to delegate the computation of CRS to  $P$ . In simulation, the simulator will compute, commit to, and prove correctness of  $\text{CRS} = (PP, \kappa)$ , with  $\kappa = \bar{\mathbf{P}}(d, \rho_3)$ .  $V$  cannot compute  $\kappa$  itself, even though the computation takes only polynomial time in  $n$ , since  $d$  cannot be revealed to  $V$  in order to ensure the indistinguishability of the simulation. On the other hand, to ensure the “privacy” of the CRS computation,  $V$  delegates this computation via obfuscation.



**Circuit P** =  $\mathbf{P}^{n, c_3, PP, K, \rho_{\text{CRSGen}}}$ : On input  $(d', \rho')$  where  $d' \in \{0, 1\}^n$  and  $\rho' \in \{0, 1\}^n$ , does:

- (a) Check if  $c_3 = \text{com}(d'; \rho')$ ; if not, output  $\perp$ .
- (b) Otherwise output  $\kappa = \text{CRSGen}(PP, K, d'; \rho_{\text{CRSGen}})$ .

**Circuit Q** =  $\mathbf{Q}^{n, c_3, \kappa}$ : On input  $(d', \rho')$  where  $d' \in \{0, 1\}^n$  and  $\rho' \in \{0, 1\}^n$ , does:

- (a) Check if  $c_3 = \text{com}(d'; \rho')$ ; if not, output  $\perp$ .
- (b) Otherwise output  $\kappa$ .

The above circuits are padded to their maximum size.

**Fig. 1.** Circuits used in the construction and proof of CZK protocol  $\langle P, V \rangle$

**Phase 5—Final Proof:**  $P$  gives the final proof:

- (a)  $P$  gives a *WISSP* proof of the statement that either  $x \in L$  OR there exists  $\pi \in \{0, 1\}^n$ ,  $\text{CRS} \in \{0, 1\}^n$ ,  $\rho_4$ , such that,

**CRS Consistency:**  $c_4 = \text{com}(\text{CRS}; \rho_4)$ ,

**Proof Verification:**  $\pi$  verifies w.r.t.  $\text{CRS}$ ,  $V_{\text{cert}}(1^n, \text{CRS}, \pi) = 1$ .

$V$  accepts if the proof is accepting.

NOTE: In simulation, the simulator computes proof  $\pi \stackrel{\$}{\leftarrow} P_{\text{cert}}(1^k, D, q, \text{CRS})$ , and succeed in the final proof by using  $\pi$  and  $\text{CRS}, \rho_4$  generated in the last phase as “trapdoor” witness.

**Theorem 1.** Assume indistinguishability obfuscation for  $\mathbf{P}/\text{poly}$ , an injective pseudo-random generator, and collision resistant hash functions that are super-polynomially secure. Then, the above protocol  $\langle P, V \rangle$  is a concurrent  $\mathcal{ZK}$  argument system for NP.

The completeness of the protocol follows from the completeness of the WIUA argument of knowledge, *WISSP*, and the  $\mathcal{ZK}$  argument. In the next subsection, we describe the concurrent zero knowledge simulator. The analysis of the simulator and the proof of concurrent  $\mathcal{ZK}$  property, as well as the soundness proof, are provided in the full version of the paper [19].

### 3.1 Construction of the Simulator

The goal of our simulator is to try to “commit to its own code” and prove about its own execution using  $\mathbf{P}$ -certificates in a way that prevents a blow-up in the running-time. Note that the only expensive part of this process is the generation of the  $\mathbf{P}$ -certificates  $\pi$ ; the rest of the computation has *a-priori* bounded complexity (depending only on the size and running-time of  $V^*$ ). To take advantage of this observation, we thus have the simulator only commit to an oracle program that generates prover messages (in identically the same way as the actual simulator), but getting certificates  $\pi$  from the  $\mathbf{P}$ -certificate oracle.

To describe the actual simulator  $S$ , let us first describe two “helper” simulators  $S_1, S_2$ . Roughly speaking,  $S_1$  is an interactive machine that simulates prover messages in a “right” interaction with  $V^*$ . Additionally,  $S_1$  excepts to have access to oracle  $\mathcal{O}_{Vcert}$  on the “left”, in particular, at any point, it can send a CRS string  $\text{CRS}$  and gets back the  $\pi = \mathcal{O}_{Vcert}(\text{CRS})$  the unique accepting certificate w.r.t. this CRS (or  $\perp$ , if such a certificate does not exist); the oracle will be simulated by  $S_2$ , who provides these “left” certificates.

Let us turn to a formal description of the  $S_1$  and  $S_2$ . To simplify the exposition, we assume w.l.o.g that  $V^*$  has its non-uniform advice  $z$  hard-coded, and is deterministic (as it can always get its random tape as non-uniform advice).

On a high-level,  $S_1(1^n, x, M, s, \ell)$  acts as a prover in a “right” interaction, communicating with a concurrent verifier  $V^*$ , while accessing oracle on the “left”. (The input  $x$  is the statement to be proved, the input  $M$  will later be instantiated with the code of  $S_1$ , and the input  $(s, \ell)$  is used to generate the randomness for  $S_1$ ;  $s$  is the seed for the forward secure pseudorandom generator  $g$ , and  $\ell$  is the number of  $n$ -bit long blocks to be generated using  $g$ .) A communication round in the “right” interaction with  $V^*$  refers to a verifier message (sent by  $V^*$ ) followed by a prover message (sent by  $S_1$ ).

PROCEDURE OF SIMULATOR  $S_1$ : Let us now specify how  $S_1$  generates prover messages in its “right” interaction with  $V^*$ .  $S_1^{\mathcal{O}_{Vcert}}(1^n, x, M, s, \ell)$  acts as follows:

**Generate Randomness:** Upon invocation,  $S_1$  generates its “random-tape” by expanding the seed  $s$ ; more specifically, let  $(s_\ell, s_{\ell-1}, \dots, s_1), (q_\ell, q_{\ell-1}, \dots, q_1)$  be the output of  $g(s, \ell)$ . We assume without loss of generality that  $S_1$  only needs  $n$  bits of randomness to generate any prover message (it can always expand these  $n$  bits into a longer sequence using a PRG); in order to generate its  $j^{\text{th}}$  prover message, it uses  $q_j$  as randomness.

**Simulate Phase 1—Commit to Its Own Code:** Upon receiving a hash function  $h_i$  in session  $i$  during the  $j^{\text{th}}$  communication round,  $S_1$  provides a commitment  $c_i$  to (the hash of) the deterministic oracle machine  $\tilde{S}_1(1^n, \alpha, s') = \text{wrap}(M(1^n, x, M, s', \alpha), V^*, \alpha)$ , where  $\text{wrap}(A, B, \alpha)$  is the program that lets  $A$  communicate with  $B$  for  $\alpha$  rounds, while allowing  $A$  to access oracle  $\mathcal{O}_{Vcert}$ , and finally outputting  $B$ ’s message in the  $j^{\text{th}}$  communication round. NOTE: That is,  $\tilde{S}_1(1^n, \alpha, s', \tau)$  emulates  $\alpha$  rounds of an execution between  $S_1$  and  $V^*$  where  $S_1$  expands out the seed  $s'$  into  $\alpha$  blocks of randomness and additionally have access to  $\mathcal{O}_{Vcert}$ .

**Simulate Phase 2—Commit to the Trapdoor Statement:** Upon receiving a challenge  $r_i$  in session  $i$  during the  $j^{\text{th}}$  communication round,  $S_1$  needs to commit to the “trapdoor” statement it will later prove in the final proof. To do so, it prepares statement  $q_i = (\text{Emulator}^n, (\tilde{S}_1, (1^n, j, s_j), \tau_{j-1}), r_i)$ , where  $\tau_{j-1}$  is the list of oracle answers received by  $S_1$  in the first  $j - 1$  communication rounds.

NOTE: That is, the “trapdoor” statement is that the execution of  $\tilde{S}_1(1^n, j, s_j)$ , emulated by  $\text{Emulator}^n$ , outputs  $r$ , when its  $k^{\text{th}}$  oracle queries is answered using  $\tau_{j-1, k}$ ; additionally, the validity of each answer is checked by  $\text{Emulator}^n$  (i.e., the answer must be an accepting proof w.r.t. the query CRS string).

By construction of  $\tilde{S}_1$ , this means after  $j$  communication rounds between  $S_1$  and  $V^*$ , where  $S_1$  uses randomness expanded out from  $s_j$ , and oracle answers  $\tau_{j-1}$ ,  $V^*$  outputs  $r_i$  in the  $j^{\text{th}}$  communication round. Note that since we only require  $\tilde{S}_1$  to generate the  $j^{\text{th}}$  verifier message, giving him the seed  $(s_j, j)$  as input suffices to generate all prover messages in rounds  $j' < j$ . It follows from the consistency requirement of the forward secure PRG that  $\tilde{S}_1$  using  $(s_j, j)$  as seed will generate the exact same random sequence for the  $j - 1$  first blocks as if running  $\tilde{S}_1$  using  $(s, \ell)$  as seed. Therefore, the “trapdoor” statement holds.

In later communication rounds, when  $S_1$  receives a message from  $V^*$  belonging to the WIUA in Phase 2 of session  $i$ ,  $S_1$  proves honestly that it knows the statement  $q_i$  it is committing to in session  $i$ , and the statement is correctly formatted and consistent with the program  $\tilde{S}_1$  committed to in Phase 1 of session  $i$ .

**Simulate Phase 3—Process the Trapdoor Statement:** Upon receiving a public parameter  $PP_i$  in session  $i$  during the  $j^{\text{th}}$  communication round,  $S_1$  needs to commit to the digest  $d_i$  of the “trapdoor” statement  $q_i$  of session  $i$ . To do so, it computes honestly  $d_i \stackrel{s}{\leftarrow} \text{PreGen}(PP_i, q_i)$  and commits to  $d_i$  using  $\text{com}$ , and randomness  $\rho_i$ .

In later communication rounds, when  $S_1$  receives a message from  $V^*$  belonging to the WIUA in Phase 3 of session  $i$ ,  $S_1$  proves honestly that it knows  $d_i$  committed to in Phase 3 of session  $i$  and it is computed correctly w.r.t.  $PP_i$  and a statement  $q_i$  committed to in Phase 2 of session  $i$ .

**Simulate Phase 4—Compute the CRS:** Upon receiving an obfuscated program  $\Lambda_i$ ,  $S_1$  acts as an honest verifier of the  $\mathcal{ZK}$  argument to verify that  $PP_i$  and  $\Lambda_i$  in session  $i$  are correctly generated. Upon receiving the last message of the  $\mathcal{ZK}$  argument, in the  $j^{\text{th}}$  communication round,  $S_1$  needs to commit to the  $\text{CRS}_i$  of session  $i$ . To do so, it computes  $\kappa_i = \Lambda_i(d_i, \rho_i)$ . If the output is  $\perp$ ,  $S_1$  aborts. Otherwise, it commits to  $\text{CRS}_i = (PP_i, \kappa_i)$  using  $\text{com}$ .

In later communication rounds, when  $S_1$  receives a message from  $V^*$  belonging to the  $WISSP$  in Phase 4 of session  $i$ ,  $S_1$  proves honestly that it knows  $\kappa_i$  committed to in Phase 4 of session  $i$  and it is computed correctly w.r.t.  $\Lambda_i$  and a digest  $d_i$  committed to in Phase 3 of session  $i$ .

**Simulate Phase 5—Prove the Trapdoor Statement Using P-certificate:** Upon receiving the last message from  $V^*$  in Phase 4 of session  $i$ , during the  $j^{\text{th}}$  communication round,  $S_1$  needs to prove in the  $WISSP$  proof that there is a **P**-certificate that verifies the validity of the “trapdoor” statement  $q_i$  w.r.t. the CRS string  $\text{CRS}_i$  committed to in Phase 4 of session  $i$ . To do so, it sends query  $\text{CRS}_i$  to its oracle  $\mathcal{O}_{V_{\text{cert}}}$ , and obtains answer  $\pi_i$ . It aborts if  $\pi_i = \perp$ . Otherwise,  $S_1$  provides an honest  $WISSP$  that  $V_{\text{cert}}(1^n, \text{CRS}_i, \pi_i) = 1$  w.r.t.  $\text{CRS}_i$  which is the committed value in Phase 4 of session  $i$ .

PROCEDURE OF SIMULATOR  $S_2$ :  $S_2(1^n, x, M, s, \ell)$  internally emulates  $\ell$  messages of an execution between  $S_1(1^n, x, M, s, \ell)$  and  $V^*$ , and simulates the oracle  $\mathcal{O}_{V_{\text{cert}}}$  for  $S_1$ . In a communication round  $j$  when  $S_1$  sends an oracle query  $\text{CRS}_i$  for a session  $i$ ,  $S_2$  generates a certificate  $\pi_i$  of the statement  $q_i = (\text{Emulator}^n, (\tilde{S}_1,$

$(1^n, j', s_{j'}, \tau_{j'-1}, r_{j'})$  w.r.t.  $\text{CRS}_i$ , that is,  $\pi_i \stackrel{\$}{\leftarrow} \text{P}_{\text{cert}}(1^n, D, q_i, \text{CRS}_i)$  (where  $j'$  is the round in which the challenge  $r_i$  is sent by  $V^*$ ,  $q_i$  and  $\text{CRS}_i$  are generated by  $S_1$  (emulated internally by  $S_2$ ) in Phase 2 and 4 of session  $i$ ).  $S_2$  checks if indeed  $\text{V}_{\text{cert}}(1^n, \text{CRS}_i, \pi_i) = 1$ , it outputs fail if this is not the case, and otherwise, feeds  $\pi_i$  to  $S_1$ . Finally,  $S_2$  outputs its view (which in particular, contains the view of  $V^*$ ) at the end of the execution.

**PROCEDURE OF THE FINAL SIMULATOR  $S$ :** The final simulator  $S(1^n, x)$  simply runs  $S_2(1^n, x, S_1, s, T(n + |x|))$ , where  $s$  is a uniformly random string of length  $n$  and  $T(n + |x|)$  is a polynomial upper-bound on the number of messages sent by  $V^*$  given the common input  $1^n, x$ , and extracts out and outputs, the view of  $V^*$  from the output of  $S_2$ . (In case that  $S_2$  outputs fail,  $S$  outputs fail as well.)

Due to the lack of space, the analysis of the simulator, including its running time, and the correctness of its output distribution is provided in the full version of the paper [19], which also contains the soundness proof.

## References

1. Ananth, P., Boneh, D., Garg, S., Sahai, A., Zhandry, M.: Differing-inputs obfuscation and applications (2013)
2. Barak, B.: How to go beyond the black-box simulation barrier. In: FOCS, pp. 106–115 (2001)
3. Barak, B.: Constant-round coin-tossing with a man in the middle or realizing the shared random string model. In: FOCS, Washington, DC, USA, 2002, pp. 345–355. IEEE Computer Society (2002)
4. Barak, B., Goldreich, O.: Universal arguments and their applications. *SIAM J. Comput.* **38**(5), 1661–1694 (2008)
5. Barak, B., Goldreich, O., Goldwasser, S., Lindell, Y.: Resetably-sound zero-knowledge and its applications. In: FOCS, pp. 116–125 (2001)
6. 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)
7. Barak, B., Sahai, A.: How to play almost any mental game over the net - concurrent composition via super-polynomial simulation. In: FOCS, pp. 543–552 (2005)
8. Bellare, M., Palacio, A.: Towards plaintext-aware public-key encryption without random oracles. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 48–62. Springer, Heidelberg (2004)
9. Bitansky, N., Garg, S., Lin, H., Pass, R., Telang, S.: Succinct randomized encodings and obfuscations. Manuscript (subsuming an early version appearing as Succinct Garbling Schemes and Applications [Lin-Pass, Eprint Report 2014/766]) (2014)
10. Bitansky, N., Paneth, O.: From the impossibility of obfuscation to a new non-black-box simulation technique. In: FOCS, pp. 223–232 (2012)
11. Boyle, E., Chung, K.-M., Pass, R.: On extractability obfuscation. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 52–73. Springer, Heidelberg (2014)
12. Boyle, E., Pass, R.: Limits of extractability assumptions with distributional auxiliary input. Cryptology ePrint Archive, Report 2013/703 (2013). <http://eprint.iacr.org/>

13. Canetti, R., Holmgren, J., Jain, A., Vaikuntanathan, V.: Indistinguishability obfuscation of iterated circuits and ram programs. Cryptology ePrint Archive, Report 2014/769 (2014)
14. Canetti, R., Kilian, J., Petrank, E., Rosen, A.: Black-box concurrent zero-knowledge requires  $\tilde{\omega}(\log n)$  rounds. In: STOC, pp. 570–579 (2001)
15. Canetti, R., Lin, H., Paneth, O.: Public-coin concurrent zero-knowledge in the global hash model. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 80–99. Springer, Heidelberg (2013)
16. Chung, K., Lin, H., Pass, R.: Constant-round concurrent zero knowledge from p-certificates. In: 54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26–29 October 2013, Berkeley, CA, USA, pp. 50–59 (2013)
17. Chung, K.-M., Ostrovsky, R., Pass, R., Venkatasubramanian, M., Visconti, I.: 4-Round resettably-sound zero knowledge. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 192–216. Springer, Heidelberg (2014)
18. Chung, K., Pass, R., Seth, K.: Non-black-box simulation from one-way functions and applications to resettable security. In: Symposium on Theory of Computing Conference, STOC 2013, 1–4 June 2013, Palo Alto, CA, USA, pp. 231–240 (2013)
19. Chung, K.-M., Lin, H., Pass, R.: Constant-round concurrent zero-knowledge from indistinguishability obfuscation. Cryptology ePrint Archive, Report 2014/991 (2014). <http://eprint.iacr.org/>
20. Chung, K.-M., Ostrovsky, R., Pass, R., Visconti, I.: Simultaneous resettable security from one-way functions (2013)
21. Damgård, I.B.: Towards practical public key systems secure against chosen ciphertext attacks. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 445–456. Springer, Heidelberg (1992)
22. Damgård, I.B.: Efficient concurrent zero-knowledge in the auxiliary string model. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 418–430. Springer, Heidelberg (2000)
23. Deng, Y., Goyal, V., Sahai, A.: Resolving the simultaneous resettable security conjecture and a new non-black-box simulation strategy. In: FOCS, pp. 251–260 (2009)
24. Dwork, C., Naor, M., Sahai, A.: Concurrent zero-knowledge. J. ACM **51**(6), 851–898 (2004)
25. Dwork, C., Sahai, A.: Concurrent zero-knowledge: reducing the need for timing constraints. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 177–190. Springer, Heidelberg (1998)
26. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: Proceedings of FOCS 2013 (2013)
27. Garg, S., Gentry, C., Halevi, S., Wichs, D.: On the implausibility of differing-inputs obfuscation and extractable witness encryption with auxiliary input. Technical report, Cryptology ePrint Archive, Report 2013/860, 6 (2013)
28. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Mitzenmacher, M. (ed.) 41st Annual ACM Symposium on Theory of Computing, 31 May - 2 June 2009, Bethesda, Maryland, USA, pp. 169–178. ACM Press (2009)
29. Gentry, C., Lewko, A., Sahai, A., Waters, B.: Indistinguishability obfuscation from the multilinear subgroup elimination assumption. Cryptology ePrint Archive, Report 2014/309 (2014). <http://eprint.iacr.org/>
30. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM J. Comput. **18**(1), 186–208 (1989)

31. Goyal, V.: Non-black-box simulation in the fully concurrent setting. In: Boneh, D., Roughgarden, T., Feigenbaum, J. (eds.) 45th Annual ACM Symposium on Theory of Computing, 1–4 June 2013, Palo Alto, CA, USA, pp. 221–230. ACM Press (2013)
32. Goyal, V., Jain, A.: On the round complexity of covert computation. In: STOC, pp. 191–200 (2010)
33. Gupta, D., Sahai, A.: On constant-round concurrent zero-knowledge from a knowledge assumption. Cryptology ePrint Archive, Report 2012/572 (2012). <http://eprint.iacr.org/>
34. Hada, S., Tanaka, T.: On the existence of 3-round zero-knowledge protocols. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 408–423. Springer, Heidelberg (1998)
35. Ishai, Y., Kushilevitz, E.: Randomizing polynomials: a new representation with applications to round-efficient secure computation. In: 41st Annual Symposium on Foundations of Computer Science, 12–14 November 2000, Redondo Beach, California, USA, pp. 294–304. IEEE Computer Society Press (2000)
36. Ishai, Y., Pandey, O., Sahai, A.: Public-coin differing-inputs obfuscation and its applications. Cryptology ePrint Archive, Report 2014/942 (2014). <http://eprint.iacr.org/>
37. Kilian, J., Petrank, E.: Concurrent and resettable zero-knowledge in poly-logarithmic rounds. In: STOC, pp. 560–569 (2001)
38. Kilian, J., Petrank, E., Rackoff, C.: Lower bounds for zero knowledge on the internet. In: FOCS, pp. 484–492 (1998)
39. Koppula, V., Lewko, A.B., Waters, A.B.: Indistinguishability obfuscation for Turing machines with unbounded memory. Cryptology ePrint Archive, Report 2014/925 (2014). <http://eprint.iacr.org/>
40. Lindell, Y.: Bounded-concurrent secure two-party computation without setup assumptions. In: STOC, pp. 683–692 (2003)
41. Micali, S.: Computationally sound proofs. *SIAM J. Comput.* **30**(4), 1253–1298 (2000)
42. Naor, M.: On cryptographic assumptions and challenges. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 96–109. Springer, Heidelberg (2003)
43. Pandey, O., Prabhakaran, M., Sahai, A.: Obfuscation-based non-black-box simulation and four message concurrent zero knowledge for NP. Cryptology ePrint Archive, Report 2013/754 (2013). <http://eprint.iacr.org/>
44. Pass, R.: Simulation in quasi-polynomial time, and its application to protocol composition. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656. Springer, Heidelberg (2003)
45. Pass, R.: Bounded-concurrent secure multi-party computation with a dishonest majority. In: STOC, New York, NY, USA, pp. 232–241. ACM (2004)
46. Pass, R., Rosen, A.: Bounded-concurrent secure two-party computation in a constant number of rounds. In: FOCS, pp. 404–413 (2003)
47. Pass, R., Rosen, A.: How to simulate using a computer virus. Unpublished manuscript (2003)
48. Pass, R., Rosen, A.: Concurrent non-malleable commitments. In: FOCS, pp. 563–572 (2005)
49. Pass, R., Rosen, A.: New and improved constructions of non-malleable cryptographic protocols. In: STOC, pp. 533–542 (2005)
50. Pass, R., Rosen, A., Tseng, W.-L.D.: Public-coin parallel zero-knowledge for NP. *J. Cryptology* **26**, 1–10 (2011)

51. Pass, R., Seth, K., Telang, S.: Indistinguishability obfuscation from semantically-secure multilinear encodings. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 500–517. Springer, Heidelberg (2014)
52. Pass, R., Tseng, W.-L.D., Venkatasubramanian, M.: Concurrent zero-knowledge, revisited. *J. Cryptology* **27**, 45–66 (2012)
53. Pass, R., Venkatasubramanian, M.: Private coins versus public coins in zero-knowledge proof systems. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 588–605. Springer, Heidelberg (2010)
54. Popper, K.: *Conjectures and Refutations: The Growth of Scientific Knowledge*. Routledge, New York (1963)
55. Prabhakaran, M., Rosen, A., Sahai, A.: Concurrent zero knowledge with logarithmic round-complexity. In: FOCS, pp. 366–375 (2002)
56. Richardson, R., Kilian, J.: On the concurrent composition of zero-knowledge proofs. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 415–432. Springer, Heidelberg (1999)
57. Rivest, R.L., Adleman, L., Dertouzos, M.L.: On data banks and privacy homomorphisms (1978)
58. Rosen, A.: A note on the round-complexity of concurrent zero-knowledge. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 451–468. Springer, Heidelberg (2000)

# Indistinguishability Obfuscation from Compact Functional Encryption

Prabhanjan Ananth<sup>1</sup> and Abhishek Jain<sup>2</sup>(✉)

<sup>1</sup> University of California, Los Angeles, USA  
prabhanjan@cs.ucla.edu

<sup>2</sup> Johns Hopkins University, Baltimore, USA  
abhishek@cs.jhu.edu

**Abstract.** The arrival of indistinguishability obfuscation (*iO*) has transformed the cryptographic landscape by enabling several security goals that were previously beyond our reach. Consequently, one of the pressing goals currently is to construct *iO* from well-studied standard cryptographic assumptions.

In this work, we make progress in this direction by presenting a reduction from *iO* to a natural form of public-key functional encryption (FE). Specifically, we construct *iO* for general functions from any single-key FE scheme for  $\text{NC}^1$  that achieves selective, indistinguishability security against sub-exponential time adversaries. Further, the FE scheme should be *compact*, namely, the running time of the encryption algorithm must only be a polynomial in the security parameter and the input message length (and not in the function description size or its output length).

We achieve this result by developing a novel *arity amplification technique* to transform FE for single-ary functions into FE for multi-ary functions (aka multi-input FE). Instantiating our approach with known, non-compact FE schemes, we obtain the first constructions of multi-input FE for constant-ary functions based on standard assumptions.

Finally, as a result of independent interest, we construct a compact FE scheme from randomized encodings for Turing machines and learning with errors assumption.

## 1 Introduction

The ability to cryptographically obfuscate computer programs holds great prospects for securing the future digital world. While general-purpose program

---

P. Ananth—Research supported in part from a DARPA/ONR PROCEED award, NSF Frontier Award 1413955, NSF grants 1228984, 1136174, 1118096, and 1065276. This material is based upon work supported by the Defense Advanced Research Projects Agency through the U.S. Office of Naval Research under Contract N00014-11-1-0389. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense, the National Science Foundation, or the U.S. Government.

A. Jain—The author is partly funded by NSF CNS-1414023. Supported in part by a DARPA Safeware grant and NSF CNS-1414023.



obfuscation remained an elusive goal for several decades, this changed recently with the seminal work of Garg et al. [26] who gave the first candidate construction of indistinguishability obfuscation [8] ( $iO$ ) for  $P/poly$ . Since then,  $iO$  has been used to realize several advanced cryptographic primitives that were previously beyond our reach, including deniable encryption [45], collusion-resistant functional encryption [26], round-optimal multiparty computation [25], and so on. Indeed, by now,  $iO$  has been established as a central hub of cryptography.

The tremendous appeal of  $iO$  motivates the goal of constructing it from well-studied, standard cryptographic assumptions. However, not much is known in this direction. The security of candidate  $iO$  constructions, initiated by [7]<sup>1</sup>, is proven in the “generic graded encoding model” and lacks a reduction in the standard model. The recent works of Pass et al. [43] and Gentry et al. [33] seek to rectify this situation by constructing  $iO$  from various assumptions on multilinear maps [24]. In particular, Pass et al. [43] reduce the security of their construction to an “uber assumption” on multilinear maps while Gentry et al. [33] provide a reduction to the “multilinear subgroup elimination assumption” (stated in their paper) on composite-order multilinear maps [22].

Till date, these remain the only known constructions of general-purpose  $iO$ . Further, all of them rely on a common cryptographic primitive, namely, multilinear maps. This is an unsatisfactory situation, especially in light of several recent attacks on multilinear maps [13, 21, 23, 30]. This calls for new constructions of  $iO$  from other, more familiar cryptographic primitives.

## 1.1 This Work

In this work, we make progress in this direction by providing a new construction of  $iO$  based on a natural form of functional encryption (FE). Along the way, we also obtain new results on multi-input FE [35] that significantly improve upon the prior results.

*I. Indistinguishability Obfuscation from Compact FE.* Our main result is a reduction from  $iO$  to any public-key functional encryption scheme that satisfies a natural “compactness” requirement on the encryption algorithm. Specifically, we give a construction of  $iO$  for  $P/poly$  from any public-key FE scheme for  $NC^1$  that satisfies the following requirements:

- *Security:* It supports *one* key query and achieves selective, indistinguishability security against sub-exponential time adversaries.
- *Compactness:* For any input message  $x$ , the running time of the encryption algorithm is polynomial in the security parameter and the size of  $x$ . In particular, it does not depend on the circuit description size or the output length of any function  $f$  supported by the scheme.<sup>2</sup> We call such an FE scheme *compact*.

<sup>1</sup> See the full version for a comprehensive list of works.

<sup>2</sup> The compactness requirement can be further relaxed.

We stress that we do *not* require function hiding property [1, 11] from the underlying FE. Indeed, function-hiding public-key FE already implies  $iO$ .

*On the use of Sub-exponential Hardness.* Our reliance on sub-exponential hardness of the underlying FE scheme is similar in spirit to the use of sub-exponential hardness assumptions in the work of Gentry et al. [33]. Indeed, as discussed in their paper, the use of sub-exponential hardness assumptions “seems” inherent to realizing  $iO$ . We note, however, that to the best of our knowledge, no formal proof supporting this intuition is known.

*On the Existence of compact FE.* While public-key FE is an extremely well-studied notion, somewhat surprisingly, compact FE has remained largely unexplored. Previously, Goldwasser et al. [36] studied the notion of “succinct” FE which, informally speaking, requires that the size of any ciphertext must be independent of the function description size. We note, however, that this notion does not preclude dependence on the function output length. Indeed, [36] focuses on functions with single bit output, and their construction does not achieve our desired compactness property for the case of functions with long output.

Compact FE with simulation-based security is known to be impossible for general functions [2, 20]. Concretely, in the case of adaptive simulation security, the impossibility result holds for a single key and message query. In the selective security case, it holds for a single key and unbounded message queries.<sup>3</sup> However, we stress that for our main result, we only require the underlying compact FE scheme to satisfy *indistinguishability security* in the selective model.

Presently, the only known constructions of compact FE for general functions rely on  $iO$  [26, 46].<sup>4</sup> In contrast, *non-compact* FE can be based on LWE [36], or even semantically-secure public-key encryption [37, 44].

We hope that this work will bring attention to the natural goal of compactness in FE and that it will be realized from standard complexity assumptions in the future. With this view, we believe that the results in this work open new doors to the eventual goal of realizing  $iO$  from well-studied cryptographic assumptions, possibly avoiding multilinear maps altogether.

*II. A Technique for Arity Amplification.* At the heart of our results is a novel technique for *arity amplification* in secret-key multi-input functional encryption (MiFE), a notion introduced by Goldwasser et al. [35]. Specifically, we show how to transform a selectively-secure secret-key MiFE scheme for  $i$ -ary functions into another selectively-secure<sup>5</sup> secret-key MiFE scheme for  $(i+1)$ -ary functions. Interestingly, we achieve this by “knitting together” a secret-key FE scheme for

<sup>3</sup> A related notion of reusable garbled circuits with output-size independence was recently studied by Gentry et al. [31]. They proved an analogous impossibility result for this notion in the case of simulation security.

<sup>4</sup> The compact FE constructions of [26, 46], in fact, achieve stronger security than what we require. Specifically, they achieve security against unbounded key queries, while we only require security against a single key query.

<sup>5</sup> Unless stated otherwise, we only consider selectively-secure (Mi)FE schemes in the subsequent discussion.

$i$ -ary functions with a public-key FE scheme for 1-ary functions. In order to prove the security of our transformation, we build on program puncturing techniques that were first introduced by Sahai and Waters [45] in the context of  $iO$  and recently developed in the context of secret-key FE by Brakerski and Segev [16] and Komargodski et al. [40].

Starting from a secret key FE scheme for single-ary functions (aka single-input FE) and applying our transformation *iteratively*, we obtain a secret-key MiFE scheme for multi-ary functions. This iterated procedure is sensitive to the efficiency of the underlying single-input FE and yields different end results depending upon whether the underlying FE scheme is compact or not.

More concretely, given a compact *single-key* FE scheme for  $NC^1$ , we first convert it into a compact FE scheme for general functions that supports an a priori bounded polynomial number of key queries. This process involves multiple steps, including the key query amplification step of Gorbunov et al. [37] and the generic transformation from [4, 31] for boosting the function family from  $NC^1$  to general functions.

Then, instantiating our iterated approach with a sub-exponentially secure *compact* FE scheme that supports (say)  $q$  number of key queries, we obtain a secret-key MiFE scheme for polynomial-arity functions that supports  $q$  key queries and  $q$  message queries. Instantiating this result for the case of  $q = 2$  and combining it with the MiFE to  $iO$  transformation of Goldwasser et al. [35], we obtain  $iO$  for general functions.

*III. MiFE for Functions with Small Arity from Standard Assumptions.* We also analyze our transformation for the case when the underlying FE scheme is *non-compact*. Recall that in such a scheme, the running time of the encryption algorithm may depend upon the function description size [37, 44] or its output length [36].

*Bounded-Message Security from Standard Assumptions.* Starting with a non-compact FE scheme that supports an a priori bounded polynomial (say)  $q$  number of key queries, we obtain a secret-key MiFE scheme for *constant-ary* functions that supports  $q$  message and  $q$  key queries. Instantiating the underlying FE scheme with [37, 44], we obtain the above result based on *semantically secure public-key encryption*.<sup>6</sup> This significantly improves over the state of the art in this area in terms of security assumptions. In particular, prior constructions of such an MiFE scheme either rely upon  $iO$  [35] or lack a security proof in the standard model [10].

*Unbounded-Message Security from  $iO$ .* Starting with a non-compact FE scheme that achieves security against unbounded key queries, we obtain a secret-key

<sup>6</sup> At the cost of further decreasing the efficiency of encryption and restricting our attention to a single key query, we can, in fact, obtain this result based on only *one-way functions*. This requires a slight modification in our construction and proof. In particular, to obtain this result, we must replace the underlying public-key FE with a secret-key FE and then leverage the “one-shot” proof technique discussed in Sect. 1.2. We defer the details to the full version of the paper.

MiFE scheme for constant-ary functions that supports unbounded message and key queries.

Presently, known constructions of public-key FE with security against unbounded collusions rely upon  $iO$  and one-way functions [26, 46] or specific assumptions on composite-order multilinear maps [27]. Then, instantiating the underlying FE scheme in our construction with [26], we obtain a secret-key MiFE scheme for functions with constant arity that supports *unbounded* number of message and key queries based on  $iO$  and one-way functions. Previously, such a MiFE scheme [35] was only known based on differing-inputs obfuscation [3, 8, 14].

*On the Optimality of Our Results.* It is easy to see that a secret-key MiFE scheme for 2-ary functions that supports a single key query and *unbounded* message queries already implies a secret-key single-input FE scheme that supports *unbounded* key and message queries. This observation is already implicit in [35].

In light of the above, we note that our results on secret-key MiFE with bounded message queries are essentially *optimal*.

*IV. Compact FE from Randomized Encodings for Turing Machines.* Our final contribution is a construction of a single-key, compact FE scheme from the learning with errors (LWE) assumption and randomized encodings (RE) [6, 38] for Turing machines where the size of the encoding only depends on the size of the Turing machine (TM) and not on its running time or the output length. Combining this with our reduction from  $iO$  to compact FE, we get a construction of  $iO$  for general circuits from sub-exponentially secure RE for Turing machines and LWE.

Randomized encodings for circuits are known to exist from only one-way functions [47]. In contrast, the problem of RE for TMs has received far less attention. Recently, a few works [28, 29, 42] construct RE for RAM programs from only one-way functions; however, the size of the garbled RAM program in these schemes is proportional to the (worst-case) running time of the underlying RAM program. Even more recently, [9, 18, 41] give constructions of RE for TMs where the encoding size is independent of the running time of TM. However, all of these results are based on  $iO$ .

We hope that our work will bring more attention to this natural goal, and that it can be realized from standard cryptographic assumptions in the future. This result is presented in the full version [5].

## 1.2 Our Techniques

**Main Goal: Arity Amplification.** The starting point of our  $iO$  construction is the recent work of Goldwasser et al. [35] who showed a transformation from secret-key MiFE to  $iO$ . Concretely, [35] proved that secret-key MiFE for  $(n + 1)$ -ary functions that supports a *single* key query and 2 message queries implies  $iO$  for all functions with input length  $n$ . Very roughly, in order to obfuscate a function  $f$  with input length  $n$ , their idea is to use the first MiFE ciphertext to

hide the function and use the remaining  $n$  positions to encode  $f$ 's input domain à la Yao's garbled circuits [47]. This, coupled with a secret key for the universal circuit yields an indistinguishability obfuscation of  $f$ .

Given their result, our goal of constructing general-purpose  $iO$  from public-key single-input FE reduces to the task of constructing secret-key MiFE scheme for *polynomial-ary* functions from a public-key FE for *single-ary* functions. To help the presentation, we ignore the succinctness requirements on the underlying FE for now, and revisit it later.

At a first glance, it is not clear at all how to proceed towards realizing the above goal.

*Knitting together Two FE Instances.* Towards that end, let us first consider a weaker goal of constructing secret-key MiFE for 2-ary functions. Roughly speaking, our main idea is to “knit” together an instance of a *secret-key* single-input FE scheme with an instance of *public-key* single-input FE to obtain a secret-key MiFE for 2-ary functions. Here, the importance of using *both* a secret-key FE and a public-key FE will become clear once we explain our approach.

More concretely, the 2-ary MiFE scheme is constructed as follows:

- The master secret key of the 2-ary scheme consists of a key pair  $(pk, msk)$  of the underlying public-key FE scheme as well as a master secret key  $msk'$  of the underlying secret-key FE scheme. Further, a key for a function  $f$  is computed as a key  $K_f$  of the underlying public-key FE scheme for  $f$ .
- In order to encrypt a message  $m_1$  corresponding to the *first* position, we generate (using  $msk'$ ) a function key of the underlying secret-key FE scheme for the following function  $\mathcal{G}_{[m_1, K, pk]}^{enc}$ : it contains the message  $m_1$ , a key for a pseudorandom function (PRF)  $K$ , and the public key  $pk$  hardwired in its description. On input a message  $(m_2, tag)$ ,  $\mathcal{G}_{[m_1, K, pk]}^{enc}$  outputs an encryption (using  $pk$ ) of the combined message  $m_1 || m_2$  w.r.t. the underlying public-key FE. Here, the randomness  $r$  for encryption is derived as  $r \leftarrow PRF_K(tag)$ .

A message  $m_2$  corresponding to the *second* position is encrypted (along with a random tag) using the encryption algorithm of the underlying secret-key FE scheme.

- In order to decrypt a pair of ciphertexts  $(c_1, c_2)$  using a function key  $K_f$ , we first decrypt  $c_2$  using  $c_1$  (recall that  $c_1$  corresponds to a function key of the secret key FE scheme) to produce a new ciphertext  $\tilde{c}$  corresponding to the underlying public-key FE scheme. Finally, we decrypt  $\tilde{c}$  using  $K_f$  to get the desired output.

The correctness of the above construction is easy to verify. A careful reader, however, may immediately notice a security problem. Note that in order to prove security, we must ensure that the first ciphertext hides the message  $m_1$  and the PRF key  $K$ . However, this is not necessarily guaranteed by the above construction.

We solve this problem by building upon the recent elegant result of Brakerski and Segev [16] who give a generic transformation from any single-input secret-key FE scheme into another secret-key FE scheme that satisfies *function hiding*.

Specifically, instead of using a standard secret-key FE, we will use a function-hiding secret-key FE in the above construction. We then rely upon the function-hiding property of the function key to argue that  $m_1$  and  $K$  remain hidden. As we will see later, this technique, when generalized to the MiFE setting, is vital to our overall approach.

We highlight another subtle point in the above construction: suppose that we want the 2-ary MiFE scheme to support  $q \geq 2$  message queries. Then, since the function keys of the underlying secret-key FE scheme act as ciphertxts in the 2-ary MiFE scheme, we need the underlying secret-key FE scheme to, in fact, support  $q$  key queries. To obtain such an FE scheme, we leverage [37] to transform a single-key FE scheme into a  $q$ -key FE scheme. We refer the reader to the full version for more details.

*Overview of Proof Strategy.* Proving the security of the above construction turns out to be quite non-trivial. Suppose that we wish to prove security for  $q$  message queries (for each position), say  $\{x_i^0, y_i^0\}_{i=1}^q, \{x_i^1, y_i^1\}_{i=1}^q$ . Further, for simplicity, let us restrict our attention to a single function key query  $f$ . One plausible proof strategy would be to construct a sequence of roughly  $q$  hybrids where at any step  $i \in [q]$ , we switch from  $(x_i^0, y_i^0)$  to  $(x_i^1, y_i^1)$ . However, note that in the case of MiFE, an adversary can compute “cross-terms” from the challenge message pairs. That is, the adversary is allowed to compute  $(x_i^b, y_i^b)$  for any  $i, j \in [q]$ . Indeed, this is why the security definition of MiFE requires that  $f(x_i^0, y_j^0) = f(x_i^1, y_j^1)$  for all  $i, j \in [q]$ . However, note that in the above proof strategy, the adversary might end up computing  $f(x_i^1, y_j^0)$  which will allow him to distinguish between two successive hybrids.

A plausible solution to overcome the above problem is to argue indistinguishability in *one shot*. That is, instead of arguing indistinguishability one message-pair at a time, we instead switch all the challenge message pairs corresponding to challenge bit 0 with the ones corresponding to challenge bit 1. Implementing this strategy successfully, however, will require “hardwiring” and “unhardwiring” of the (public-key) encryption of *all* the  $q^2$  message pairs  $(x_i^b, y_j^b)$  (each of which corresponds to a different output) in the challenge ciphertxts for the first position that correspond to function keys of the underlying secret-key FE scheme. While this is tolerable for the case of arity 2 (and more generally for constant arity), it quickly becomes prohibitive for large arity. Indeed, for arity  $n = \text{poly}(\lambda)$ , the number of possible outputs (and therefore the message pair combinations) is exponential.

We solve the above problems by carefully employing a “one-input-at-a-time” strategy where we consider roughly  $q^2$  intermediate hybrids (and  $q^n$  in the case of arity  $n$ ; see below). Very briefly, our proof involves careful hardwiring and unhardwiring of the (public-key) encryption of each of the  $q^2$  message pairs  $(x_i^b, y_j^b)$ , *one at a time*, in the challenge ciphertxts for the first position that correspond to function keys of the underlying secret-key FE scheme. Furthermore, we crucially ensure that the adversary cannot learn an output of the form  $f(x_i^0, y_j^1)$  at any point in the hybrids. In order to implement these ideas, we rely upon *program puncturing techniques* that were originally introduced in the context of *iO* [45]

and recently developed in the secret-key FE setting by [16, 40]. In particular, as in the work of [40], we rely on function hiding property of the underlying secret-key FE scheme to argue indistinguishability of these core hybrids. We finally note that our proof strategy bears resemblance to the proof methodology in several recent works [9, 18, 19, 32, 33, 41].

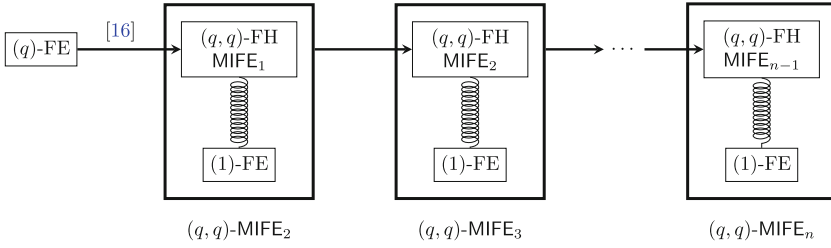
Note that in the above proof strategy, it was crucial that we use a *public-key* FE in our construction. To see this, suppose we were to replace the public-key FE with an instance of a secret-key FE, referred to as  $\mathcal{FE}$  (while the other secret-key FE instance used in the construction is referred to as  $\mathcal{FE}'$ ). Note that now, the challenge ciphertexts corresponding to the first position would contain the master secret key (say)  $\text{msk}$  of  $\mathcal{FE}$ . Then, in order to execute the aforementioned proof strategy, it would seem that we need to somehow “puncture”  $\text{msk}$  such that it allows encryption all messages except a select message (say)  $x_i^b \| y_j^b$ . Furthermore, the punctured  $\text{msk}$  *should not allow generation of any function keys*, except  $K_f$ . However, it is not clear how to realize such a notion of secret-key FE. By using public-key FE, we are able to bypass the above difficulties since by definition, the public key does not need to be hidden.

*Climbing the Arity Ladder.* The above approach can be generalized to transform a secret-key MiFE scheme for  $i$ -ary functions into a secret-key MiFE scheme for  $(i + 1)$ -ary functions. Concretely, this transformation consists of two steps: first, by using ideas from [16], we add function privacy property to the  $i$ -ary MiFE scheme. Next, we combine the resultant scheme with a “fresh” instance of a public-key single-input FE scheme to obtain an  $(i + 1)$ -ary MiFE scheme.

In more detail, as in the 2-ary case, the ciphertext corresponding to the first position will consist of a function key of the underlying (function private)  $c$ -ary MiFE scheme for the function  $\mathcal{G}_{[m_1, K, pk]}^{\text{enc}}$  which is defined similarly to the 2-ary case, except that here it takes as input messages  $m_2, \dots, m_{i+1}$  (along with random tags) and outputs an encryption (using  $\text{pk}$ ) of the combined message  $m_1 \| \dots \| m_{i+1}$  w.r.t. the underlying public-key FE. The ciphertexts corresponding to remaining  $i$  positions will correspond to ciphertexts of the underlying  $c$ -ary MiFE scheme. The function key for a function  $f$  in the  $c + 1$ -ary scheme will correspond to a key  $K_f$  for the same function  $f$  of the underlying public-key single-input FE scheme.

By applying the above ideas iteratively, we can transform a secret-key single-input FE into a secret-key multi-input FE. Our iterated construction is depicted in Fig. 1. The security of the construction follows along the same lines as discussed above.

*The Role of Compactness.* Upon “unrolling” our construction of  $n$ -ary MiFE scheme, one can observe that it involves  $n$  instances of a single-input FE scheme. Specifically, in the  $n$ -ary MiFE scheme, each of the ciphertexts corresponding to the first  $n - 1$  positions corresponds to a function key of (a different instance of) a single-input FE, while the ciphertext corresponding to the  $n$ th position corresponds to a ciphertext of a single-input FE scheme. The function key at position  $n - 1$  computes an encryption corresponding to the function key at



**Fig. 1.** The Iterated Construction.  $(q)$ -FE denotes a single-input public-key FE scheme that supports  $q$  key queries.  $(q_1, q_2)$ -MIFE <sub>$i$</sub>  denotes a secret-key MiFE scheme for  $i$ -ary functions that supports  $q_1$  key and  $q_2$  message queries. Finally,  $FH$  refers to function hiding.

position  $n - 2$  which in turn computes an encryption corresponding to the function key at position  $n - 3$ , and so on.

With the above view, it is easy to see that the complexity of the above construction becomes prohibitive for  $n = \omega(1)$  when it is instantiated with a non-succinct FE scheme. On the other hand, instantiating the construction with a succinct FE scheme allows us to go all the way to  $n = \text{poly}(\lambda)$ .

We remark that the above discussion is oversimplified. We refer the reader to the technical parts of the paper (and the full version [5]) for more details.

## 2 Preliminaries

Throughout the paper, we denote the security parameter by  $\lambda$ . We assume that the reader is familiar with basic cryptographic concepts [34].

Given a PPT sampling algorithm  $A$ , we use  $x \stackrel{\$}{\leftarrow} A$  to denote that  $x$  is the output of  $A$  when the randomness is sampled from the uniform distribution.

*Punctured Pseudorandom Function Families.* The works of [12, 15, 39] constructed a strengthening of PRF families that is commonly known as *punctured pseudorandom function families*. Unlike the standard notion of PRFs, this primitive is accompanied by a puncturing algorithm that takes as input  $x$ , a PRF key  $K$  and outputs a punctured key  $K_x$  that allows one to evaluate the output of PRF on any input other than  $x$ . The security guarantee states that the output of PRF on  $x$  is indistinguishable from random even if the adversary gets a key punctured on  $x$ .

### 2.1 Indistinguishability Obfuscation

Here we recall the notion of indistinguishability obfuscation ( $iO$ ) that was defined by Barak et al. [8].

**Definition 2.1 (Indistinguishability Obfuscator ( $iO$ )).** A uniform PPT algorithm  $iO$  is called an *indistinguishability obfuscator* for a circuit class  $\{\mathcal{C}_\lambda\}$ ,



where  $\mathcal{C}_\lambda$  consists of circuits  $C$  of the form  $C : \{0, 1\}^\lambda \rightarrow \{0, 1\}$ , if the following holds:

- **Completeness:** For every  $\lambda \in \mathbb{N}$ , every  $C \in \mathcal{C}_\lambda$ , every input  $x \in \{0, 1\}^\lambda$  (i.e., it belongs to the input space of  $C$ ), we have that

$$\Pr[C'(x) = C(x) : C' \leftarrow iO(\lambda, C)] = 1.$$

- **Indistinguishability:** For any PPT distinguisher  $D$ , there exists a negligible function  $\text{negl}(\cdot)$  such that the following holds: for all sufficiently large  $\lambda \in \mathbb{N}$ , for all pairs of circuits  $C_0, C_1 \in \mathcal{C}_\lambda$  such that  $C_0(x) = C_1(x)$  for all inputs  $x$ , we have:

$$\left| \Pr[D(iO(\lambda, C_0)) = 1] - \Pr[D(iO(\lambda, C_1)) = 1] \right| \leq \text{negl}(\lambda)$$

## 2.2 Public-Key Functional Encryption

*Syntax.* Let  $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda \in \mathbb{N}}$  and  $\mathcal{Y} = \{\mathcal{Y}_\lambda\}_{\lambda \in \mathbb{N}}$  be ensembles where each  $\mathcal{X}_\lambda, \mathcal{Y}_\lambda$  are sets of size, functions in  $\lambda$ . Let  $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda \in \mathbb{N}}$  be an ensemble where each  $\mathcal{F}_\lambda$  is a finite collection of functions. Each function  $f \in \mathcal{F}_\lambda$  takes as input a string  $x \in \mathcal{X}_\lambda$  and outputs  $f(x) \in \mathcal{Y}_\lambda$ .

A public-key functional encryption (FE) scheme FE for  $\mathcal{F}$  consists of four algorithms (FE.Setup, FE.KeyGen, FE.Enc, FE.Dec):

- **Setup.** FE.Setup( $1^\lambda$ ) is a PPT algorithm that takes as input a security parameter  $\lambda$  and outputs a public key, (master) secret key pair (FE.pk, FE.msk).
- **Key Generation.** FE.KeyGen(FE.msk,  $f$ ) is a PPT algorithm that takes as input a master secret key FE.msk, a function  $f \in \mathcal{F}_\lambda$  and outputs a functional key FE.sk $_f$ .
- **Encryption.** FE.Enc(FE.pk,  $x$ ) is a PPT algorithm that takes as input a public key FE.pk, a message  $x \in \mathcal{X}_\lambda$  and outputs a ciphertext ct.
- **Decryption.** FE.Dec(FE.sk $_f$ , ct) is a deterministic algorithm that takes as input a functional key FE.sk $_f$ , a ciphertext ct and outputs a string  $y$ .

The correctness property guarantees that the output of the decryption on input a functional key of  $f \in \mathcal{F}_\lambda$  and a ciphertext of  $x \in \mathcal{X}_\lambda$  yields  $f(x) \in \mathcal{Y}_\lambda$ .

*Selective Security.* We recall indistinguishability-based selective security for FE. This security notion is modeled as a game between the challenger and the adversary where the adversary can request functional keys and ciphertexts from the challenger. Specifically, the adversary can submit function queries  $f$  to the challenger and receive corresponding functional keys. It can also submit a message query of the form  $(x_0, x_1)$  and in response, the challenger encrypts message  $x_b$  and sends the ciphertext back to the adversary. The adversary wins the game if she can guess  $b$  with probability significantly greater than  $1/2$  and if  $f(x_0) = f(x_1)$  for all function queries  $f$ . The only constraint here is that the adversary has to declare the challenge messages at the beginning of the game itself.

The term  $(q_{\text{key}}, \mu)$ -secure FE scheme refers to the setting where the adversary can request up to  $q_{\text{key}}$  queries and he can succeed in the game with probability at most  $\mu$ .

**Compactness.** We now define the notion of *compact* FE that will play a central role in our main result on *iO*. In a compact FE scheme, the running time of the encryption algorithm only depends on the security parameter and the input message length. In particular, it is independent of the complexity of the function family supported by the FE scheme. Note that a direct consequence of this is that the size of the public key must also be independent of the complexity of the function family.

**Definition 2.2 (Compact FE).** Let  $p(\cdot)$  be a polynomial. A  $(q_{\text{key}}, \mu)$ -selectively secure public-key FE scheme  $\text{FE} = (\text{FE.Setup}, \text{FE.KeyGen}, \text{FE.Enc}, \text{FE.Dec})$ , defined for an input space  $\mathcal{X} = \{\mathcal{X}_\lambda\}$  and function space  $\mathcal{F} = \{\mathcal{F}_\lambda\}$  is said to be **compact** if for all  $\lambda \in \mathbb{N}$ , the size of any public key  $\text{FE.pk}$  is  $p(\lambda)$ , where  $(\text{FE.msk}, \text{FE.pk}) \leftarrow \text{FE.Setup}(1^\lambda)$ , and the running time of the encryption algorithm  $\text{FE.Enc}$ , on input  $1^\lambda$ ,  $\text{FE.pk}$  and a message  $x \in \mathcal{X}_\lambda$ , is  $p(\lambda, q_{\text{key}}, |x|)$ .

*Remark 2.3.* We can define the notion of unbounded compact FE in the same manner as above except that we now allow the number of key queries made by the adversary in the security game to be an arbitrary polynomial.

### 3 Function Private Multi-input Functional Encryption (MiFE)

The concept of multi-input functional encryption was proposed by Goldwasser et al. [35]. Standard FE only allows for computing on a single ciphertext, i.e., it only supports *single-ary* functions. In contrast, multi-input functional encryption (MiFE) allows for (joint) computation over multiple ciphertexts. In other words, it supports *multi-ary* functions.

Analogous to standard FE, one can consider MiFE in two settings, namely, public-key and secret-key setting.<sup>7</sup> In this work, we will restrict our attention to the *secret-key* setting.

The security notion we are interested in is stronger than the one considered in Goldwasser et al. [35]. We expect the functional keys to hide the function it is associated with. This concept, termed as *function privacy* was previously considered in the single ary private key FE setting by Brakerski-Segev [16]. We extend their notion to the multi-input functional encryption setting as well.

We first present the syntax of a MiFE scheme and later we formalize the function privacy property.

<sup>7</sup> Goldwasser et al. [35] also define a more general notion of MiFE where there is different encryption key for each input position. When the adversary knows all (resp., none of) the encryption keys, then this notion captures the public-key (resp., secret-key) setting.

*Syntax.* Let  $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda \in \mathbb{N}}$  and  $\mathcal{Y} = \{\mathcal{Y}_\lambda\}_{\lambda \in \mathbb{N}}$  be ensembles where each  $\mathcal{X}_\lambda, \mathcal{Y}_\lambda$  are sets of size, functions in  $\lambda$ . Let  $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda \in \mathbb{N}}$  be an ensemble where each  $\mathcal{F}_\lambda$  is a finite collection of  $n$ -ary functions. Each function  $f \in \mathcal{F}_\lambda$  takes as input strings  $x_1, \dots, x_n$ , where each  $x_i \in \mathcal{X}_\lambda$ , and outputs  $f(x_1, \dots, x_n) \in \mathcal{Y}_\lambda$ .

An MiFE scheme  $\text{MIFE}_n$  for  $n$ -ary functions  $\mathcal{F}$  consists of four algorithms ( $\text{MIFE}_n.\text{Setup}, \text{MIFE}_n.\text{KeyGen}, \text{MIFE}_n.\text{Enc}, \text{MIFE}_n.\text{Dec}$ ) described below:

- **Setup.**  $\text{MIFE}_n.\text{Setup}(1^\lambda)$  is a PPT algorithm that takes as input a security parameter  $\lambda$  and outputs the master secret key  $\text{MIFE}_n.\text{msk}$ .
- **Key Generation.**  $\text{MIFE}_n.\text{KeyGen}(\text{MIFE}_n.\text{msk}, f)$  is a PPT algorithm that takes as input the master secret key  $\text{MIFE}_n.\text{msk}$  and a function  $f \in \mathcal{F}_\lambda$ . It outputs a functional key  $\text{MIFE}_n.\text{sk}_f$ .
- **Encryption.**  $\text{MIFE}_n.\text{Enc}(\text{MIFE}_n.\text{msk}, m, i)$  is a PPT algorithm that takes as input the master secret key  $\text{MIFE}_n.\text{msk}$ , a message  $x \in \mathcal{X}_\lambda$  and an index  $i \in [n]$ . It outputs a ciphertext  $\text{MIFE}_n.\text{ct}$ .  
Here index  $i$  signals to the encryption algorithm that message  $x$  corresponds to the  $i^{\text{th}}$  input of functions  $f \in \mathcal{F}_\lambda$ .
- **Decryption.**  $\text{MIFE}_n.\text{Dec}(\text{MIFE}_n.\text{sk}_f, \text{MIFE}_n.\text{ct})$  is a deterministic algorithm that takes as input a functional key  $\text{MIFE}_n.\text{sk}_f$  and a ciphertext  $\text{MIFE}_n.\text{ct}$ . It outputs a value  $y \in \mathcal{Y}_\lambda$ .

*Remark 3.1.* From now on, we use the phrase “encryption of  $m$  in the  $i^{\text{th}}$  position” to refer to the process of executing  $\text{MIFE}_n.\text{Enc}$  on the input  $(\text{MIFE}_n.\text{msk}, m, i)$ .

*Correctness.* There exists a negligible function  $\text{negl}(\cdot)$  such that for all sufficiently large  $\lambda \in \mathbb{N}$ , every  $n$ -ary function  $f \in \mathcal{F}_\lambda$  and input tuple  $(x_1, \dots, x_n) \in \mathcal{X}_\lambda^n$

$$\Pr \left[ \begin{array}{l} \text{MIFE}_n.\text{msk} \leftarrow \text{MIFE}_n.\text{Setup}(1^\lambda) ; \text{MIFE}_n.\text{sk}_f \leftarrow \text{MIFE}_n.\text{KeyGen}(\text{MIFE}_n.\text{msk}, f) ; \\ \text{MIFE}_n.\text{Dec}(\text{MIFE}_n.\text{sk}_f, \{\text{MIFE}_n.\text{Enc}(\text{MIFE}_n.\text{msk}, x_i, i)\}_{i=1}^n) \neq f(x_1, \dots, x_n) \end{array} \right]$$

is at most  $\text{negl}(\lambda)$ . In the above expression, the probability is taken over the random coins of all the algorithms.

We present the function privacy definition below. Similar to the single ary setting, we can consider two security notions – selective and adaptive. We first give the selective security definition since this is the definition we are going to consider throughout this paper.

**Definition 3.2 (Selective Function Private MiFE).** *A secret-key MiFE scheme  $\text{MIFE}_n$  for  $n$ -ary functions  $\mathcal{F}$  is  $(q_{\text{key}}, q_{\text{msg}}, \mu)$ -selective function private if for any PPT adversary  $\mathcal{A}$ , there exists a function  $\mu(\lambda)$  such that for all sufficiently large  $\lambda \in \mathbb{N}$ , the advantage of  $\mathcal{A}$  is*

$$\text{Adv}_{\mathcal{A}}^{\text{MIFE}_n} = \left| \Pr[\text{Expt}_{\mathcal{A}}^{\text{MIFE}_n}(1^\lambda, 0) = 1] - \Pr[\text{Expt}_{\mathcal{A}}^{\text{MIFE}_n}(1^\lambda, 1) = 1] \right| \leq \mu(\lambda),$$

where for each  $b \in \{0, 1\}$  and  $\lambda \in \mathbb{N}$ , the experiment  $\text{Expt}_{\mathcal{A}}^{\text{MIFE}_n}(1^\lambda, b)$  is defined below:

1. **Message Queries:**  $\mathcal{A}$  submits  $q_{\text{msg}}$  number of queries,  $\left\{ \left( (x_{1,0}^j, x_{1,1}^j), \dots, (x_{n,0}^j, x_{n,1}^j) \right) \right\}_{j \in [q_{\text{msg}}]}$ , with  $x_{i,0}^j \in \mathcal{X}_\lambda$ , to the challenger  $C$ .
2.  $C$  computes  $\text{MIFE}_n.\text{msk} \leftarrow \text{MIFE}_n.\text{Setup}(1^\lambda)$ . It then computes  $\text{MIFE}_n.\text{ct}_i^j \leftarrow \text{MIFE}_n.\text{Enc}(\text{MIFE}_n.\text{msk}, x_{i,b}^j)$  for all  $i \in [n]$  for all  $j \in [q_{\text{msg}}]$ . The challenger  $C$  then sends  $\left\{ (\text{MIFE}_n.\text{ct}_1^j, \dots, \text{MIFE}_n.\text{ct}_n^j) \right\}_{j \in [q_{\text{msg}}]}$  to the adversary  $\mathcal{A}$ .
3. **Function Queries:** The following is repeated up to  $q_{\text{key}}$  times:  $\mathcal{A}$  submits a function query  $(f_0, f_1) \in \mathcal{F}_\lambda^2$  to  $C$ . The challenger  $C$  computes  $\text{MIFE}_n.\text{sk}_f \leftarrow \text{MIFE}_n.\text{KeyGen}(\text{MIFE}_n.\text{msk}, f_b)$  and sends it to  $\mathcal{A}$ .
4. If there exists a function query  $(f_0, f_1)$  and a challenge message query  $((x_{1,0}, \dots, x_{n,0}), (x_{1,1}, \dots, x_{n,1}))$  such that  $f_0(x_{1,0}, \dots, x_{n,0}) \neq f_1(x_{1,1}, \dots, x_{n,1})$ , then the output of the experiment is set to  $\perp$ . Otherwise, the output of the experiment is set to  $b'$ , where  $b'$  is the output of  $\mathcal{A}$ .

*Remark 3.3.* When  $\mu$  is a negligible function in the security parameter, then we omit it from the notation and simply refer to  $(q_{\text{key}}, q_{\text{msg}})$ -function privacy of MiFE.

*Constructing Function Private MiFE.* In the single ary setting, Brakerski-Segev [16] gave a generic transformation that converts any secret key single ary FE into a function private secret key single ary FE. We observe that techniques, similar to those used in Brakerski-Segev, can be adapted to obtain a transformation from any  $i$ -ary MiFE into a function private  $i$ -ary MiFE in the secret key setting. We defer the technical details to the full version.

## 4 Our Transformation: From $c$ -ary to $(c + 1)$ -ary MiFE

In this section, we show how to transform a secret-key MiFE scheme for  $c$ -ary functions into a MiFE scheme for  $(c + 1)$ -ary functions, for  $c \geq 1$ .

Our transformation proceeds in two steps:

1. Starting with an MiFE scheme for  $c$ -ary functions, we first apply the function privacy transformation (mentioned towards the end of Sect. 3) to obtain a function private MiFE scheme  $\text{MIFE}_c$  for  $c$ -ary functions.
2. Next, we convert  $\text{MIFE}_c$  into an MiFE scheme  $\text{MIFE}_{c+1}$  for  $c+1$ -ary functions. We refer to this step as the *arity amplification* step.

We now describe the arity amplification step. We construct an MiFE scheme for  $c + 1$ -ary functions  $\text{MIFE}_{c+1}$  with function space  $\mathcal{F}^{c+1}$  and message space  $\mathcal{X}^{c+1}$ .

*Notation.* We use the following tools in our transformation: (a) A function private MiFE scheme for  $c$ -ary functions, denoted as  $\text{MIFE}_c = (\text{MIFE}_c.\text{Setup}, \text{MIFE}_c.\text{KeyGen}, \text{MIFE}_c.\text{Enc}, \text{MIFE}_c.\text{Dec})$ . Let  $\mathcal{F}^{\text{fp},c}$  and  $\mathcal{X}^{\text{fp},c}$  be the associated function space and message space, respectively. (b) A public-key FE scheme for single-ary functions, denoted as  $\text{FE} = (\text{FE}.\text{Setup}, \text{FE}.\text{KeyGen}, \text{FE}.\text{Enc}, \text{FE}.\text{Dec})$ .

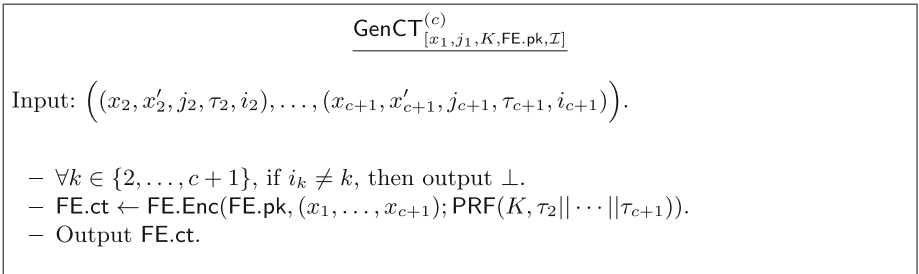
Let  $\mathcal{F}^{\text{fe}}$  and  $\mathcal{X}^{\text{fe}}$  be the associated function space and message space, respectively.  
 (c) A puncturable pseudorandom function family, denoted as  $F = \text{PRF}_K(\cdot)$ .

*Setup*  $\text{MIFE}_{c+1}.\text{Setup}(1^\lambda)$ : On input a security parameter  $\lambda$ , sample a master secret key  $\text{MIFE}_c.\text{msk} \leftarrow \text{MIFE}_c.\text{Setup}(1^\lambda)$  of  $\text{MIFE}_c$  and a key pair  $(\text{FE.pk}, \text{FE.msk}) \leftarrow \text{FE.Setup}(1^\lambda)$  of FE. Output  $\text{MIFE}_{c+1}.\text{msk} = (\text{MIFE}_c.\text{msk}, \text{FE.pk}, \text{FE.msk})$ .

*Key Generation*  $\text{MIFE}_{c+1}.\text{KeyGen}(\text{MIFE}_{c+1}.\text{msk}, f)$ : On input master secret key  $\text{MIFE}_{c+1}.\text{msk}$  and a function  $f \in \mathcal{F}^{c+1}$ , parse  $\text{MIFE}_{c+1}.\text{msk} = (\text{MIFE}_c.\text{msk}, \text{FE.pk}, \text{FE.msk})$ . Sample a functional key  $\text{FE.sk}_f \leftarrow \text{FE.KeyGen}(\text{FE.msk}, f)$  for function  $f$ . Output  $\text{MIFE}_{c+1}.\text{sk}_f = \text{FE.sk}_f$ .

*Encryption*  $\text{MIFE}_{c+1}.\text{Enc}(\text{MIFE}_{c+1}.\text{msk}, x, i)$ : On input master secret key  $\text{MIFE}_{c+1}.\text{msk}$ , message  $x \in \mathcal{X}^{c+1}$  and index  $i$ , parse  $\text{MIFE}_{c+1}.\text{msk} = (\text{MIFE}_c.\text{msk}, \text{FE.pk}, \text{FE.msk})$ .

1. If  $i = 1$ , then draw a PRF key  $K \in \{0, 1\}^\lambda$  at random. Initialize the index vector  $\mathcal{I} = (0, \dots, 0)$ . Compute  $\text{MIFE}_c.\text{sk}_G \leftarrow \text{MIFE}_c.\text{KeyGen}(\text{MIFE}_c.\text{msk}, G)$  where the circuit  $G = \text{GenCT}_{[x, 1, K, \text{FE.pk}, \mathcal{I}]}^{(c)} \in \mathcal{F}^{\text{fp}, c}$  is described in Fig. 2. Output the ciphertext  $\text{MIFE}_{c+1}.\text{ct} = \text{MIFE}_c.\text{sk}_G$ .
2. Else, if  $2 \leq i \leq c + 1$ , then perform the following steps:
  - If the input message  $x$  is of the form  $(x_1, x_2, 1, \tau, i - 1)$  then compute  $\text{MIFE}_{c+1}.\text{ct} \leftarrow \text{MIFE}_c.\text{Enc}(\text{MIFE}_c.\text{msk}, (x_1, x_2, 1, \tau, i), i)$
  - Else, choose a tag  $\tau \in \{0, 1\}^\lambda$  at random. Compute  $\text{MIFE}_{c+1}.\text{ct} \leftarrow \text{MIFE}_c.\text{Enc}(\text{MIFE}_c.\text{msk}, (x, x, 1, \tau, i), i)$ .
 Output the ciphertext  $\text{MIFE}_{c+1}.\text{ct}$ .



**Fig. 2.** Description of  $\text{GenCT}^c$ .

*Decryption*  $\text{MIFE}_{c+1}.\text{Dec}(\text{MIFE}_{c+1}.\text{sk}_f, \text{MIFE}_{c+1}.\text{ct}_1, \dots, \text{MIFE}_{c+1}.\text{ct}_{c+1})$ : On input  $(\text{MIFE}_{c+1}.\text{sk}_f, \text{MIFE}_{c+1}.\text{ct}_1, \dots, \text{MIFE}_{c+1}.\text{ct}_{c+1})$ , perform the following steps:

1. Parse: (a)  $\text{MIFE}_{c+1}.sk_f = \text{FE}.sk_f$ , (b)  $\text{MIFE}_{c+1}.ct_1 = \text{MIFE}_c.sk_G$ , and (c)  $\text{MIFE}_{c+1}.ct_i = \text{MIFE}_c.ct_{i-1}$  for all  $i \neq 1$ , where  $\text{MIFE}_c.ct_{i-1}$  denotes the ciphertext corresponding to  $(i-1)^{th}$  position in  $\text{MIFE}_c$ .
2. Next, compute  $\text{FE}.ct^* \leftarrow \text{MIFE}_c.\text{Dec}(\text{MIFE}_c.sk_G, \text{MIFE}_c.ct_1, \dots, \text{MIFE}_c.ct_c)$ .
3. Finally, compute  $y \leftarrow \text{FE}.\text{Dec}(\text{FE}.sk_f, \text{FE}.ct^*)$ . Output  $y$ .

This completes the description of the scheme.

*Correctness.* We now argue the correctness of  $\text{MIFE}_{c+1}$ . Let  $\text{MIFE}_c.sk$  be a valid functional key for the function  $\text{GenCT}[x_1, j_1, K, \text{FE}.pk, \mathcal{I}]$  w.r.t.  $\text{MIFE}_c$ . For  $i \in [c]$ , let  $\text{MIFE}_c.ct_i$  be a valid encryption of  $x_{i+1}$  w.r.t.  $\text{MIFE}_c$ . By the correctness of  $\text{MIFE}_c.\text{Enc}$ , we have that the output of  $\text{MIFE}_c.\text{Dec}(\text{MIFE}_c.sk_{\text{GenCT}}, \text{MIFE}_c.ct_1, \dots, \text{MIFE}_c.ct_c)$  is  $\text{FE}.ct^*$ , where  $\text{FE}.ct^*$  is a valid encryption of  $(x_1, \dots, x_{c+1})$  w.r.t.  $\text{FE}$ . Further, from the correctness of  $\text{FE}$ , it follows that the output of  $\text{FE}.\text{Dec}(\text{FE}.sk_f, \text{FE}.ct^*)$  is  $f(x_1, \dots, x_{c+1})$ , where  $\text{FE}.sk_f$  is a valid functional key of  $f$  w.r.t.  $\text{FE}$ . The proof of security can be found in the full version [5].

## 5 Multi-input FE from Single-input FE

In Sect. 4, we gave a general transformation from a secret-key MiFE scheme for  $c$ -ary functions to another secret-key MiFE scheme for  $c+1$ -ary functions. Using this transformation, we now give a construction of a secret-key MiFE scheme for functions with  $n = \text{poly}(\lambda)$  arity. Later we will use this construction to obtain our main result on  $iO$ . We will also consider different instantiations of this construction which yield new results on constant-ary MiFE from standard assumptions.

We construct a  $n$ -ary  $(q, q)$ -secure MiFE scheme  $\text{MIFE}_n$ . To obtain this construction we start with a  $q$ -secure<sup>8</sup> public-key FE scheme. This implies a single-ary  $(q, q)$ -secure secret-key MiFE scheme,  $\text{MIFE}_1$ .

**(Iterated)** Construction of  $\text{MIFE}_n$  (Informal description):

Repeat the following two steps for  $c = 1, \dots, n$ :

1. **Function Privacy Transformation:** Using the MiFE function-privacy transformation (mentioned towards the end of Sect. 3), convert the  $(q, q)$ -secure MiFE scheme  $\text{MIFE}_c$ , obtained in the previous iteration, into a function-private  $(q, q)$ -secure MiFE scheme  $\text{MIFE}_c^{\text{fp}}$ , also supporting  $c$ -arity functions.
2. **Arity Amplification:** The function-private  $c$ -ary  $(q, q)$ -secure MiFE scheme  $\text{MIFE}_c^{\text{fp}}$  obtained in the previous step is then transformed into a  $c+1$ -ary  $(q, q)$ -secure MiFE scheme, using the transformation presented in Sect. 4. In this step, we additionally use a  $q$ -secure public-key FE scheme  $\text{FE}$ .

<sup>8</sup> Throughout this section, we only consider selectively secure public-key FE and secret key MiFE schemes. For simplicity of notation, we omit the use of the word “selective” in the rest of this section and assume that it is implicit.

The efficiency properties of the underlying public-key FE scheme determines the value of  $n$  that we can achieve in the above construction. Consequently, we consider two different instantiations of the underlying public-key FE scheme that yield different results. We discuss these instantiations next.

*iO from Compact FE.* We start by stating our main result for secret-key MiFE for polynomial-arity functions.

**Theorem 5.1.** *For all  $n = \text{poly}(\lambda)$ , the above proposed scheme  $\text{MiFE}_n$  is  $(q, q)$ -secure for any polynomial  $q$ , assuming that FE is  $\left(1, \frac{1}{(64q)^{(n+1)^2} \cdot 2^\lambda}\right)$ -selectively secure compact public-key FE scheme.*

The core non triviality in the above theorem is to argue that the size of parameters does not grow exponentially with the number of iterations. At a high level, this is because the compactness of FE ensures that the growth of the parameters at the  $i^{\text{th}}$  level ( $i$ -ary MiFE) depends only on the security parameter, level  $i$  and the message length. The actual calculations can be found in the appropriate section in the full version.

We now invoke a theorem by [35] that shows how to obtain iO for functions of input length  $n$  from a  $n$ -ary MiFE for a specific function family. We thus have the following main theorem.

**Theorem 5.2.** *Assuming the  $\left(2, \frac{1}{(128)^{n^2} 2^\lambda}\right)$ -security of compact public-key selectively secure FE public key FE scheme for polynomial time computable functions, the scheme iO is an indistinguishability obfuscation scheme for  $P/\text{poly}$ .*

*Constant ary MiFE.* If we restrict our attention to just constant ary MiFE then we can obtain a construction based on public key encryption encryption schemes. We state the result formally below.

**Theorem 5.3.** *For any constant  $n$ , the above proposed scheme  $\text{MiFE}_n$  is  $(q, q)$ -selectively secure assuming that FE is a  $q$ -selectively secure (not necessarily compact) public-key FE scheme.*

Combining Theorem 5.3 with [37,44], we obtain the following result.

**Corollary 5.4.** *For any polynomial  $q = q(\lambda)$ , there exists a  $(q, q)$ -selectively secure secret-key MiFE scheme for constant-arity functions, assuming the existence of semantically-secure public-key encryption.*

The reason why we can only achieve constant arity is because the growth of parameters in this case could be exponential. If we start from any public key FE scheme, the size of the parameters at each level grows proportional to the size of the parameters at the next level. This stems from the fact that the FE scheme that we start off with could be such that the encryption complexity might depend on the function complexity. And hence, the number of iterations that can be performed is just a constant. A detailed explanation is provided in the full version.

## References

1. Agrawal, S., Agrawal, S., Badrinarayanan, S., Kumarasubramanian, A., Prabhakaran, M., Sahai, A.: Function private functional encryption and property preserving encryption: new definitions and positive results. IACR Cryptology ePrint Archive 2013/744 (2013)
2. Agrawal, S., Gorbunov, S., Vaikuntanathan, V., Wee, H.: Functional encryption: new perspectives and lower bounds. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II [17]. LNCS, vol. 8043, pp. 500–518. Springer, Heidelberg (2013). [http://dx.doi.org/10.1007/978-3-642-40084-1\\_28](http://dx.doi.org/10.1007/978-3-642-40084-1_28)
3. Ananth, P., Boneh, D., Garg, S., Sahai, A., Zhandry, M.: Differing-inputs obfuscation and applications. IACR Cryptology ePrint Archive 2013/689 (2013). <http://eprint.iacr.org/2013/689>
4. Ananth, P., Brakerski, Z., Segev, G., Vaikuntanathan, V.: The trojan method in functional encryption: from selective to adaptive security, generically. IACR Cryptology ePrint Archive 2014/917 (2014). <http://eprint.iacr.org/2014/917>
5. Ananth, P., Jain, A.: Indistinguishability obfuscation from compact functional encryption. Technical report, Cryptology ePrint Archive, report 2015/173 (2015)
6. Applebaum, B., Ishai, Y., Kushilevitz, E.: Computationally private randomizing polynomials and their applications. *Comput. Complexity* **15**(2), 115–162 (2006)
7. Barak, B., Garg, S., Kalai, Y.T., Paneth, O., Sahai, A.: Protecting obfuscation against algebraic attacks. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 221–238. Springer, Heidelberg (2014)
8. 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). [http://dx.doi.org/10.1007/3-540-44647-8\\_1](http://dx.doi.org/10.1007/3-540-44647-8_1)
9. Bitansky, N., Garg, S., Lin, H., Pass, R., Telang, S.: Succinct randomized encodings and their applications. In: STOC (2015)
10. Boneh, D., Lewi, K., Raykova, M., Sahai, A., Zhandry, M., Zimmerman, J.: Semantically secure order-revealing encryption: multi-input functional encryption without obfuscation. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 563–594. Springer, Heidelberg (2015)
11. Boneh, D., Raghunathan, A., Segev, G.: Function-private identity-based encryption: hiding the function in functional encryption. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 461–478. Springer, Heidelberg (2013)
12. Boneh, D., Waters, B.: Constrained pseudorandom functions and their applications. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part II. LNCS, vol. 8270, pp. 280–300. Springer, Heidelberg (2013)
13. Boneh, D., Wu, D.J., Zimmerman, J.: Immunizing multilinear maps against zeroizing attacks. IACR Cryptology ePrint Archive 2014/930 (2014). <http://eprint.iacr.org/2014/930>
14. Boyle, E., Chung, K.-M., Pass, R.: On extractability obfuscation. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 52–73. Springer, Heidelberg (2014). [http://dx.doi.org/10.1007/978-3-642-54242-8\\_3](http://dx.doi.org/10.1007/978-3-642-54242-8_3)
15. Boyle, E., Goldwasser, S., Ivan, I.: Functional signatures and pseudorandom functions. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 501–519. Springer, Heidelberg (2014)



16. Brakerski, Z., Segev, G.: Function-private functional encryption in the private-key setting. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015, Part II. LNCS, vol. 9015, pp. 306–324. Springer, Heidelberg (2015)
17. Canetti, R., Garay, J.A. (eds.): CRYPTO 2013, Part II. LNCS, vol. 8043. Springer, Heidelberg (2013). <http://dx.doi.org/10.1007/978-3-642-40084-1>
18. Canetti, R., Holmgren, J., Jain, A., Vaikuntanathan, V.: Indistinguishability obfuscation of iterated circuits and RAM programs. In: STOC (2015)
19. Canetti, R., Lin, H., Tessaro, S., Vaikuntanathan, V.: Obfuscation of probabilistic circuits and applications. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015, Part II. LNCS, vol. 9015, pp. 468–497. Springer, Heidelberg (2015)
20. De Caro, A., Iovino, V., Jain, A., O’Neill, A., Paneth, O., Persiano, G.: On the achievability of simulation-based security for functional encryption. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II [17]. LNCS, vol. 8043, pp. 519–535. Springer, Heidelberg (2013). [http://dx.doi.org/10.1007/978-3-642-40084-1\\_29](http://dx.doi.org/10.1007/978-3-642-40084-1_29)
21. Cheon, J.H., Han, K., Lee, C., Ryu, H., Stehlé, D.: Cryptanalysis of the multilinear map over the integers. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 3–12. Springer, Heidelberg (2015)
22. Coron, J.-S., Lepoint, T., Tibouchi, M.: Practical multilinear maps over the integers. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 476–493. Springer, Heidelberg (2013)
23. Coron, J., Lepoint, T., Tibouchi, M.: Cryptanalysis of two candidate fixes of multilinear maps over the integers. IACR Cryptology ePrint Archive 2014/975 (2014). <http://eprint.iacr.org/2014/975>
24. Garg, S., Gentry, C., Halevi, S.: Candidate multilinear maps from ideal lattices. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 1–17. Springer, Heidelberg (2013). [http://dx.doi.org/10.1007/978-3-642-38348-9\\_1](http://dx.doi.org/10.1007/978-3-642-38348-9_1)
25. Garg, S., Gentry, C., Halevi, S., Raykova, M.: Two-round secure MPC from indistinguishability obfuscation. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 74–94. Springer, Heidelberg (2014)
26. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: 54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26–29 October 2013, Berkeley, CA, USA, pp. 40–49. IEEE Computer Society (2013). <http://doi.ieeecomputersociety.org/10.1109/FOCS.2013.13>
27. Garg, S., Gentry, C., Halevi, S., Zhandry, M.: Fully secure functional encryption without obfuscation. IACR Cryptology ePrint Archive 2014/666 (2014). <http://eprint.iacr.org/2014/666>
28. Garg, S., Lu, S., Ostrovsky, R., Scafuro, A.: Garbled RAM from one-way functions. In: STOC (2015)
29. Gentry, C., Halevi, S., Lu, S., Ostrovsky, R., Raykova, M., Wichs, D.: Garbled RAM revisited. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 405–422. Springer, Heidelberg (2014)
30. Gentry, C., Halevi, S., Maji, H.K., Sahai, A.: Zeroizing without zeroes: cryptanalyzing multilinear maps without encodings of zero. IACR Cryptology ePrint Archive 2014/929 (2014). <http://eprint.iacr.org/2014/929>
31. Gentry, C., Halevi, S., Raykova, M., Wichs, D.: Outsourcing private RAM computation. IACR Cryptology ePrint Archive 2014/148 (2014). <http://eprint.iacr.org/2014/148>
32. Gentry, C., Lewko, A., Waters, B.: Witness encryption from instance independent assumptions. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 426–443. Springer, Heidelberg (2014)

33. Gentry, C., Lewko, A.B., Sahai, A., Waters, B.: Indistinguishability obfuscation from the multilinear subgroup elimination assumption. IACR Cryptology ePrint Archive 2014/309 (2014). <http://eprint.iacr.org/2014/309>
34. Goldreich, O.: Foundations of Cryptography: Volume 2, Basic Applications, vol. 2. Cambridge University Press, Cambridge (2009)
35. Goldwasser, S., Gordon, S.D., Goyal, V., Jain, A., Katz, J., Liu, F.-H., Sahai, A., Shi, E., Zhou, H.-S.: Multi-input functional encryption. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 578–602. Springer, Heidelberg (2014). [http://dx.doi.org/10.1007/978-3-642-55220-5\\_32](http://dx.doi.org/10.1007/978-3-642-55220-5_32)
36. Goldwasser, S., Kalai, Y.T., Popa, R.A., Vaikuntanathan, V., Zeldovich, N.: Reusable garbled circuits and succinct functional encryption. In: Boneh, D., Roughgarden, T., Feigenbaum, J. (eds.) Symposium on Theory of Computing Conference, STOC 2013, 1–4 June 2013, Palo Alto, CA, USA, pp. 555–564. ACM (2013). <http://doi.acm.org/10.1145/2488608.2488678>
37. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Functional encryption with bounded collusions via multi-party computation. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 162–179. Springer, Heidelberg (2012)
38. Ishai, Y., Kushilevitz, E.: Randomizing polynomials: a new representation with applications to round-efficient secure computation. In: 41st Annual Symposium on Foundations of Computer Science, FOCS 2000, 12–14 November 2000, Redondo Beach, California, USA, pp. 294–304 (2000)
39. Kiayias, A., Papadopoulos, S., Triandopoulos, N., Zacharias, T.: Delegatable pseudorandom functions and applications. In: Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security, pp. 669–684. ACM (2013)
40. Komargodski, I., Segev, G., Yogev, E.: Functional encryption for randomized functionalities in the private-key setting from minimal assumptions. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015, Part II. LNCS, vol. 9015, pp. 352–377. Springer, Heidelberg (2015)
41. Koppula, V., Lewko, A.B., Waters, B.: Indistinguishability obfuscation for turing machines with unbounded memory. In: STOC (2015)
42. Lu, S., Ostrovsky, R.: How to garble RAM programs? In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 719–734. Springer, Heidelberg (2013)
43. Pass, R., Seth, K., Telang, S.: Indistinguishability obfuscation from semantically-secure multilinear encodings. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 500–517. Springer, Heidelberg (2014)
44. Sahai, A., Seyalioglu, H.: Worry-free encryption: functional encryption with public keys. In: Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010, 4–8 October 2010, Chicago, Illinois, USA, pp. 463–472 (2010)
45. Sahai, A., Waters, B.: How to use indistinguishability obfuscation: deniable encryption, and more. In: Shmoys, D.B. (ed.) Symposium on Theory of Computing, STOC 2014, 31 May–03 June 2014, New York, NY, USA, pp. 475–484. ACM (2014). <http://doi.acm.org/10.1145/2591796.2591825>
46. Waters, B.: A punctured programming approach to adaptively secure functional encryption. Cryptology ePrint Archive, report 2014/588 (2014)
47. Yao, A.C.C.: How to generate and exchange secrets (extended abstract). In: FOCS, pp. 162–167 (1986)

# **Pseudorandomness**

# Efficient Pseudorandom Functions via On-the-Fly Adaptation

Nico Döttling<sup>1</sup> and Dominique Schröder<sup>2</sup>(✉)

<sup>1</sup> Department of Computer Science, Aarhus University, Aarhus, Denmark  
nico.doettling@cs.au.dk

<sup>2</sup> Department of Computer Science, Saarland University, CISPA,  
Saarbrücken, Germany  
ds@ca.cs.uni-saarland.de

**Abstract.** Pseudorandom functions (PRFs) are one of the most fundamental building blocks in cryptography with numerous applications such as message authentication codes and private key encryption. In this work, we propose a new framework to construct PRFs with the overall goal to build efficient PRFs from standard assumptions with an almost tight proof of security. The main idea of our framework is to start from a PRF for any small domain (i.e. poly-sized domain) and turn it into an  $\ell$ -bounded pseudorandom function, i.e., into a PRF whose outputs are pseudorandom for the first  $\ell$  distinct queries to  $F$ . In the second step, we apply a novel technique which we call *on-the-fly adaptation* that turns any bounded PRF into a fully-fledged (large domain) PRF. Both steps of our framework have a tight security reduction, meaning that any successful attacker can be turned into an efficient algorithm for the underlying hard computational problem without any significant increase in the running time or loss of success probability.

Instantiating our framework with specific number theoretic assumptions, we construct a PRF based on  $k$ -LIN (and thus DDH) that is faster than all known constructions, which reduces almost tightly to the underlying problem, and which has shorter keys. Instantiating our framework with general assumptions, we construct a PRF with very flat circuits whose security tightly reduces to the security of some small domain PRF.

---

N. Döttling—The authors acknowledge support from the Danish National Research Foundation and The National Science Foundation of China (under the grant 61061130540) for the Sino-Danish Center for the Theory of Interactive Computation, within which part of this work was performed; and also from the CFEM research center (supported by the Danish Strategic Research Council) within which part of this work was performed.

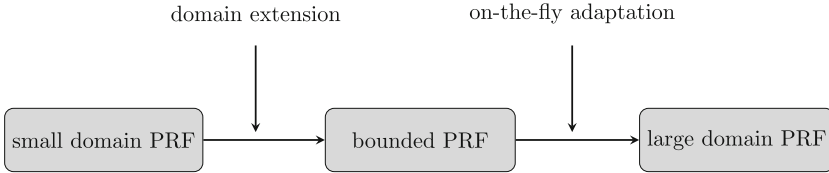
N. Döttling—Supported by European Research Commission Starting Grant no. 279447.

D. Schröder—Supported by the German Federal Ministry of Education and Research (BMBF) through funding for the Center for IT-Security, Privacy and Accountability (CISPA, [www.cispa-security.org](http://www.cispa-security.org)) and also by an Intel Early Career Faculty Honor Program Award.

**Keywords:** Pseudorandom functions · Efficient reductions · DDH · K-LIN · LWE

## 1 Introduction

Goldreich, Goldwasser, and Micali (GGM) introduced pseudorandom functions (PRFs) in 1984 [13]. Roughly, a PRF is a keyed deterministic function, whose output is indistinguishable from a random function. PRFs are one of the most fundamental building blocks in cryptography, with numerous applications such as private-key encryption, message authentication codes, key derivation, and many more, e.g., [2, 14, 22, 29, 32]. In this work, we propose a novel framework to construct PRFs with the overall goal of constructing *efficient* PRFs based on *standard assumptions* with an almost *tight* proof of security. The basic idea of this framework is to transform a PRF for a small domain (i.e., poly-size) into a fully fledged domain that handles large input spaces. This transformation tightly reduces to the underlying small domain PRF. The main steps of our framework and the novel techniques are shown in Fig. 1. We begin with a PRF that works over a small domain, say  $\{0, 1\}^{\log \ell}$ , and which can be evaluated very efficiently in time  $\text{poly}(\lambda, \log \ell)$ , for some parameter  $\ell$ .



**Fig. 1.** Overview of the main steps and the techniques.

The first step in our framework is to extend the domain of a small-domain PRF into a *bounded pseudorandom function* (bPRF). A function  $F$  is an  $\ell$ -bounded pseudorandom function (for an  $\ell \leq \text{poly}(\lambda)$ ), if the outputs of  $F$  are pseudorandom for the first  $\ell$  distinct queries to  $F$  and if  $F$  can be computed “super efficiently” (i.e., in time  $\text{poly}(\lambda, \log(\ell))$ ). In some sense, this primitive can be seen as the computational analogue to  $\ell$ -wise independent functions.

The second step in our framework is a reduction technique we call *on-the-fly adaptation*. The goal of this technique is to construct a PRF  $F$  in which we can dynamically embed an  $\ell$ -bounded PRF  $F_\ell$  for every  $\ell$  that grows at most polynomially. Now assume we have a PPT distinguisher  $\mathcal{D}$  that distinguishes  $F$  from a truly random function. Since  $\mathcal{D}$  is efficient, it sends at most  $q = \text{poly}(\lambda)$  queries to its oracle (for an a-priori unknown  $q$ ). On-the-fly adaptation allows us to turn this distinguisher against  $F$  into a distinguisher  $\mathcal{D}'$  against a bounded PRF  $F_q$  that has the same advantage.

We will demonstrate this idea with a simple on-the-fly adaptation technique that works for any bounded PRF. The basic idea of this technique is to compute  $F$  as a sum of functions  $F_\ell$ , for an exponentially increasing  $\ell$ . An important point is that all  $F_\ell$  have the *same domain*. The function  $F$  is computed by

$$F(K, x) = \bigoplus_{i=0}^t F_{2^i}(K_{2^i}, x),$$

where  $K = (K_{2^i})_{i=1, \dots, t}$ . If we choose the parameter  $t = \omega(\log(\lambda))$  slightly super-logarithmic, we will be able to embed any  $F_\ell$  into  $F$ . Notice that  $F$  can be computed efficiently, as we required that bounded PRFs can be computed in time  $\text{poly}(\lambda, \log(\ell))$ . To illustrate the main idea, assume that there exists a distinguisher  $\mathcal{D}$  that makes at most  $q = \text{poly}(\lambda)$  queries distinguishes  $F$  from a truly random function. We will provide a reduction that turns this distinguisher into a distinguisher against the small domain PRF  $F_{2^{\lceil \log q \rceil}}$ . Observe that we can express  $F$  by

$$F(K, x) = \bigoplus_{i=0}^{\log q - 1} F_{2^i}(K_{2^i}, x) \oplus F_{2^{\lceil \log q \rceil}}(K_{2^{\lceil \log q \rceil}}, x) \oplus \bigoplus_{i=\log q + 1}^t F_{2^i}(K_{2^i}, x).$$

The reduction can now replace the middle term  $F_{2^{\lceil \log q \rceil}}(K_{2^{\lceil \log q \rceil}}, x)$  by its own oracle and provide to  $\mathcal{D}$  an oracle  $\mathcal{O}'$  that computes the function

$$\mathcal{O}'(x) = \bigoplus_{i=0}^{\log q - 1} F_{2^i}(K_{2^i}, x) \oplus \mathcal{O}(x) \oplus \bigoplus_{i=\log q + 1}^t F_{2^i}(K_{2^i}, x).$$

Clearly, if  $\mathcal{O}$  computes the function  $F_{2^{\lceil \log q \rceil}}$ , then  $\mathcal{O}'$  computes the function  $F$ . On the other hand, if  $\mathcal{O}$  is a random function, then  $\mathcal{O}'$  also is a random function. This reduction is tight. Notice that it is crucial for this technique to work that all  $F_\ell$  have the same input domain. Basically the domain extension step in our framework is geared towards *equalizing* the domains of small domain PRFs. This generic technique is similar to a transformation from non-adaptive to adaptive pseudorandom functions by Berman and Haitner [4]. The construction of [4] however yields no tight security proof (which was not the purpose of that work), as their construction does not start from bounded PRFs.

We will now discuss domain extension for arbitrary PRFs and provide a simple domain extension technique that uses only linear functions to pre- and post-process a small domain PRF. This, together with the generic on-the-fly adaptation technique described above yields a PRF construction from any small domain PRF. We will discuss an instantiation of this general construction based on LWE.

**Domain Extension for Arbitrary PRFs.** The problem of domain extension for pseudorandom functions was first considered by Levin [20]. Levin showed that if the domain of a certain PRF is already sufficiently large, then it can be

extended by using a universal hash function to hash larger inputs into the domain of this PRF. However, this technique is vulnerable to a “birthday attack”, which means that after a certain number of queries there is a high probability of finding a collision in the hash function. Levin’s technique also fails for small domain PRFs, i.e., PRFs with domains of polynomial size. Jain, Pietrzak, and Tentes [19] provided a domain extension technique which also works for small domains, but has an unfavorable security loss in this case. Moreover, as mentioned by the authors, their technique does not seem to be directly applicable to efficient PRF such as the one’s based on DDH [19]. The work of Jain et al. [19] was refined by Chandran and Garg [8]. Berman et al. [5] also showed how to bypass the birthday barrier via Cuckoo hashing.

We provide a simple general domain extension technique that preserves the parallel complexity of an underlying small domain PRF. This domain extension technique is inspired by the construction of universal hash functions by Ishai et al. [18] and can be seen as an amplified version of Levin’s trick. For a small domain pseudorandom function  $\text{PRF}_\ell : \{0, 1\}^{\log(2\lambda\ell)} \rightarrow \mathcal{Y}$ , we construct a large domain bounded PRF  $F_\ell : \mathcal{X} \rightarrow \mathcal{Y}$  by

$$F_\ell(K', x) = \bigoplus_{j=1}^{\lambda} \text{PRF}_\ell(K, \text{BIN}(j) \| H_j(x)),$$

where  $K' = (K, H_1, \dots, H_\lambda)$ ,  $H_1, \dots, H_\lambda \leftarrow_{\S} \mathcal{H}$  are randomly chosen universal hash functions from a family  $\mathcal{H}$  that maps  $\mathcal{X}$  to  $\{0, 1\}^{\log(2\ell)}$  and  $\text{BIN}(j)$  is the  $\log(\lambda)$  bit binary representation of an integer  $j \in \{1, \dots, \lambda\}$ .

## 1.1 A General Transformation

Above we described our on-the-fly adaptation technique that works for any bounded PRF. Combining this technique with a general domain extension technique, we obtain large domain pseudorandom function with almost tight security (i.e., only a logarithmic loss) from any suitable small domain PRF. In a nutshell, a small domain PRF is suitable for this technique if its security loss only depends on the size of its input domain, but not (polynomially) on the number queries a distinguisher sends<sup>1</sup>. The computational problems from which PRFs with such a small security loss can be constructed usually have one feature in common: they support a statistical random self-reduction. Candidate PRFs with this property are PRFs based on the LWE [28, 30] problem, such as the PRF of Banerjee, Peikert, and Rosen [1]. Using the BPR PRF as small domain PRF in our general construction, we obtain a large domain PRF which is secure under a weaker assumption, which has a tighter proof of security, and a shallower evaluation circuit than instantiating the BPR scheme with a large domain directly.

<sup>1</sup> The Naor Reingold PRF would be such a suitable PRF as its security reduction only loses a factor of  $n$ . However, as discussed above we provide a much more efficient direct construction based on the NR PRF.

In the remaining part of this section, we discuss more efficient instantiations based on DDH and  $k$ -LIN. Here, we exploit specific number theoretic properties in order to improve the efficiency and security of the resulting PRF.

### 1.2 Efficient PRFs Based on DDH and $k$ -LIN

One appealing property of our framework is that it yields several new constructions of PRFs based on weak standard assumptions, such as  $k$ -LIN (and thus DDH) with an almost tight proof of security. A tight reduction means that a successful attacker can be turned into an efficient algorithm for the hard computational problem without any significant increase in the running time or significant loss of success probability<sup>2</sup>. We will provide a specific on-the-fly adaptation technique that exploits algebraic properties of the underlying number theoretic assumptions. We can thus avoid the blow up of the general on-the-fly adaptation technique described in the last paragraph and obtain PRFs that improve upon known constructions in terms of efficiency, security, and key-size.

**Instantiation Based on DDH.** In the following we discuss our construction based on the DDH assumption. Our underlying small domain PRF is the Naor-Reingold PRF based on DDH [26]. For an input domain  $\{0, 1\}^n$ , the Naor Reingold PRF  $\text{NR} : \mathcal{K}_n \times \{0, 1\}^n \rightarrow \mathbb{G}$  is defined by

$$\text{NR}_n(K, x) = g^a \prod_{j=0}^{n-1} s_j^{x_j},$$

where  $K = (a, s_0, \dots, s_{n-1})$  and  $a, s_0, \dots, s_{n-1} \leftarrow_{\mathcal{S}} \mathbb{Z}_p$ . In the first step, we turn the small domain PRF  $\text{NR}_{\log(\ell)}$ , which has a domain of size  $\ell$  into an  $\ell$ -bounded PRF that has input domain  $\mathbb{Z}_p$ , i.e., a large input domain. In contrast to our generic construction (that we will discuss later), we can exploit specific number theoretic properties in order to improve the efficiency, the tightness of the security reduction, and the key-size. The bounded PRF  $F_\ell : \mathcal{K} \times \mathbb{Z}_p \rightarrow \mathbb{G}$  is defined as follows:

$$F_\ell(K, x) = g^a \prod_{j=0}^{\log(\ell)-1} (s_j + x^{2^j}). \tag{1}$$

We will briefly discuss why security of this PRF tightly reduces to the security of  $\text{NR}_{\log(\ell)}$ . Expanding the exponent of  $F_\ell(K, x)$  yields

$$a \prod_{j=0}^{\log(\ell)-1} (s_j + x^{2^j}) = \sum_{\mathbf{c} \in \{0,1\}^{\log(\ell)}} \underbrace{\left( a \prod_{j=0}^{\log(\ell)-1} s_j^{1-c_j} \right)}_{E(\mathbf{c})} x^{\sum_{j=1}^{\log(\ell)} c_j 2^j} \tag{2}$$

Now, observe that the term  $E(\mathbf{c})$  on the right side is an exponent of the Naor Reingold PRF  $\text{NR}_{\log(\ell)}$ . Specifically, it holds that  $g^{E(\mathbf{c})} = \text{NR}_{\log(\ell)}(-\mathbf{c})$

<sup>2</sup> Usually even a polynomially-bounded increase/loss is considered as significant, if the polynomial may be large. An increase/loss by a small constant factor is not considered as significant.



(where  $\neg \mathbf{c}$  is the bitwise negation of  $\mathbf{c}$ ). Changing the sum on the right side of (2) to run over all  $j = 0, \dots, 2^{\lceil \ell \rceil} - 1$  and setting  $\mathbf{c} = \text{BIN}(j)$  (where  $\text{BIN}(j)$  is the  $\log(\ell)$ -bit binary representation of  $j$ ), we get that  $F_\ell$  can be equivalently computed as

$$F_\ell(K, x) = \prod_{j=0}^{2^{\lceil \log(\ell) \rceil} - 1} (\text{NR}_{\log(\ell)}(K, \neg \text{BIN}(j)))^{x^j}. \tag{3}$$

Notice that this expression can still be efficiently computed as long as  $\ell \leq \text{poly}(\lambda)$ . Now, observe that if we replace  $\text{NR}_{\log(\ell)}$  by a random function in this expression, then  $F_\ell$  becomes an (information theoretic)  $\ell$ -wise independent function. We can therefore use this alternative description of  $F_\ell$  to show that it is an  $\ell$ -bounded PRF.

The observation that functions of the form as  $F_\ell$  in (3) can be computed in time  $\log \ell$  via a closed form as (1) was previously made by Benadbbas, Gennaro, and Vahlis for the Naor-Reingold PRF [3] and Fiore and Gennaro for the Lewko-Waters PRF [12]. The fact that  $F_\ell$  is a bounded PRF was independently observed by Hazay [15].

In the last step, we apply an “in-place” on-the-fly adaptation technique to this function. We will not use the generic technique described above, but one that exploits the specific algebraic properties of  $F_\ell$ . We define the full fledged PRF  $F$  by

$$F(K, x) = g^a \prod_{j=0}^{t-1} (s_j + x^{2^j}),$$

where the parameter  $t = \omega(\log(\lambda))$  is chosen slightly super-logarithmic. Now, notice that we can embed the bounded PRF  $F_\ell$  (for any  $\ell \leq \text{poly}(\lambda)$ ) into  $F$  by

$$F(K, x) = \left( g^a \prod_{j=0}^{t_{\log(\ell)} - 1} (s_j + x^{2^j}) \right)^{\prod_{j=\log(\ell)}^{t-1} (s_j + x^{2^j})} = (F_\ell(K_\ell, x))^{\prod_{j=\log(\ell)}^{t-1} (s_j + x^{2^j})}.$$

In the security proof, we replace  $F_\ell$  by a truly random function. The main part of the proof consists in showing that the exponent  $\prod_{j=\log(\ell)}^{t-1} (s_j + x^{2^j})$  only accounts for a negligible error.

*Comparison to Naor-Reingold* [25]. Our full fledged PRF with input domain  $\mathbb{Z}_p$  improves upon the Naor-Reingold PRF (NR-PRF) in terms of tightness of the security reduction and compactness. In contrast to the NR-PRF, the loss of our security reduction is only a factor of  $\log(q)$  (where  $q = \text{poly}(\lambda)$ ) is the number of queries required by the distinguisher  $\mathcal{D}$ ), compared to a factor  $n$  for the NR-PRF. Our PRF is very compact as it only requires  $\omega(\log(\lambda))$   $\mathbb{Z}_p$  elements for its key, whereas the Naor-Reingold needs  $n$   $\mathbb{Z}_p$  elements. Since the exponentiation is the dominating factor in the computation of both PRFs, the costs to evaluate both functions is roughly the same.

**Instantiation Based on  $k$ -LIN.** In the main body, we directly provide a PRF construction based on a family of weaker computational problems known as  $k$ -LIN [17, 31]. The decisional  $k$ -linear assumption becomes (generically) weaker as the parameter  $k$  grows, where the instance  $k = 1$  corresponds to DDH and  $k = 2$  to the linear assumption [6]. The main motivation for these assumptions is that groups are known where the DDH assumption is easy, but the computational Diffie Hellman problem is supposedly hard [16]. It is thus desirable to have constructions of cryptographic primitives based on the decisional  $k$ -linear assumption instead of DDH. Our generalized PRF is defined as follows: Let  $k \geq 1$  be a positive integer,  $\mathbb{G} = \langle g \rangle$  be a cyclic group of prime order  $p$  and  $t = \omega(\log(\lambda))$ . The function  $F : \mathcal{K} \times \mathbb{Z}_p \rightarrow \mathbb{G}^k$  is defined by

$$F(K, x) = g^{\mathbf{a}^\top \cdot \prod_{j=0}^{t-1} (\mathbf{S}_j + x^{2^j} \cdot \mathbf{I})},$$

where  $K = (\mathbf{a}, \mathbf{S}_0, \dots, \mathbf{S}_{t-1})$  with  $\mathbf{a} \leftarrow_{\S} \mathbb{Z}_p^k$ ,  $\mathbf{S}_0, \dots, \mathbf{S}_{t-1} \leftarrow_{\S} \mathbb{Z}_p^{k \times k}$  and  $\mathbf{I}$  the identity matrix. Clearly, if we only need a single group element as output, we can truncate the exponent and perform only 1 exponentiation.

*Comparison to Lewko-Waters* [21]. Our PRF improves upon the Lewko-Waters PRF (LW-PRF) in terms of efficiency, tightness of the security reduction, and compactness. A single evaluation of the LW-PRF involves  $n$  matrix multiplications and a single exponentiation. In our case, the computation requires only  $t = \omega(\log(\lambda))$  matrix multiplications and a single exponentiation. For larger  $k$ , the cost of the matrix multiplication dominates the cost of the exponentiation, so in this case our construction is more efficient. The security reduction of Lewko and Waters loses a factor of  $k \cdot n$  while our reduction only loses a factor of  $k \log q$ . The keys of the LW-PRF consist of  $n$   $k \times k$  matrices over  $\mathbb{Z}_p$ , while ours consists merely of  $t = \omega(\log \lambda)$  such matrices.

### 1.3 Other Related Work

Many number-theoretic PRF constructions follow the GGM paradigm [13], such as [1, 21, 25]. Naor and Reingold introduced pseudorandom synthesizer (PRS) that can be used to construct parallel computable pseudorandom function [1, 24]. A PRF construction that is not based on either the GGM or synthesizers paradigm is the PRF of Dodis-Yampolskiy, which is in fact a direct construction, but whose security is closely related to its underlying bilinear  $q$ -type assumption [10]. Recently, Chase and Meiklejohn showed that this  $q$ -type assumption can be reduced to the subgroup hiding assumption in composite order groups [9]. The PRF of Naor, Reingold, and Rosen is a clever variant of the Naor-Reingold PRF that is secure under the factoring assumption [27]. The work of Boneh, Montgomery, and Raghunathan combines a generalization of the GGM tree with the Dodis-Yampolskiy PRF to get a large-domain (simulateable) verifiable random function [7].

## 2 Preliminaries

Throughout this paper, we will use  $\lambda$  to denote the security parameter. We will denote the concatenation of two bit strings  $x$  and  $y$  by  $x\|y$ . We will generally assume that logarithms are rounded to the next biggest integer, i.e., when we write  $\log(\ell)$  we actually mean  $\lceil \log(\ell) \rceil$ . To avoid confusion, we will sometimes still write  $\lceil \log(\ell) \rceil$ , e.g. when we write  $2^{\lceil \log(\ell) \rceil}$  to indicate that this can be different from  $\ell$ .

**Definition 1 (Pseudorandom Functions).** *Let  $\mathcal{X}_\lambda$  and  $\mathcal{Y}_\lambda$  be two finite sets depending on  $\lambda$ . We say that an efficiently computable keyed function  $\text{PRF} : \mathcal{K}_\lambda \times \mathcal{X}_\lambda \rightarrow \mathcal{Y}_\lambda$  with key-space  $\mathcal{K}_\lambda$  is a pseudorandom function (PRF), if it holds for every PPT oracle distinguisher  $\mathcal{D}$  that*

$$|\Pr[\mathcal{D}^{\text{PRF}(K,\cdot)}(1^\lambda) = 1] - \Pr[\mathcal{D}^R(1^\lambda) = 1]| \leq \text{negl}(\lambda),$$

where  $K \leftarrow_{\S} \mathcal{K}_\lambda$  and  $R : \mathcal{X}_\lambda \rightarrow \mathcal{Y}_\lambda$  is a randomly chosen function. If  $|\mathcal{X}| \leq \text{poly}(\lambda)$ , then we say that PRF is a small-domain PRF, otherwise we call PRF a large-domain PRF.

We will usually omit the  $\lambda$  subscript in the definition of  $\mathcal{K}$ ,  $\mathcal{X}$  and  $\mathcal{Y}$ . Moreover, we will henceforth implicitly assume that distinguisher gets  $1^\lambda$  as an additional input without explicitly stating this.

As mentioned in the outline, bounded pseudorandom functions can be seen as a computational analogue of limited-wise independent functions. Basically, the difference between true PRFs and bounded PRFs manifests itself in their security guarantee. While a distinguisher against a true PRF can query the PRF an a-priori unbounded number of times, a distinguisher against an  $\ell$ -bounded PRF can query the PRF with at most  $\ell$  distinct queries.

**Definition 2 (Bounded Pseudorandom Functions).** *Let  $\mathcal{X}$  and  $\mathcal{Y}$  be finite sets (depending on  $\lambda$ ). We say that a keyed function  $F_\ell : \mathcal{K}_\ell \times \mathcal{X} \rightarrow \mathcal{Y}$  parametrized by a parameter  $\ell$  is a bounded pseudorandom function (bPRF), if  $F_\ell$  is computable in time  $\text{poly}(\lambda, \log(\ell))$  and if it holds for all efficiently computable  $\ell^* = \ell(\lambda) \leq \text{poly}(\lambda)$  and all  $\ell^*$ -query distinguishers  $\mathcal{D}$  (i.e. distinguishers that send at most  $\ell^*$  distinct queries) that*

$$|\Pr[\mathcal{D}^{F_\ell(K,\cdot)} = 1] - \Pr[\mathcal{D}^R = 1]| \leq \text{negl}(\lambda),$$

where  $K \leftarrow_{\S} \mathcal{K}_\ell$  and  $R : \mathcal{X} \rightarrow \mathcal{Y}$  is a randomly chosen function.

Notice that in the definition of bounded PRFs we allow the key-space to depend on  $\ell$ , but  $\mathcal{X}$  and  $\mathcal{Y}$  are independent of  $\ell$ . Moreover, as we require that  $F_\ell$  is computable in time  $\text{poly}(\lambda, \log(\ell))$ , we implicitly also require that  $|\mathcal{K}_\ell| \leq \text{poly}(\lambda, \log(\ell))$ . Requiring that  $F_\ell$  can be computed in time  $\text{poly}(\lambda, \log(\ell))$  allows us to evaluate  $F_\ell$  for super-polynomial  $\ell$ , while we only require security for  $\ell^*$  which are at most polynomial.

The following lemma states that if a function  $F$  outputs uniformly random outputs under *benign* inputs, then the statistical distance from  $F$  to a uniformly random function  $F'$  can be bounded by the probability that a non-adaptive sequence of inputs is not benign. Intuitively, an adaptive distinguisher  $\mathcal{D}$  learns nothing about the set of bad inputs unless it finds such an input by chance, as otherwise the function  $F$  reveals no information about the set of bad inputs. This lemma is a simplified version of a more general statement due to Maurer [23].

**Lemma 1.** *Let  $\mathcal{X}$  and  $\mathcal{Z}$  be two finite sets. Let  $F_{K,\text{aux}} : \mathcal{X} \rightarrow \mathcal{Z}$  be a function that takes two additional parameters  $K \in \mathcal{K}$  and  $\text{aux} \in \text{AUX}$ . Let  $\text{good}(\cdot, \cdot)$  be a predicate with the following property: If  $\text{good}(\{x_1, \dots, x_i\}, \text{aux})$  holds, then  $F_{K,\text{aux}}(x_1), \dots, F_{K,\text{aux}}(x_i)$  are distributed uniformly at random over the choice of  $K \leftarrow_{\S} \mathcal{K}$ . Let  $\mathcal{D}$  be a (possibly unbounded) distinguisher that makes at most  $\ell$  distinct queries,  $K \leftarrow_{\S} \mathcal{K}$ ,  $\text{aux} \leftarrow_{\S} \text{AUX}$  and let  $F'$  be a uniformly chosen function from  $\mathcal{X}$  to  $\mathcal{Z}$ . Then it holds that*

$$|\Pr[\mathcal{D}^{F_{K,\text{aux}}} = 1] - \Pr[\mathcal{D}^{F'} = 1]| \leq \max_S \Pr[\neg \text{good}(S, \text{aux})],$$

where  $S$  runs over all subsets of  $\mathcal{X}$  of size at most  $\ell$ .

A proof of Lemma 1 can be found in the full version.

### 3 A Generic Construction

In this section, we will first provide an efficient construction of  $\ell$ -bounded pseudorandom function any small domain PRF with input space of (polynomial) size  $n \cdot \ell$ . Security of the  $\ell$ -bounded PRF follows tightly from the underlying small domain PRF. Second, we will provide a general construction of a PRF from  $\ell$ -bounded PRFs, where security also follows tightly.

#### 3.1 Bounded PRFs via Domain Extension of Small Domain PRFs

We will need universal hash functions for our domain extension technique.

**Definition 3 (Universal Hash Functions).** *Let  $\mathcal{X}$  and  $\mathcal{Y}$  be finite sets. We say that a family  $\mathcal{H}$  of functions from  $\mathcal{X}$  to  $\mathcal{Y}$  is a family of universal hash functions, if it holds for all  $x \neq x' \in \mathcal{X}$  that  $\Pr[H(x) = H(x')] \leq 1/|\mathcal{Y}|$ , where the probability is taken over the random choice of  $H \leftarrow_{\S} \mathcal{H}$ .*

Universal hash functions can be constructed very efficiently, see e.g., [18].

**Construction 1.** Let  $\text{PRF}_\ell : \mathcal{K}_\ell \times \{0, 1\}^{\log(2\lambda\ell)} \rightarrow \{0, 1\}^m$  be a keyed function with key space  $\mathcal{K}_\ell$ . Let  $\mathcal{H}_\ell$  be a family of universal hash functions that map  $\mathcal{X}$  to  $\{0, 1\}^{\log(2\ell)}$ . Let  $\text{BIN}(j)$  denote the  $\log(\lambda)$  bit binary representation of a number  $j \in \{1, \dots, \lambda\}$ . We define the keyed function  $F_\ell : \mathcal{K}' \times \mathcal{X} \rightarrow \{0, 1\}^m$  with key space  $\mathcal{K}'_\ell = \mathcal{H}^\lambda \times \mathcal{K}_\ell$  by

$$F_\ell(K', x) = \bigoplus_{j=1}^\lambda \text{PRF}_\ell(K, \text{BIN}(j) \| H_j(x)),$$

where  $H_j \leftarrow_{\S} \mathcal{H}_\ell$  for  $j = 1, \dots, \lambda$ ,  $K \leftarrow_{\S} \mathcal{K}_\ell$  and  $K' = (H_1, \dots, H_\lambda, K)$ .

The following theorem states that  $F_\ell$  is an  $\ell$ -bounded pseudorandom function if  $\text{PRF}_\ell$  is a pseudorandom function.

**Theorem 1.** Let  $\text{PRF}_\ell$  and  $F_\ell$  be as in Construction 1. If  $\text{PRF}_\ell$  is a pseudorandom function, then  $F_\ell$  is an  $\ell$ -bounded pseudorandom function. More specifically, assume there exists an  $\ell^* \leq \text{poly}(\lambda)$  and an  $\ell^*$ -query PPT distinguisher  $\mathcal{D}$  that distinguishes  $F_{\ell^*}$  from a truly random function with advantage  $\epsilon$ , then there exists a PPT distinguisher  $\mathcal{D}'$  with essentially the same runtime as  $\mathcal{D}$  that distinguishes  $\text{PRF}_{\ell^*}$  from a truly random function with advantage at least  $\epsilon - \ell^* \cdot 2^{-\lambda}$ .

The proof of Theorem 1 will be given in the full version.

### 3.2 PRFs via On-the-Fly Adaptation of Bounded PRFs

In this section we provide a generic on-the-fly adaptation technique which converts a bounded PRF into a standard PRF.

**Construction 2.** Let  $t = \omega(\log(\lambda))$  be slightly super-logarithmic. For a given parameter  $\ell$ , let  $F_\ell : \mathcal{K}_\ell \times \mathcal{X} \rightarrow \{0, 1\}^m$  be a keyed function with corresponding key space  $\mathcal{K}_\ell$ . Define the function  $F : \mathcal{K} \times \mathcal{X} \rightarrow \{0, 1\}^m$  with key-space  $\mathcal{K} = \prod_{i=0}^t \mathcal{K}_{2^i}$  by

$$F(K, x) = \bigoplus_{i=0}^t F_{2^i}(K_{2^i}, x),$$

where  $K_{2^i} \leftarrow_{\S} \mathcal{K}_{2^i}$  for  $i = 1, \dots, t$  and  $K = (K_{2^i})_{i=1, \dots, t}$ .

We will now show that  $F$  is in fact a pseudorandom function.

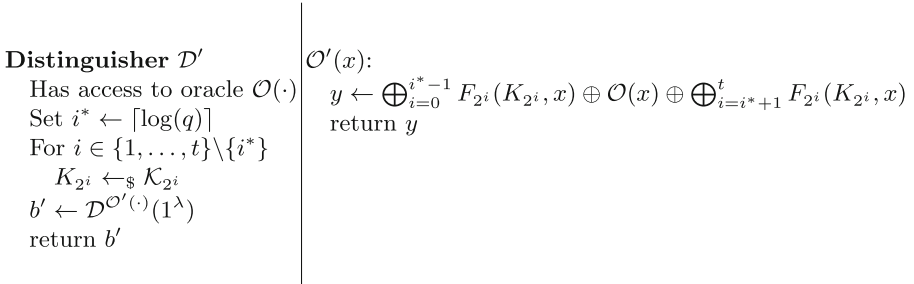
**Theorem 2.** Let  $F_\ell$  and  $F$  be as in Construction 2. Assume that  $F_\ell$  is an  $\ell$ -bounded PRF for every efficiently computable  $\ell = \ell(\lambda)$ . Then  $F$  is a pseudorandom function. Specifically, if  $\mathcal{D}$  is a PPT distinguisher against  $F$  with advantage  $\epsilon$  that makes at most  $q = \text{poly}(\lambda)$  distinct queries, then there exists a PPT distinguisher  $\mathcal{D}'$  (with essentially the same runtime as  $\mathcal{D}$ ) with advantage  $\epsilon$  against  $F_{\ell^*}$ , where  $\ell^* = 2^{\lceil \log(q) \rceil} \leq 2q = \text{poly}(\lambda)$ .

*Proof.* Let  $\mathcal{D}$  be a PPT distinguisher against  $F$  with advantage  $\epsilon$  that makes at most  $q$  distinct queries. Note that since  $q = \text{poly}(\lambda)$  and  $t = \omega(\log(\lambda))$ , it holds  $\log(q) \leq t$  (for a sufficiently large  $\lambda$ ). We will now construct an  $\ell^*$ -query distinguisher  $\mathcal{D}'$  against  $F_{\ell^*}$ , which is given in Fig. 2.

Notice first that  $\mathcal{D}'$  sends at most  $q \leq 2^{\lceil \log(q) \rceil} = \ell^*$  queries to its oracle, as  $\mathcal{D}$  sends at most  $q$  oracle queries. We will now analyze the distinguishing advantage of  $\mathcal{D}'$ . First, assume that  $\mathcal{D}'$ 's oracle  $\mathcal{O}$  implements the function  $F_{\ell^*}(K, \cdot)$  for a randomly chosen  $K \leftarrow_{\S} \mathcal{K}_{\ell^*}$ . Then, the oracle  $\mathcal{O}$  provided by  $\mathcal{D}'$  to  $\mathcal{D}$  implements exactly the function  $F(K, \cdot)$  for a randomly chosen  $K \leftarrow_{\S} \mathcal{K}$ . On the other hand, if  $\mathcal{O}$  behaves like a uniformly random function  $R'$ , then the oracle  $\mathcal{O}'$  also implements a uniformly random function  $R$ , as  $R'$  is independent of the  $K_{2^i}$ . Consequently, it holds that

$$\begin{aligned} \text{Adv}(\mathcal{D}') &= |\Pr[\mathcal{D}'^{F_{\ell^*}(K_{\ell^*}, \cdot)} = 1] - \Pr[\mathcal{D}'^{R'} = 1]| \\ &= |\Pr[\mathcal{D}^F = 1] - \Pr[\mathcal{D}^R = 1]| = \epsilon, \end{aligned}$$

i.e.  $\mathcal{D}'$  distinguishes  $F_{\ell}$  from a uniformly random function  $R'$  with advantage  $\epsilon$ . This concludes the proof.



**Fig. 2.** The distinguisher  $\mathcal{D}'$

### 3.3 Instantiations

Combining Theorems 1 and 2 yields the following

**Theorem 3.** *Let  $t = \omega(\log(\lambda))$ . Let  $\text{PRF}_{\ell} : \{0, 1\}^{\log(2\lambda\ell)} \rightarrow \{0, 1\}^m$  be a small domain PRF, let  $\mathcal{H}_{\ell} : \mathcal{X} \rightarrow \{0, 1\}^{\log(2\ell)}$  be a family of universal hash functions. Define the keyed function  $F : \mathcal{K} \times \mathcal{X} \rightarrow \{0, 1\}^m$  by*

$$F(K, x) = \bigoplus_{i=1}^t \bigoplus_{j=1}^{\lambda} \text{PRF}_{2^i}(K_{2^i}, \text{BIN}(j) \| H_{2^i, j}(x)),$$

where  $K_{2^i} \leftarrow_{\S} \mathcal{K}_{2^i}$  for  $i = 1, \dots, t$  and  $H_{2^i, j} \leftarrow_{\S} \mathcal{H}_{2^i}$  for  $i = 1, \dots, t$  and  $j = 1, \dots, \lambda$ .

If  $\text{PRF}_{\ell}$  is a PRF for every  $\ell = \text{poly}(\lambda)$ , then  $F$  is a PRF. More specifically, if there exists a distinguisher  $\mathcal{D}$  that makes at most  $q = \text{poly}(\lambda)$  queries and distinguishes  $F$  with advantage  $\epsilon$ , then there exists a distinguisher  $\mathcal{D}'$  with essentially the same runtime as  $\mathcal{D}$  that distinguishes  $\text{PRF}_{2^{\lceil q \rceil}}$  with advantage  $\epsilon - q \cdot 2^{-\lambda}$ .

We will briefly discuss efficiency aspects of the construction provided in Theorem 3. First of all notice that the transformation preserves the parallel complexity of the underlying small domain PRF. Moreover, the pre- and post-processing steps are entirely linear, i.e. the computation of universal hash functions and XOR-ing the results.

We will now discuss an instantiation of this PRF using a small domain PRF based on lattice problems. As already mentioned in the introduction, the main purpose of our constructions is obtaining PRFs from standard assumptions that are as tight as possible. Since the construction in the last section allows reducing the security of the constructed large domain PRF to the security of an *adversary specific* small domain PRF, we need a family of small domain PRFs with security as tight as possible. The Naor-Reingold PRF with domain  $\{0, 1\}^n$  allows for a security loss of a factor of  $n$ , while the security loss of a comparable GGM PRF is  $q \cdot n$ . This holds because the DDH problem possesses a *statistical random self-reduction* which allows to compute an arbitrary number of DDH samples from a given sample. The learning with errors (LWE) problem enjoys a similar property, which is stated explicitly in the assumption.

**Definition 4 (Decisional LWE [28,30]).** Let  $p = p(\lambda)$  be a modulus,  $k = k(\lambda) = \text{poly}(\lambda)$  be a positive integer and  $\chi_r = D_{\mathbf{Z}, r}$  be a gaussian distribution with noise parameter  $r$ . Let  $\mathbf{s} \leftarrow_{\S} \mathbb{Z}_p^k$  be chosen uniformly at random. The goal of the LWE( $p, n, \chi_r$ ) problem is to distinguish an arbitrary number of samples  $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e)$  where  $\mathbf{a} \leftarrow_{\S} \mathbb{Z}_p^k$  and  $e \leftarrow_{\S} \chi_{\alpha}$  from samples  $(\mathbf{a}, u)$  where  $u \leftarrow_{\S} \mathbb{Z}_p$  is chosen uniformly at random.

Banerjee, Peikert and Rosen [1] constructed a PRF based on the LWE problem. The PRF has a structure which is similar to the Lewko-Waters PRF but uses a rounding operation instead of exponentiation. Let  $p_1 \gg p_2$ . For an  $x \in \mathbb{Z}_{p_1}$  define  $\lfloor x \rfloor_{p_2} = \lceil (p_2/p_1) \cdot x \rceil \bmod p_2$ . For vectors  $\mathbf{x} \in \mathbb{Z}_{p_1}^k$  define  $\lfloor \cdot \rfloor_{p_2}$  component-wise. We can now state the BPR PRF.

**Theorem 4.** Let  $n = n(\lambda)$  be a positive integer,  $r = r(n)$  be a noise parameter,  $k = k(\lambda) = \text{poly}(\lambda)$  be a positive integer and let  $p_1, p_2$  be moduli such that  $p_1 \geq p_2 \cdot n \cdot (Cr\sqrt{k})^n \cdot k^{\omega(1)}$ , where  $C$  is a universal constant. The keyed function  $\text{BPR}_n : \mathcal{K}_n \times \{0, 1\}^n \rightarrow \mathbb{Z}_{p_2}^k$  with key space  $\mathcal{K}_n = \mathbb{Z}_{p_2}^k \times (\mathbb{Z}_{p_2}^{k \times k})^n$  is defined by

$$\text{BPR}_n(K, x) = \left[ \mathbf{a}^\top \prod_{j=1}^n \mathbf{S}_j^{x_j} \right]_{p_2},$$

where  $\mathbf{a} \leftarrow_{\S} \mathbb{Z}_{p_1}^k$  and  $\mathbf{S}_1, \dots, \mathbf{S}_n \leftarrow_{\S} \chi_{\alpha}^{k \times k}$  and  $K = (\mathbf{a}, \mathbf{S}_1, \dots, \mathbf{S}_n)$ .

Assume that  $\text{LWE}(p_1, k, \chi_r)$  is hard. Then  $\text{BPR}_n$  is a pseudorandom function. Specifically, if there exists a distinguisher  $\mathcal{D}$  that distinguishes  $\text{BPR}_n$  with advantage  $\epsilon$  from a random function, then there exists a distinguisher  $\mathcal{D}'$  with essentially the same runtime as  $\mathcal{D}$  that distinguishes  $\text{LWE}(p_1, k, \chi_r)$  with advantage  $\epsilon/(k \cdot n)$ .

Observe that in Theorem 4 the underlying hardness assumption changes when we increase the input length  $n$ . More specifically, the smaller the term  $p_1/r$  is, the harder the underlying LWE problem  $\text{LWE}(p_1, k, \chi_r)$  becomes. The term  $p_1/r$  is dominated by  $(Cr\sqrt{k})^n$ , thus we aim towards minimizing  $n$ . Observe that we can fix a modulus  $p_2$  for the whole family  $\text{BPR}_n$ , therefore all functions in this family have the same output domain. Plugging the  $\text{BPR}_n$  as small domain PRF in the construction of Theorem 3 yields that  $n$  never becomes larger than  $\log(q)$  for some  $q = \text{poly}(n)$ . Thus we can base the security of the PRF in Theorem 3 on  $\text{LWE}(p_1, k, \chi_r)$  with  $p_1 = p_2 \cdot n \cdot (Cr\sqrt{k})^{\log(2\lambda q)} \cdot k^{\omega(1)}$ , which is slightly super-polynomial (instead of sub-exponential). Moreover, since the  $\text{BPR}_n$  loses only a factor  $k \cdot n$  in its security reduction to LWE, the resulting PRF from Theorem 3 loses only a factor of  $k \cdot \log(2\lambda q)$ . We remark that using the more efficient and tighter Ring-LWE based PRF of [1], the security reduction to Ring-LWE loses only a factor of  $\log(2\lambda q)$ .

While the construction from Theorem 3 preserves the parallel complexity of the small domain PRF, the overall complexity of evaluating the PRF may actually increase. We consider it an interesting problem to find a PRF construction which enjoys similar properties as the  $k$ -LIN based construction in Sect. 4, i.e. one improves the underlying small domain PRF in all aspects, in particular key size and evaluation complexity.

## 4 A Direct Construction from the $k$ -LIN Problem

In this section, we will provide our efficient constructions of number-theoretic PRFs. As discussed above, we will first develop a specialized domain extension technique and then construct a large domain PRF using a tailor-made on-the-fly adaptation strategy.

### 4.1 Preliminaries

In this section, we will generally index vectors of length  $n$  with indices  $0, \dots, n-1$ . We will denote the identity matrix in  $\mathbb{Z}_p^{k \times k}$  by  $\mathbf{I}$ . For vectors  $\mathbf{a} \in \mathbb{Z}_p^k$  we define exponentiation component-wise, i.e.  $g^{\mathbf{a}} = (g^{a_0}, \dots, g^{a_{k-1}})$ . The decisional  $k$ -linear assumption ( $k$ -LIN) [17, 31] generalizes the decisional DDH problem. The decisional  $k$ -Linear assumption becomes (generically) weaker when the parameter



$k$  grows, where the instance  $k = 1$  corresponds to DDH and  $k = 2$  to the linear assumption [6]. The main motivation for these assumptions is that groups are known, where the DDH assumption is easy, but the computational Diffie Hellman problem is supposedly hard [16].

**Definition 5 (Decisional  $k$ -LIN Problem).** *Let  $\mathbb{G}$  be a cyclic group of prime order  $p$ . Let  $g_0, g_1, \dots, g_k \leftarrow_{\S} \mathbb{G}$  and  $s_1, \dots, s_k, r \leftarrow_{\S} \mathbb{Z}_p$  be chosen uniformly at random. The goal of the  $k$ -LIN problem in  $\mathbb{G}$  is to distinguish the distributions*

$$(g_0, \dots, g_k, g_1^{s_1}, \dots, g_k^{s_k}, g_0^{\sum_{i=1}^k s_i}) \text{ and } (g_0, \dots, g_k, g_1^{s_1}, \dots, g_k^{s_k}, g_0^r).$$

We will use the PRF construction of Lewko and Waters [21] as underlying small domain PRF in our construction.

**Theorem 5.** *Let  $k \geq 1$  be a positive integer,  $\mathbb{G} = \langle g \rangle$  be a cyclic group of prime order  $p$  and  $n = n(\lambda)$  be a positive integer. Define the keyed function  $\text{LW}_n : \mathcal{K}_n \times \{0, 1\}^n \rightarrow \mathbb{G}$  with key space  $\mathcal{K}_n = \mathbb{Z}_p^k \times (\mathbb{Z}_p^{k \times k})^n$  by*

$$\text{LW}_n(K, x) = g^{\mathbf{a}^\top \cdot \prod_{j=0}^{n-1} \mathbf{S}_j^{x_j}},$$

where  $\mathbf{a} \leftarrow_{\S} \mathbb{Z}_p^k$ ,  $\mathbf{S}_0, \dots, \mathbf{S}_{n-1} \leftarrow_{\S} \mathbb{Z}_p^{k \times k}$  and  $K = (\mathbf{a}, \mathbf{S}_0, \dots, \mathbf{S}_{n-1})$ . If the  $k$ -LIN problem is hard in  $\mathbb{G}$ , then  $\text{LW}_n$  is a pseudorandom function. More specifically, assume that there exists a PPT distinguisher  $\mathcal{D}$  that distinguishes  $\text{LW}_n$  with advantage  $\epsilon$  from a random function. Then there exists a PPT distinguisher  $\mathcal{D}'$  that distinguishes the  $k$ -LIN problem with advantage  $\epsilon/(k \cdot n)$ .

The Lewko-Waters PRF  $\text{LW}$  as described in the construction in Theorem 5 outputs  $k$  group elements and therefore requires  $k$  exponentiations. We can truncate the output of the  $\text{LW}$  PRF to a single group element, thereby only requiring a single exponentiation.

## 4.2 A Bounded PRF From $k$ -LIN

We will now provide an efficient construction of a bounded PRF from  $k$ -LIN. The security of this bounded PRF tightly reduces to the security of a small domain  $\text{LW}$  PRF and therefore to  $k$ -LIN with only a logarithmic loss.

**Construction 3.** *Let  $k \geq 1$  be a positive integer,  $\mathbb{G} = \langle g \rangle$  be a cyclic group of prime order  $p$ . The keyed function  $F_\ell : \mathcal{K}_\ell \times \mathbb{Z}_p \rightarrow \mathbb{G}$  with key space  $\mathcal{K}_\ell = \mathbb{Z}_p^k \times (\mathbb{Z}_p^{k \times k})^{\log(\ell)}$  is defined by*

$$F_\ell(K_\ell, x) = g^{\mathbf{a}^\top \cdot \prod_{j=0}^{\log(\ell)-1} (\mathbf{S}_j + x^{2^j} \cdot \mathbf{I})},$$

where  $\mathbf{a} \leftarrow_{\S} \mathbb{Z}_p^k$ ,  $\mathbf{S}_0, \dots, \mathbf{S}_{\log(\ell)-1} \leftarrow_{\S} \mathbb{Z}_p^{k \times k}$  and  $K_\ell = (\mathbf{a}, \mathbf{S}_0, \dots, \mathbf{S}_{\log(\ell)-1})$

For a bit  $b \in \{0, 1\}$  let  $\neg b = 1 - b$  denote the negation of  $b$ . For a bit-vector  $\mathbf{c} \in \{0, 1\}^m$  let  $\neg \mathbf{c}$  denote the bit-wise negation of  $\mathbf{c}$ . We will need the following technical lemma.

**Lemma 2.** *Let  $p$  be a prime integer. It holds for all  $r \in \mathbb{N}_{>0}$ , all matrices  $\mathbf{S}_0, \dots, \mathbf{S}_{r-1} \in \mathbb{Z}_p^{k \times k}$  and all  $x \in \mathbb{Z}_p$  that*

$$\prod_{j=0}^{r-1} (\mathbf{S}_j + x^{2^j} \mathbf{I}) = \sum_{\mathbf{c} \in \{0,1\}^r} \left( \prod_{j=0}^{r-1} \mathbf{S}_j^{-c_j} \right) x^{\sum_{j=0}^{r-1} c_j 2^j}.$$

The proof of Lemma 2 works by inductively expanding the left side of the equation and can be found in the full version of this paper. We will now show that the function  $F_\ell$  given in Construction 3 is a bounded PRF.

**Theorem 6.** *Assume that the  $k$ -LIN problem is hard in  $\mathbb{G}$ . Then the function  $F_\ell$  defined in Construction 3 is a bounded PRF. More specifically let  $\ell^* \leq \text{poly}(\lambda)$  and assume that  $\mathcal{D}$  is an  $\ell^*$ -query PPT distinguisher with advantage  $\epsilon$  against the pseudorandomness of  $F_{\ell^*}$ . Then there exists a distinguisher  $\mathcal{D}'$  (with essentially the same runtime as  $\mathcal{D}$ ) with advantage  $\frac{\epsilon}{k \cdot \log(\ell^*)}$  against  $k$ -LIN.*

*Proof.* First observe that  $F_\ell$  can be computed in time  $\text{poly}(\lambda, \log(\ell))$ . Notice that  $\text{LW}_{\log(\ell)}$  and  $F_\ell$  have identical key-spaces. Let  $K = (\mathbf{a}, \mathbf{S}_0, \dots, \mathbf{S}_{\log(\ell)-1})$  be a key for  $F_\ell$ . It follows immediately by Lemma 2 that we can compute  $F_\ell$  by

$$\begin{aligned} F_\ell(K, x) &= g^{\mathbf{a}^\top \cdot \prod_{j=0}^{\log(\ell)-1} (\mathbf{S}_j + x^{2^j} \cdot \mathbf{I})} \\ &= g^{\mathbf{a}^\top \cdot \sum_{\mathbf{c} \in \{0,1\}^{\log(\ell)}} \left( \prod_{i=0}^{\log(\ell)-1} \mathbf{S}_i^{-c_i} \right) x^{\sum_{i=0}^{\log(\ell)-1} c_i 2^i}} \\ &= \prod_{\mathbf{c} \in \{0,1\}^{\log(\ell)}} g^{\mathbf{a}^\top \cdot \left( \prod_{i=0}^{\log(\ell)-1} \mathbf{S}_i^{-c_i} \right) x^{\sum_{i=0}^{\log(\ell)-1} c_i 2^i}} \\ &= \prod_{\mathbf{c} \in \{0,1\}^{\log(\ell)}} \left( \text{LW}_{\log(\ell)}(K, \neg \mathbf{c}) \right)^{x^{\sum_{i=0}^{\log(\ell)-1} c_i 2^i}} \end{aligned}$$

For an integer  $j \in \{0, \dots, 2^{\lceil \log(\ell) \rceil} - 1\}$  let  $\text{BIN}(j)$  denote the  $\log(\ell)$  bit binary representation of  $j$ , i.e. it holds that  $j = \sum_{i=0}^{\log(\ell)-1} \text{BIN}(j)_i 2^i$ . Thus, it holds that

$$F_\ell(K, x) = \prod_{j=0}^{2^{\lceil \log(\ell) \rceil} - 1} \left( \text{LW}_{\log(\ell)}(K, \neg \text{BIN}(j)) \right)^{x^j}. \quad (4)$$

Now, let  $\ell^* \leq \text{poly}(\lambda)$  and assume that  $\mathcal{D}$  is a  $\ell^*$ -query PPT distinguisher that distinguishes  $F_{\ell^*}$  with advantage  $\epsilon$  from a random function. We will construct a PPT distinguisher  $\mathcal{D}'$  that distinguishes  $\text{LW}_{\log(\ell^*)}$  from a random function with advantage  $\epsilon$ . Since the function table of a function  $\{0,1\}^{\log(\ell^*)} \rightarrow \mathbb{G}^k$  has size  $2^{\lceil \log(\ell^*) \rceil} \cdot k \log(|\mathbb{G}|) \leq 2\ell^* k \log(|\mathbb{G}|) = \text{poly}(\lambda)$ , we can assume that  $\mathcal{D}'$ 's input is an explicit function table

<p><b>Distinguisher <math>\mathcal{D}'</math></b>                  Input: Function <math>T : \{0, 1\}^{\log(\ell^*)} \rightarrow \mathbb{G}^k</math>,                  encoded as a function table  <math>b' \leftarrow \mathcal{D}^{\mathcal{O}(\cdot)}(1^\lambda)</math>                  return <math>b'</math></p>	<p><math>\mathcal{O}(x)</math>:  <math>y \leftarrow \prod_{j=0}^{2^{\lceil \log(\ell^*) \rceil} - 1} (T(-\text{BIN}(j)))^{x^j}</math>                  return <math>y</math></p>
--	--

First observe that  $\mathcal{D}'$  is efficient as  $\mathcal{D}$  is efficient and the oracle  $\mathcal{O}$  can be implemented efficiently (as  $2^{\lceil \log(\ell^*) \rceil} \leq 2\ell^*$ ). We will now analyze the advantage of  $\mathcal{D}'$ . If  $\mathcal{D}'$ 's input  $T$  is a function  $\text{LW}_{\log(\ell^*)}(K, \cdot)$  for a randomly chosen  $K \leftarrow_{\S} \mathcal{K}_{\log(\ell^*)}$ , then clearly by (4) it holds that the oracle  $\mathcal{O}$  implements exactly  $F_{\ell^*}(K, \cdot)$ . On the other hand, if  $T$  implements a random function  $R' : \{0, 1\}^{\log(\ell^*)} \rightarrow \mathbb{G}^k$ , then we can express  $R'$  by  $R'(-\text{BIN}(j)) = g^{\mathbf{a}_j^\top}$  for all  $j = 0, \dots, 2^{\lceil \log(\ell^*) \rceil} - 1$ , where the  $\mathbf{a}_0, \dots, \mathbf{a}_{2^{\lceil \log(\ell^*) \rceil} - 1} \leftarrow_{\S} \mathbb{G}^k$  are chosen uniformly at random. Thus, in this case the function computed by  $\mathcal{O}$  is

$$\begin{aligned} \mathcal{O}(x) &= \prod_{j=0}^{2^{\lceil \log(\ell^*) \rceil} - 1} g^{\mathbf{a}_j^\top x^j} \\ &= g^{\sum_{j=0}^{2^{\lceil \log(\ell^*) \rceil} - 1} \mathbf{a}_j^\top x^j}, \end{aligned}$$

which is an  $\ell^*$ -wise independent function. To see this, note that  $g$ -exponentiation is an isomorphism and the function in the exponent  $\sum_{j=0}^{2^{\lceil \log(\ell^*) \rceil} - 1} \mathbf{a}_j^\top x^j$  is a random polynomial of degree  $2^{\lceil \log(\ell^*) \rceil} - 1 \geq \ell^* - 1$ , which is an  $\ell^*$ -wise independent function. Thus, from the view of  $\mathcal{D}$  the oracle  $\mathcal{O}$  implements a random function  $R$ , as  $\mathcal{D}$  sends at most  $\ell^*$  distinct queries. We conclude

$$\begin{aligned} \text{Adv}(\mathcal{D}') &= |\Pr[\mathcal{D}'^{\text{LW}_{\log(\ell^*)}(K, \cdot)} = 1] - \Pr[\mathcal{D}'^R = 1]| \\ &= |\Pr[\mathcal{D}^{F_{\ell^*}(K, \cdot)} = 1] - \Pr[\mathcal{D}^R = 1]| = \epsilon. \end{aligned}$$

By Theorem 5, the distinguisher  $\mathcal{D}'$  yields a distinguisher  $\mathcal{D}''$  with advantage  $\frac{\epsilon}{k \log(\ell^*)}$  against  $k$ -LIN.

### 4.3 In-Place On-the-Fly Adaptation

While the general on-the-fly adaptation strategy we will provide in Sect. 3.2 needs to replicate the the underlying bounded PRF  $t$  times, we will now provide a specific on-the-fly adaptation technique for the bounded PRF  $F_\ell$  provided in the last paragraph that involves no expansion whatsoever. Due to the special algebraic structure of  $F_\ell$ , this on-the-fly adaptation can be done in-place. To obtain an unbounded PRF from the bounded PRF of Construction 3, we will set

the upper limit of the product in the exponent from  $\log(\ell)$  to some  $t = \omega(\log(\lambda))$ . We thereby ensure that  $t$  is large enough that we can embed  $F_{\ell^*}$  in this PRF for any  $\ell^* \leq \text{poly}(\lambda)$ .

**Construction 4.** Let  $k \geq 1$  be a positive integer and  $\mathbb{G} = \langle g \rangle$  be a cyclic group of prime order  $p$ . Let  $t = \omega(\log(\lambda))$ . The keyed function  $F : \mathcal{K} \times \mathbb{Z}_p \rightarrow \mathbb{G}$  with key space  $\mathcal{K} = \mathbb{Z}_p^k \times (\mathbb{Z}_p^{k \times k})^t$  is defined by

$$F(K, x) = g^{\mathbf{a}^\top \cdot \prod_{j=0}^{t-1} (\mathbf{S}_j + x^{2^j} \cdot \mathbf{I})},$$

where  $\mathbf{a} \leftarrow_{\S} \mathbb{Z}_p^k$ ,  $\mathbf{S}_0, \dots, \mathbf{S}_{t-1} \leftarrow_{\S} \mathbb{Z}_p^{k \times k}$  and  $K = (\mathbf{a}, \mathbf{S}_0, \dots, \mathbf{S}_{t-1})$ .

We still need the following auxiliary lemma which states that a randomly chosen matrix from  $\mathbb{Z}_p^{k \times k}$  has full rank, except with small probability.

**Lemma 3.** Let  $p$  be a prime and  $\mathbf{S} \leftarrow_{\S} \mathbb{Z}_p^{k \times k}$  be chosen uniformly at random. Then it holds that

$$\Pr[\text{rank}(\mathbf{S}) < k] \leq \frac{1}{p-1}.$$

The proof of Lemma 3 is standard.

**Theorem 7.** Assume that the  $k$ -LIN problem is hard in  $\mathbb{G}$ . Then the function  $F$  defined in Construction 4 is a PRF. More specifically assume that  $\mathcal{D}$  is PPT distinguisher that makes at most  $q = \text{poly}(\lambda)$  queries and distinguishes  $F$  with advantage  $\epsilon$  from a uniformly random function. Then there exists a PPT distinguisher  $\mathcal{D}^*$  (with essentially the same runtime as  $\mathcal{D}$ ) with advantage  $\frac{1}{k \cdot \log(q)} \cdot \left( \epsilon - \frac{qt}{p-1} \right)$  against  $k$ -LIN in  $\mathbb{G}$ .

*Proof.* Let  $\mathcal{D}$  be a distinguisher with advantage  $\epsilon$  against the pseudorandomness of  $F$  which makes at most  $q = \text{poly}(n)$  queries. Note that since  $q = \text{poly}(\lambda)$  and  $t = \omega(\log(\lambda))$ , it holds  $\log(q) \leq t-1$  (for a sufficiently large  $\lambda$ ). We will define 3 hybrid experiments. In hybrid  $i$   $\mathcal{D}$  is given access to a function  $F^{(i)} : \mathbb{Z}_p \rightarrow \mathbb{G}^k$ .

- Hybrid  $\mathfrak{H}_1$ : In this experiment  $\mathcal{D}$  is given oracle access to the function  $F^{(1)}$  given by  $F^{(1)}(x) = F(K, x)$  for a randomly chosen  $K \leftarrow_{\S} \mathcal{K}$ .
- Hybrid  $\mathfrak{H}_2$ : In this experiment  $\mathcal{D}$  is given oracle access to the function  $F^{(2)}$  defined by

$$F^{(2)}(x) = g^{\mathbf{r}(x)^\top \cdot \prod_{j=\log(q)}^t (\mathbf{S}_j + x^{2^j} \mathbf{I})},$$

where  $\mathbf{r} : \mathbb{Z}_p \rightarrow \mathbb{Z}_p^k$  is a uniformly random function and  $\mathbf{S}_{\log(q)}, \dots, \mathbf{S}_{t-1} \leftarrow_{\S} \mathbb{Z}_p^{k \times k}$ .

- Hybrid  $\mathfrak{H}_3$ : In this experiment  $\mathcal{D}$  is given oracle access to a uniformly random function  $F^{(3)}$ .

Clearly, it holds that

$$|\Pr[\mathcal{D}^{F^{(1)}} = 1] - \Pr[\mathcal{D}^{F^{(3)}} = 1]| \geq \epsilon.$$

Define

$$\begin{aligned} \epsilon_1 &= |\Pr[\mathcal{D}^{F^{(1)}} = 1] - \Pr[\mathcal{D}^{F^{(2)}} = 1]| \\ \epsilon_2 &= |\Pr[\mathcal{D}^{F^{(2)}} = 1] - \Pr[\mathcal{D}^{F^{(3)}} = 1]|. \end{aligned}$$

By the triangle inequality it holds that

$$\epsilon \leq \epsilon_1 + \epsilon_2.$$

We will first show that  $\epsilon_2 \leq qt/(p - 1)$ . Define

$$\mathbf{M}(x) = \prod_{j=\log(q)}^{t-1} (\mathbf{S}_j + x^{2^j} \mathbf{I}),$$

and observe that  $F^{(2)}(x) = g^{\mathbf{r}^\top(x) \cdot \mathbf{M}(x)}$ . Now, if it holds for distinct  $x_1, \dots, x_q \in \mathbb{Z}_p$  that  $\text{rank}(\mathbf{M}(x_i)) = k$  for  $i = 1, \dots, q$ , then  $\mathbf{r}^\top(x_1) \cdot \mathbf{M}(x_1), \dots, \mathbf{r}^\top(x_q) \cdot \mathbf{M}(x_q)$  are distributed independently and uniformly at random. Thus it also holds that  $F^{(2)}(x_1), \dots, F^{(2)}(x_q)$  are distributed independently and uniformly at random. We can define the predicate  $\text{good}(\{x_1, \dots, x_q\}, \mathbf{M})$  to be true if and only if it holds  $\text{rank}(\mathbf{M}(x_i)) = k$  for  $i = 1, \dots, q$ . Applying Lemma 1 yields

$$\begin{aligned} \epsilon_2 &= |\Pr[\mathcal{D}^{F^{(2)}} = 1] - \Pr[\mathcal{D}^{F^{(3)}} = 1]| \\ &\leq \max_{x_1, \dots, x_\ell} \Pr[\neg \text{good}(\{x_1, \dots, x_q\}, \mathbf{M})] \\ &= \max_{x_1, \dots, x_\ell} \Pr[\exists i : \text{rank}(\mathbf{M}(x_i)) < k]. \end{aligned}$$

For a fixed  $x$  it holds that  $\text{rank}(\mathbf{M}(x)) < k$  if there exists a  $j \in \{\log q, \dots, t - 1\}$  with  $\text{rank}(\mathbf{S}_j + x^{2^j} \mathbf{I}) < k$ . Since  $\mathbf{S}_j$  is chosen uniformly at random it holds by Lemma 3 that

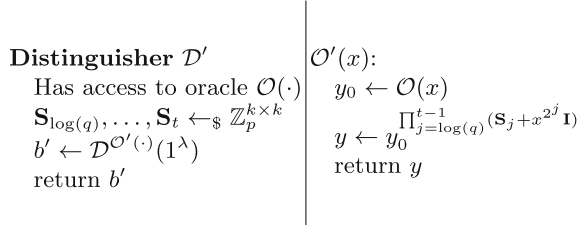
$$\Pr[\text{rank}(\mathbf{S}_j + x^{2^j} \mathbf{I}) < k] = \Pr[\text{rank}(\mathbf{S}_j) < k] \leq \frac{1}{p - 1}.$$

By a union bound over the  $j$  it holds that  $\Pr[\text{rank}(\mathbf{M}(x)) < k] \leq \frac{t}{p - 1}$ . By another union bound over  $i = 1, \dots, q$  it holds that

$$\Pr[\exists i : \text{rank}(\mathbf{M}(x_i)) < k] \leq \frac{qt}{p - 1}$$

We conclude  $\epsilon_2 \leq qt/(p - 1)$  and therefore  $\epsilon_1 \geq \epsilon - qt/(p - 1)$ .

Now let  $\ell^* = 2^{\lceil \log(q) \rceil}$ . We will now construct a PPT distinguisher  $\mathcal{D}'$  that distinguishes the bounded PRF  $F_{\ell^*}$  with advantage  $\epsilon_2$ . The distinguisher  $\mathcal{D}'$  is given in Fig. 3.



**Fig. 3.** The distinguisher  $\mathcal{D}'$

First assume that  $\mathcal{D}'$ 's oracle  $\mathcal{O}$  implements the function  $F_{\ell^*}(K_{\ell^*}, x) = g^{\mathbf{a}^\top \prod_{j=0}^{\log(q)-1} (\mathbf{S}_j + x^{2^j} \mathbf{I})}$  where  $K_\ell = (\mathbf{a}, \mathbf{S}_0, \dots, \mathbf{S}_{\log(q)-1})$  is a uniformly chosen key for  $F_{\ell^*}$ . Then the oracle  $\mathcal{O}'$  implements the function

$$\begin{aligned} \mathcal{O}'(x) &= \left( g^{\mathbf{a}^\top \prod_{j=0}^{\log(q)-1} (\mathbf{S}_j + x^{2^j} \mathbf{I})} \right)^{\prod_{j=\log(q)}^{t-1} (\mathbf{S}_j + x^{2^j} \mathbf{I})} \\ &= g^{\left( \mathbf{a}^\top \prod_{j=0}^{\log(q)-1} (\mathbf{S}_j + x^{2^j} \mathbf{I}) \right) \cdot \prod_{j=\log(q)}^{t-1} (\mathbf{S}_j + x^{2^j} \mathbf{I})} \\ &= g^{\mathbf{a}^\top \prod_{j=0}^{t-1} (\mathbf{S}_j + x^{2^j} \mathbf{I})}. \end{aligned}$$

Thus  $\mathcal{O}'$  implements exactly  $F^{(1)}$ . On the other hand, if  $\mathcal{D}'$ 's oracle  $\mathcal{O}$  implements a random function  $R$  with  $R(x) = g^{\mathbf{r}(x)^\top}$ , where  $\mathbf{r} : \mathbb{Z}_p \rightarrow \mathbb{Z}_p^k$  is a random function, then the oracle  $\mathcal{O}'$  implements the function

$$\begin{aligned} \mathcal{O}'(x) &= \left( g^{\mathbf{r}^\top(x)} \right)^{\prod_{j=\log(q)}^{t-1} (\mathbf{S}_j + x^{2^j} \mathbf{I})} \\ &= g^{\mathbf{r}^\top(x) \cdot \prod_{j=\log(q)}^{t-1} (\mathbf{S}_j + x^{2^j} \mathbf{I})}. \end{aligned}$$

Thus  $\mathcal{O}'(x)$  implements exactly  $F^{(2)}$ . We conclude that

$$\begin{aligned} \text{Adv}(\mathcal{D}') &= |\Pr[\mathcal{D}'^{F_{\ell^*}(K_{\ell^*}, \cdot)} = 1] - \Pr[\mathcal{D}'^R = 1]| \\ &= |\Pr[\mathcal{D}^{F^{(1)}} = 1] - \Pr[\mathcal{D}^{F^{(2)}} = 1]| = \epsilon_1 \geq \epsilon - \frac{qt}{p-1}. \end{aligned}$$

By Theorem 6 this yields a distinguisher  $\mathcal{D}^*$  with advantage  $\frac{1}{k \cdot \log(q)} \cdot \left( \epsilon - \frac{qt}{p-1} \right)$  against  $k$ -LIN in  $\mathbb{G}$ . This concludes the proof.

**PRF with Shorter Keys.** Escala et al. [11] suggested a framework that generalizes Diffie-Hellman like decisional assumptions and proposed a variant of the Lewko-Waters PRF with short keys based on the so-called Matrix-DDH (MDDH) assumption. The proof of Theorem 6 immediately generalizes to this setting. Theorem 7 also holds in this setting, given that the distribution of *aggregated transformation matrices*  $\mathbf{T}$  corresponding to the matrix distribution  $\mathcal{D}_{\ell,k}$  (c.f. [11], Sect. 5.3) used in the MDDH problem satisfies  $\Pr[\text{rank}(\mathbf{T} + x \cdot \mathbf{I}) < k] \leq \text{negl}$  for all  $x \in \mathbb{Z}_p$ .

**Acknowledgements.** We thank Max Rabkin and the reviewers of CRYPTO 2015 for their helpful comments and feedback.

## References

1. Banerjee, A., Peikert, C., Rosen, A.: Pseudorandom functions and lattices. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 719–737. Springer, Heidelberg (2012)
2. Bellare, M., Goldwasser, S.: New paradigms for digital signatures and message authentication based on non-interactive zero knowledge proofs. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 194–211. Springer, Heidelberg (1990)
3. Benabbas, S., Gennaro, R., Vahlis, Y.: Verifiable delegation of computation over large datasets. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 111–131. Springer, Heidelberg (2011)
4. Berman, I., Haitner, I.: From non-adaptive to adaptive pseudorandom functions. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 357–368. Springer, Heidelberg (2012)
5. Berman, I., Haitner, I., Komargodski, I., Naor, M.: Hardness preserving reductions via cuckoo hashing. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 40–59. Springer, Heidelberg (2013)
6. 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)
7. Boneh, D., Montgomery, H.W., Raghunathan, A.: Algebraic pseudorandom functions with improved efficiency from the augmented cascade. In: Al-Shaer, E., Keromytis, A.D., Shmatikov, V. (eds.) ACM CCS 10, pp. 131–140. ACM Press, Chicago (2010)
8. Chandran, N., Garg, S.: Balancing output length and query bound in hardness preserving constructions of pseudorandom functions. In: Meier, W., Mukhopadhyay, D. (eds.) INDOCRYPT 2014. LNCS, vol. 8885, pp. 89–103. Springer, Berlin (2014)
9. Chase, M., Meiklejohn, S.: Déjà Q: Using dual systems to revisit q-type assumptions. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 622–639. Springer, Heidelberg (2014)
10. Dodis, Y., Yampolskiy, A.: A verifiable random function with short proofs and keys. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 416–431. Springer, Heidelberg (2005)
11. Escala, A., Herold, G., Kiltz, E., Ràfols, C., Villar, J.: An algebraic framework for diffie-hellman assumptions. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 129–147. Springer, Heidelberg (2013)

12. Fiore, D., Gennaro, R.: Publicly verifiable delegation of large polynomials and matrix computations, with applications. In: Yu, T., Danezis, G., Gligor, V.D. (eds.) ACM CCS 12, pp. 501–512. ACM Press, Raleigh (2012)
13. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions (extended abstract). In: 25th FOCS, 24–26 October 1984, pp. 464–479. IEEE Computer Society Press, Singer Island (1984)
14. 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)
15. Hazay, C.: Oblivious polynomial evaluation and secure set-intersection from algebraic prfs. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015, Part II. LNCS, vol. 9015, pp. 90–120. Springer, Heidelberg (2015)
16. Herranz, J., Hofheinz, D., Kiltz, E.: The kurosawa-desmedt key encapsulation is not chosen-ciphertext secure. Cryptology ePrint Archive, Report 2006/207 (2006). <http://eprint.iacr.org/>
17. Hofheinz, D., Kiltz, E.: Secure hybrid encryption from weakened key encapsulation. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 553–571. Springer, Heidelberg (2007)
18. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Cryptography with constant computational overhead. In: Ladner, R.E., Dwork, C. (eds.) 40th ACM STOC, pp. 433–442. ACM Press, Canada (2008)
19. Jain, A., Pietrzak, K., Tentes, A.: Hardness preserving constructions of pseudorandom functions. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 369–382. Springer, Heidelberg (2012)
20. Levin, L.: One way functions and pseudorandom generators. *Combinatorica* **7**(4), 357–363 (1987)
21. Lewko, A.B., Waters, B.: Efficient pseudorandom functions from the decisional linear assumption and weaker variants. In: Al-Shaer, E., Jha, S., Keromytis, A.D. (eds.) ACM CCS 09, pp. 112–120. ACM Press, Chicago (2009)
22. Luby, M.: Pseudorandomness and Cryptographic Applications. Princeton University Press, Princeton, NJ, USA (1994)
23. Maurer, U.M.: Indistinguishability of random systems. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 110–132. Springer, Heidelberg (2002)
24. Naor, M., Reingold, O.: Synthesizers and their application to the parallel construction of pseudo-random functions. In: 36th FOCS, 23–25 October 1995, pp. 170–181. IEEE Computer Society Press, Milwaukee (1995)
25. Naor, M., Reingold, O.: Number-theoretic constructions of efficient pseudo-random functions. In: 38th FOCS, 19–22 October 1997, pp. 458–467. IEEE Computer Society Press, Miami Beach (1997)
26. Naor, M., Reingold, O.: On the construction of pseudo-random permutations: Luby-Rackoff revisited (extended abstract). In: 29th ACM STOC, 4–6 May 1997, pp. 189–199. ACM Press, El Pas (1997)
27. Naor, M., Reingold, O., Rosen, A.: Pseudo-random functions and factoring (extended abstract). In: 32nd ACM STOC, 21–23 May 2000, pp. 11–20. ACM Press, Portland (2000)
28. Peikert, C.: Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In: Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, 31 May - 2 June 2009, pp. 333–342 (2009)
29. Razborov, A.A., Rudich, S.: Natural proofs. In: 26th ACM STOC, 23–25 May 1994, pp. 204–213. ACM Press, Montréal (1994)



30. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, 22–24 May 2005, pp. 84–93 (2005)
31. Shacham, H.: A cramer-shoup encryption scheme from the linear assumption and from progressively weaker linear variants. Cryptology ePrint Archive, Report 2007/074 (2007). <http://eprint.iacr.org/>
32. Valiant, L.G.: A theory of the learnable. *Commun. ACM* **27**(11), 1134–1142 (1984). <http://doi.acm.org/10.1145/1968.1972>

# The Iterated Random Permutation Problem with Applications to Cascade Encryption

Brice Minaud and Yannick Seurin<sup>(✉)</sup>

ANSSI, Paris, France

brice.minaud@gmail.com, yannick.seurin@m4x.org

**Abstract.** We introduce and study the *iterated random permutation problem*, which asks how hard it is to distinguish, in a black-box way, the  $r$ -th power of a random permutation from a uniformly random permutation of a set of size  $N$ . We show that this requires  $\Omega(N)$  queries (even for a two-sided, adaptive adversary). As a direct application of this result, we show that cascading a block cipher with the same key cannot degrade its security (as a pseudorandom permutation) more than negligibly.

**Keywords:** Iterated random permutation problem · Block cipher · Pseudorandom permutation · Cascade encryption

## 1 Introduction

A SIMPLE QUESTION. Assume that, as a cautious and slightly paranoid cryptographer, you are not at ease with using AES (say, with 256-bit keys) as is. Instead, you define the block cipher `myAES` as

$$\text{myAES}(k, x) \stackrel{\text{def}}{=} \text{AES}(k, \text{AES}(k, x)),$$

that is, you encipher the plaintext  $x$  twice with the same key  $k$ , hoping that this will increase security. After all, this seems like a cheap, “black-box” way of doubling the number of rounds of AES-256, and it is heuristically well established that increasing the number of rounds of a cipher improves its resistance to various attacks. <sup>1</sup> How can you be sure that the security of your new custom block cipher does not suddenly collapse, becoming *much worse* than the security of AES-256? This seems quite implausible, but can we hope to *formally prove that this cannot happen*?

CASCADE ENCRYPTION. This question is obviously related to what is called *cascade encryption* (or *multiple encryption*), i.e., self-composition of a block cipher.

---

<sup>1</sup> For example, the traditional UNIX password protection mechanism `crypt` uses DES iterated 25 times. However this is in a hashing context and hence not directly relevant to our work.

Given a block cipher  $E$ , the cascade of length  $r$  associated with  $E$  encrypts a message  $x$  as

$$E'_{k_1, \dots, k_r}(x) \stackrel{\text{def}}{=} (E_{k_r} \circ \dots \circ E_{k_1})(x).$$

Cascade encryption has been extensively studied in the setting where the keys  $(k_1, \dots, k_r)$  for the  $r$  calls to the underlying block cipher  $E$  are independent: there are results in the computational setting [LR86, Mye99, MT09, Tes11], in the information-theoretic setting (where only computationally unbounded adversaries are considered) [Vau98, Vau99, Vau03, MP04, MPR07, CPS14], and in the ideal cipher model (in the context of key-length extension) [ABCV98, BR06, GM09, Lee13, DLMS14]. In particular, it is known that cascading a given block cipher with independent keys is *security-amplifying*: if  $E$  is a  $(q, t, \varepsilon)$ -pseudorandom permutation<sup>2</sup> (PRP), then the  $r$ -fold cascade with independent keys for the  $r$  calls to  $E$  is a  $(q', t', r\varepsilon^r)$ -PRP [Tes11], with  $q' \simeq q$  and  $t' \simeq t$ . In the information-theoretic setting, the following slightly weaker result has been shown: if  $E$  is a  $(q, \varepsilon)$ -PRP, then the  $r$ -fold cascade of  $E$  with independent keys is a  $(q, 2^{r-1}\varepsilon^r)$ -PRP [Vau98, Vau99].

On the other hand, virtually nothing is known regarding the security of cascade encryption when the keys used for each call to the underlying block cipher are *not* independent.<sup>3</sup> Not only is it not known whether this might amplify security (and indeed, proving even a tiny security amplification result for cascade encryption without increasing the total key-length would be a major breakthrough), but there is absolutely no guarantee that this might not in some cases *dramatically deteriorate* security.

**OUR RESULT.** In this short paper, we prove that *cascading a block cipher with the same key cannot degrade its security beyond negligible*. By security, we mean the standard notion of (strong) pseudorandomness, defined as follows.

**Definition 1 (Strong Pseudorandom Permutation (SPRP)).** *Let  $E$  be a block cipher with key space  $K$  and message space  $S$ , and  $\text{Perm}(S)$  be the set of all permutations of  $S$ . Let  $\mathcal{D}$  be a distinguisher with oracle access to a permutation and its inverse, and returning a single bit. The SPRP-advantage of  $\mathcal{D}$  against  $E$  is defined as*

$$\text{Adv}_E^{\text{SPRP}}(\mathcal{D}) = \left| \Pr \left[ P \leftarrow_{\S} \text{Perm}(S) : \mathcal{D}^{P, P^{-1}} = 1 \right] - \Pr \left[ k \leftarrow_{\S} K : \mathcal{D}^{E_k, (E_k)^{-1}} = 1 \right] \right|.$$

<sup>2</sup> A block cipher  $E = (E_k)_{k \in K}$  with key space  $K$  is a  $(q, t, \varepsilon)$ -PRP if any adversary making at most  $q$  oracle queries and running in time at most  $t$  can distinguish  $E_k$  (for a random key  $k$ ) from a uniformly random permutation with advantage at most  $\varepsilon$ . See also Definition 1 below.

<sup>3</sup> This setting is sometimes called *product encryption* [Sha49, MM93], *cascade encryption* being reserved to the case where the keys are independent. Yet since the wording *product encryption* carries the idea of iterating a very weak round function rather than a entire block cipher, we will not use it here.

For integers  $q$  and  $t$ , the SPRP-advantage of  $E$  is defined as

$$\mathbf{Adv}_E^{\text{SPRP}}(q, t) = \max_{\mathcal{D}} \mathbf{Adv}_E^{\text{SPRP}}(\mathcal{D}),$$

where the maximum is taken over all distinguishers making at most  $q$  oracle queries and running in time at most  $t$ .  $E$  is a  $(q, t, \varepsilon)$ -SPRP if  $\mathbf{Adv}_E^{\text{SPRP}}(q, t) \leq \varepsilon$ .

A block cipher is deemed secure if its SPRP-advantage is “small” for all “reasonable” parameters  $q$  and  $t$ . We show the following.

**Theorem 1.** *Let  $E$  be a block cipher with message space of size  $N$ , and  $r > 0$  be an integer. Let  $E^r$  be the block cipher obtained by  $r$ -fold self-composition of  $E$  with the same key. (Note that  $E$  and  $E^r$  have the same message and key spaces.) Then*

$$\mathbf{Adv}_{E^r}^{\text{SPRP}}(q, t) \leq \mathbf{Adv}_E^{\text{SPRP}}(rq, t') + \frac{(2r + 1)q}{N},$$

with  $t' = \mathcal{O}(t)$ .

Hence, cascade encryption with the same key does not hurt security beyond negligible, or, to phrase it more positively, it can only improve the security of a given PRP. Theorem 1 follows straightforwardly from a purely information-theoretic result that we now expose in details.

THE ITERATED RANDOM PERMUTATION PROBLEM. Let  $S$  be a set of size  $N > 0$ , and let  $\text{Perm}(S)$  be the group of all permutations of  $S$ . For a permutation  $P \in \text{Perm}(S)$  and an integer  $r \geq 1$ , we denote  $P^r$  the  $r$ -fold self-composition of  $P$ . Consider an adversary (later called distinguisher)  $\mathcal{D}$  having two-sided oracle access to an element  $P \in \text{Perm}(S)$ : it can either query  $P(x)$  and receive the corresponding image  $y$ , or query  $P^{-1}(y)$  and receive the corresponding antecedent  $x$ . We assume that  $\mathcal{D}$  makes at most  $q$  (adaptive queries) before outputting a bit  $b$ . The *iterated random permutation problem* asks how many queries  $q$  needs  $\mathcal{D}$  to distinguish with a noticeable probability the following two situations:

1. a permutation  $P$  is drawn at random from  $\text{Perm}(S)$ , and  $\mathcal{D}$  is given oracle access to  $P$  and  $P^{-1}$ ;
2. a permutation  $P$  is drawn at random from  $\text{Perm}(S)$ , and  $\mathcal{D}$  is given oracle access to  $P^r$  and  $(P^r)^{-1}$ .

In other words, defining the advantage of  $\mathcal{D}$  for the iterated random permutation problem as

$$\mathbf{Adv}_{P, P^r}(\mathcal{D}) = \left| \Pr \left[ P \leftarrow_{\S} \text{Perm}(S) : \mathcal{D}^{P, P^{-1}} = 1 \right] - \Pr \left[ P \leftarrow_{\S} \text{Perm}(S) : \mathcal{D}^{P^r, (P^r)^{-1}} = 1 \right] \right|,$$

and the best advantage at  $q$  queries as

$$\mathbf{Adv}_{P, P^r}(q) = \max_{\mathcal{D}} \mathbf{Adv}_{P, P^r}(\mathcal{D}),$$

where the maximum is taken over all distinguishers making at most  $q$  queries, we ask how  $q$  must grow with  $N$  for  $\mathbf{Adv}_{P,P^r}(q)$  to be constant, say  $\mathbf{Adv}_{P,P^r}(q) \geq 1/2$ . We show that this requires  $q = \Omega(N/r)$ . More precisely, we have the following theorem.

**Theorem 2.** *For any integer  $q$ , one has*

$$\mathbf{Adv}_{P,P^r}(q) \leq \frac{(2r + 1)q}{N}.$$

This theorem is proved in Sect. 2. Theorem 1 follows from Theorem 2 by a simple hybrid argument that we give for completeness.

*Proof of Theorem 1.* Let  $\mathcal{D}$  be a distinguisher against the strong pseudorandomness of  $E^r$  making at most  $q$  oracle queries and running in time at most  $t$ . By definition,

$$\begin{aligned} \mathbf{Adv}_{E^r}^{\text{sprp}}(\mathcal{D}) &= \left| \Pr \left[ P \leftarrow_{\S} \text{Perm}(S) : \mathcal{D}^{P,P^{-1}} = 1 \right] \right. \\ &\quad \left. - \Pr \left[ k \leftarrow_{\S} K : \mathcal{D}^{(E_k)^r, ((E_k)^r)^{-1}} = 1 \right] \right| \\ &\leq \mathbf{Adv}_{P,P^r}(\mathcal{D}) + \mathbf{Adv}_{P^r,E^r}(\mathcal{D}) \\ &\leq \frac{(2r + 1)q}{N} + \mathbf{Adv}_{P^r,E^r}(\mathcal{D}), \end{aligned}$$

where the last inequality follows from Theorem 2 and where

$$\begin{aligned} \mathbf{Adv}_{P^r,E^r}(\mathcal{D}) &= \left| \Pr \left[ P \leftarrow_{\S} \text{Perm}(S) : \mathcal{D}^{P^r, (P^r)^{-1}} = 1 \right] \right. \\ &\quad \left. - \Pr \left[ k \leftarrow_{\S} K : \mathcal{D}^{(E_k)^r, ((E_k)^r)^{-1}} = 1 \right] \right|. \end{aligned}$$

Consider the following distinguisher  $\mathcal{D}'$  against the strong pseudorandomness of  $E$ . It has oracle access to some permutation oracle  $O$  (which is either a random permutation  $P$  or  $E_k$  for a random key  $k$ ) and works as follows: it runs  $\mathcal{D}$ , answering each oracle query of  $\mathcal{D}$  by querying its own oracle  $r$  times to return  $O^r(x)$  for a direct query or  $(O^r)^{-1}(y)$  for an inverse query, and outputting the same decision as  $\mathcal{D}$ . Clearly, the SPRP-advantage of  $\mathcal{D}'$  against  $E$  is exactly  $\mathbf{Adv}_{P^r,E^r}(\mathcal{D})$ , and  $\mathcal{D}'$  makes at most  $rq$  queries and runs in time  $t' = \mathcal{O}(t)$ . Hence

$$\mathbf{Adv}_{P^r,E^r}(\mathcal{D}) \leq \mathbf{Adv}_E^{\text{sprp}}(rq, t'),$$

which concludes the proof. □

*Remark 1.* It can be noted in the proof above that even when  $\mathcal{D}$  is non-adaptive (i.e.,  $\mathcal{D}$  chooses all its queries at the beginning of the security experiment and issues them all at once),  $\mathcal{D}'$  seems to inherently have to query its oracle adaptively. And indeed, Theorem 1 does not extend to the non-adaptive

variant of (strong) pseudorandomness. This can be seen from the following simple example: Consider a (single-key) Even-Mansour cipher [EM97], defined by  $E_k(x) = k \oplus P(k \oplus x)$ , where  $P$  is a public (efficiently computable and invertible) permutation. Assume that  $P$  is an involution (i.e.,  $P^2$  is the identity). Then, the block cipher  $E^2$  obtained by composing  $E$  twice with the same key is highly insecure (even against non-adaptive adversaries making one single query) since it is equal to the identity for any key. On the other hand, modeling  $P$  as a public random involution oracle, it can be shown [DKS12] that  $E$  is secure against non-adaptive distinguishers making at most  $q = 2^{n/2}$  encryption/decryption queries and evaluating  $P$  on at most  $t = 2^{n/2}$  values.<sup>4</sup> This shows that, unlike what Theorem 1 ensures for adaptive security, cascading with the same key can completely ruin security against non-adaptive distinguishers.

We also exhibit a distinguisher whose advantage matches the upper bound of Theorem 2 (up to some constant term which depends on  $r$ ), establishing the following lower bound.

**Theorem 3.** *For  $q \leq N/r$ , one has*

$$\mathbf{Adv}_{P,Pr}(q) \geq \frac{q}{2N} - \frac{r}{N}.$$

The adversary that we use to arrive at Theorem 3 simply picks a random message  $x \in S$  and travels along the cycle on which this point lies, hoping to cycle back to  $x$ . Details of the analysis can be found in Sect. 3. A different attack, based on the search of a fixed point, has been analyzed by Courtois *et al.* [BAC12].

PERSPECTIVES. A natural question is whether it is possible to prove any kind of security amplification for cascade encryption with non-independent keys, which in its full generality would take the form

$$E'_k(x) = (E_{f_r(k)} \circ \dots \circ E_{f_1(k)})(x),$$

where the  $f_i$ 's are permutations of the key space of  $E$ .<sup>5</sup> However, in the particular scenario where the same key is reused (i.e., all  $f_i$ 's are equal to the identity), this clearly requires additional assumptions on the underlying block cipher  $E$ , as indicated (again) by the simple example of a single-key Even-Mansour cipher  $E_k(x) = k \oplus P(k \oplus x)$ , where  $P$  is a public (efficiently computable and invertible) permutation. There is a generic<sup>6</sup> attack on any block cipher of this class requiring

<sup>4</sup> But note that  $E$  can be distinguished from random by an *adaptive* adversary making two queries; namely, denoting  $O$  the adversary's oracle, it queries  $y := O(x)$ ,  $y' := O(y)$ , and checks whether  $y' = x$ .

<sup>5</sup> Remark that, seeing  $E$  as a *round function* rather than a full-fledged block cipher and  $(f_1, \dots, f_r)$  as a key-schedule, this is exactly how most modern block ciphers are designed.

<sup>6</sup> In this context, an attack is said to be generic if it only uses the inner permutation  $P$  as a black-box.

$q = 2^{n/2}$  queries to the encryption/decryption oracle and  $t = 2^{n/2}$  evaluations of the inner permutation  $P$  [Dae91, DKS12]. Note that for any  $r > 1$ , the  $r$ -fold cascade with the same key  $E^r$  is again a one-round single-key Even-Mansour cipher, with inner permutation  $P^r$ , so that it can be generically attacked with  $q = 2^{n/2}$  queries to the encryption/decryption oracle and  $t = r2^{n/2}$  evaluations of  $P$ . Hence, under the assumption that  $P$  is such that the best attack against  $E$  is the generic one, composition with the same key does not amplify the security of such a block cipher. The same argument applies if the  $f_i$ 's are of the form  $f_i(k) = k \oplus c_i$  for public constants  $c_i$ . Indeed, this yields again a one-round single-key Even-Mansour cipher with inner permutation

$$P'(x) = c_r \oplus P(c_r \oplus c_{r-1} \oplus P(c_{r-1} \oplus \dots \oplus c_1 \oplus P(c_1 \oplus x) \dots)).$$

Besides, slide attacks [BW99] show that iterating a truly weak cipher cannot make it arbitrarily strong, independently of the number of iterations. For instance, in the information-theoretic setting, if  $E$  is so weak that it can be distinguished from random using a single plaintext/ciphertext pair with advantage  $1 - 2^{-n/2}$ , then  $E^r$  can be distinguished from random using  $2^{n/2}$  queries with constant probability of success, regardless of the value of  $r$ .<sup>7</sup>

We leave open the problem whether it is possible to find assumptions on the block cipher  $E$  (e.g. resistance to related-key attacks, resistance to key-dependent messages attacks, etc.) sufficient to prove that cascading with non-independent keys is security amplifying.

## 2 Proof of the Main Result

In this section, we prove Theorem 2. We rely on the game-playing framework, and we assume some familiarity of the reader with this technique (see [Sho04, BR06] for more details).

In all the following, given a non-empty set  $S$ , we denote  $\text{Card}(S)$  the number of elements in  $S$ . Let  $\text{Cycl}(S)$  denote the set of *cyclic permutations* of  $S$ , i.e., the subset of  $\text{Perm}(S)$  consisting of permutations with a single cycle. Overall, we will consider the following four games:

- $G_P$ , which gives access to  $P$  and  $P^{-1}$  for  $P \leftarrow_{\S} \text{Perm}(S)$ ;
- $G_{P^r}$ , which gives access to  $P^r$  and  $(P^r)^{-1}$  for  $P \leftarrow_{\S} \text{Perm}(S)$ ;
- $G_C$ , which gives access to  $C$  and  $C^{-1}$  for  $C \leftarrow_{\S} \text{Cycl}(S)$ ;
- $G_{C^r}$ , which gives access to  $C^r$  and  $(C^r)^{-1}$  for  $C \leftarrow_{\S} \text{Cycl}(S)$ .

<sup>7</sup> Indeed, given  $2^{n/2}$  plaintext/ciphertext pairs  $(p, c)$  for  $(E_k)^r$ , the distinguisher against  $E$  can be used to recognize so-called *slid pairs*  $((p, c), (p', c'))$  satisfying  $E_k(p) = p'$ , and hence  $E_k(c) = c'$ . By the birthday paradox, such a slid pair is ensured to exist with constant probability when making  $2^{n/2}$  random queries to  $(E_k)^r$ . Hence, the distinguisher between  $(E_k)^r$  and a random permutation can count the number of plaintext/ciphertext pairs  $((p, c), (p', c'))$ , such that the distinguisher against  $E$  outputs 1 on both inputs  $(p, p')$  and  $(c, c')$ : the expected result is roughly 1 for a random permutation and 2 for  $(E_k)^r$ .

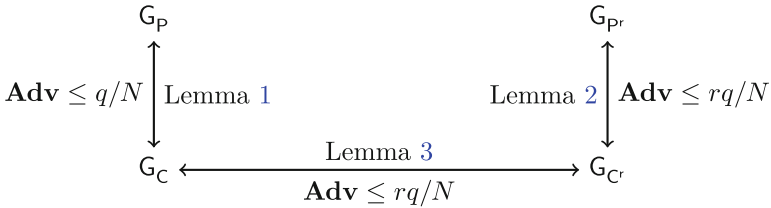
Each game provides two interfaces to the distinguisher, denoted  $Q$  and  $Q^{-1}$ , for querying the underlying permutation respectively in the direct and inverse direction. For example, the formal definition of  $G_P$  is:

```

1  Game  $G_P$ :
2  Initialization:
3     $P \leftarrow_{\S} \text{Perm}(S)$ 
4  procedure  $Q(x)$ :
5    return  $P(x)$ 
6  procedure  $Q^{-1}(y)$ :
7    return  $P^{-1}(y)$ 

```

For any games  $G, H$ , we write  $\text{Adv}_{G,H}(q)$  to denote the maximal advantage attainable by distinguishers between  $G$  and  $H$  within  $q$  queries. We say that two games  $G$  and  $H$  are *equivalent* (within  $q$  queries) if  $\text{Adv}_{G,H}(q) = 0$ . Our goal is to bound  $\text{Adv}_{G_P, G_{Pr}}(q)$ . The layout of the proof is summarized by the following picture:



**Lemma 1.**

$$\text{Adv}_{G_P, G_C}(q) \leq \frac{q}{N}.$$

*Proof.* We start with some useful definitions. A *partial permutation graph*  $(V, E)$  of size  $N$  is a directed graph (with loops allowed) with set of vertices  $V$  of size  $N$  and set of edges  $E \subset V^2$ , where each vertex has out- and in-degree 0 or 1. Given a partial permutation graph  $(V, E)$  containing no cycles and a vertex  $z \in V$ , the *source* of  $z$ , denoted  $\text{So}(z)$ , is the unique  $x \in V$  with in-degree 0 such that there is a path from  $x$  to  $z$  (with the convention that  $\text{So}(z) = z$  if  $z$  has in-degree 0), and the *sink* of  $z$ , denoted  $\text{Si}(z)$ , is the unique  $y \in V$  with out-degree 0 such that there is a path from  $z$  to  $y$  (with the convention that  $\text{Si}(z) = z$  if  $z$  has out-degree 0). The existence and uniqueness of  $\text{So}(z)$  and  $\text{Si}(z)$  when  $(V, E)$  is acyclic are straightforward to prove.

We consider *lazily sampled* versions of  $G_P$  and  $G_C$ . To describe the lazy sampling procedure, we assume that  $G_P$  internally maintains a partial permutation graph over  $V = S$  (with initially no edge). This graph represents the current state of the sampling process. We let  $E \subset S^2$  denote the (time-dependent) set of edges of the graph. We also let  $X$  be the set of vertices with out-degree 1 and  $Y$  be the set of vertices with in-degree 1, these two sets being time-dependent as well. Slightly abusing notation, for  $x \in X$ , we denote  $E(x)$  the unique  $y \in S$  such that  $(x, y) \in E$ , and for  $y \in Y$ , we denote  $E^{-1}(y)$  the unique  $x \in S$  such that  $(x, y) \in E$ . The lazy sampled version of  $G_P$  is as follows:



```

1 Game  $G_p^{\text{lazy}}$ :
2 Variables:
3   Set of edges  $E$ , initially empty
4 procedure  $Q(x)$ :
5   if  $x \notin X$  then
6      $y \leftarrow_{\S} S \setminus Y$ 
7      $E := E \cup \{(x, y)\}$ 
8   return  $E(x)$ 
9 procedure  $Q^{-1}(y)$ :
10  if  $y \notin Y$  then
11     $x \leftarrow_{\S} S \setminus X$ 
12     $E := E \cup \{(x, y)\}$ 
13  return  $E^{-1}(y)$ 

```

*Claim.*  $G_p$  and  $G_p^{\text{lazy}}$  are equivalent (for any number  $q$  of queries).

*Proof.* This is a folklore result (see e.g. [BR06, Sect. 7.4]). Proving it amounts to showing, with the previous notation, that if  $P \leftarrow_{\S} \text{Perm}(S)$  agrees with a partial permutation graph  $(S, E)$ , for  $x \in S \setminus X$ , then  $P(x)$  is uniformly distributed over  $S \setminus Y$ . Equivalently, for any  $x, x_1, \dots, x_n$  pairwise distinct in  $S$ , and  $y_A, y_B, y_1, \dots, y_n$  pairwise distinct in  $S$ , we have

$$\begin{aligned} & \text{Card}\{P \in \text{Perm}(S) : P(x) = y_A, P(x_1) = y_1, \dots, P(x_n) = y_n\} \\ &= \text{Card}\{P \in \text{Perm}(S) : P(x) = y_B, P(x_1) = y_1, \dots, P(x_n) = y_n\}. \end{aligned}$$

To see this, observe that left-hand side composition with transposition  $(y_A \ y_B)$  is a bijection between the two sets. The reasoning for an inverse query is similar. ■

Similarly, the lazy version of  $G_C$  is:

```

1 Game  $G_C^{\text{lazy}}$ :
2 Variables:
3   Set of edges  $E$ , initially empty
4 procedure  $Q(x)$ :
5   if  $x \notin X$  then
6      $y \leftarrow_{\S} S \setminus (Y \cup \{\text{So}(x)\})$ 
7      $E := E \cup \{(x, y)\}$ 
8   return  $E(x)$ 
9 procedure  $Q^{-1}(y)$ :
10  if  $y \notin Y$  then
11     $x \leftarrow_{\S} S \setminus (X \cup \{\text{Si}(y)\})$ 
12     $E := E \cup \{(x, y)\}$ 
13  return  $E^{-1}(y)$ 

```

*Claim.*  $G_C$  and  $G_C^{\text{lazy}}$  are equivalent (for any number  $q$  of queries).

*Proof.* Here, we must show that for any partial permutation graph  $(S, E)$  containing no cycle, with the previous notation and letting  $X = \{x_1, \dots, x_n\}$ ,  $Y = \{y_1, \dots, y_n\}$ ,  $x \in S \setminus X$  and  $y_A, y_B \in S \setminus (Y \cup \{\text{So}(x)\})$ , we have

$$\begin{aligned} & \text{Card}\{C \in \text{Cycl}(S) : C(x) = y_A, C(x_1) = y_1, \dots, C(x_n) = y_n\} \\ &= \text{Card}\{C \in \text{Cycl}(S) : C(x) = y_B, C(x_1) = y_1, \dots, C(x_n) = y_n\}. \end{aligned}$$

Once again, we prove this equality by building a bijection between the two sets. This bijection is:  $C \mapsto (y_A \ y_B) \circ C \circ (\text{Si}(y_A) \ \text{Si}(y_B))$ , where  $(a \ b)$  denotes the transposition swapping  $a$  and  $b$ . If  $C$  is seen as a cyclic graph, this bijection

swaps the position of the longest chain starting from  $y_A$  in  $E$  with the longest chain starting from  $y_B$ . Thus it preserves the cyclic structure and is an involutive bijection between the two sets considered. The reasoning for an inverse query is similar. ■

From the lazy sampling versions of the games, it becomes apparent that  $G_C^{\text{lazy}}$  and  $G_P^{\text{lazy}}$  are identical, unless the event  $[Q(x) = \text{So}(x) \text{ or } Q^{-1}(y) = \text{Si}(y)]$  happens for some query in  $G_P^{\text{lazy}}$ . More precisely, we can rewrite  $G_C^{\text{lazy}}$  using a flag **bad** as follows:

```

1  Game  $G_C^{\text{lazy2}}$ :
2  Variables:
3  Set of edges  $E$ , initially empty
4  bad  $\leftarrow$  false
5  procedure  $Q(x)$ :
6    if  $x \notin X$  then
7       $y \leftarrow_{\S} S \setminus Y$ 
8      if  $y = \text{So}(x)$  then
9        bad  $\leftarrow$  true
10      $y \leftarrow_{\S} S \setminus (Y \cup \{\text{So}(x)\})$ 
11      $E := E \cup \{(x, y)\}$ 
12     return  $E(x)$ 
13 procedure  $Q^{-1}(y)$ :
14 if  $y \notin Y$  then
15    $x \leftarrow_{\S} S \setminus X$ 
16   if  $x = \text{Si}(y)$  then
17     bad  $\leftarrow$  true
18    $x \leftarrow_{\S} S \setminus (X \cup \{\text{Si}(y)\})$ 
19    $E := E \cup \{(x, y)\}$ 
20   return  $E^{-1}(y)$ 

```

Clearly,  $G_C^{\text{lazy}}$  and  $G_C^{\text{lazy2}}$  are equivalent (this technique is called *resampling*, see [BR06, Sect. 7.2]). Moreover,  $G_P^{\text{lazy}}$  and  $G_C^{\text{lazy2}}$  are syntactically identical unless **bad** is set to **true**. By the fundamental lemma of game-playing (see [BR06, Lemma 2]), one has

$$\text{Adv}_{G_P^{\text{lazy}}, G_C^{\text{lazy2}}}(q) \leq \max_{\mathcal{D}} \Pr \left[ \mathcal{D} \text{ sets } \mathbf{bad} \text{ to } \mathbf{true} \text{ in } G_C^{\text{lazy2}} \right],$$

where the maximum is taken over all distinguishers making at most  $q$  queries.

For any distinguisher  $\mathcal{D}$ , the probability that **bad** is set to **true** at the  $i$ -th query of  $\mathcal{D}$  in  $G_C^{\text{lazy2}}$  is exactly  $1/(N - i)$ . Hence, we finally obtain

$$\text{Adv}_{G_P, G_C}(q) = \text{Adv}_{G_P^{\text{lazy}}, G_C^{\text{lazy2}}}(q) \leq 1 - \prod_{i=0}^{q-1} \left( 1 - \frac{1}{N - i} \right) = \frac{q}{N}. \quad \square$$

**Lemma 2.**

$$\text{Adv}_{G_{Pr}, G_{Cr}}(q) \leq \frac{rq}{N}.$$

*Proof.* Any distinguisher between  $P^r$  and  $C^r$  can be used to distinguish between  $P$  and  $C$  at the cost of multiplying the number of queries by  $r$ . More formally, given a distinguisher  $\mathcal{D}$  between  $P^r$  and  $C^r$  making at most  $q$  queries, consider the distinguisher  $\mathcal{D}'$  with oracle access to some permutation oracle  $O$  (which is either  $P$  or  $C$ ) working as follows: it runs  $\mathcal{D}$ , answering each oracle query of  $\mathcal{D}$  by querying its own oracle  $r$  times to return  $O^r(x)$  for a direct query or

$(O^r)^{-1}(y)$  for an inverse query, and outputting the same decision as  $\mathcal{D}$ . Clearly, the advantage of  $\mathcal{D}'$  in distinguishing  $\mathsf{G}_P$  and  $\mathsf{G}_C$  is equal to the advantage of  $\mathcal{D}$  in distinguishing  $\mathsf{G}_{Pr}$  and  $\mathsf{G}_{Cr}$ , and  $\mathcal{D}'$  makes at most  $rq$  queries if  $\mathcal{D}$  makes at most  $q$  queries. Hence, by Lemma 1,

$$\mathbf{Adv}_{\mathsf{G}_{Pr}, \mathsf{G}_{Cr}}(q) \leq \mathbf{Adv}_{\mathsf{G}_P, \mathsf{G}_C}(rq) \leq \frac{rq}{N}. \quad \square$$

**Lemma 3.**

$$\mathbf{Adv}_{\mathsf{G}_C, \mathsf{G}_{Cr}}(q) \leq \frac{rq}{N}.$$

*Proof.* Let  $d = \gcd(N, r)$ . The key observation is that  $\mathsf{G}_{Cr}$  is equivalent to querying a random permutation with  $d$  cycles of equal length.<sup>8</sup> This follows from the fact that the mapping  $C \mapsto C^r$  sends  $\text{Cycl}(S)$  onto the set of permutations with exactly  $d$  cycles of the same length, and that each such permutation has the same number of preimages in  $\text{Cycl}(S)$  under this mapping (the interested reader can refer to Appendix A where we prove this claim). In particular, if  $d = 1$  (when  $N$  and  $r$  are coprime), the games  $\mathsf{G}_C$  and  $\mathsf{G}_{Cr}$  are identical and we are done. If  $d > 1$ , we need to upper bound the advantage of an adversary distinguishing between a random permutation with a single cycle, and a random permutation with  $d$  cycles of equal length.

We now describe a new game  $\mathsf{G}_{Cr}^*$ , which we claim is an equivalent description of  $\mathsf{G}_{Cr}$ .

```

1  Game  $\mathsf{G}_{Cr}^*$ :
2  Initialization:
3     $C \leftarrow_{\S} \text{Cycl}(S)$ 
4     $s_0 \leftarrow_{\S} S$ 
5    for  $i < d$ ,  $s_i = C^{N/d}(s_{i-1})$ 
6  procedure  $\mathsf{Q}(x)$ :
7    if  $x = s_i$  for some  $i$  then
8      return  $C(s_{(i-1) \bmod d})$ 
9    else
10   return  $C(x)$ 
11 procedure  $\mathsf{Q}^{-1}(y)$ :
12 if  $y = C(s_i)$  for some  $i$  then
13   return  $s_{(i+1) \bmod d}$ 
14 else
15   return  $C^{-1}(y)$ 

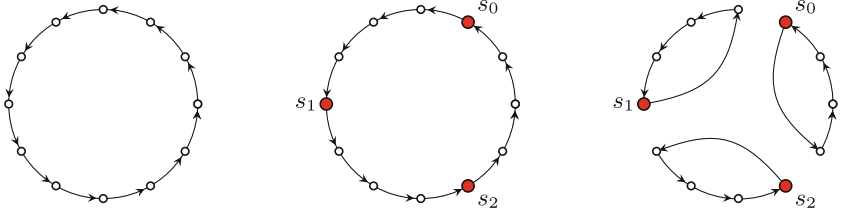
```

Intuitively,  $\mathsf{G}_{Cr}^*$  may be pictured as shown on Fig. 1.

We show in Appendix A that the sampling process underlying game  $\mathsf{G}_{Cr}^*$  is also equivalent to sampling a random permutation with  $d$  cycles of equal length. Meanwhile, we define the game  $\mathsf{G}_C^*$  as being identical to  $\mathsf{G}_{Cr}^*$ , except queries  $\mathsf{Q}(x)$  (resp.  $\mathsf{Q}^{-1}(y)$ ) simply return  $C(x)$  (resp.  $C^{-1}(y)$ ): the  $s_i$ 's play no special role. This corresponds to step 2 in the picture above. The point is that  $\mathsf{G}_C^*$  is clearly an equivalent description of  $\mathsf{G}_C$  (since procedures  $\mathsf{Q}$  and  $\mathsf{Q}^{-1}$  are syntactically the same in both games), while  $\mathsf{G}_{Cr}^*$  is an equivalent description of  $\mathsf{G}_{Cr}$  (indeed, by the two claims proved in Appendix A, they are both equivalent to querying a random permutation with  $d$  cycles of length  $N/d$ ).

Thus  $\mathbf{Adv}_{\mathsf{G}_C, \mathsf{G}_{Cr}}(q) = \mathbf{Adv}_{\mathsf{G}_C^*, \mathsf{G}_{Cr}^*}(q)$ . The following claim completes the proof.

<sup>8</sup> When we say “a random permutation with some property”, more formally we mean “a uniformly random element among permutations with this property”.



1. Pick a random cycle.
2. Pick a random point  $s_0$ . Split into  $d$  equal chains.
3. Redirect the  $s_i$ 's to form  $d$  cycles.

**Fig. 1.** Representation of the game  $G_{Cr}^*$ .

*Claim.*

$$\mathbf{Adv}_{G_C^*, G_{Cr}^*}(q) \leq \frac{dq}{N}.$$

*Proof.* The only difference between  $G_C^*$  and  $G_{Cr}^*$  occurs when  $Q(s_i)$  is queried for some  $i$  (or  $Q^{-1}(C(s_i))$  for backward queries). So  $\mathbf{Adv}_{G_C, G_{Cr}}(q)$  is upper bounded by the advantage of an adversary playing the following game: she queries  $G_C^*$ , and wins iff one of the queries is an  $s_i$  (or  $C(s_i)$  for a backward query). We now prove that the advantage of such an adversary is at most  $dq/N$ .

To show this, we give extra information to the adversary: we grant her full knowledge of the cycle  $C$  before queries begin. Clearly this can only increase her advantage. The point is that queries no longer provide any new information. Thus the game becomes equivalent to the adversary simply trying to guess one of the  $s_i$ 's within  $q$  tries.

Notice that the position of the  $s_i$ 's in the cycle  $C$  is essentially defined modulo  $a = N/d$ . Guessing the position of one of the  $s_i$ 's in the cycle amounts to guessing a value modulo  $a$ . Thus the game is equivalent to guessing a value among  $a$  possibilities, within  $q$  tries. The advantage of an adversary in this game is:

$$1 - \prod_{i=0}^{q-1} \left(1 - \frac{1}{a-i}\right) = 1 - \frac{a-q}{a} = 1 - \frac{N-dq}{N} = \frac{dq}{N}.$$

By the previous reasoning, this is an upper bound for  $\mathbf{Adv}_{G_C^*, G_{Cr}^*}(q)$ . ■

Thus, we have

$$\mathbf{Adv}_{G_C, G_{Cr}}(q) = \mathbf{Adv}_{G_C^*, G_{Cr}^*}(q) \leq \frac{dq}{N} \leq \frac{rq}{N}. \quad \square$$

The proof of Theorem 2 is now complete. Combining Lemmas 1, 2, and 3, we obtain

$$\mathbf{Adv}_{G_p, G_{pr}}(q) \leq \mathbf{Adv}_{G_p, G_C}(q) + \mathbf{Adv}_{G_C, G_{Cr}}(q) + \mathbf{Adv}_{G_{Cr}, G_{pr}}(q) \leq \frac{(2r+1)q}{N}.$$

### 3 A Matching Attack

In this section, we describe a simple attack matching the bound in Theorem 2 within a constant factor, when the number of iterations  $r$  is constant. Our attack uses the following distinguisher  $\mathcal{D}_{\text{cycle}}$  between  $\mathbf{G}_P$  and  $\mathbf{G}_{P^r}$ . It makes  $q$  queries to the interface  $\mathbf{Q}$  (corresponding to  $P$  in  $\mathbf{G}_P$  and  $P^r$  in  $\mathbf{G}_{P^r}$ ), ignoring  $\mathbf{Q}^{-1}$ .

```

1  Distinguisher  $\mathcal{D}_{\text{cycle}}^{\mathbf{Q}}(q)$ 
2   $s_0 \leftarrow_{\S} S$ 
3  for  $i$  in  $\{0, \dots, q-1\}$ :
4     $s_{i+1} \leftarrow \mathbf{Q}(s_i)$ 
5  end for
6  if all  $s_i$ 's are distinct
7    return 0
8  else
9    return 1
    
```

Thus,  $\mathcal{D}_{\text{cycle}}^{\mathbf{Q}}$  returns 1 iff the point  $s_0 \leftarrow_{\S} S$  belongs to a cycle of length at most  $q$ . We have the following result (from which Theorem 3 is a direct application).

**Lemma 4.** *Assume  $q \leq N/r$ . Then*

$$C(r) \frac{q}{N} - \frac{r}{N} \leq \mathbf{Adv}_{\mathbf{G}_P, \mathbf{G}_{P^r}}(\mathcal{D}_{\text{cycle}}) \leq C(r) \frac{q}{N} + \frac{r}{N} \quad \text{with } C(r) = \sum_{d|r} \frac{\phi(d)}{d} - 1$$

where  $d|r$  denotes “ $d$  divides  $r$ ”, and  $\phi$  is Euler’s totient function. Moreover  $C(r) \geq 1/2$  for  $r \geq 2$ .

*Proof.* By definition:

$$\mathbf{Adv}_{\mathbf{G}_P, \mathbf{G}_{P^r}}(\mathcal{D}_{\text{cycle}}) = \left| \Pr \left[ P \leftarrow_{\S} \text{Perm}(S) : \mathcal{D}_{\text{cycle}}^{P^r} = 1 \right] - \Pr \left[ P \leftarrow_{\S} \text{Perm}(S) : \mathcal{D}_{\text{cycle}}^P = 1 \right] \right|.$$

We now set out to compute these two probabilities.

If we pick a random point in a random permutation on  $N$  points, and look at the length of the cycle it belongs to, all lengths  $1 \leq k \leq N$  are equally probable. This is a standard result. It can be shown, for instance, using  $\mathbf{G}_P^{\text{lazy}}$ : if we choose  $s_0 \leftarrow_{\S} S$  and query  $q$  times along a chain, and assume the first  $i$  queries do not create a cycle, then the probability that the next query does is exactly  $1/(N-i)$ . Thus, the probability that  $s_0$  belongs to a cycle of length  $k$  is precisely

$$\prod_{i=0}^{k-1} \left( 1 - \frac{1}{N-i} \right) \cdot \frac{1}{N-k} = \frac{1}{N}.$$

As a consequence, one has

$$\Pr \left[ P \leftarrow_{\S} \text{Perm}(S) : \mathcal{D}_{\text{cycle}}^P(q) = 1 \right] = \frac{q}{N}.$$

We now turn to the case where  $\mathcal{D}_{\text{cycle}}$  interacts with  $P^r$  instead of  $P$ . We let

$$p \stackrel{\text{def}}{=} \Pr[P \leftarrow_{\S} \text{Perm}(S) : \mathcal{D}_{\text{cycle}}^{P^r}(q) = 1].$$

First, we recall two classic equalities regarding the totient function:

$$\sum_{d|n} \phi(d) = n \quad (1) \qquad \phi(n) = n \prod_{p|n, p \in \mathbb{P}} \left(1 - \frac{1}{p}\right) \quad (2)$$

where  $\mathbb{P}$  is the set of prime numbers. Now let  $k$  be the length of the cycle containing  $s_0$ . In  $P^r$  this cycle is broken up into  $d = \gcd(k, r)$  cycles of length  $k/d$ . Hence  $\mathcal{D}_{\text{cycle}}(q)$  detects a cycle iff  $q \geq k/d$ . Since all lengths  $k$  are equally probable, we have

$$\begin{aligned} p &= \frac{1}{N} \text{Card} \left\{ k \leq N : q \geq \frac{k}{\gcd(k, r)} \right\} \\ &= \frac{1}{N} \text{Card} \{ k \leq N : \exists d|r, \gcd(k, r) = d \text{ and } k \leq dq \} \\ &= \frac{1}{N} \text{Card} \{ k : \exists d|r, \gcd(k, r/d) = 1 \text{ and } k \leq \min(q, N/d) \} \quad \text{with } k \leftarrow k/d \\ &= \frac{1}{N} \text{Card} \{ k : \exists d|r, \gcd(k, r/d) = 1 \text{ and } k \leq q \} \quad \text{using } q \leq N/r \\ &= \frac{1}{N} \sum_{d|r} \text{Card} \{ k : \gcd(k, d) = 1 \text{ and } k \leq q \} \text{ since } d \mapsto r/d \text{ is 1-to-1 over } d|r \\ &\geq \frac{1}{N} \sum_{d|r} \text{Card} \left\{ k : \gcd(k, d) = 1 \text{ and } k \leq d \left\lfloor \frac{q}{d} \right\rfloor \right\} \\ &= \frac{1}{N} \sum_{d|r} \phi(d) \left\lfloor \frac{q}{d} \right\rfloor \\ &\geq \frac{1}{N} \sum_{d|r} \phi(d) \left( \frac{q}{d} - 1 \right) \\ &= \frac{q}{N} \sum_{d|r} \frac{\phi(d)}{d} - \frac{r}{N} \quad \text{by (1)}. \end{aligned}$$

One can upper bound  $p$  in a very similar manner, and we obtain the main inequality.

Finally, we show that  $C(r) \geq 1/2$  for  $r \geq 2$ . In fact it holds that for all  $r$ ,  $C(r) \geq 1 - 1/r$ . To see this, observe that if  $r > 2$  is not prime, we have:

$$\begin{aligned} C(r) &= \sum_{d|r} \frac{\phi(d)}{d} - 1 \\ &= \frac{\phi(1)}{1} + \sum_{d|r, 1 < d < r} \frac{\phi(d)}{d} + \prod_{p|r, p \in \mathbb{P}} \left(1 - \frac{1}{p}\right) - 1 \quad \text{by (2)} \end{aligned}$$

$$\begin{aligned} &\geq \sum_{d|r, 1 < d < r} \frac{\phi(d)}{d} + \left(1 - \sum_{p|r, p \in \mathbb{P}} \frac{1}{p}\right) \quad \text{by the union bound} \\ &\geq 1 \quad \text{since } \{p \in \mathbb{P} : p|r\} \subseteq \{1 < d < r : d|r\}. \end{aligned}$$

On the other hand, if  $r$  is prime then  $C(r) = 1 - 1/r$ , hence this is the lower bound. □

**Corollary 1.** *For constant  $r$ , the best distinguisher between  $G_P$  and  $G_{P^r}$  has advantage  $\Theta(q/N)$  as  $N \rightarrow \infty$  and  $q \leq N$  is any function of  $N$ .*

*Proof.* Theorem 2 shows that the advantage is  $\mathcal{O}(q/N)$ . Theorem 3 shows that it is  $\Omega(q/N)$  if  $q \leq N/r$ . On the other hand if  $q > N/r$ , as the advantage can only increase with  $q$ , it is at least  $C(r) \frac{N/r}{N} + o(1) \geq \frac{1}{2r} + o(1) = \Omega(1) = \Omega(q/N)$ . Hence overall the advantage is  $\Theta(q/N)$ . □

For concreteness, if  $r = 2$ , Theorem 3 exhibits a distinguisher with advantage  $0.5q/N$  (under the assumption  $q < N/2$ ), while the main theorem upper bounds the advantage of any such distinguisher by  $5q/N$ . Note that if  $r$  is not constant, the behavior is more complex; informally, only cycles whose length is not coprime with  $r$  are affected by the transformation  $P \mapsto P^r$ . In particular, if  $r$  is prime and  $r > N$ ,  $P \mapsto P^r$  is a permutation of  $\text{Perm}(S)$  and  $G_P$  is indistinguishable from  $G_{P^r}$ .

The problem of finding a tight bound for variable  $r$  is interesting from a purely theoretical standpoint, although we do not know of a situation where such a result would be applicable.

## A Omitted Proofs

We prove here two claims that we used in the proof of Lemma 3. We denote  $\text{Cycl}_d(S)$  the set of permutations of  $S$  with exactly  $d$  cycles of length  $N/d$  (note that  $\text{Cycl}(S) = \text{Cycl}_1(S)$ ).

*Claim.* Let  $S$  be a set of size  $N$ ,  $r \geq 1$  be an integer, and  $d = \text{gcd}(N, r)$ . Let  $\phi$  be the mapping

$$\begin{aligned} \phi : \text{Cycl}(S) &\rightarrow \text{Perm}(S) \\ P &\mapsto P^r. \end{aligned}$$

Then  $\phi(\text{Cycl}(S)) = \text{Cycl}_d(S)$  and all permutations in  $\text{Cycl}_d(S)$  have exactly the same number of preimages by  $\phi$ .

*Proof.* First, we show that for any  $C \in \text{Cycl}(S)$ ,  $\phi(C) \in \text{Cycl}_d(S)$ . Let  $a = N/d$ . Denote

$$C = (x_1 \ x_2 \ \dots \ x_N).$$

Then it is easy to see that  $C^r$  is the product of  $d$  disjoint cycles  $C_i$ ,  $1 \leq i \leq d$ , with

$$C_i = (x_i \ x_{(i+r) \bmod N} \ x_{(i+2r) \bmod N} \ \dots \ x_{(i+(a-1)r) \bmod N}).$$

For  $A \in \text{Cycl}_d(S)$ , we denote  $\phi^{-1}(A)$  the set of preimages of  $A$  by  $\phi$ . We now show that for any  $A, B \in \text{Cycl}_d(S)$ ,  $|\phi^{-1}(A)| = |\phi^{-1}(B)|$ . For  $P \in \text{Perm}(S)$ , we denote  $f_P$  the conjugation by  $P$ , namely  $f_P(Q) = P \circ Q \circ P^{-1}$ . Since  $A$  and  $B$  have the same cycle structure, they belong to the same conjugacy class, i.e., there exists a permutation  $P$  such that  $f_P(A) = B$ . Hence, for any  $C \in \phi^{-1}(A)$ ,  $f_P(C) \in \text{Cycl}(S)$  since conjugation preserves the cycle structure, and one has

$$f_P(C)^r = (P \circ C \circ P^{-1})^r = P \circ C^r \circ P^{-1} = P \circ A \circ P^{-1} = B.$$

This implies that  $f_P(\phi^{-1}(A)) \subseteq \phi^{-1}(B)$ , and hence  $|\phi^{-1}(A)| \leq |\phi^{-1}(B)|$  since  $f_P$  is one-to-one. By symmetry,  $|\phi^{-1}(A)| = |\phi^{-1}(B)|$ . ■

*Claim.* Let  $\psi$  denote the mapping which sends a pair  $(C, s_0) \in \text{Cycl}(S) \times S$  to the permutation defined by game  $\mathbf{G}_{\mathbf{C}^*}$ . Then  $\psi(\text{Cycl}(S) \times S) = \text{Cycl}_d(S)$  and all permutations in  $\text{Cycl}_d(S)$  have exactly the same number of preimages by  $\psi$ .

*Proof.* The fact that  $\psi(C, s_0) \in \text{Cycl}_d(S)$  for any  $(C, s_0) \in \text{Cycl}(S) \times S$  is clear. We now show that for any  $A, B \in \text{Cycl}_d(S)$ ,  $|\psi^{-1}(A)| = |\psi^{-1}(B)|$ . As in the previous claim, there exists a permutation  $P$  such that  $f_P(A) = B$ . We show that for any  $(C, s_0) \in \psi^{-1}(A)$ ,  $(f_P(C), P(s_0)) \in \psi^{-1}(B)$ . First,  $f_P(C) \in \text{Cycl}(S)$  since conjugation preserves the cycle structure. For  $i < d$ , let  $s_i = C^{iN/d}(s_0)$ . By definition of  $\psi$ ,  $A(s_i) = C(s_{(i-1) \bmod d})$  and  $A(x) = C(x)$  for  $x \notin \{s_0, \dots, s_{d-1}\}$ . Let  $s'_0 = P(s_0)$  and for  $i < d$ ,  $s'_i = f_P(C)^{iN/d}(s'_0) = P(s_i)$ . Then

$$\begin{aligned} \psi(f_P(C), P(s_0))(s'_i) &= f_P(C)(s'_{(i-1) \bmod d}) \\ &= P \circ C(s_{(i-1) \bmod d}) = P \circ A(s_i) = f_P(A)(s'_i) = B(s'_i), \end{aligned}$$

and for  $x \notin \{s'_0, \dots, s'_{d-1}\}$ , since  $P^{-1}(x) \notin \{s_0, \dots, s_{d-1}\}$ , one has

$$\psi(f_P(C), P(s_0))(x) = f_P(C)(x) = P \circ C \circ P^{-1}(x) = P \circ A \circ P^{-1}(x) = B(x),$$

which shows that  $\psi(f_P(C), P(s_0)) = B$ . Hence, the image of  $\psi^{-1}(A)$  by the one-to-one mapping  $(C, s_0) \mapsto (f_P(C), P(s_0))$  is a subset of  $\psi^{-1}(B)$ , thus  $|\psi^{-1}(A)| \leq |\psi^{-1}(B)|$ . By symmetry,  $|\psi^{-1}(A)| = |\psi^{-1}(B)|$ . ■

## References

[ABCV98] 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. 390–407. Springer, Heidelberg (1998)

[BAC12] Bard, G.V., Van Ault, S., Courtois, N.T.: Statistics of random permutations and the cryptanalysis of periodic block ciphers. *Cryptologia* **36**(3), 240–262 (2012)

[BR06] 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). <http://eprint.iacr.org/2004/331>



- [BW99] Biryukov, A., Wagner, D.: Slide attacks. In: Knudsen, L.R. (ed.) FSE 1999. LNCS, vol. 1636, pp. 245–259. Springer, Heidelberg (1999)
- [CPS14] Cogliati, B., Patarin, J., Seurin, Y.: Security amplification for the composition of block ciphers: simpler proofs and new results. In: Joux, A., Youssef, A. (eds.) SAC 2014. LNCS, vol. 8781, pp. 129–146. Springer, Heidelberg (2014)
- [Dae91] Daemen, J.: Limitations of the even-mansour construction. In: Matsumoto, T., Imai, H., Rivest, R.L. (eds.) ASIACRYPT 1991. LNCS, vol. 739, pp. 495–498. Springer, Heidelberg (1993)
- [DKS12] Dunkelman, O., Keller, N., Shamir, A.: Minimalism in cryptography: the even-mansour scheme revisited. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 336–354. Springer, Heidelberg (2012)
- [DLMS14] Dai, Y., Lee, J., Mennink, B., Steinberger, J.: The security of multiple encryption in the ideal cipher model. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 20–38. Springer, Heidelberg (2014)
- [EM97] Even, S., Mansour, Y.: A construction of a cipher from a single pseudorandom permutation. *J. Cryptology* **10**(3), 151–162 (1997)
- [GM09] Gaži, P., Maurer, U.: Cascade encryption revisited. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 37–51. Springer, Heidelberg (2009)
- [Lee13] Lee, J.: Towards key-length extension with optimal security: cascade encryption and xor-cascade encryption. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 405–425. Springer, Heidelberg (2013)
- [LR86] Luby, M., Rackoff, C.: Pseudo-random permutation generators and cryptographic composition. In: Symposium on Theory of Computing - STOC 1986, pp. 356–363. ACM (1986)
- [MM93] Maurer, U.M., Massey, J.L.: Cascade ciphers: the importance of being first. **6**(1), 55–61 (1993)
- [MP04] Maurer, U.M., Pietrzak, K.: Composition of random systems: when two weak make one strong. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 410–427. Springer, Heidelberg (2004)
- [MPR07] Maurer, U.M., Pietrzak, K., Renner, R.S.: Indistinguishability amplification. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 130–149. Springer, Heidelberg (2007). <http://eprint.iacr.org/2006/456>
- [MT09] Maurer, U., Tessaro, S.: Computational indistinguishability amplification: tight product theorems for system composition. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 355–373. Springer, Heidelberg (2009)
- [Mye99] Myers, S.: On the development of block-ciphers and pseudo-random function generators using the composition and XOR operators. Ph.D. thesis, University of Toronto (1999)
- [Sha49] Shannon, C.: Communication theory of secrecy systems. *Bell Syst. Tech. J.* **28**(4), 656–715 (1949)
- [Sho04] Shoup, V.: Sequences of games: a tool for taming complexity in security proofs. IACR ePrint Archive, Report 2004/332 (2004). <http://eprint.iacr.org/2004/332.pdf>
- [Tes11] Tessaro, S.: Security amplification for the cascade of arbitrarily weak PRPs: tight bounds via the interactive hardcore lemma. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 37–54. Springer, Heidelberg (2011)

- [Vau98] Vaudenay, S.: Provable security for block ciphers by decorrelation. In: Meinel, C., Morvan, M. (eds.) STACS 1998. LNCS, vol. 1373, pp. 249–275. Springer, Heidelberg (1998)
- [Vau99] Vaudenay, S.: Adaptive-attack norm for decorrelation and super-pseudorandomness. In: Heys, H.M., Adams, C.M. (eds.) SAC 1999. LNCS, vol. 1758, pp. 49–61. Springer, Heidelberg (2000)
- [Vau03] Vaudenay, S.: Decorrelation: a theory for block cipher security. *J. Cryptology* **16**(4), 249–286 (2003)

# The Exact PRF Security of Truncation: Tight Bounds for Keyed Sponges and Truncated CBC

Peter Gazi<sup>1</sup>, Krzysztof Pietrzak<sup>1</sup> (✉), and Stefano Tessaro<sup>2</sup>

<sup>1</sup> IST, Klosterneuburg, Austria

{gazi,pietrzak}@ist.ac.at

<sup>2</sup> UC, Santa Barbara, USA

tessaro@cs.ucsb.edu

**Abstract.** This paper studies the concrete security of PRFs and MACs obtained by keying hash functions based on the sponge paradigm. One such hash function is KECCAK, selected as NIST’s new SHA-3 standard.

In contrast to other approaches like HMAC, the exact security of keyed sponges is not well understood. Indeed, recent security analyses delivered concrete security bounds which are far from existing attacks.

This paper aims to close this gap. We prove (nearly) exact bounds on the concrete PRF security of keyed sponges using a random permutation. These bounds are tight for the most relevant ranges of parameters, i.e., for messages of length (roughly)  $\ell \leq \min\{2^{n/4}, 2^r\}$  blocks, where  $n$  is the state size and  $r$  is the desired output length; and for  $\ell \leq q$  queries (to the construction or the underlying permutation). Moreover, we also improve standard-model bounds.

As an intermediate step of independent interest, we prove tight bounds on the PRF security of the *truncated* CBC-MAC construction, which operates as plain CBC-MAC, but only returns a prefix of the output.

## 1 Introduction

Hash functions are popular building blocks for message-authentication codes (MACs) and pseudorandom functions (PRFs) [23]. The latter are *keyed* functions with the property that their outputs (under a secret key) are indistinguishable from random, except with a (small) distinguishing gap  $\varepsilon$ . PRFs are not only good MACs, but can also be used in a variety of other contexts, from symmetric encryption to key derivation. The to-date most widely used hash-based PRF construction is HMAC [4], and a large body of works has studied its concrete security under different assumptions [3, 18, 21, 26].

It is very likely that hash-based MACs and PRFs will remain popular as the upcoming SHA-3 hash function will replace older designs like MD5, SHA-1 and SHA-256. In contrast to legacy functions, the SHA-3 [1] competition winner KECCAK [10] follows the *sponge* paradigm by Bertoni *et al.* [11]. A key property of sponges is that they resist *extension attacks*, and this enables much simpler approaches than HMAC to derive a PRF. For example, it is suggested (e.g. in [11]) that one may simply pre-pend the key to the message.

OUR CONTRIBUTIONS, IN A NUTSHELL. This paper studies the exact security (i.e., how large is the best distinguishing gap  $\varepsilon$ ?) of keyed sponge constructions. The existing indistinguishability security proof [11], as well as recent targeted analyses [2, 8] yield upper bounds on  $\varepsilon$  for several keying approaches. However, it is not clear that these bounds are the best possible ones. For example, they all degrade *quadratically* in the message length, yet no known generic attacks seem to exploit the message length at all.

In this work, we show that the concrete security of keyed sponges is far superior to what was previously proved, and in particular only minimally depends on the message length. We provide a nearly exact characterization of the PRF security of keyed sponges in the model where the underlying  $n$ -bit permutation is random and the adversary is allowed to issue queries to it. We consider both variants where the key is processed as part of the input (as in HMAC) or where the initialization value takes the role of key (akin to NMAC). Our bounds are *tight* for messages whose length does not exceed (roughly)  $\min\{2^r, 2^{n/4}\}$  blocks, where  $r$  is the output length of the constructions and  $n$  is the underlying block length – a constraint satisfied in all envisioned application scenarios.<sup>1</sup>

The key to our results is a tight analysis of truncated CBC, the construction operating as plain CBC-MAC *without* prefix-free encoding, but only returning a subset of the output bits.

SECURITY OF KEYED SPONGES. The sponge construction relies on an invertible permutation  $\pi$  on  $n$ -bit strings.<sup>2</sup> For a parameter  $b < n$ , it pads the message  $M$  into  $b$ -bit blocks  $M[1], \dots, M[\ell]$ , and keeps a state  $S_i \parallel T_i$ , where  $S_i \in \{0, 1\}^b$  and  $T_i \in \{0, 1\}^{n-b}$ . It outputs the first  $r$  bits of  $S_\ell$  for some<sup>3</sup>  $r \leq b$ , where

$$S_0 \parallel T_0 \leftarrow \kappa^n, \quad S_i \parallel T_i \leftarrow \pi((S_{i-1} \oplus M[i]) \parallel T_{i-1}) \text{ for } i = 1, \dots, \ell.$$

We first consider the keyed construction **GSponge** which sets  $S_0 \parallel T_0$  to equal the  $n$ -bit key value. We prove that when this key is secret and random, no attacker making  $q_C$  queries of length at most  $\ell < 2^{n/4}$   $b$ -bit blocks to **GSponge** using a *random* permutation  $\pi$ , and  $q_\pi$  queries to  $\pi$  itself (and to its inverse  $\pi^{-1}$ ), can distinguish it from a random function, except with distinguishing gap roughly

$$\varepsilon(q_C, q_\pi, \ell) = O\left(\frac{q_C^2 + q_C q_\pi + \ell q_C}{2^{n-r}} + \frac{\ell q_C^2 + \ell q_C q_\pi}{2^n}\right).$$

<sup>1</sup> For SHA-3, we have  $r \geq 224$  and  $n = 1600$ , and thus processing messages exceeding these lengths is practically impossible.

<sup>2</sup> Naming consistency with the TCBC setting below forces us to deviate from the usual naming in the literature on sponges.

<sup>3</sup> The sponge paradigm also allows for outputs of  $r > b$  bits obtained by repeated application of  $\pi$ , an option that does not occur for any of the SHA-3 parameters, and that we will not consider for simplicity in the present paper.

The ideal-permutation model is common for sponge-based constructions, and was used in [2, 8, 11]. For comparison, the previously best known bound was dominated by a term of much larger magnitude  $O((\ell^2 q_C^2 + \ell q_C q_\pi)/2^{n-r})$ .<sup>4</sup>

The salient feature of our new bound is that *the length  $\ell$  only affects terms with denominator  $2^n$ , or appears in a term  $\ell q_C/2^{n-r}$  linear in  $q_C$* . Therefore, the terms with denominator  $2^{n-r}$  are the dominating ones when  $\ell \leq \min\{2^{n/4}, 2^r\}$ , and in this case, our bound simply becomes of the order  $O(\frac{q_C^2 + q_C q_\pi + \ell q_C}{2^{n-r}})$ . We also show that this is tight for  $\max\{q_C, q_\pi\} \geq \ell$ , which is a very common scenario. We leave the question of proving tightness of the remaining terms (or, alternatively, of improving our bound) as a challenging open problem.

Our generalized analysis also shows that *with respect to PRF security*, we are not constrained to any block length  $b < n$  – we could well XOR  $n$ -bit message blocks to the *whole* state. Shorter block lengths can then be enforced by the padding function setting some of the bits to 0 (e.g. the last  $n - b$  bits). Note that full,  $n$ -bit blocks were already used in the design of the sponge-based MAC construction *donkeySponge* [9], which is implicitly covered by our result.

BLACK-BOX KEYING. In most scenarios, black-box keying by pre-pending a key to the message is more desirable than altering the initial value. We provide a complete analysis of key-prepend for arbitrary key-length  $b \cdot w$  (for simplicity, we assume that the key length fits exactly in  $w$  blocks). Our results are in terms of the overall number of queries  $q = q_\pi + \ell \cdot q_C$  made to the permutation. We distinguish two cases: If  $q^2 \leq 2^{n-b}$ , then the additional keying step is secure as long as  $q \leq 2^{bw}$ . In contrast, in the high-query regime, the keying step is secure as long as  $q \leq 2^{bw/2}$ , which effectively requires doubling the key length to achieve a similar security level as in the previous case. (This gap is due to the fact that the high- $q$  regime enables meet-in-the-middle attacks.)

We note that a similar analysis was given in [2] concurrent to our work, but their initial proof was incorrect for  $w \geq 2$ . The current version of [2] uses the results from this paper to obtain a correct bound.

STANDARD-MODEL BOUNDS. We also show improved *standard-model* security of keyed sponges under an assumption on the permutation  $\pi$  introduced by Chang *et al.* [13] and further considered in [2]. The assumption is that a block cipher built from the permutation  $\pi$  as  $E_K^\pi(X) = (0^b \parallel K) \oplus \pi(X \oplus (0^b \parallel K))$  for  $X \in \{0, 1\}^n$  and  $K \in \{0, 1\}^{n-b}$ , where  $b$  is the block length, is a pseudorandom permutation. (Note that this construction is essentially a low-entropy single-key version of the Even-Mansour cipher [19, 20]).

OUR APPROACH: TRUNCATED CBC. Our analysis of keyed sponges builds on top of a result of independent interest – a tight analysis of truncated CBC. In particular,

<sup>4</sup> We note that the recently proved bound of Andreeva *et al.* [2] is slightly more general and modular, as discussed in the full version [22]. In particular, it uses a somewhat different parametrization of the attacker complexity for the second term  $\ell q_C q_\pi / 2^{n-r}$ , which converges to the above in the worst case, but which can make the term smaller (and incomparable to ours) in some scenarios.

our standard-model bounds on sponges are a direct corollary of our truncated-CBC analysis, whereas our bounds in the random permutation model are obtained by a modification of the proof for truncated CBC.

In its *basic* form, the *cipher block-chaining* mode (or CBC, for short) [15, 28] uses a block cipher  $E$  with  $n$ -bit block size. The input  $M \in \{0, 1\}^*$  is first padded into  $n$ -bit blocks  $M = M[1] \dots M[\ell]$ , and then for a key  $K$ ,  $\text{CBC}_K(M)$  outputs the value  $Y_\ell$  resulting from the following iterative computation:  $Y_0 \leftarrow \text{IV}$  and  $Y_i \leftarrow E_K(Y_{i-1} \oplus M[i])$  for all  $i \in [\ell]$ , where  $\text{IV}$  is the initialization value, e.g.,  $\text{IV} = 0^n$ . The basic CBC construction is only secure for messages of *equal* length  $\ell$  [5]. Otherwise, one can easily mount an extension attack.

Three (variants of) solutions prevent extension attacks: The first one is prefix-free encoding of messages [33]. The second outputs  $E_{K'}(\text{CBC}_K(M))$ , under a key  $K'$  independent from  $K$ . (This has been used in EMAC, developed as part of the RACE project [35]). Also, combinations of these ideas have been used in other constructions, like XCBC [12], TMAC [27], and OMAC [24]. The third solution, considered in this paper, is to use *truncation*, i.e., to only output the first  $r < n$  bits of the output. While the first two variants have been extensively analyzed [5–7, 25, 29–31, 33, 34, 36], we are not aware of any explicit analysis of truncated CBC having ever been published,<sup>5</sup> let alone a tight one.

We prove that no attacker making  $q$  queries of length at most  $\ell < 2^{n/4}$  to TCBC using a random permutation can distinguish it from a random function, except with distinguishing gap  $\varepsilon(q, \ell) = O\left(\frac{q(q+\ell)}{2^{n-r}} + \frac{\ell q^2}{2^n}\right)$ . This implies security when the random permutation is replaced by a secure block cipher which is a good PRP. The second term matches the one from the best known analysis of prefix-free CBC [6], whereas we prove that the first term is *tight* for  $q \geq \ell$ .

OUR TECHNIQUES. The analysis of TCBC immediately appears harder than that of related constructions. Existing proofs are based on “Bad event analyses”: For example, for encrypted MAC (as in EMAC), one defines the bad event that for two distinct query messages  $M, M'$ ,  $\text{CBC}^\pi(M)$  and  $\text{CBC}^\pi(M')$  collide, where  $\text{CBC}^\pi$  denotes (plain) CBC-MAC using a random permutation  $\pi$ . It is not hard to prove that as long as no such collision occurs, the outputs  $\pi'(\text{CBC}^\pi(M))$  are indistinguishable from random for an independent permutation  $\pi'$ , and the distinguishing advantage is upper-bounded by the probability of such collisions.<sup>6</sup> This implies indistinguishability when  $\pi$  and  $\pi'$  are replaced by  $E_K$  and  $E_{K'}$ , respectively, for a block cipher  $E$  and independent keys  $K$  and  $K'$ . Similarly, for prefix-free CBC the bad event is that in the evaluation of  $\text{CBC}^\pi(M)$ , the *last* internal query to  $\pi$  is not *fresh*, i.e., it was already made within the same or an earlier evaluation of  $\text{CBC}^\pi$ .

For TCBC, however, if we make a query  $M$ , resulting into output  $Y$  (consisting of the first  $r$  bits of  $\text{CBC}^\pi(M)$ ), we cannot prevent a later query  $M'$ ,

<sup>5</sup> Implicitly, the techniques from sponge analyses [2, 8, 11] yield non-tight bounds of order  $O(\ell^2 q^2 / 2^{n-r})$ .

<sup>6</sup> This notwithstanding, proving bounds on the collision probability is far from trivial [6, 34].

with output  $Y'$ , where  $M'$  is a prefix of  $M$ . Previous machinery only tells us that  $\text{CBC}^\pi(M)$  and  $\text{CBC}^\pi(M')$  are unlikely to collide, but this is insufficient to argue randomness and independence of  $Y$  and  $Y'$ . Moreover, the last query to  $\pi$  within the evaluation of  $\text{CBC}^\pi(M')$  cannot be fresh, as the same query was made *earlier* within the evaluation of  $\text{CBC}^\pi(M)$ . One cannot swap the order of these queries either, as the choice of  $M'$  may well depend *adaptively* on  $Y$ .

To deal with this, our proof will crucially use Patarin’s H-coefficient technique [32], as recently revisited by Chen and Steinberger [14]: We fix a (deterministic) adversary  $\mathcal{A}$  and a *compatible* transcript  $(M_1, Y_1), \dots, (M_q, Y_q)$  (i.e.,  $\mathcal{A}$  indeed would ask such queries  $M_1, \dots, M_q$  if fed with the corresponding answers  $Y_1, \dots, Y_q$ ) and then compare the probabilities that such a transcript would indeed occur with  $\mathcal{A}$  in the real and in the ideal world, respectively. It is easy to see that the latter ideal-world probability is exactly  $2^{-rq}$ , as all outputs of a random functions on (distinct) inputs  $M_1, \dots, M_q$  are random.

However, the real world (where TCBC is evaluated), is far more complex. We are going to show the probability that  $\Pr[\text{TCBC}^\pi(M_i) = Y_i]$  is at least  $(1 - \varepsilon)2^{-rq}$ , for some small  $\varepsilon$ , as long as  $\pi$  is uniformly distributed, conditioned on the following being true:

- For every message  $M_i$ , the value  $Z_i \leftarrow \text{CBC}^\pi(M_i)$  is *unique*. (This is equivalent to stating that the  $\pi$ -query leading to the value  $Z_i$  in the evaluation of  $M_i$  is unique). Recall that the actual output on input  $M_i$  consists of the first  $r$  bits of  $Z_i$ .
- For every message  $M_i$ , and every message  $M_j$  such that  $M_i$  is a prefix of  $M_j$ , the value  $Z_{i,j} \leftarrow \text{CBC}^\pi(M_i \parallel m)$  is unique, where  $m$  is the first  $n$ -bit block in  $M_j$  after the end of  $M_i$ .

It turns out that those conditions are satisfied also except with some small probability  $\delta$ . The actual indistinguishability bound happens to be  $\varepsilon + \delta$  by the H-coefficient method, but determining both values will be at the core of the proof. While an upper bound on  $\delta$  follows by using techniques from [6, 34], upper-bounding  $\varepsilon$  will require new techniques.

Our security proof for sponges is very similar, and will essentially rely on the argument that with good probability (roughly  $\ell q_\pi q_C / 2^n$ ), queries to  $\pi$  made in the evaluation of the sponge queries and direct queries to  $\pi$  by the attacker are disjoint. However, while this is fairly simple to show when the sponge construction is keyed by setting the initial value  $(S_0, T_0)$  to be an  $n$ -bit secret key, proving the same statement when the key is input through several absorbing steps turns out to be more involved. We also give a security proof for this more complex setting using techniques inspired by [17].

**STANDARD-MODEL ANALYSIS.** A recent paper by Chang *et al.* [13] also provides a security analysis of variants of sponge constructions in the *standard* model. We note that (a simple twist of) their very elegant trick reduces the security of the sponge construction with a random IV as the key (this is the construction GSponge) to the security of TCBC for a random permutation *and* the PRP

security against  $\ell q$  queries of the block cipher  $E^\pi$  described above. Our bounds for TCBC directly yield improved standard-model bounds.

Their technique was generalized further in the recent work of Andreeva *et al.* [2]. Beyond the modularity, the main technical contribution of their work is to reduce (in some contexts) the quantity  $\ell q$  in the reduction to the security of  $E^\pi$ . Their contribution is completely orthogonal to ours, and their techniques can be applied in our context.

## 2 Preliminaries

We denote  $[n] := \{1, \dots, n\}$ . Moreover, for a finite set  $\mathcal{S}$  (e.g.,  $\mathcal{S} = \{0, 1\}$ ), we let  $\mathcal{S}^n, \mathcal{S}^+$  and  $\mathcal{S}^*$  be the sets of sequences of elements of  $\mathcal{S}$  of length  $n$ , of arbitrary (but non-zero) length, and of arbitrary length, respectively (with  $\varepsilon$  denoting the empty sequence). We denote by  $S[i]$  the  $i$ -th element of  $S \in \mathcal{S}^n$  for all  $i \in [n]$ . Similarly, we denote by  $S[i \dots j]$ , for every  $1 \leq i \leq j \leq n$ , the sub-sequence consisting of  $S[i], S[i + 1], \dots, S[j]$ , with the convention that  $S[i \dots i] = S[i]$ . Moreover, we denote by  $S \parallel S'$  the concatenation of two sequences in  $\mathcal{S}^*$ , and also, we let  $S \mid T$  be the usual prefix-of relation:  $S \mid T \Leftrightarrow (\exists S' \in \mathcal{S}^* : S \parallel S' = T)$ .

We also let  $\text{Fcs}(m, n)$  be the set of functions mapping  $m$ -bit strings to  $n$ -bit strings, and let  $\text{Perm}(n) \subseteq \text{Fcs}(n, n)$  be the set of *permutations* on the set of  $n$ -bit strings. We use the shorthand  $\text{Fcs}(*, n)$  to denote the set of functions from  $\{0, 1\}^*$  to  $\{0, 1\}^n$ . Finally, we denote the event that an adversary  $\mathcal{A}$ , given access to an oracle  $\mathcal{O}$ , outputs a value  $y$ , as  $\mathcal{A}^{\mathcal{O}} \Rightarrow y$ .

PSEUDORANDOM FUNCTIONS. We consider *keyed* functions  $F : \{0, 1\}^\kappa \times \{0, 1\}^* \rightarrow \{0, 1\}^r$  taking a  $\kappa$ -bit key, arbitrary long messages  $M \in \{0, 1\}^*$  as inputs, and returning an  $r$ -bit output. In particular, we denote as  $F_K$  the map such that  $F(K, \cdot) = F_K(\cdot)$ . We are typically interested in the security of  $F$  as a *pseudorandom function* (or PRF, for short) [23]. This is defined via the following advantage measure, involving an adversary  $\mathcal{A}$ , such that

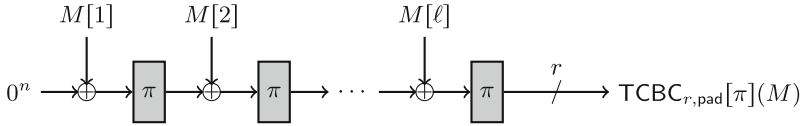
$$\text{Adv}_{\text{F}}^{\text{prf}}(\mathcal{A}) := \left| \Pr \left[ K \xleftarrow{\$} \{0, 1\}^\kappa : \mathcal{A}^{F_K} \Rightarrow 1 \right] - \Pr \left[ f \xleftarrow{\$} \text{Fcs}(*, n) : \mathcal{A}^f \Rightarrow 1 \right] \right|.$$

We consider constructions  $C[\pi] : \{0, 1\}^* \rightarrow \{0, 1\}^r$  invoking a permutation  $\pi \in \text{Perm}(n)$  (we sometimes write  $C^\pi$  instead of  $C[\pi]$ ), and denote by  $C$  the resulting keyed function where the key is a permutation  $\pi \in \text{Perm}(n)$  (i.e., there are  $2^n!$  key values).

For our analysis of keyed sponges, we are also going to consider constructions  $F^\pi : \{0, 1\}^\kappa \times \{0, 1\}^* \rightarrow \{0, 1\}^r$  invoking a *public* randomly chosen permutation  $\pi \xleftarrow{\$} \text{Perm}(n)$ , i.e., one that can be evaluated *directly* by the adversary. For this case, we use the following notation to express the PRF advantage of  $\mathcal{A}$  in the so-called *ideal permutation model*:

$$\begin{aligned} \text{Adv}_{\text{F}, \pi}^{\text{prf}}(\mathcal{A}) := & \left| \Pr \left[ K \xleftarrow{\$} \{0, 1\}^\kappa, \pi \xleftarrow{\$} \text{Perm}(n) : \mathcal{A}^{F_{K, \pi}^{\pi, \pi^{-1}}} \Rightarrow 1 \right] - \right. \\ & \left. - \Pr \left[ f \xleftarrow{\$} \text{Fcs}(*, r), \pi \xleftarrow{\$} \text{Perm}(n) : \mathcal{A}^{f, \pi, \pi^{-1}} \Rightarrow 1 \right] \right|. \end{aligned}$$





**Fig. 1. Truncated CBC**  $\text{TCBC}_{r,\text{pad}}[\pi]$ . Here,  $M[1], \dots, M[\ell]$  are  $n$ -bit blocks resulting from applying the padding scheme  $\text{pad}$  to the input message  $M \in \{0, 1\}^*$ .

MACS AND UNPREDICTABILITY. It is appropriate to note that good PRFs also yield good message-authentication codes (MACs). A concrete security bound for unforgeability can be obtained from our PRF bounds via a standard argument.

### 3 Truncated CBC and its Security

This first part of the paper deals with the concrete security of truncated CBC (TCBC). On top of being of independent interest, the TCBC analysis of this section will be instrumental to analyze the security of keyed sponges in Sect. 5 below. First off, our analysis of keyed sponges in the ideal permutation model will rely on a modification of the proof for TCBC. Second, our standard-model proofs for keyed sponges will directly apply the TCBC result in a black-box way.

TRUNCATED CBC. We fix two parameters  $r < n$  and a *padding scheme*  $\text{pad} : \{0, 1\}^* \rightarrow (\{0, 1\}^n)^+$ , uniquely encoding arbitrary strings into non-empty sequences of  $n$ -bit blocks. (We are *not* requiring the padding to be prefix-free.) The canonical approach computes  $\text{pad}(M)$  by appending a single 1-bit to  $M$ , and then sufficiently many 0’s to reach a length which is a multiple of  $n$ .<sup>7</sup>

The (plain) CBC construction for padding scheme  $\text{pad}$ , using  $\pi \in \text{Perm}(n)$ , computes  $\text{CBC}_{\text{pad}}^\pi(M)$  by first producing  $n$ -bit blocks  $M[1], \dots, M[\ell] \leftarrow \text{pad}(M)$ , and then outputs  $S_\ell$ , where

$$S_0 \leftarrow \text{IV}, \quad S_i \leftarrow \pi(M[i] \oplus S_{i-1}) \quad \text{for all } i = 1, \dots, \ell. \tag{1}$$

Then, truncated CBC (or TCBC, for short) on input  $M \in \{0, 1\}^*$ , outputs the first  $r < n$  bits of CBC evaluated on input  $M$ , i.e.,

$$\text{TCBC}_{r,\text{pad}}^\pi(M) = (\text{CBC}_{\text{pad}}^\pi(M)) [1 \dots r].$$

Also cf. Fig. 1 for a pictorial representation.

SECURITY ANALYSIS. The following theorem characterizes the concrete PRF security of the TCBC construction in the case where  $\pi$  is randomly sampled from  $\text{Perm}(n)$ . By a standard argument, this implies that TCBC is a secure PRF when  $\pi$  is instantiated with a block cipher which is secure as a pseudorandom permutation (PRP).

<sup>7</sup> In this case,  $\text{pad}(M)$  consist of  $\ell = \lceil \frac{|M|+1}{n} \rceil$   $n$ -bit blocks.

**Theorem 1 (Security of TCBC).** *Let  $\mathcal{A}$  be a prf-adversary making at most  $q$  queries, each of length at most  $\ell < 2^{n/4}$   $n$ -bit blocks (after padding). Let  $\text{TCBC} = \text{TCBC}_{r,\text{pad}}[\pi]$  for a random permutation  $\pi \in \text{Perm}(n)$ . Then, for any  $t \geq 1$ ,*

$$\text{Adv}_{\text{TCBC}}^{\text{prf}}(\mathcal{A}) \leq (6t + 17) \frac{\ell q^2}{2^n} + \frac{8n \cdot q^2}{2^{n-r}} + \frac{8q\ell}{2^{n-r}} + \frac{2q}{2^n} + \frac{136\ell^4 q^2}{2^{2n}} + \frac{2q^{t+1}\ell^{t+1}}{2^{nt}}. \quad (2)$$

The proof of Theorem 1 is found below in Sect. 4, where we also give high-level overviews of the individual components of the proof. Here, we first discuss the bound and its tightness.

DISCUSSION OF THE BOUND. First off, note that  $q < 2^{(n-r)/2}$  for the above bound to be negligible. We stress in particular that under the constraints  $\ell < 2^{n/4}$ , the first three terms are the leading ones: Indeed,  $2q/2^n$  is always negligible if the other terms are, and the second last term is for sure negligible as long as  $\ell < 2^{n/4}$ . For the final term, note that  $q\ell < 2^{3n/4}$  for the previous terms to be negligible, and the term becomes negligible for  $t \geq 4$ .

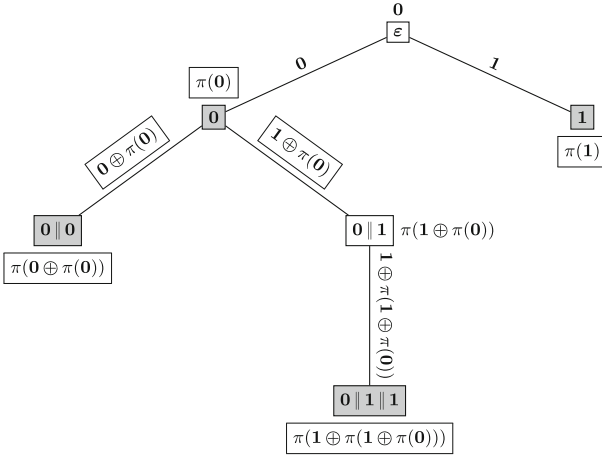
Given this, the most important point is that when additionally  $\ell < 2^r$ , the bound is of the order  $O((q^2 + q\ell)/2^{n-r})$ , and thus only mildly depends on the length. In the full version, we also show how to break TCBC with a  $q$ -query prf-adversary achieving distinguishing advantage roughly  $\Omega(q^2/2^{n-r})$ . The attack works regardless of the permutation  $\pi$  used to instantiate TCBC. Therefore, the bound is tight when additionally  $q \geq \ell$ . We leave it as an open question to determine tightness for other parameter cases.

## 4 Proof of Theorem 1

We start with the high level overview of the proof of Theorem 1, which relies on Patarin’s H-coefficient technique [32], for which we give a self-contained introduction below. (The notation we use is consistent with the recent revisited version of the framework by Chen and Steinberger [14]).

ROADMAP. Sections 4.1 and 4.2 first introduce the notational framework to precisely describe interactions between  $\mathcal{A}$  and the given system – i.e., either  $\text{TCBC}[\pi]$  for  $\pi \stackrel{\$}{\leftarrow} \text{Perm}(n)$  or a truly random function  $f \stackrel{\$}{\leftarrow} \text{Fcs}(*, n)$ . Then, Sect. 4.3 will review the H-coefficient method, and apply it to our setting. Finally, Sect. 4.4 will state and explain the individual probabilistic lemmas composing the rest of the proof, and combine them into the theorem.

SIMPLIFYING ASSUMPTION. Throughout the proof, we assume that (1)  $\mathcal{A}$  is deterministic, (2) it makes *exactly*  $q$  queries, and (3) it never repeats the same query twice. All these assumptions are without loss of generality for an information-theoretic indistinguishability analysis, since any (possibly randomized) adversary making at most  $q$  queries can be transformed into one satisfying these constraints and achieving advantage which is at least as large.



**Fig. 2. (Reduced) message tree.** Message tree for permutation  $\pi \in \text{Perm}(n)$  and four messages  $M_1 = \mathbf{0}$ ,  $M_2 = \mathbf{0} \parallel \mathbf{0}$ ,  $M_3 = \mathbf{0} \parallel \mathbf{1} \parallel \mathbf{1}$ , and  $M_4 = \mathbf{1}$ , where  $\mathbf{b} = b^n$  for  $b \in \{0, 1\}$ . The gray vertices correspond to these four messages. Labels are represented in proximity of the vertices and the edges they are assigned to (as a function of  $\pi$ ) and we let  $\lambda(\varepsilon) = \mathbf{0} = \text{IV}$ . The boxed labels are omitted in the reduced message tree.

**4.1 Message Trees**

We start by introducing some graph-theoretic concepts – the *message tree*, and its reduced version – which capture the inherent combinatorial structure of any  $q$  messages  $M_1, \dots, M_q$  queried by the attacker, as well as the internal values computed while these messages are processed by TCBC. Then, we will put these concepts to work to define transcripts describing the adversary’s interaction with either of TCBC or a random function  $f$ .

We stress that our transcripts will release more information than what is actually seen by the adversary  $\mathcal{A}$ : This information will make the proof simpler, and will not help substantially in distinguishing TCBC from random.

THE MESSAGE TREE. Let  $q \geq 1$ ,  $\pi \in \text{Perm}(n)$ , and let  $M_1, \dots, M_q \in (\{0, 1\}^n)^+$  represent the padded versions of the messages. These  $q$  messages induce a labeled tree  $T^\pi(M_1, \dots, M_q) = (V, E, \lambda, \gamma)$  – called the *message tree*, and often simply denoted as  $T$  or  $T^\pi$ , whenever parameters are clear from the context – defined as follows:

- The set  $V$  of vertices of the tree is  $V := \{M' \in (\{0, 1\}^n)^* : \exists i \in [q] : M' \mid M_i\}$ , where  $\mid$  is the prefix-of partial ordering of strings. In particular, note that the empty string  $\varepsilon$  is a vertex.
- The set  $E \subseteq V \times V$  of edges is  $E := \{(M, M') : \exists m \in \{0, 1\}^n : M' = M \parallel m\}$ .
- We label vertices and edges recursively. Concretely, we define  $\lambda : V \rightarrow \{0, 1\}^n$  and  $\gamma : E \rightarrow \{0, 1\}^n$ . We start with  $\lambda(\varepsilon) = \text{IV}$ . Then, for every vertex  $M \parallel m \in$

$V$  where  $M \in V$  and  $m \in \{0, 1\}^n$ , we set

$$\lambda(M \parallel m) = \pi(\lambda(M) \oplus m).$$

Moreover, we let  $\gamma((M, M \parallel m)) = \lambda(M) \oplus m$ .

An example of a message tree is given in Fig. 2. Note that the vertex labels  $\lambda(M)$  are exactly the values of  $\text{CBC}[\pi](M)$  while the edge labels correspond to the inputs on which  $\pi$  is invoked. Strictly speaking, edge labels are redundant as they can be reconstructed from the vertex labels and  $V$ , but their explicit definition will occasionally simplify descriptions.

For every vertex  $M \in V$  (where possibly  $M \notin \{M_1, \dots, M_q\}$ ), we let  $\mathcal{M}_M$  be the set of  $n$ -bit blocks  $m$  such that  $(M, M \parallel m) \in E$  and  $D_M = |\mathcal{M}_M|$  be the out-degree of vertex  $M$ . It is convenient to denote  $D_i = D_{M_i}$  and  $\mathcal{M}_i = \mathcal{M}_{M_i}$  for all  $i \in [q]$ . A very useful fact we repeatedly use below is that

$$\sum_{i=1}^q D_i < q. \tag{3}$$

This is because every edge  $(M_i, M_i \parallel m)$  can be uniquely mapped to the shortest message  $M_j$  such that  $M_i \parallel m$  is a prefix of  $M_j$ .

THE REDUCED MESSAGE TREE. An abridged version of the above tree, called the *reduced* message tree and denoted  $\overline{T}^\pi = \overline{T}^\pi(M_1, \dots, M_q)$ , will be used in the definition of transcripts below. In particular,  $\overline{T}^\pi$  is obtained from  $T^\pi(M_1, \dots, M_q) = (V, E, \lambda, \gamma)$  as follows. First, we check whether the following condition is true for the given labels  $\lambda$  and  $\gamma$ , and if so, we let  $\overline{T}^\pi = \star$ :

- There exists  $i \in [q]$  and  $M \in V \setminus \{M_i\}$  such that  $\lambda(M_i) = \lambda(M)$ ; or
- For some  $i \in [q]$  and  $m \in \mathcal{M}_i$ , there exists  $M \in V \setminus \{M_i \parallel m\}$  such that  $\lambda(M_i \parallel m) = \lambda(M)$ .

This condition is met when a label of an actual message in  $\{M_1, \dots, M_q\}$ , or of one of its successor vertices, collides with some other label. (Labels not associated with messages are allowed to collide with each other).

If the above condition is not true, we are going to selectively delete some labels from  $T$  (setting them to  $\perp$ ) to obtain a new vertex- and edge-labeled tree, which is the value taken by  $\overline{T}$ . Specifically,

- For all  $i \in [q]$ , we let  $\lambda(M_i) = \perp$ .
- For all  $i \in [q]$  and all  $m \in \mathcal{M}_i$ , we let  $\gamma(M_i, M_i \parallel m) = \perp$ .

In other words, we remove the information necessary to recover the values  $\lambda(M_i)$  for all  $i \in [q]$ .<sup>8</sup>

In Fig. 2, we explicitly show what is omitted when computing the reduced message tree in the case where the tree is not reduced to equal  $\star$ .

<sup>8</sup> Note, however, that some information about these values can be deduced from the rest of the labels using the fact that  $\pi$  is a permutation. As we will implicitly see below, this information is irrelevant.

### 4.2 Interactions and Transcripts

We call a sequence of query/answer pairs  $(M_1, Y_1), \dots, (M_q, Y_q)$  *valid* if the adversary  $\mathcal{A}$  asks indeed queries  $M_1, \dots, M_q$  when fed with answers  $Y_1, \dots, Y_q$  to its queries. (Since  $\mathcal{A}$  is deterministic, the first query  $M_1$  only depends on  $\mathcal{A}$ , the second query only depends on  $\mathcal{A}$  and the first answer  $Y_1$ , etc.) Moreover, a valid *transcript* has the form

$$\tau = ((M_1, Y_1), \dots, (M_q, Y_q), \overline{T}^\pi(M_1, \dots, M_q)),$$

where  $(M_1, Y_1), \dots, (M_q, Y_q)$  is valid,  $\pi : \{0, 1\}^n \rightarrow \{0, 1\}^n$  is a permutation, and  $\overline{T}^\pi(M_1, \dots, M_q)$  is the reduced message tree for  $M_1, \dots, M_q$  with respect to  $\pi$ . We differentiate between the ways in which such valid transcripts are generated in the real and in the ideal worlds, respectively, by defining corresponding distributions  $T_{\text{real}}$  and  $T_{\text{ideal}}$  over the set of valid transcripts:

**Real World.** The transcript  $T_{\text{real}}$  for the adversary  $\mathcal{A}$  is obtained by sampling  $\pi \xleftarrow{\$} \text{Perm}(n)$ , and letting

$$T_{\text{real}} = ((M_1, Y_1), \dots, (M_q, Y_q), \overline{T}^\pi(M_1, \dots, M_q)),$$

where we execute  $\mathcal{A}$ , which asks queries  $M_1, \dots, M_q$  answered with  $Y_i = \text{TCBC}[\pi](M_i)$  for all  $i \in [q]$ , and we let  $\overline{T}^\pi(M_1, \dots, M_q)$  be the corresponding reduced message tree. Note that because  $\mathcal{A}$  is fixed and deterministic,  $T_{\text{real}}$  only depends on  $\pi$ , and thus we occasionally write  $T_{\text{real}}(\pi)$  for the corresponding map.

**Ideal World.** The transcript  $T_{\text{ideal}}$  for the adversary  $\mathcal{A}$  is obtained similarly to the above. However, here we sample *both* a random permutation  $\pi \xleftarrow{\$} \text{Perm}(n)$  and  $q$  independent random values  $Y_1, \dots, Y_q \xleftarrow{\$} \{0, 1\}^r$ , and let

$$T_{\text{ideal}} = T_{\text{ideal}}(Y_1, \dots, Y_q, \pi) = ((M_1, Y_1), \dots, (M_q, Y_q), \overline{T}^\pi(M_1, \dots, M_q)),$$

where  $M_1, \dots, M_q$  are the queries asked when executing  $\mathcal{A}$  and answering each query  $M_i$  with  $Y_i$ , for all  $i \in [q]$ . We stress that here we are *augmenting* the ideal world with an additional *independent* random permutation  $\pi$  which does not actually exist in the original *prf* distinguishing game. This is in order to make real- and ideal-world transcripts alike. In particular, the tree  $\overline{T}^\pi$  is generated according to the permutation  $\pi$ .

Note that the range of  $T_{\text{real}}$  is included in the range of  $T_{\text{ideal}}$  by definition, and that the range of  $T_{\text{ideal}}$  is easily seen to contain all valid transcripts.

### 4.3 The “H-Coefficient Method”: Good and Bad Transcripts

We upper bound the advantage  $\mathcal{A}$  in distinguishing  $\text{TCBC}[\pi]$  for  $\pi \xleftarrow{\$} \text{Perm}(n)$  from a random function using the statistical distance of the transcripts, i.e.,

$$\text{Adv}_{\text{TCBC}}^{\text{prf}}(\mathcal{A}) \leq \text{SD}(T_{\text{real}}, T_{\text{ideal}}) = \frac{1}{2} \sum_{\tau} |\Pr[T_{\text{real}} = \tau] - \Pr[T_{\text{ideal}} = \tau]|, \quad (4)$$

where the sum is over all valid transcripts. This is because a distinguisher for  $T_{\text{real}}$  and  $T_{\text{ideal}}$ , whose optimal advantage is exactly  $\text{SD}(T_{\text{real}}, T_{\text{ideal}})$ , can always output the same decision bit as  $\mathcal{A}$ , ignoring any extra information provided by the transcript.

To this end, we are going to use Patarin’s H-coefficient method [32], recently revisited in [14]. Concretely, this means that we need to partition the set of possible transcripts into *good* transcripts GT and *bad* transcripts BT to enable effective usage of the following lemma.

**Lemma 1 (The H-Coefficient Method).** *Let  $\delta, \varepsilon \in [0, 1]$  be such that:*

- (a)  $\Pr [T_{\text{ideal}} \in \text{BT}] \leq \delta.$
- (b) *For all  $\tau \in \text{GT}$ ,  $\frac{\Pr[T_{\text{real}}=\tau]}{\Pr[T_{\text{ideal}}=\tau]} \geq 1 - \varepsilon.$*

*Then,  $\text{Adv}_{\text{T CBC}}^{\text{prf}}(\mathcal{A}) \leq \text{SD}(T_{\text{real}}, T_{\text{ideal}}) \leq \varepsilon + \delta.$*

More verbally, we require that with very high probability (i.e.,  $1 - \delta$ ) a generated transcript *in the ideal world* is going to be in GT, and moreover, for each such good transcript, the probabilities that it occurs in the real and in the ideal worlds are *roughly* the same, i.e., at most a *multiplicative* factor  $1 - \varepsilon$  apart.

TRANSCRIPT-DEPENDENT QUANTITIES. Concretely, a transcript  $\tau$  will be defined as “good” if the associated reduced message tree  $\bar{T} = (V, E, \gamma, \lambda)$  is not  $\star$  and not “too degenerate”. This requires introducing two relevant quantities. Before doing so, however, we first note that  $\bar{T}$  defines a partial permutation  $\bar{\pi}$  on the  $n$ -bit strings such that  $\bar{\pi}(\gamma(e)) = \lambda(v)$  for every edge  $e$  with end-node  $v$  with  $\gamma(e), \lambda(v) \neq \perp$ , and  $\bar{\pi}(x) = \perp$  for all other inputs.

We will make use of the following quantities, which connect the outputs  $Y_1, \dots, Y_q$  with  $\bar{T}$ .

**Definition 1.** *Let  $\tau = ((M_1, Y_1), \dots, (M_q, Y_q), \bar{T} = (V, E, \gamma, \lambda))$  be a valid transcript with associated partial permutation  $\bar{\pi}$ . Then, for all  $i \in [q]$  we define:*

- $N_i^{(1)}(\tau)$  *is the number of  $x \in \{0, 1\}^n$  with  $\bar{\pi}(x) \neq \perp$  and  $\bar{\pi}(x)[1 \dots r] = Y_i.$*
- $N_i^{(2)}(\tau)$  *is defined as*

$$N_i^{(2)}(\tau) := |\{z \in \{0, 1\}^n : z[1 \dots r] = Y_i \wedge \exists e \in E, m \in \mathcal{M}_i : \gamma(e) = z \oplus m\}|.$$

*Moreover, for  $a \in \{1, 2\}$ , let  $N^{(a)} = \sum_{i=1}^q N_i^{(a)}$ . If  $\bar{T} = \star$ , then these values are set to 0.*

Let us give some intuition on how the above quantities behave for an ideal-world transcript. Note that  $\bar{\pi}$  is defined on at most  $q \cdot \ell$  values, and the value  $\bar{\pi}(x)$ , when first defined, is obtained by sampling a (nearly) uniform random  $n$ -bit string. Thus the expectation of  $N_i^{(1)}$  is roughly  $q\ell/2^r$ , and in turn,  $N^{(1)}$  should be roughly  $q^2\ell/2^r$ .

Also, note that  $N_i^{(2)}$  is the number of  $n$ -bit strings  $z$  which are consistent with  $Y_i$  in their first  $r$  bits which have additionally the property that for some

message block  $m \in \mathcal{M}_i$ ,  $z \oplus m$  is the (non- $\perp$ ) label of an edge in the reduced message tree. Here, the intuition is that every edge label  $\gamma(e)$  in the partial tree is uniform (this won't be quite true, but let us assume it is), and therefore the expectation of  $N_i^{(2)}$  should be (roughly)  $D_i q \ell / 2^r$ , and thus, the expectation of  $N^{(2)}$  should also be roughly  $q^2 \ell / 2^r$ , using  $\sum_i D_i \leq q$ .

**GOOD TRANSCRIPTS.** We require that in a good transcript  $\tau$  the actual values of  $N^{(1)}$  and  $N^{(2)}$  are not too far off their (heuristic) expected values we mentioned above. Moreover, we also want that the reduced message tree is not degenerate, i.e., even though we can't see them, we want the guarantee that the labels of the actual messages (and their successors) are unique – the failure to satisfy this would be signalled by  $\overline{T} = \star$  by definition.

**Definition 2 (Good Transcripts).** *Let  $\tau = ((M_1, Y_1), \dots, (M_q, Y_q), \overline{T})$  be a valid transcript. We say that the transcript is good (and thus  $\tau \in \text{GT}$ ) if the following properties are true (for  $t \geq 1$  as in the theorem statement):*

- (1)  $\overline{T} \neq \star$ .
- (2)  $N^{(1)}(\tau) \leq 3q(qt\ell/2^r + n)$ .
- (3)  $N^{(2)}(\tau) \leq (2n + 1)q^2 + (3t + 1)q^2 \ell / 2^r + 8q^2 \ell^4 / 2^{n+r}$ .

We denote as GT the set of all good transcripts, and BT the set of all *bad* transcripts, i.e., transcripts which can possibly occur (i.e., they are in the range of  $T_{\text{ideal}}$ ) and are not good. More specifically, we denote by  $\text{BT}_i$  the set of all bad transcripts that do not satisfy the  $i$ -th property in the definition of a good transcript above, hence we have  $\text{BT} = \bigcup_{i=1}^3 \text{BT}_i$ .

#### 4.4 High-Level Lemmas and Putting Pieces Together

**BOUNDING THE RATIO.** In Sect. 4.5 below, we are going to prove the following lemma.

**Lemma 2.** *For all good transcripts  $\tau \in \text{GT}$ ,*

$$\frac{\Pr[T_{\text{real}} = \tau]}{\Pr[T_{\text{ideal}} = \tau]} \geq 1 - \left( \frac{N^{(1)} + N^{(2)}}{2^{n-r}} + \frac{2q^2}{2^{n-r}} \right). \tag{5}$$

**BOUNDING PROBABILITY OF BAD TRANSCRIPTS.** We now upper bound the probabilities that a transcript sampled according to  $T_{\text{ideal}}$  is bad via the following lemmas, proved in the full version [22] for lack of space.

**Lemma 3 (Bad-Transcript Analysis for  $\text{BT}_1$ ).**  $\Pr[T_{\text{ideal}} \in \text{BT}_1] \leq 16\ell q^2 / 2^n + 128\ell^4 q^2 / 2^{2n}$ .

**Lemma 4 (Bad-Transcript Analysis for  $\text{BT}_2$ ).** *For  $t \geq 1$  as in the theorem statement,  $\Pr[\text{BT}_2] \leq q/2^n + (q \cdot \ell)^{t+1} / 2^{nt}$ .*

**Lemma 5 (Bad-Transcript Analysis for  $\text{BT}_3$ ).** *For all  $t \geq 1$  as in the theorem statement,  $\Pr[\text{BT}_3] \leq q/2^n + 8q\ell / 2^{n-r} + (q \cdot \ell)^{t+1} / 2^{nt}$ .*

The proof of Lemma 3 above uses and extends techniques inherited from the work of [6] and in particular their analysis of prefix-free CBC. The proof requires some extra work, since we are considering non-prefix free messages.

One would expect that the proofs of Lemmas 4 and 5 follow by application of a simple Chernoff-like argument. Unfortunately, more work is required: First off, the sampled values are not uniform, but only close to uniform. But more importantly, Lemma 5 requires to prove a concentration bound on a series of random variables (the edge labels) which are defined adaptively by an iterative process when computing the reduced message tree. Our technique will essentially show that most of the edge labels will exhibit a high degree of independence, and only a small number of them will be defined by “recycled values” when generating the tree.

COMBINING PIECES. Therefore, we can apply Lemma 1 using  $\varepsilon$  and  $\delta$  extracted from the above lemmas. In particular,

$$\varepsilon = \frac{N^{(1)} + N^{(2)}}{2^{n-r}} + \frac{2q^2}{2^{n-r}} \leq \frac{(6t + 1)\ell q^2}{2^n} + \frac{8nq^2}{2^{n-r}} + \frac{8q^2\ell^4}{2^{2n}},$$

and

$$\delta = \frac{2q}{2^n} + \frac{8q\ell}{2^{n-r}} + \frac{16\ell q^2}{2^n} + \frac{128\ell^4 q^2}{2^{2n}} + 2 \frac{(q \cdot \ell)^{t+1}}{2^{nt}}.$$

In particular, we simplify

$$\varepsilon + \delta \leq (6t + 17) \frac{\ell q^2}{2^n} + \frac{8n \cdot q^2}{2^{n-r}} + \frac{8q\ell}{2^{n-r}} + \frac{2q}{2^n} + \frac{136\ell^4 q^2}{2^{2n}} + \frac{2q^{t+1}\ell^{t+1}}{2^{nt}}.$$

#### 4.5 Lower Bounding the Probability Ratio (Proof of Lemma 2)

We fix a good transcript  $\tau = ((M_1, Y_1), \dots, (M_q, Y_q), \bar{T}) \in \text{GT}$ , where  $\bar{T} = (V, E, \lambda, \gamma) \neq \star$ . To start with, we define the set  $\Omega[\tau]$  of  $\pi$ 's consistent with  $\tau$  in the real world, i.e.,

$$\Omega[\tau] := \{\pi \in \text{Perm}(n) : T_{\text{real}}(\pi) = \tau\}.$$

Moreover, let  $\Omega'[\tau]$  be the set of permutations  $\pi$  which are consistent with the labels of the reduced message tree  $\bar{T}$ , however  $\text{TCBC}^\pi(M_i)$  does not need to equal  $Y_i$  for all  $i$ . More formally,

$$\Omega'[\tau] := \left\{ \pi \in \text{Perm}(n) : \bar{T}^\pi(M_1, \dots, M_q) = \bar{T} \right\}.$$

Now, we define

$$\bar{p}(\tau) := \frac{|\Omega[\tau]|}{|\Omega'[\tau]|} = \Pr \left[ \pi \stackrel{\$}{\leftarrow} \Omega'[\tau] : \pi \in \Omega[\tau] \right].$$

This is the probability that a random permutation  $\pi$  consistent with the constraints on the *reduced* message tree also yields  $\text{TCBC}^\pi(M_i) = Y_i$  for all  $i \in [q]$ .<sup>9</sup>

<sup>9</sup> Note that sampling such a  $\pi$  is *not* the same as sampling a random  $\pi$  which is consistent with  $\bar{\pi}$ . The latter may allow for some permutations which are not possibly generating a message tree which can be reduced to  $\bar{T}$ .



The following claim will reduce lower bounding the probability ratio to lower bounding  $\bar{p}(\tau)$  for  $\tau \in \text{GT}$ , and its proof is omitted here.

*Claim (1).* For all good transcripts  $\tau \in \text{GT}$ ,  $\frac{\Pr[\text{T}_{\text{real}}=\tau]}{\Pr[\text{T}_{\text{ideal}}=\tau]} = 2^{r \cdot q} \cdot \bar{p}(\tau)$ .

It is easy to see that the ordering of  $(M_1, Y_1), \dots, (M_q, Y_q)$  does not affect  $\bar{p}(\tau)$ , and we therefore assume without loss of generality that it is prefix-preserving, i.e., if  $M_i \mid M_j$ , then  $i < j$ . Let  $e_i$  be the edge leading to  $M_i$ .

To study  $\bar{p}(\tau)$ , we consider an *iterative process* where we extend  $\bar{\pi}$  defined by  $\bar{T}$  as above, setting the values of  $\bar{\pi}(\gamma(e_i)) = \lambda(M_i)$  for  $i = 1, \dots, q$  one after the other in this order. Moreover, upon setting  $\lambda(M_i) = \bar{\pi}(\gamma(e_i)) \leftarrow Z_i$ , for all  $m \in \mathcal{M}_i$ , we do the following:

- We set  $\gamma(M_i, M_i \parallel m) \leftarrow Z_i \oplus m$
- If we know the value  $\lambda(M_i \parallel m)$ , we set  $\bar{\pi}(Z_i \oplus m) \leftarrow \lambda(M_i \parallel m)$ .<sup>10</sup>

Note that depending on the choice of the  $Z_i$ 's, the resulting  $\bar{\pi}$  may or may not be a partial permutation, or we may overwrite values, etc. We will of course be only interested in sequences of  $Z_i$ 's which maintain the permutation property.

To this end, let  $\mathcal{L} = \mathcal{L}(\bar{T}, (M_1, Y_1), \dots, (M_q, Y_q))$  be the set of sequences  $(z_1, \dots, z_q)$  of *distinct*  $q$  values such that  $z_i[1 \dots r] = Y_i$  for all  $i \in [q]$  and when assigning  $\lambda(M_i) \leftarrow z_i$  for all  $i \in [q]$  in the above process, at the end of the process the labels  $\lambda(M_i) = z_i$  are unique (i.e., no other vertex has the same label) and moreover, for all  $i \in [q]$  and all  $m \in \mathcal{M}_i$ , we also have that  $\lambda(M_i \parallel m)$  is a unique label.

The following claim is proved in the full version [22] and reduces the problem of lower bounding  $\bar{p}(\tau)$  to that of lower bounding the size of  $\mathcal{L}$ .

*Claim (2).* For all good transcripts  $\tau \in \text{GT}$ ,  $\bar{p}(\tau) \geq \frac{|\mathcal{L}|}{2^{nq}}$ .

THE LOWER BOUND ON  $|\mathcal{L}|$ . Here, to lower bound  $|\mathcal{L}|$ , we go through the above process, and assuming  $z_1, \dots, z_{i-1}$  have been fixed, we see how many ways we still have to fix  $z_i$  satisfying the invariant that it is still possible to reach sequence  $(z_1, \dots, z_q) \in \mathcal{L}$ . In particular, at every step, we are going to exclude values  $z_i$  with the following properties:

- (1)  $z_i[1 \dots r] \neq Y_i$
- (2) There exists  $1 \leq j < i$  such that  $z_j = z_i$ .
- (3) There exists  $M \notin \{M_1, \dots, M_q\}$  with  $\lambda(M) = z_i$ .
- (4) There exists  $1 \leq j < i$ ,  $m' \in \mathcal{M}_j$ ,  $m \in \mathcal{M}_i$  such that  $m \oplus z_i = m' \oplus z_j$ .
- (5) There exists a  $n$ -bit value  $m \in \mathcal{M}_i$  and an edge  $e \in E$  with tail node not in  $\{M_1, \dots, M_q\}$  such that  $\gamma(e) = z_i \oplus m$ .

It is clear that we reach a sequence in  $\mathcal{L}$  if at every step we pick a non-excluded value. In particular, note that (4) and (5) are necessary for us to ensure that the edge labels leading to successor vertices of  $M_i$  are *fresh*, which is necessary to ensure that the sequence is in  $\mathcal{L}$ .

<sup>10</sup> Note that if for some  $m \in \mathcal{M}_i$ , we have  $\lambda(M_i \parallel m) = \perp$ , then  $(M_i, m) = e_j$  for  $j > i$ , and will be set later in the process.

Now, for every  $i$ , note that due to condition (1) there are initially  $2^{n-r}$  possible values for  $z_i$ , i.e., all strings with the first  $r$  bits equal to  $Y_i$ . However, we need to remove all strings satisfying any of (2)–(5) above. These can be counted as follows:

- (2) There are at most  $i \leq q$  such values.
- (3) In order for  $M$  to be such that  $\lambda(M) = z_i$ , we need to have  $\lambda(M)[1 \dots r] = Y_i$ , but we know that there are at most  $N_i^{(1)}$  such vertices by definition.
- (4) Note that for every  $j \in [i - 1]$ , there are exactly  $D_j$  possible values  $m' \in \mathcal{M}_j$  which can be combined with a value  $m \in \mathcal{M}_i$  (there are  $D_i$  of those) to get a possible “forbidden” value  $z_i = z_j \oplus m \oplus m'$ , and thus we need to exclude  $D_i \cdot \sum_{j=1}^{i-1} D_j \leq q \cdot D_i$  possible values.
- (5) This is exactly the definition of  $N_i^{(2)}$ .

Therefore, we can now lower bound  $|\mathcal{L}|$  as

$$\begin{aligned}
 |\mathcal{L}| &\geq \prod_{i=1}^q (2^{n-r} - N_i^{(1)} - N_i^{(2)} - q - q \cdot D_i) \\
 &\geq 2^{q \cdot (n-r)} \cdot \left( 1 - \frac{N^{(1)} + N^{(2)}}{2^{n-r}} - \frac{2q^2}{2^{n-r}} \right), \tag{6}
 \end{aligned}$$

where we used the fact that  $\prod_i (1 - x_i) \geq 1 - \sum_i x_i$ , and that  $\sum_{i=1}^q q \cdot D_i \leq q^2$ .

## 5 Security Analysis of Sponge-Based PRFs

In this final section, we turn to discussing security of sponge-based PRFs. We first discuss the constructions considered in this section.

**SPONGE-BASED MAC.** As in the TCBC case above, we fix parameters  $n, r$  and an injective padding scheme  $\text{pad} : \{0, 1\}^* \rightarrow (\{0, 1\}^n)^+$ . Then, the construction  $\text{Sponge} = \text{Sponge}_{r, \text{pad}}[\pi] : \{0, 1\}^\kappa \times \{0, 1\}^* \rightarrow \{0, 1\}^r$ , using a permutation  $\pi \in \text{Perm}(n)$ , on input  $M \in \{0, 1\}^*$  and key  $K \in \{0, 1\}^\kappa$ , first computes  $K[1] \dots K[w]M[1] \dots M[\ell] \leftarrow \text{pad}(K \parallel M)$ . Then, it outputs  $S_\ell[1 \dots r]$  (the first  $r$  bits of  $S_\ell$ ), where

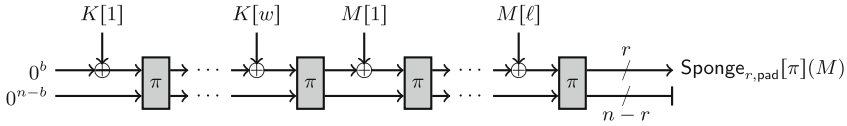
$$\begin{aligned}
 V_0 &\leftarrow 0^n, & V_i &\leftarrow \pi(K[i] \oplus V_{i-1}) \text{ for } i = 1, \dots, w, \\
 S_0 &\leftarrow V_w, & S_i &\leftarrow \pi(M[i] \oplus S_{i-1}) \text{ for } i = 1, \dots, \ell.
 \end{aligned}$$

We are explicitly assuming (for simplicity) that the (padded) keys and the actual message end up in different blocks, and hence our naming conventions.<sup>11</sup>

Different from the actual hash-function instantiations, the presented **Sponge** construction is *more general* in that it allows for processing  $n$ -bit input blocks in the absorption phase. We can retrieve the originals sponge construction and

<sup>11</sup> Our results can be extended to the more general case, but we avoid the notational overhead in this version of the paper.

SHA-3 instantiations as special case — shorter blocks can be enforced by the padding function  $\text{pad}$ , which we only require to be injective, but an added benefit of our analysis is that it shows that such shorter blocks are not necessary. The construction  $\text{Sponge}_{r,\text{pad}}[\pi]$  using a customary padding  $\text{pad}_b$  that enforces  $b$ -bit blocks is depicted in Fig. 3.



**Fig. 3. Sponge construction.** Representation of  $\text{Sponge}_{r,\text{pad}_b}[\pi]$  used with a padding scheme  $\text{pad}_b$  that enforces  $b$ -bit blocks.

We also consider a variant of the construction – called  $\text{GSponge}$  – that takes an  $n$ -bit key  $K$  and differs from  $\text{Sponge}$  in that it directly sets  $S_0 \leftarrow K$  instead of absorbing the key to obtain  $V_w$ . The construction is similar to some other MAC designs such as  $\text{donkeySponge}$  [9] and  $\text{Pelican}$  [16].

SECURITY ANALYSIS OF  $\text{GSponge}$ . We prove the following theorem:

**Theorem 2 (Security of  $\text{GSponge}$ ).** *Let  $\mathcal{A}$  be a prf-adversary in the ideal-permutation model, making at most  $q_\pi$  queries to  $\pi$  and at most  $q_C$  queries of length at most  $\ell < 2^{n/4}$  blocks to the construction (either  $\text{GSponge}_{r,\text{pad}}[\pi]$  for a random  $n$ -bit key  $K$  or a random function). Then, for all  $t \geq 1$ ,*

$$\text{Adv}_{\text{GSponge}_{r,\text{pad},\pi}}^{\text{prf}}(\mathcal{A}) \leq \frac{(6t + 17)\ell q_C^2 + 7\ell q_\pi q_C + 2q_C}{2^n} + \frac{6nq_C^2 + 8\ell q_C + q_\pi q_C}{2^{n-r}} + \frac{136\ell^4 q_C^2}{2^{2n}} + \frac{2(\ell q_C)^{t+1}}{2^{nt}}. \tag{7}$$

This bound substantially improves the previously known bound from [8], which was of the order  $O(\frac{\ell^2 q_C^2 + \ell q_C q_\pi}{2^{n-r}})$ . (We discuss the subtleties of the bound in [2] in detail in the full version [22]). For sufficiently large  $t$  and for  $\ell < 2^{n/4}$ , the first two terms are the leading terms. If we additionally assume that  $\ell < 2^r$ , then the bound becomes of order  $O(\frac{q_C^2 + q_C q_\pi + \ell q_C}{2^{n-r}})$ . In the full version [22], we prove that this is tight when additionally  $\max\{q_\pi, q_C\} \geq \ell$ .

The proof of Theorem 2 adapts the proof strategy of Theorem 1 to the setting of sponges. The  $\text{GSponge}$  and  $\text{TCBC}$  constructions are in fact the same, with the main difference that the initial value in  $\text{GSponge}$  is set to a random secret key and the underlying permutation  $\pi$  can be evaluated by the adversary. Intuitively, however, one can show that thanks to the random secret key, no internal permutation query in a construction query intersects with a direct permutation query by the attacker, except with probability  $O(\ell \cdot q_C q_\pi / 2^n)$ . Conditioned on the event that such intersection does not occur, the distinguishing bound (in terms of construction queries) is roughly the same as the one of  $\text{TCBC}$ .

REPLACING THE UNIFORM KEY. We now address security of the **Sponge** construction when using the customary padding  $\text{pad}_b$ , where the  $(\kappa = w \cdot b)$ -bit key  $K$  is first split into  $w$   $b$ -bit blocks as  $K[1] \cdots K[w]$ , each of them is padded with  $n - b$  trailing zeroes and absorbed by the construction, as depicted in Fig. 3. The proof of the following theorem is found in the full version [22], and relies on a detailed analysis of the key absorption mechanism which shows that the behaviors of **GSponge** and **Sponge** are indistinguishable given enough key material.

**Theorem 3 (Security of Sponge).** *Let  $\mathcal{A}$  be a prf-adversary in the ideal-permutation model, making at most  $q_\pi$  queries to  $\pi$  and at most  $q_C$  queries of length at most  $\ell < 2^{n/4}$  blocks to the construction (either  $\text{Sponge}_{r,\text{pad}_b}[\pi]$  with the padding  $\text{pad}_b$  and a random  $(w \cdot b)$ -bit key, or a random function). Then, for all  $t \geq 1$ , and  $q = q_\pi + \ell q_C < 2^{n-b}$ , we have*

$$\text{Adv}_{\text{Sponge}_{r,\text{pad}_b},\pi}^{\text{prf}}(\mathcal{A}) \leq A_t(q_C, q_\pi, \ell) + \frac{wq}{2^n} + \min \left\{ \frac{q}{2^{\frac{b - \log_2(3n) - 1}{2} w}}, \frac{q}{2^{bw}} + \frac{q^2}{2^{n-b}} \right\},$$

where  $A_t$  denotes the expression on the right-hand side of inequality (7). Moreover, if  $w = 1$  then one can replace the whole min-term by  $q/2^{bw}$ .

We remark that our proof is highly involved for the case where  $q^2 > 2^{n-b}$ , where  $q = q_\pi + q_C \cdot \ell$  is the overall number of queries to  $\pi$  in the experiment, and requires an adaptation of combinatorial techniques proposed in [17].

THE STANDARD-MODEL BOUNDS. We combine an approach by Chang *et al.* [13] (also used in [2]) with our improved bound for TCBC. In particular, we measure security of the underlying permutation  $\pi$  in terms of the advantage  $\text{Adv}_\pi^{(r,\oplus)\text{-prp}}(\mathcal{B})$  of an adversary in distinguishing the map  $M \mapsto (0^r \parallel K) \oplus \pi(M \oplus (0^r \parallel K))$  under a random secret key  $K \xleftarrow{\$} \{0, 1\}^{n-r}$  from  $\tau \xleftarrow{\$} \text{Perm}(n)$ . In the full version [22], we prove and discuss the following theorem.

**Theorem 4 (Standard-Model Security of GSponge).** *Let  $\pi \in \text{Perm}(n)$  and  $\text{pad} : \{0, 1\}^* \rightarrow (\{0, 1\}^n)^+$  a padding scheme. Let  $\mathcal{A}$  be a prf-adversary making at most  $q$  queries, each of length at most  $\ell < 2^{n/4}$   $n$ -bit blocks (after padding). Then, there exists an  $(\oplus, r)$ -prp-adversary  $\mathcal{B}$  such that for any  $t \geq 1$ ,*

$$\text{Adv}_{\text{GSponge}_{r,\text{pad}[E]}}^{\text{prf}}(\mathcal{A}) \leq \text{Adv}_\pi^{(r,\oplus)\text{-prp}}(\mathcal{B}) + B(q, \ell, n, r, t),$$

where  $\mathcal{B}$  has  $\text{Time}(\mathcal{B}) = \text{Time}(\mathcal{A}) + O(q \cdot \ell)$  and makes at most  $q \cdot \ell$  permutation queries, and  $B(q, \ell, n, r, t)$  is the term on the right-hand-side of Theorem 1.

**Acknowledgments.** We thank Mihir Bellare for insightful feedback, and Bart Mennink for technical comments. Gaži and Pietrzak’s work was partly funded by the European Research Council under an ERC Starting Grant (259668-PSPC). Tessaro’s research was partially supported by NSF grant CNS-1423566, by a gift from the Gareatis Foundation, and by the Culler Chair. Part of this work was done while the third author was visiting IST Austria.

## References

1. SHA-3 standard. National Institute of Standards and Technology (NIST), Draft FIPS Publication 202, U.S. Department of Commerce, April 2014
2. Andreeva, E., Daemen, J., Mennink, B., Van Assche, G.: Security of keyed sponge constructions using a modular proof approach. In: FSE 2015. LNCS (2015, to appear)
3. Bellare, M.: New proofs for NMAC and HMAC: security without collision-resistance. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 602–619. Springer, Heidelberg (2006)
4. 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)
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., Pietrzak, K., Rogaway, P.: Improved security analyses for CBC MACs. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 527–545. Springer, Heidelberg (2005)
7. Bernstein, D.J.: A short proof of the unpredictability of cipher block chaining (2005). <http://cr.yp.to/antiforgery/easycbc-20050109.pdf>
8. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: On the security of the keyed sponge construction. In: Symmetric Key Encryption Workshop (SKEW), February 2011
9. Bertoni, G., Daemen, J., Peeters, M.: Permutation-based encryption, authentication and authenticated encryption. In: Directions in Authenticated Ciphers (2012)
10. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Keccak. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 313–314. Springer, Heidelberg (2013)
11. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: On the indistinguishability of the sponge construction. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 181–197. Springer, Heidelberg (2008)
12. Black, J., Rogaway, P.: CBC MACs for arbitrary-length messages: the three-key constructions. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 197–215. Springer, Heidelberg (2000)
13. Chang, D., Dworkin, M., Hong, S., Kelsey, J., Nandi, M.: A keyed sponge construction with pseudorandomness in the standard model. In: Proceedings of the Third SHA-3 Candidate Conference (2012)
14. Chen, S., Steinberger, J.: Tight security bounds for key-alternating ciphers. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 327–350. Springer, Heidelberg (2014)
15. Computer data authentication. National Bureau of Standards, NBS FIPS PUB 113, U.S. Department of Commerce, May 1985
16. Daemen, J., Rijmen, V.: The mac function pelican 2.0. Cryptology ePrint Archive, Report 2005/088 (2005). <http://eprint.iacr.org/>
17. Dai, Y., Lee, J., Mennink, B., Steinberger, J.: The security of multiple encryption in the ideal cipher model. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 20–38. Springer, Heidelberg (2014)
18. Dodis, Y., Ristenpart, T., Steinberger, J.P., Tessaro, S.: To hash or not to hash again? (In)differentiability results for  $h^2$  and HMAC. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 348–366. Springer, Heidelberg (2012)

19. Dunkelman, O., Keller, N., Shamir, A.: Minimalism in cryptography: the even-mansour scheme revisited. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 336–354. Springer, Heidelberg (2012)
20. Even, S., Mansour, Y.: A construction of a cipher from a single pseudorandom permutation. *J. Cryptology* **10**(3), 151–162 (1997)
21. Gaži, P., Pietrzak, K., Rybár, M.: The exact PRF-security of NMAC and HMAC. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 113–130. Springer, Heidelberg (2014)
22. Gaži, P., Pietrzak, K., Tessaro, S.: The exact PRF security of truncation: Tight bounds for keyed sponges and truncated CBC. *Cryptology ePrint Archive*, Report 2015/053 (2015). Full version of this paper
23. 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)
24. Iwata, T., Kurosawa, K.: OMAC: one-key CBC MAC. In: Johansson, T. (ed.) FSE 2003. LNCS, vol. 2887, pp. 129–153. Springer, Heidelberg (2003)
25. Iwata, T., Kurosawa, K.: Stronger security bounds for OMAC, TMAC, and XCBC. In: Johansson, T., Maitra, S. (eds.) INDOCRYPT 2003. LNCS, vol. 2904, pp. 402–415. Springer, Heidelberg (2003)
26. Kobitz, N., Menezes, A.: Another look at HMAC. *Cryptology ePrint Archive*, Report 2012/074 (2012). <http://eprint.iacr.org/2012/074>
27. Kurosawa, K., Iwata, T.: TMAC: two-key CBC MAC. In: Joye, M. (ed.) CT-RSA 2003. LNCS, vol. 2612, pp. 33–49. Springer, Heidelberg (2003)
28. Information technology security techniques message authentication codes (macs) part 1: Mechanisms using a block cipher. ISO/IEC 9797–1 (1999)
29. Maurer, U.M.: Indistinguishability of random systems. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 110–132. Springer, Heidelberg (2002)
30. Minematsu, K., Matsushima, T.: New bounds for PMAC, TMAC, and XCBC. In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 434–451. Springer, Heidelberg (2007)
31. Nandi, M.: A simple and unified method of proving indistinguishability. In: Barua, R., Lange, T. (eds.) INDOCRYPT 2006. LNCS, vol. 4329, pp. 317–334. Springer, Heidelberg (2006)
32. Patarin, J.: The “Coefficients H” technique (invited talk). In: Avanzi, R.M., Keliher, L., Sica, F. (eds.) SAC 2008. LNCS, vol. 5381, pp. 328–345. Springer, Heidelberg (2009)
33. Petrank, E., Rackoff, C.: CBC MAC for real-time data sources. *J. Cryptology* **13**(3), 315–338 (2000)
34. Pietrzak, K.: A tight bound for EMAC. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4052, pp. 168–179. Springer, Heidelberg (2006)
35. Vandewalle, J., Chaum, D., Fumy, W., Jansen, C.J.A., Landrock, P., Roelofsen, G.: A european call for cryptographic algorithms: RIPE; race integrity primitives evaluation. In: Quisquater, J.-J., Vandewalle, J. (eds.) EUROCRYPT 1989. LNCS, vol. 434, pp. 267–271. Springer, Heidelberg (1990)
36. Vaudenay, S.: Decorrelation over infinite domains: the encrypted CBC-MAC case. In: Stinson, D.R., Tavares, S. (eds.) SAC 2000. LNCS, vol. 2012, pp. 189–201. Springer, Heidelberg (2001)

# An Algebraic Framework for Pseudorandom Functions and Applications to Related-Key Security

Michel Abdalla<sup>(✉)</sup>, Fabrice Benhamouda<sup>(✉)</sup>, and Alain Passelègue<sup>(✉)</sup>

ENS, CNRS, INRIA, and PSL, École normale supérieure, 45 Rue d'Ulm,  
75230 Paris Cedex 05, France

{michel.abdalla,fabrice.ben.hamouda,alain.passelegue}@ens.fr

**Abstract.** In this work, we provide a new algebraic framework for pseudorandom functions which encompasses many of the existing algebraic constructions, including the ones by Naor and Reingold (FOCS'97), by Lewko and Waters (CCS'09), and by Boneh, Montgomery, and Raghunathan (CCS'10), as well as the related-key-secure pseudorandom functions by Bellare and Cash (Crypto'10) and by Abdalla *et al.* (Crypto'14). To achieve this goal, we introduce two versions of our framework. The first, termed linearly independent polynomial security, states that the values  $(g^{P_1(\vec{a})}, \dots, g^{P_q(\vec{a})})$  are indistinguishable from a random tuple of the same size, when  $P_1, \dots, P_q$  are linearly independent multivariate polynomials of the secret key vector  $\vec{a}$ . The second, which is a natural generalization of the first framework, additionally deals with constructions based on the decision linear and matrix Diffie-Hellman assumptions. In addition to unifying and simplifying proofs for existing schemes, our framework also yields new results, such as related-key security with respect to arbitrary permutations of polynomials. Our constructions are in the standard model and do not require the existence of multilinear maps.

## 1 Introduction

Pseudorandom functions (PRFs), originally defined by Goldreich, Goldwasser, and Micali [19], are one of the most fundamental primitives in cryptography. Informally speaking, a function is said to be pseudorandom if its outputs are indistinguishable from that of a random function with respect to a computationally bounded adversary which only has black-box access to it. Hence, even if the adversary can control the inputs on which the function is computed and see the corresponding outputs, he or she should still not be able to distinguish this function from a perfectly random one.

Due to their simplicity and security properties, pseudorandom functions have been used in numerous applications, including symmetric encryption, authentication, and key exchange. In particular, since pseudorandom functions can be used to model real-world block-ciphers, such as AES [3], they are also extremely useful for the security analysis of protocols that rely on these primitives.

**Number-Theoretic Constructions.** Despite its elegance, the original construction of pseudorandom functions by Goldreich, Goldwasser, and Micali based on pseudorandom generators was not very efficient. In order to improve its efficiency while still being able to prove its security under reasonable complexity assumptions, Naor and Reingold [27] proposed a new construction based on the Decisional Diffie-Hellman assumption (DDH) [27]. Let  $\vec{a} = (a_0, \dots, a_n) \in \mathbb{Z}_p^{n+1}$  be the key and  $x = x_1 \parallel \dots \parallel x_n \in \{0, 1\}^n$  be the input of the PRF. Let  $g$  be a fixed public generator of a group  $\mathbb{G}$  of prime order  $p$ . The Naor-Reingold PRF is then defined as

$$\text{NR}(\vec{a}, x) = \left[ a_0 \prod_{i=1}^n a_i^{x_i} \right]$$

where for any  $a \in \mathbb{Z}_p$ ,  $[a]$  stands for  $g^a$ , as defined in [18].

As mentioned in [17], the algebraic nature of the Naor-Reingold PRF has led to many applications, such as verifiable random functions [2, 22], distributed PRFs [27], and related-key-secure PRFs [8], which are hard to obtain from generic PRFs. Hence, due to its importance, several other extensions of the Naor-Reingold PRF have been proposed [17, 26] based on different assumptions, such as the Decision Linear assumption (DLin) [15] and the  $d$ -DDHI assumption [17, 20].

In this work, our main contribution is to further extend the above line of work by providing a *generic algebraic framework* for building pseudorandom functions. In particular, all of the algebraic constructions mentioned above can be seen as particular instantiations of our framework. In addition, our framework is general enough that it captures and extends other constructions such as the related-key-secure PRF constructions by Bellare and Cash [8] (BC) and by Abdalla *et al.* [1] (ABPP).

**Linearly Independent Polynomial Security.** To obtain our results, our first contribution is to introduce a new notion of linearly independent polynomial (LIP) security. Informally, it states that the values  $([P_1(\vec{a})], \dots, [P_q(\vec{a})])$  are indistinguishable from a random tuple of the same size, when  $P_1, \dots, P_q$  are linearly independent multivariate polynomials of degree at most  $d$  in any indeterminate and  $\vec{a}$  is the PRF secret key vector. The new notion is based on a new MDDH assumption [18] over the underlying group  $\mathbb{G}$ , denoted  $\mathcal{E}_{1,d}$ -MDDH, which can be (tightly) reduced to either DDH or DDHI depending on value of  $d$ .

In order to illustrate the usefulness of the new notion, we show in Sect. 4 how to use it to provide alternative security proofs for the Naor-Reingold PRF [27] and the PRF by Boneh, Montgomery, and Raghunathan (BMR) in [17] as well as generalizations of both these PRFs, that we call *weighted NR* and *weighted BMR*. Intuitively, all these PRFs are defined over a prime order group  $\mathbb{G} = \langle g \rangle$  as a function  $F$  that takes a key  $\vec{a}$  and an input  $x$  and outputs an element in  $\mathbb{G}$  of the shape  $F(\vec{a}, x) = [P_x(\vec{a})]$  where the polynomial  $P_x$  depends on  $x$ . Hence, to prove the security of such constructions, we just need to prove that all polynomials  $P_x$ , for any entries  $x$ , are linearly independent.



We would like to remark that the actual formulation of the LIP security in Sect. 3 includes a value  $a' \in \mathbb{Z}_p$  multiplying each  $P_i(\vec{a})$  term, which allows for the use of different generators in the PRF constructions. While we could dispense with  $a'$  in the case where  $a'$  and the  $a_i$  values in  $\vec{a}$  are scalars, we opted to use it to be consistent with the case in which these values are matrices, as in Sect. 6.

**Applications to Related-Key Security.** Related-key attacks (RKAs) were first introduced by Biham and Knudsen [11, 24] and consider the setting in which an adversary could force a given cryptographic primitive to execute under a different but related key. Over the years, such attacks became more predominant and several related-key attacks have been proposed against existing block-ciphers (e.g., [12, 13, 23]). Since these attacks are quite powerful and hard to defend against, Bellare and Kohno [9] introduced a formal treatment of these attacks in the context of PRFs and pseudorandom permutations (PRPs) to better understand if and how one could achieve security in the presence of related-key attacks. One of their main observations is that certain classes of related-key attacks are impossible to protect against and, hence, their goal was to identify the set of classes  $\Phi$  for which one could design secure RKA-PRFs and RKA-PRPs.

Let  $F: \mathcal{K} \times \mathcal{D} \rightarrow \mathcal{R}$  be a family of functions for a security parameter  $\kappa$ , and let  $\Phi = \{\phi: \mathcal{K} \rightarrow \mathcal{K}\}$  be a set of related-key deriving (RKD) functions on the key space  $\mathcal{K}$ . Let  $G: \mathcal{K} \times \mathcal{D} \rightarrow \mathcal{R}$  be a random function and let  $K \in \mathcal{K}$  be a random target key. Informally, in the RKA security model of [9],  $F$  is said to be a  $\Phi$ -RKA-PRF if no polynomial-time adversary can distinguish the output of  $F(\phi(K), x)$  from the output of  $G(\phi(K), x)$ , for pairs  $(\phi, x)$  of its choice, with non-negligible probability.

Our second contribution is to show that the new LIP security notion can be used to prove directly the related-key security of certain constructions. In particular, we show that a particular case of our weighted BMR PRF construction is secure against permutations of the secret key. In these attacks, the attacker can obtain the output of the PRF with respect to any key that is a permutation of the original one.

To understand why RKA security can follow from the LIP security notion, let  $F$  be a PRF defined over a prime-order group  $\mathbb{G} = \langle g \rangle$  that takes a key  $\vec{a}$  and an input  $x$  and outputs  $F(\vec{a}, x) = [P_x(\vec{a})]$ . Let  $\Phi$  be a class of RKD functions, where functions  $\vec{\phi} = (\phi_1, \dots, \phi_n) \in \Phi$  are such that  $\phi_i$  are multivariate polynomials in  $\mathbb{Z}_p[T_1, \dots, T_n]$ . Then, for a RKD function  $\vec{\phi}$  and an input  $x$ , the PRF outputs  $F(\vec{\phi}(\vec{a}), x) = [P_{\vec{\phi}, x}(\vec{a})]$ , where the polynomial  $P_{\vec{\phi}, x}(\vec{T}) = P_x(\vec{\phi}(\vec{T})) = P_x(\phi_1(\vec{T}), \dots, \phi_n(\vec{T}))$  depends on  $\vec{\phi}$  and  $x$ , with  $\vec{T} = (T_1, \dots, T_n)$ . Hence, when all polynomials  $P_{\vec{\phi}, x}$  are linearly independent, the LIP security notion directly shows that  $F$  is  $\Phi$ -RKA-secure.

**Related-Key Security with Respect to Unique-Input Adversaries.**

Unfortunately, the case in which the polynomials  $P_{\vec{\phi}, x}$  are all linearly independent is not so easy to instantiate as we would like, and we have only been able to directly obtain RKA security for very restricted classes. Hence, to overcome these restrictions, our third contribution is to further extend our results in Sect. 5.2

to deal with the case where polynomials are only linearly independent when all the inputs  $x$  are distinct. This scenario is similar to the one considered in [1]. In particular, our new algebraic framework extends the one from [1] and provides constructions for new and larger classes of RKD functions. More precisely, we build in Sect. 5.2 RKA-PRFs against classes of permutations of univariate polynomials. Furthermore, in the full version, we also consider classes of univariate polynomials and multivariate affine RKD functions.

For simplicity, the results in Sect. 5.2 only hold with respect to PRFs of the form  $[P_x(\vec{a})]$  where  $P_x$  is a polynomial that depends on  $x$ . However, a more general framework which does not make this assumption is described in the full version.

**An Algebraic Framework for Non-commutative Structures.** Finally, our last contribution is to extend the LIP security notion to work under weaker assumptions than DDH, such as DLin. As we point out in Sect. 6, the main difficulty in this case is that the key values  $a_i$ 's may be matrices, which do not necessarily commute. To address this issue, we introduce natural conditions on the order of indeterminates which makes non-commutative and commutative polynomials behave in a similar manner. Through the new generalization, we not only deal with cases already covered by the LIP security notion, but we also capture PRFs based on the DLin and MDDH assumptions [18].

**Further Discussions.** In addition to the foundational work of Goldreich, Goldwasser, and Micali [19], several other frameworks for constructing PRFs have appeared in the literature, including [7, 17, 28] to name a few.

In [28], Naor and Reingold proposed the notion of pseudorandom synthesizers and provided several instantiations for it based on different complexity assumptions. Informally speaking, a pseudorandom synthesizer is a two-variable function,  $S(\cdot, \cdot)$ , so that, for polynomially many random and independent input assignments  $(x_1, \dots, x_m)$  and  $(y_1, \dots, y_m)$ , the set of values  $\{S(x_i, y_j)\}$  are computationally indistinguishable from uniform for  $i$  and  $j$  in  $\{1, \dots, m\}$ .

In [7], Bellare, Canetti, and Krawczyk provide a framework for building variable-length input PRFs from fixed-length input ones, known as the cascade construction. In their framework, one obtains a larger-domain PRF  $F'$  simply by partitioning the input  $x$  into a number  $n$  of small blocks  $x_1, \dots, x_n$  matching the domain of the underlying PRF  $F$  and using the output of  $F$  on key  $k_i$  and input  $x_i$  as the secret key  $k_{i+1}$  for the next stage. Since their framework requires the output of the underlying PRF to be at least as long as the secret key, it cannot be applied to PRFs with very small domains.

To circumvent the restrictions of the cascade construction, Boneh, Montgomery, and Raghunathan proposed an extension in [17], known as the augmented cascade construction, in which supplemental secret information is provided in every iteration. Unlike the cascade construction, its security does not follow from the standard security of the underlying PRF, requiring it to meet a new notion called parallel security.

While these frameworks are more general than ours and capable of handling different complexity assumptions (e.g., [6]), they are more combinatorial

in nature and do not fully exploit the algebraic nature of the underlying PRFs. In particular, it is not clear how to extend them to the RKA setting, which is one of the main applications of our new algebraic framework. Moreover, even in the standard PRF setting, our framework seems to possess complementary features compared to the existing ones. Notably, it only requires the verification of an algebraic condition (such as testing the linear independence of the polynomials) for each instantiation, which is generally easier to prove.

**Other Related Work.** It is worth mentioning that in the context of related-key security, Lewi, Montgomery and Raghunathan [25] designed RKA-PRFs for similar classes of polynomial RKD functions. However, unlike their constructions, ours do not require multilinear maps. Also, our constructions are proven fully RKA-secure while theirs are only proven unique-input RKA-secure.

## 2 Definitions

**Notations and Conventions.** We denote by  $\kappa$  the security parameter. Let  $F: \mathcal{K} \times \mathcal{D} \rightarrow \mathcal{R}$  be a function that takes a key  $K \in \mathcal{K}$  and an input  $x \in \mathcal{D}$  and returns an output  $F(K, x) \in \mathcal{R}$ . The set of all functions  $F: \mathcal{K} \times \mathcal{D} \rightarrow \mathcal{R}$  is then denoted by  $\text{Fun}(\mathcal{K}, \mathcal{D}, \mathcal{R})$ . Likewise,  $\text{Fun}(\mathcal{D}, \mathcal{R})$  denotes the set of all functions mapping  $\mathcal{D}$  to  $\mathcal{R}$ . If  $S$  is a set, then  $|S|$  denotes its size. We denote by  $s \xleftarrow{\$} S$  the operation of picking at random  $s$  in  $S$ . If  $\vec{x}$  is a vector then we denote by  $|\vec{x}|$  its length, so  $\vec{x} = (x_1, \dots, x_{|\vec{x}|})$ . For a binary string  $x$ , we denote its length by  $|x|$  so  $x \in \{0, 1\}^{|x|}$ ,  $x_i$  its  $i$ -th bit, so  $x = x_1 \parallel \dots \parallel x_n$ . We extend these notations to any  $d$ -ary string  $x$ , for  $d \geq 2$ . For a matrix  $\mathbf{A}$  of size  $k \times m$ , we denote by  $a_{i,j}$  the coefficient of  $\mathbf{A}$  in the  $i$ -th row and the  $j$ -th column. For a vector  $\vec{\phi} = (\phi_1, \dots, \phi_n)$  of  $n$  functions from  $S_1$  to  $S_2$  with  $|\vec{\phi}| = n$  and  $\vec{a} \in S_1$ , we denote by  $\vec{\phi}(\vec{a})$  the vector  $(\phi_1(\vec{a}), \dots, \phi_n(\vec{a})) \in S_2^n$ . We denote by  $\mathbb{Z}_p[T_1, \dots, T_n]$  the ring of multivariate polynomials in indeterminates  $T_1, \dots, T_n$ . For a polynomial  $P \in \mathbb{Z}_p[T_1, \dots, T_n]$ , we denote  $P(T_1, \dots, T_n)$  by  $P(\vec{T})$  and by  $P(\vec{a})$  the evaluation of  $P$  by setting  $\vec{T}$  to  $\vec{a}$ , meaning that we set  $T_1 = a_1, \dots, T_n = a_n$ . For  $F: \mathcal{K} \times \mathcal{D} \rightarrow \mathcal{R}$  and for a vector  $\vec{x}$  over  $\mathcal{D}$ , we denote by  $F(K, \vec{x})$  the vector  $(F(K, x_1), \dots, F(K, x_{|\vec{x}|}))$ . We denote by  $\mathfrak{S}_n$  the set of all permutations of  $\{1, \dots, n\}$ .

Finally, we often implicitly consider a multiplicative group  $\mathbb{G} = \langle g \rangle$  with public generator  $g$  of order  $p$  and we denote by  $[\![g]\!]a$ , or simply  $[a]$  if there is no ambiguity about the generator, the element  $g^a$ , for any  $a \in \mathbb{Z}_p$ . Similarly, if  $\mathbf{A}$  is a matrix in  $\mathbb{Z}_p^{k \times m}$ ,  $[\mathbf{A}]$  is a matrix  $\mathbf{U} \in \mathbb{G}^{k \times m}$ , such that  $u_{i,j} = [a_{i,j}]$  for  $i = 1, \dots, k$  and  $j = 1, \dots, m$ .

**Games [10].** Most of our definitions and proofs use the code-based game-playing framework, in which a game has an **Initialize** procedure, procedures to respond to adversary oracle queries, and a **Finalize** procedure. To execute a game  $G$  with an adversary  $\mathcal{A}$ , we proceed as follows. First, **Initialize** is executed and its outputs become the input of  $\mathcal{A}$ . When  $\mathcal{A}$  executes, its oracle queries are

answered by the corresponding procedures of  $G$ . When  $\mathcal{A}$  terminates, its outputs become the input of **Finalize**. The output of the latter, denoted  $G^{\mathcal{A}}$  is called the output of the game, and we let “ $G^{\mathcal{A}} \Rightarrow 1$ ” denote the event that this game output takes the value 1. The running time of an adversary by convention is the worst case time for the execution of the adversary with any of the games defining its security, so that the time of the called game procedures is included.

**PRFs** [8, 19]. The advantage of an adversary  $\mathcal{A}$  in attacking the standard PRF security of a function  $F: \mathcal{K} \times \mathcal{D} \rightarrow \mathcal{R}$  is defined via

$$\text{Adv}_F^{\text{prf}}(\mathcal{A}) = \Pr \left[ \text{PRFReal}_F^{\mathcal{A}} \Rightarrow 1 \right] - \Pr \left[ \text{PRFRand}_F^{\mathcal{A}} \Rightarrow 1 \right].$$

Game  $\text{PRFReal}_F$  first picks  $K \xleftarrow{\$} \mathcal{K}$  and responds to oracle query  $\mathbf{Fn}(x)$  via  $F(K, x)$ . Game  $\text{PRFRand}_F$  first picks  $f \xleftarrow{\$} \text{Fun}(\mathcal{D}, \mathcal{R})$  and responds to oracle query  $\mathbf{Fn}(x)$  via  $f(x)$ .

**RKA-PRFs** [8, 9]. Let  $F: \mathcal{K} \times \mathcal{D} \rightarrow \mathcal{R}$  be a function and  $\Phi \subseteq \text{Fun}(\mathcal{K}, \mathcal{K})$ . The members of  $\Phi$  are called RKD (Related-Key Deriving) functions. An adversary is said to be  $\Phi$ -restricted if its oracle queries  $(\phi, x)$  satisfy  $\phi \in \Phi$ . The advantage of a  $\Phi$ -restricted adversary  $\mathcal{A}$  in attacking the RKA-PRF security of  $F$  is defined via

$$\text{Adv}_{\Phi, F}^{\text{prf-rka}}(\mathcal{A}) = \Pr \left[ \text{RKPRFReal}_F^{\mathcal{A}} \Rightarrow 1 \right] - \Pr \left[ \text{RKPRFRand}_F^{\mathcal{A}} \Rightarrow 1 \right].$$

Game  $\text{RKPRFReal}_F$  first picks  $K \xleftarrow{\$} \mathcal{K}$  and responds to oracle query  $\mathbf{RKFn}(\phi, x)$  via  $F(\phi(K), x)$ . Game  $\text{RKPRFRand}_F$  first picks  $K \xleftarrow{\$} \mathcal{K}$  and  $G \xleftarrow{\$} \text{Fun}(\mathcal{K}, \mathcal{D}, \mathcal{R})$  and responds to oracle query  $\mathbf{RKFn}(\phi, x)$  via  $G(\phi(K), x)$ . We say that  $F$  is a  $\Phi$ -RKA-secure PRF if for any  $\Phi$ -restricted adversary, its advantage in attacking the RKA-PRF security is negligible.

**Group Generators.** All our PRFs and RKA-PRFs use a cyclic group of prime order  $p$ . The generator(s) used in their construction is supposed to be public. In particular, RKD functions *cannot* modify the generator(s). Our security proofs will then start by giving the generators to the adversary.

**Hardness Assumptions.** To get a simpler and unified framework, we introduce a particular MDDH assumption [18]: the  $\mathcal{E}_{k,d}$ -MDDH assumption, defined by the matrix distribution  $\mathcal{E}_{k,d}$  which samples matrices  $\Gamma$  as follows

$$\Gamma = \begin{pmatrix} \mathbf{A}_1^0 \cdot \mathbf{A}_0 \\ \mathbf{A}_1^1 \cdot \mathbf{A}_0 \\ \vdots \\ \mathbf{A}_1^d \cdot \mathbf{A}_0 \end{pmatrix} \in \mathbb{Z}_p^{k(d+1) \times k} \quad \text{with } \mathbf{A}_0, \mathbf{A}_1 \xleftarrow{\$} \mathbb{Z}_p^{k \times k}. \tag{1}$$

The advantage of an adversary  $\mathcal{D}$  against the  $\mathcal{E}_{k,d}$ -MDDH assumption is

$$\text{Adv}_{\mathcal{G}}^{\mathcal{E}_{k,d}\text{-mddh}}(\mathcal{D}) = \Pr [\mathcal{D}(g, [\Gamma], [\Gamma \cdot \mathbf{W}])] - \Pr [\mathcal{D}(g, [\Gamma], [\mathbf{U}])],$$

**Table 1.** Security of  $\mathcal{E}_{k,d}$ -MDDH

	$k = 1$	$k = 2$	$k \geq 3$
$d = 1$	$= \mathbf{Adv}_{\mathbb{G}}^{\text{ddh}}$	$\lesssim 2 \cdot \mathbf{Adv}_{\mathbb{G}}^{\mathcal{U}_2\text{-mddh}}$	$\lesssim k \cdot \mathbf{Adv}_{\mathbb{G}}^{\mathcal{U}_k\text{-mddh}}$
$d \geq 2$	$\lesssim d \cdot \mathbf{Adv}_{\mathbb{G}}^{d\text{-ddhi}}$	generic bilinear group <sup>a</sup> ? <sup>b</sup>	

$\mathbf{Adv}_{\mathbb{G}}^{\text{ddh}}, \mathbf{Adv}_{\mathbb{G}}^{d\text{-ddhi}}$  and  $\mathbf{Adv}_{\mathbb{G}}^{\mathcal{U}_k\text{-mddh}}$  are advantages for DDH, DDHI, and  $\mathcal{U}_k$ -MDDH. This latter assumption is weaker than  $k$ -Lin;  
<sup>a</sup> Proven in the generic (symmetric) bilinear group model [14] in the full version;  
<sup>b</sup> (Trivially) secure in the generic cyclic group model [30], but nothing known about security in generic (symmetric)  $k$ -linear group model [21, 29]

where  $\mathbf{r} \xleftarrow{\$} \mathcal{E}_{k,d}, \mathbf{W} \xleftarrow{\$} \mathbb{Z}_p^{k \times 1}, \mathbf{U} \xleftarrow{\$} \mathbb{Z}_p^{k(d+1) \times 1}$ . As any MDDH assumption and as recalled in the full version, this assumption is random self-reducible, which enables us to make relatively tight proofs.

In Table 1, we summarize security results for  $\mathcal{E}_{k,d}$ -MDDH. For  $k = 1$  or  $d = 1$ , the  $\mathcal{E}_{k,d}$ -MDDH assumption is implied by standard assumptions (DDH, DDHI, or  $k$ -Lin, recalled in the full version).  $\mathcal{E}_{1,1}$ -MDDH is actually exactly DDH.

For our RKA framework, we also make use of the  $d$ -Strong Discrete Logarithm (SDL) problem given in [20] and recalled in the full version.

### 3 Linearly Independent Polynomial Security

In this section, we define a new security notion, termed linearly independent polynomial (LIP) security, which captures that, given a cyclic group  $\mathbb{G} = \langle g \rangle$  of prime order  $p$ , the hardness of distinguishing a tuple  $(y_1, \dots, y_q) = ([P_1(\vec{a}) \cdot a'], \dots, [P_q(\vec{a}) \cdot a']) \in \mathbb{G}^q$  from a random tuple in  $(y_1, \dots, y_q) \xleftarrow{\$} \mathbb{G}^q$ , where  $\vec{a}$  is a secret random vector in  $\mathbb{Z}_p^n$ ,  $a'$  is a secret random scalar in  $\mathbb{Z}_p$ , and  $P_j$  are linearly independent multivariate polynomials. Our LIP theorem (Theorem 1) shows that distinguishing these two tuples is harder than the  $\mathcal{E}_{1,d}$ -MDDH problem in  $\mathbb{G}$ , where  $d$  is the maximum degree in one indeterminate in polynomials  $P_1, \dots, P_q$ . We point out that, on the one hand, if there were a linear relation between the polynomials, i.e., if there exists  $(\lambda_1, \dots, \lambda_q) \in \mathbb{Z}_p^q \setminus \{(0, \dots, 0)\}$ , such that  $\sum_{j=1}^q \lambda_j P_j = 0$ , then it would be straightforward to break the LIP security by checking whether  $\prod_{j=1}^q y_j^{\lambda_j} = 1$  (real case) or not (random case). So the linear independence of the  $P_j$ 's is required.

On the other hand, if the polynomials  $P_j$  are linearly independent, then distinguishing the two tuples is hard in the generic group model, since in this model, the adversary can only compute linear combinations of the group elements it is given (and check for equality). The LIP security is therefore not surprising. What is surprising, is that it is possible to prove it under classical assumptions such as  $\mathcal{E}_{1,d}$ -MDDH, without an exponential blow-up.

In the following, we first consider a particular case of the LIP theorem in which the polynomials are given in their expanded form. This section not only serves as a warm-up for the sequel, but it also helps better grasp the challenges of the proof of the full theorem and gives a nice overview. Next, we formally state the LIP theorem.

### 3.1 Warm-Up: Expanded Multilinear Polynomials

As a warm-up, let us first suppose the polynomials  $P_j$  are multilinear and given in their expanded form:  $P_j \in \mathbb{Z}_p[T_1, \dots, T_n]$  and

$$P_j(\vec{T}) = \sum_{i \in \{0,1\}^n} \alpha_{j,i} T_1^{i_1} \cdots T_n^{i_n}.$$

There are  $2^n$  monomials  $T_1^{i_1} \cdots T_n^{i_n}$ , even in that restricted case. So we need to suppose that either  $n$  is logarithmic in the security parameter, or, more generally, only a polynomial (in the security parameter) number of  $\alpha_{j,i}$  are non-zero.

Let us now prove the LIP security of these polynomials. In the real case, we have:

$$y_j = [P_j(\vec{a})a'] = \left[ \sum_{i \in \{0,1\}^n} \alpha_{j,i} a_1^{i_1} \cdots a_n^{i_n} a' \right] = \prod_{i \in \{0,1\}^n} \text{NR}((a', \vec{a}), i)^{\alpha_{j,i}}, \quad (2)$$

where  $\text{NR}((a', \vec{a}), i) = [a' \prod_{k=1}^n a_k^{i_k}]$  (for  $i \in \{0, 1\}^n$ ).  $\text{NR}$  is a secure PRF under the DDH assumption, meaning that all the values  $\text{NR}((a', \vec{a}), i)$  for all  $i \in \{0, 1\}^n$  look independent and uniformly random. Let us write  $\vec{U}$  the column vector, with rows indexed by  $i \in \{0, 1\}^n$ , containing all the discrete logarithm of these values, i.e.,  $u_i = a' \prod_{k=1}^n a_k^{i_k}$ . Let us also write  $\mathbf{M}$  the  $q \times 2^n$  matrix, with columns indexed by  $i \in \{0, 1\}^n$ , defined by  $m_{j,i} = \alpha_{j,i}$ . Then we can rewrite (2) as:

$$(y_1 \cdots y_q)^\top = [\mathbf{M} \cdot \vec{U}].$$

Since the polynomials  $P_j$  are linearly independent, the rows of  $\mathbf{M}$  are linearly independent. Therefore, as  $[\vec{U}]$  looks uniformly random in  $\mathbb{G}^{2^n}$ ,  $(y_1, \dots, y_q)$  looks like a uniformly random tuple in  $\mathbb{G}^q$ . This proves the result of the LIP theorem in this multilinear case with expanded polynomial. Extending this result to non multilinear polynomial would just require slightly changing the assumption, as long as polynomials are given in their expanded form.

This result is already very useful. We will see in Sect. 4 that it enables to prove the security of the Naor-Reingold PRF and variants thereof.

**Challenges for its Extension.** Unfortunately, for certain settings such as those considered in the context of related-key security, or even for the Boneh-Montgomery-Raghuathan PRF [17], we cannot have polynomials in an expanded form, but only as a polynomial-size (in the number  $n$  of indeterminates and the maximum degree  $d$  in each indeterminate) formula (given by an abstract tree).<sup>1</sup> The problem is that the expanded version of these polynomials may be exponentially large. For example,  $(T_1 + 1) \cdots (T_n + 1)$  has  $2^n$  monomials.

Therefore, the main challenge is to prove the theorem without expanding the polynomials. This requires a much more subtle proof that we sketch here. This

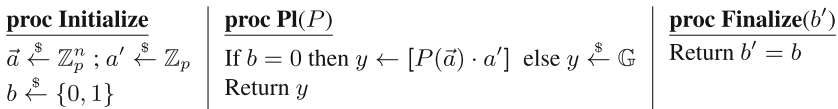
<sup>1</sup> Details on the representation of polynomials are given in the full version.

first idea is the following: instead of replacing all monomials by independent random values at once, we first fix all values  $T_2, \dots, T_n$  to randomly chosen  $a_2, \dots, a_n$ , and get polynomials in  $T_1$  only. These polynomials can be expanded without an exponential blow-up, and each monomial  $T_1, T_1^2, \dots$  can be replaced by an independent random value (instead of  $a_1, a_1^2, \dots$  for some value  $a_1$ ). Then, we can fix only  $T_3, \dots, T_n$  to randomly chosen  $a_3, \dots, a_n$ , get a polynomial in  $T_1$  and  $T_2$ , and replace all distinct monomial  $(T_1, T_1^2, T_1T_2, T_2^2, \dots)$  by independent random values. And we can continue like that until all monomials are replaced.

Obviously, if we do that so naively, we get back to the original problem: we have an exponential number of monomials. The second idea is to remark that we actually do not need to expand polynomials to replace all distinct monomials by random values and get the result, at each step of the previous idea. We could just assign random values to all polynomials (after fixing  $T_{i+1}, \dots, T_n$  to  $a_{i+1}, \dots, a_n$ ), if they are all linearly independent: this is exactly what we showed in the previous proof for expanded polynomials. And if they are not all linearly independent, we just need to take care of linear combinations, and compute the resulting value accordingly.

More precisely, for any polynomial  $P$ , let us write  $Q_P \in \mathbb{Z}_p[T_1, \dots, T_i]$  the polynomial obtained after fixing  $T_{i+1}, \dots, T_n$  to  $a_{i+1}, \dots, a_n$ . To answer the  $j$ -th query  $P_j$ , we check whether  $Q_{P_j}$  is linearly independent from  $(Q_{P_l})_{l=1, \dots, j-1}$ . If that is the case, we answer with an independent random value  $y_j$ . Otherwise, we find some linear combination between  $Q_{P_j}$  and  $(Q_{P_l})_{l=1, \dots, j-1}$ , and we write  $Q_{P_j} = \sum_{l=1}^{j-1} \lambda_l Q_{P_l}$  and outputs  $\prod_{l=1}^{j-1} y_l^{\lambda_l}$ , with  $y_l$  the output given for  $P_l$ .

The last difficulty is that this proof requires a test of linear dependence of multivariate polynomials. One way to do that would be to expand them, which is exactly what we are trying to avoid. So, instead, we use a statistical test based on the Schwartz-Zippel lemma, which basically consists in evaluating the polynomials in enough random points and looking for linear combination among the vectors of these evaluations.



**Fig. 1.** Game defining the  $(n, d)$ -LIP security for a group  $\mathbb{G}$

### 3.2 Main Theorem: LIP Security

**LIP Security.** Let  $\mathbb{G} = \langle g \rangle$  be a group of prime order  $p$ . We define the advantage of an adversary  $\mathcal{A}$  against the  $(n, d)$ -LIP security of  $\mathbb{G}$ , denoted  $\mathbf{Adv}_{\mathbb{G}}^{(n, d)\text{-lip}}(\mathcal{A})$  as the probability of success in the game defined in Fig. 1, with  $\mathcal{A}$  being restricted to make queries  $P \in \mathbb{Z}_p[T_1, \dots, T_n]$  such that for any query  $P$ , the maximum degree in one indeterminate in  $P$  is at most  $d$ , and for any sequence  $(P_1, \dots, P_q)$  of queries, the polynomials  $(P_1, \dots, P_q)$  are always *linearly independent* over  $\mathbb{Z}_p$ . Another way to look at the security definition is to consider that when  $b = 0$ ,

$\text{PI}(P)$  outputs  $[P(\vec{a})]_h = [P(\vec{a}) \cdot a']_g$ , where the generator is  $h = [a']_g$ , which is not public (but can be obtained by querying the polynomial 1), and  $g$  is a public generator.

**Theorem 1** (LIP). *Let  $\mathbb{G} = \langle g \rangle$  be a group of prime order  $p$ . Let  $\mathcal{A}$  be an adversary against the  $(n, d)$ -LIP security of  $\mathbb{G}$  that makes  $q$  oracle queries  $P_1, \dots, P_q$ . Then we can design an adversary  $\mathcal{B}$  against the  $\mathcal{E}_{1,d}$ -MDDH problem in  $\mathbb{G}$ , such that  $\text{Adv}_{\mathbb{G}}^{(n,d)\text{-lip}}(\mathcal{A}) \leq n \cdot d \cdot \text{Adv}_{\mathbb{G}}^{\mathcal{E}_{1,d}\text{-mddh}}(\mathcal{B}) + O(ndq/p)$ . The running time of  $\mathcal{B}$  is that of  $\mathcal{A}$  plus the time to perform a polynomial number (in  $q, n,$  and  $d$ ) of operations in  $\mathbb{Z}_p$  and  $\mathbb{G}$ .*

The proof is detailed in the full version.

### 4 Recovering and Extending Existing Number-Theoretic PRFs

In Table 2, we recall known number-theoretic PRFs, namely the Naor-Reingold (NR) PRF [27], its variant NR\* defined in [8], and the algebraic PRF by Boneh, Montgomery, and Raghunathan (BMR) in [17]. We also introduce weighted (extended) versions of these PRFs, namely weighted NR (WNR) and weighted BMR (WBMR), in order to construct RKA-secure PRFs for new classes of RKD functions (Sect. 5). These weighted PRFs are obtained by applying particular permutations to the key space. Then, as PRFs, it is straightforward that the security of NR and BMR implies the security of their weighted versions. However, as detailed in Sect. 5, in the RKA setting, we can prove that some of these weighted PRFs are secure against certain classes of RKD functions while both NR and BMR are not, even if we apply the BC/ABPP frameworks.

Using the LIP theorem and changing the generators used (to get PRFs of the form  $F(\vec{a}, x) = [P_x(\vec{a}) \cdot a']$ ), the security proof of WNR and WBMR is straightforward, and so is the security proof of NR, NR\*, and BMR, as particular cases of WNR and WBMR. Concretely, for WBMR<sup>w</sup>, we start by revealing the generator  $h$  to the adversary where

$$h = \left[ \left( \prod_{i=1}^n \prod_{k \in \{0, \dots, d\}} (a_i + w_i + k) \right) \cdot a' \right]_g = [P(\vec{a}) \cdot a']_g$$

which is a generator with overwhelming probability. Then, when the adversary makes a query  $x$ , it is clear that

$$\left[ \prod_{i=1}^n \frac{1}{a_i + w_i + x_i} \right]_h = \left[ \left( \prod_{i=1}^n \prod_{k \in \{0, \dots, d\} \setminus \{x_i\}} (a_i + w_i + k) \right) \cdot a' \right]_g = [P_x(\vec{a}) \cdot a']_g$$

As each polynomial  $P_x$  is null on every input  $-x'$  for  $x' \in \{0, \dots, d\}^n$ , seen as a vector of  $\mathbb{Z}_p^n$ , except when  $x' = x$ , and as  $P$  is null on all  $-x'$ ,  $P$  and  $(P_x)_x$  are linearly independent. Then, we conclude the security proof of WBMR<sup>w</sup> by applying the LIP theorem. Formal proofs are provided in the full version.



**Table 2.** Existing number-theoretic PRFs and their weighted extensions

PRF $F$	Key $\vec{a}$ Key domain $\mathcal{K}$	Domain $\mathcal{D}$	Output	$\mathbf{Adv}_F^{\text{prf}} \lesssim$
NR	$(a_0, \dots, a_n)$ $\mathcal{K} = \mathbb{Z}_p^{n+1}$	$\{0, 1\}^n$	$\left[ a_0 \prod_{i=1}^n a_i^{x_i} \right]$	$n \cdot \mathbf{Adv}_{\mathbb{G}}^{\text{ddh}}$
NR*	$(a_1, \dots, a_n)$ $\mathcal{K} = \mathbb{Z}_p^n$	$\{0, 1\}^n \setminus \{0^n\}$	$\left[ \prod_{i=1}^n a_i^{x_i} \right]$	$n \cdot \mathbf{Adv}_{\mathbb{G}}^{\text{ddh}}$
BMR	$(a_1, \dots, a_n)$ $\mathcal{K} = \mathbb{Z}_p^n$	$\{0, \dots, d\}^n$	$\left[ \prod_{i=1}^n \frac{1}{a_i + x_i} \right]$	$nd \cdot \mathbf{Adv}_{\mathbb{G}}^{d\text{-ddhi}}$
WNR $^{\vec{w}}$ $(\vec{w} \in \mathbb{Z}_p^{n+1})^*$	$(a_0, \dots, a_n)$ $\mathcal{K} = \mathbb{Z}_p^{n+1}$	if $w_0 \neq 0$ : $\{0, 1\}^n$ , else: $\{0, 1\}^n \setminus \{0^n\}$	$\left[ a_0^{w_0} \prod_{i=1}^n a_i^{w_i x_i} \right]$	$n \cdot \mathbf{Adv}_{\mathbb{G}}^{\text{ddh}\dagger}$
WBMR $^{\vec{w}}$ $(\vec{w} \in \mathbb{Z}_p^n)^\ddagger$	$(a_1, \dots, a_n)$ $\mathcal{K} = \mathbb{Z}_p^n$	$\{0, \dots, d\}^n$	$\left[ \prod_{i=1}^n \frac{1}{a_i + w_i + x_i} \right]$	$nd \cdot \mathbf{Adv}_{\mathbb{G}}^{d\text{-ddhi}}$

$\mathbb{G} = \langle g \rangle$  is a prime order group, and  $g$  is the generator used for the PRF construction;

The last column show approximate simplified bounds on the advantage  $\mathbf{Adv}_F^{\text{prf}}$  of a polynomial-time adversary against the security of the PRF  $F$ ; exact bounds can be found in the full version; Remarks: NR = WNR $^{(1, \dots, 1)}$ , NR\* = WNR $^{(0, 1, \dots, 1)}$ , and BMR = WBMR $^{(0, \dots, 0)}$ ;

<sup>a</sup> For WNR, weights are  $\vec{w} = (w_0, \dots, w_n) \in \mathbb{Z}_p^{n+1}$ ;

<sup>b</sup> When  $w_1, \dots, w_n$  are coprime to  $p - 1$ , and  $w_0$  is 0 or coprime to  $p - 1$ ;

<sup>c</sup> For WBMR, weights are  $\vec{w} = (w_1, \dots, w_n) \in \mathbb{Z}_p^n$ .

## 5 Application to Related-Key Security

In this section, we show how our theorem can be used to build RKA-secure PRFs from a PRF  $F$  defined over a prime order group  $\mathbb{G} = \langle g \rangle$  that takes a key  $\vec{a}$  and an input  $x$  and outputs a group element  $F(\vec{a}, x) = [P_x(\vec{a})]$ . Let  $\Phi$  be a class of RKD functions, where functions  $\vec{\phi} = (\phi_1, \dots, \phi_n) \in \Phi$  are such that  $\phi_i$  are multivariate polynomials in  $\mathbb{Z}_p[T_1, \dots, T_n]$ . Then, for an RKD function  $\vec{\phi}$  and an input  $x$ , the PRF outputs  $F(\vec{\phi}(\vec{a}), x) = [P_{\vec{\phi}, x}(\vec{a})]$ , where the polynomial  $P_{\vec{\phi}, x}(\vec{T}) = P_x(\vec{\phi}(\vec{T})) = P_x(\phi_1(\vec{T}), \dots, \phi_n(\vec{T}))$  depends on  $\vec{\phi}$  and  $x$ . In particular,  $P_{\text{id}, x} = P_x$  for all  $x$ , where id is the identity function.

When all polynomials  $P_{\vec{\phi}, x}$  and the constant polynomial 1 are linearly independent, the LIP theorem directly shows that  $F$  is  $\Phi$ -RKA-secure. To illustrate this, we construct in Sect. 5.1 a PRF that is secure against permutations of the secret key using this method.

However, to assume that all polynomials  $P_{\vec{\phi}, x}$  are linearly independent is a very strong property and, in general, this is not the case for all  $x$  and  $\vec{\phi}$ . Hence, in Sect. 5.2, we consider the less restrictive case where the polynomials  $P_{\vec{\phi}_1, x_1}, \dots, P_{\vec{\phi}_q, x_q}$  are linearly independent as long as the inputs  $x_1, \dots, x_q$  are distinct (in which case the adversary is said to be unique-input). More precisely, we first design a new algebraic framework that extends the one from [1], when the PRF  $F$  is of the form  $[P_x(\vec{a})]$  and the RKD functions are multivariate poly-

nomials, and then use it to construct RKA-secure PRFs from  $F$  for new and larger classes of RKD functions.

### 5.1 Direct Constructions of RKA-Secure PRFs

In this section, we show how the LIP theorem can be used to prove the  $\Phi$ -RKA-PRF security in the particular case where all polynomials  $P_{\vec{\phi},x}$  are linearly independent, for any  $\vec{\phi} \in \Phi$  and any input  $x$ .

Specifically, we consider the class  $\Phi_{\mathfrak{S}_n}$  of functions defined as  $\{\sigma \mid \sigma \in \mathfrak{S}_n\}$  such that, applying a function  $\sigma \in \Phi_{\mathfrak{S}_n}$  to a key  $\vec{a} = (a_1, \dots, a_n) \in \mathbb{Z}_p^n$  leads to the key  $\sigma(\vec{a}) = (a_{\sigma^{-1}(1)}, \dots, a_{\sigma^{-1}(n)})$ , so the  $i$ -th component of  $\vec{a}$  becomes the  $\sigma(i)$ -th component of the key  $\sigma(\vec{a})$ .

It is clear that BMR is not  $\Phi_{\mathfrak{S}_n}$ -RKA-secure, since we can distinguish BMR from a random function with only 2 queries. Indeed, let  $\text{id}$  be the identity function and  $(12)$  be the permutation which switches the first two components of the key. Then, one can just first query  $(\text{id}, 100\dots 0)$  and  $((12), 010\dots 0)$  and check whether the output of these queries are the same, which is the case in the real case while they are independent in the random case. However, we show in what follows that a particular case of WBMR, defined below, is a  $\Phi_{\mathfrak{S}_n}$ -RKA-secure PRF.

**Linear WBMR PRF.** We define  $\text{WBMR}^{\text{lin}}$  as the particular case of WBMR, where  $w_i = (i - 1)(d + 1)$ , for  $i = 1, \dots, n$ . Please refer to Table 2 for details.

**Theorem 2.** *Let  $\mathbb{G} = \langle g \rangle$  be a group of prime order  $p$  and let  $\text{WBMR}^{\text{lin}}$  be the function defined above. Then we can reduce the  $\Phi_{\mathfrak{S}_n}$ -RKA-PRF security of  $\text{WBMR}^{\text{lin}}$  to the hardness of the  $(n(d + 1) - 1)$ -DDHI problem in  $\mathbb{G}$ , with a loss of a factor  $n(n(d + 1) - 1)$ . Moreover, the time overhead of this reduction is polynomial in  $n, d$  and in the number of queries made by the adversary.*

The proof is given in the full version and is very similar to the proof of security of WBMR sketched in Sect. 4. The construction can actually be extended to also tolerates small additive factors in addition to permutations (see the full version).

### 5.2 Constructions via Unique-Input RKA-Secure PRFs

In this section, we address the less restrictive case where the polynomials  $P_{\vec{\phi}_1,x_1}, \dots, P_{\vec{\phi}_q,x_q}$  are linearly independent for any  $\vec{\phi}_1, \dots, \vec{\phi}_q$  only when the inputs  $x_1, \dots, x_q$  are all distinct. Please notice that this is the case for all the classes considered in [1, 8]. We now denote by  $M$  the “original” PRF:  $M(\vec{a}, x) = [P_x(\vec{a})]$ .

In order to build RKA-secure PRFs from such PRFs, we would like to apply the ABPP generic framework [1] that allows to transform a PRF  $M$  which is RKA-secure with respect to unique-input adversaries (UI-RKA-secure) into an RKA-secure PRF  $F$ , when  $M$  is key-collision and statistical-key-collision secure. The latter means that it is hard to find two functions  $\phi_1, \phi_2 \in \Phi$  such that

$\phi_1(K) = \phi_2(K)$ , even with access to an oracle  $(\phi, x) \mapsto f(\phi(K), x)$ , when  $f = M$  (key-collision security), and when  $f$  is a random function (statistical key-collision security). The framework consists in transforming this UI-RKA-secure PRF  $M$  into an RKA-secure PRF  $F$ , as follows:

$$F(K, x) = M(K, H(x, M(K, \vec{\omega}))),$$

where  $H$  is a compatible collision-resistant hash function, and the vector  $\vec{\omega}$  is a strong key fingerprint, meaning that it is a vector of inputs such that the vector of outputs  $M(K, \vec{\omega})$  completely defines  $K$  (recall that  $M(K, \vec{\omega}) = (M(K, \omega_1), \dots, M(K, \omega_{|\vec{\omega}|}))$ ). As defined in [8], a hash function is said to be compatible if it guarantees that the inner calls to  $M$  in the construction above will never collide with the outer calls to  $M$  even under related keys.

Unfortunately, if we consider the PRF  $\text{WNR}^{\vec{w}}$  with some  $w_i > 1$ , then it is not clear how to find a strong key fingerprint, which can be used to apply the ABPP framework. Furthermore, this ABPP framework requires to prove several non-algebraic properties (statistical or computational), namely key-collision, statistical-key-collision, and UI-RKA securities.

For this reason, we design a new algebraic framework, that generalizes the ABPP framework in the particular case of PRFs of the shape  $M(\vec{a}, x) = [P_x(\vec{a})]$  and of RKD functions which are multivariate polynomials. For completeness, a more general framework, which does not make any assumptions about the shape of a PRF, is also given in the full version. Afterwards, we use our algebraic framework to design new RKA-secure PRFs based on WNR for larger classes for which previous constructions from [1, 8] are not secure.

**An Algebraic Framework for Related-Key Security.** Here, we describe a

new framework that transforms any PRF that satisfies that  $P_{\vec{\phi}_1, x_1}, \dots, P_{\vec{\phi}_q, x_q}$  are linearly independent, for any  $\vec{\phi}_1, \dots, \vec{\phi}_q$  as long as  $x_1, \dots, x_q$  are all distinct inputs, into a RKA-secure PRF. To do so, we first introduce three new notions, termed *algebraic fingerprint*, *helper information*, and *expansion function*, and defined as follows.

**GROUP GENERATOR.** In this framework and its applications, we assume for simplicity that the generator used in the PRF construction, that is revealed to the adversary, is  $[a']$ .

**ALGEBRAIC FINGERPRINT.** In order to overcome the eventual lack of a strong key fingerprint, we introduce algebraic fingerprint, which will be used to replace  $M(K, \vec{\omega})$  in the construction in [1], where  $\vec{\omega}$  is a strong fingerprint. An algebraic fingerprint is simply an injective function  $\vec{\Omega}: \mathbb{Z}_p^n \rightarrow \mathbb{G}^m$  such that the image  $\vec{\Omega}(\vec{a})$  is a vector of group elements  $([\Omega_1(\vec{a})a'], \dots, [\Omega_m(\vec{a})a'])$  with  $\Omega_1, \dots, \Omega_m$  being polynomials in  $\mathbb{Z}_p[T_1, \dots, T_n]$  and  $a' \in \mathbb{Z}_p$ . In our applications, we will simply have  $\vec{\Omega}(\vec{a}) = ([a_1a'], \dots, [a_na'])$ , so  $m = n$  and  $\Omega_i(\vec{T}) = T_i$  for  $i = 1, \dots, n$ .

**HELPER INFORMATION.** In order to prove the security of our framework, we need to be able to compute the image of the algebraic fingerprint,  $\vec{\Omega}(\vec{\phi}(\vec{a})) = ((\Omega_1 \circ \vec{\phi})(\vec{a}), \dots, (\Omega_m \circ \vec{\phi})(\vec{a}))$ , for any related key  $\vec{\phi}(\vec{a}) \in \mathbb{Z}_p^n$ , with  $\vec{\phi} \in \Phi$ , from

some information which can somehow be made public without hurting security. We call this information a helper information, write it  $\text{Help}_\Phi(\vec{a})$ , and call  $\text{Help}_\Phi$  the helper function. We suppose that  $\text{Help}_\Phi(\vec{a}) = ([\text{help}_1(\vec{a})a'], \dots, [\text{help}_l(\vec{a})a'])$ , with  $\text{help}_1, \dots, \text{help}_l$  linearly independent polynomials which generate a vector subspace of  $\mathbb{Z}_p[T_1, \dots, T_n]$  containing the polynomials  $\Omega_i \circ \vec{\phi}$  for  $i = 1, \dots, m$ , and  $\vec{\phi} \in \Phi$ .

**HASH FUNCTION AND EXPANSION FUNCTION.** Let  $\overline{\mathcal{D}} = \mathcal{D} \times \mathbb{G}^m$  where  $\mathcal{D}$  is the domain of the PRF  $M$ , and let  $h$  be a collision-resistant hash function  $h: \overline{\mathcal{D}} \rightarrow \text{hSp}$  (definition recalled in the full version), where  $\text{hSp}$  is a large enough space. The last thing we need to define is an expansion function, which is simply an injective function  $\text{E}: \text{hSp} \rightarrow \mathcal{S} \subseteq \mathcal{D}$  such that for any sequence  $(\vec{\phi}_1, x_1), \dots, (\vec{\phi}_q, x_q)$  where  $x_1, \dots, x_q$  are distinct inputs in  $\mathcal{S}$  and  $\vec{\phi}_1, \dots, \vec{\phi}_q$  are RKD functions, polynomials  $\text{help}_1, \dots, \text{help}_l$  and polynomials  $P_{\vec{\phi}_1, x_1}, \dots, P_{\vec{\phi}_q, x_q}$  and 1 (which needs to be queried to define the generator  $[a']$ ) are linearly independent over  $\mathbb{Z}_p$  (in particular,  $\text{E}$  has to be injective).

Using these new tools, we obtain the following framework.

**Theorem 3.** *Let  $\mathbb{G}$  be a group of prime order  $p$ . We use the above definitions, with  $M: \mathbb{Z}_p^n \times \mathcal{D} \rightarrow \mathbb{G}$  defined by  $M(\vec{a}, x) = [P_x(\vec{a})]$ . Let  $d$  be an upper bound for the maximum degree in any indeterminate of polynomials in  $\{\text{help}_1, \dots, \text{help}_l\} \cup \{P_{x, \vec{\phi}} \mid x \in \mathcal{S}, \vec{\phi} \in \Phi\}$ . Define  $F: \mathbb{Z}_p^n \times \mathcal{D} \rightarrow \mathbb{G}$  by*

$$F(\vec{a}, x) = M(\vec{a}, \text{E}(h(x, \vec{\Omega}(\vec{a}))))$$

for all  $\vec{a} \in \mathbb{Z}_p^n$  and  $x \in \mathcal{D}$ . Then, we can reduce the  $\Phi$ -RKA-PRF security of  $F$  to the  $(n, d)$ -LIP security, the collision-resistance security of  $h$  without any loss, and to the [-SDLd] assumption with a loss of a factor  $2n$ . The running time overhead of this reduction is polynomial in  $n, d$  and  $q$ .

**PROOF OVERVIEW.** The proof of the above theorem is detailed in the full version and relies on the sequence of 10 games (games  $G_0 - G_9$ ). We first prove an intermediate statement whose proof is very similar to the proof of Theorem 3.1 from [1], under a notion termed extended key-collision security (that states the hardness of finding key collisions given access to PRF values and helper information) which is defined in the appendix. Afterwards, we reduce this notion to the hardness of the SDL in  $\mathbb{G}$ . Here we provide a brief overview of the proof of the intermediate statement.

We start by giving the generator used for the PRF by querying polynomial 1. Hence, the generator is simply  $[a']$ . Since we may have key collisions (i.e., two RKD functions  $\phi_1 \neq \phi_2$ , such that  $\phi_1(\vec{a}) = \phi_2(\vec{a})$ ), we start by dealing with possible collisions on the related keys in the RKAPRFReal case, using the extended key-collision notion (games  $G_0 - G_2$ ). These claws can be detected by looking for collisions on images of  $\vec{\Omega}$  for different RKD functions.

Then, in games  $G_3 - G_4$ , we deal with possible collisions on hash values in order to ensure that the inputs  $t = E(h(x, \vec{Q}(\vec{a})))$  used to compute the output  $y$  are distinct (recall that  $E$  is injective).

Then, we use the  $(n, d)$ -LIP security notion to show that it is hard to distinguish the output of  $F$  and the helper information from uniformly random values (games  $G_5 - G_6$ ).

Finally, we use once again the extended-key-collision security notion to deal with possible key collisions in the RKAPRFRand case (games  $G_7 - G_9$ ) so that  $G_9$  matches the description of the RKAPRFRand game. These key collisions can still be detected in these games by making crucial use of the helper information.

**RKA-PRFs for Permutations of Univariate Polynomial Functions.** We now apply our framework to a particular case of WNR and build the first RKA-secure PRF secure against permutations of univariate polynomials. We chose to set  $w_0$  to 0 in our construction in order to ease the readability so that the key space of the PRF stays  $\mathbb{Z}_p^n$ , but similar results can be proven with  $w_0 = 1$  or set to a prime number  $p_0 > d$  (and distinct to  $p_1, \dots, p_n$  defined below).

For  $d \geq 1$ , let  $\Phi_d$  be the class of degree at most  $d$  non-constant univariate polynomials defined as  $\Phi_d = \{\vec{\phi}: \mathbb{Z}_p^n \rightarrow \mathbb{Z}_p^n \mid \phi_i: \vec{T} \mapsto \sum_{j=0}^d \alpha_{i,j} T_i^j, (\alpha_{i,1}, \dots, \alpha_{i,d}) \neq 0^d, \forall i = 1, \dots, n\}$ . Then we consider the class  $\Phi_{\mathfrak{S}_n, d}$  of permutations of degree at most  $d$  non-constant univariate polynomials, defined as follows:

$$\Phi_{\mathfrak{S}_n, d} = \{\sigma \circ \vec{\phi} \mid (\sigma, \vec{\phi}) \in \mathfrak{S}_n \times \Phi_d\} .$$

For a key  $\vec{a} = (a_1, \dots, a_n) \in \mathbb{Z}_p^n$ , applying an RKD function  $\sigma \circ \vec{\phi} \in \Phi_{\mathfrak{S}_n, d}$ , where  $\vec{\phi} = (\phi_1, \dots, \phi_n)$  leads to the key  $(\phi_{\sigma^{-1}(1)}(\vec{a}), \dots, \phi_{\sigma^{-1}(n)}(\vec{a})) \in \mathbb{Z}_p^n$ , so the  $i$ -th component  $a_i$  of the key is changed into  $\phi_i(\vec{a})$  and becomes the  $\sigma(i)$ -th component of the related key.

Before explaining our construction, we would like to point out that, even if we just consider the simple class of permutations  $\Phi_{\mathfrak{S}_n} \subset \Phi_{\mathfrak{S}_n, 1}$  introduced in Sect. 5.1, we can already show that NR and NR\* are not  $\Phi_{\mathfrak{S}_n}$ -RKA secure, even with respect to unique-input adversaries.

Indeed, let us consider NR\*: let  $\text{id}$  be the identity function and  $(12)$  be the permutation which switches the first two components of the key. Then, the output of the queries  $(\text{id}, 100 \dots 0)$  and  $((12), 010 \dots 0)$  will be the same in the real case and independent in the random case.

In fact, we can generalize the attack above to show that there even exists a compatible collision-resistant hash function  $h$  such that the PRF that one obtains when applying the Bellare-Cash (or ABPP) transform to NR\* would not be RKA-secure with respect to the class of permutations. Indeed, let  $h'$  be a collision-resistant hash function. The counter-example for  $h$  could be as follows (where  $x_1$  and  $x_2$  are two arbitrary distinct inputs):

$$h(x, [a_1], \dots, [a_n]) = \begin{cases} 1110 \parallel h'(x_1, [a_1], \dots, [a_n]) & \text{if } x = x_1 \\ 1101 \parallel h'(x_1, [a_2], [a_1], [a_3], \dots, [a_n]) & \text{if } x = x_2 \\ 1111 \parallel h'(x, [a_1], \dots, [a_n]) & \text{otherwise.} \end{cases}$$

Note that  $h$  is a compatible collision-resistant hash function. It is easy to see that the output of the queries  $(\text{id}, x_1)$  and  $((12), x_2)$  will be the same in the real case and independent in the random case. The same kind of attack can be mounted against NR.

However, while NR and  $\text{NR}^*$  are not RKA-secure against permutations attacks, we show in what follows that a particular case of WNR, defined below, yields a  $\Phi_{\mathfrak{S}_n, d}$ -RKA-secure PRF.

**$d$ -Linear Weighted NR PRF.** Let  $d \geq 1$ . Let  $p_1 < p_2 < \dots < p_n$  be distinct prime numbers such that  $p_1 > d$ . We define  $\text{WNR}^{d\text{-lin}}$  as the particular case of WNR, where  $w_0 = 0$  and  $w_i = p_i$ . Please refer to Table 2 for details. Using standard inequalities over prime numbers, it is easy to see that we can find  $p_1, \dots, p_n$  such that  $p_n = \tilde{O}(d + n)$ .

In order to apply the framework from Theorem 3 to  $\text{WNR}^{d\text{-lin}}$  and  $\Phi_{\mathfrak{S}_n, d}$ , we define:

- $[a'] \in \mathbb{G}$  is the generator used for the PRF construction
- $\vec{\Omega}: \vec{a} \in \mathbb{Z}_p^n \mapsto ([a_1 a'], \dots, [a_n a']) \in \mathbb{G}^n$
- $\text{Help}_{\Phi_{\mathfrak{S}_n, d}}: \vec{a} \in \mathbb{Z}_p^n \mapsto ([a'], [a_1 a'], \dots, [a_1^d a'], \dots, [a_n a'], \dots, [a_n^d a']) \in \mathbb{G}^{nd+1}$
- $h$  can be any collision-resistant hash function  $h: \{0, 1\}^n \times \mathbb{G}^n \rightarrow \{0, 1\}^{n-2}$
- $\text{E}: z \in \{0, 1\}^{n-2} \mapsto 11 \parallel z \in \{0, 1\}^n$ .

We just need to prove that  $\text{E}$  satisfies the linear independence property required to apply the framework, which is done in the full version, and sketched here. We order monomials of multivariate polynomials, with any order respecting the total degree of polynomials (e.g., the graded lexicographic order). The leading monomial (i.e., the first monomial for that order) of the polynomial  $P_{\vec{\phi}, x}$  is  $T_1^{x_{\sigma(1)} p_{\sigma(1)} d_1} \dots T_n^{x_{\sigma(n)} p_{\sigma(n)} d_n}$ , with  $d_i > 0$  the degree of  $\phi_i$ . The polynomials for the helper information ( $\text{help}_k$ ) are  $T_i^j$ . Therefore, the leading monomials of  $\text{help}_1, \dots, \text{help}_l, P_{\vec{\phi}_1, x_1}, \dots, P_{\vec{\phi}_q, x_q}, 1$  are all distinct, when  $x_1, \dots, x_q$  are distinct inputs. This means that the matrix whose columns correspond to monomials (ordered as specified above) and whose rows correspond to the polynomials  $\text{help}_1, \dots, \text{help}_l, P_{\vec{\phi}_1, x_1}, \dots, P_{\vec{\phi}_q, x_q}, 1$  (ordered according to their leading monomial) is in echelon form. Hence, the latter polynomials are linearly independent. Finally, by combining Theorem 3 and the LIP theorem, we obtain the following theorem.

**Theorem 4.** *Let  $\vec{\Omega}$ ,  $h$  and  $\text{E}$  be defined as above. Define  $F: \mathbb{Z}_p^n \times \{0, 1\}^n \rightarrow \mathbb{G}$  by  $F(\vec{a}, x) = \text{WNR}^{d\text{-lin}}(\vec{a}, \text{E}(h(x, \vec{\Omega}(\vec{a}))))$ , for all  $\vec{a} \in \mathbb{Z}_p^n$  and  $x \in \{0, 1\}^n$ . Then we can reduce the  $\Phi_{\mathfrak{S}_n, d}$ -RKA-PRF security of  $F$  to the hardness of the  $p_n d$ -DDHI problem in  $\mathbb{G}$  and the  $p_n d$ -SDL problem in  $\mathbb{G}$ , respectively with a loss of a factor  $np_n d$  and of a factor  $n$ , and to the CR security of  $h$ . Moreover, the time overhead of this reduction is polynomial in  $n, d, p_n$  and in the number of queries made by the adversary attacking the  $\Phi_{\mathfrak{S}_n, d}$ -RKA-PRF security of  $F$ .*

## 6 Extension to PRFs in Symmetric Bilinear Groups

### 6.1 High-Level Overview of Existing Constructions and Challenges

All the previous constructions (of classical PRF and RKA-secure PRF) require at least DDH to hold. In particular, they are insecure if there exists a symmetric pairing  $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ . In this section, we investigate how to adapt our linearly independent polynomials framework and the corresponding LIP theorem to handle constructions of PRFs under weaker assumptions, which may hold in symmetric bilinear groups.

The first algebraic PRF based on DLin is the Lewko-Waters PRF [26], which is defined as follows:

$$\text{LW}(\vec{A}, x) = \left[ \prod_{i=1}^n A_i^{x_i} \cdot A' \right],$$

with  $\vec{A} = (A_1, \dots, A_n)$  being a vector of  $n$  uniformly random matrices in  $\mathbb{Z}_p^{2 \times 2}$  and  $A'$  a uniformly random matrix in  $\mathbb{Z}_p^{2 \times m}$ , for some  $m \geq 1$ .  $A'$  was actually in  $\mathbb{Z}_p^{2 \times 1}$  (i.e.,  $m = 1$ ) in [26] (with only the first group element being returned). This PRF is secure under DLin, and even under a weaker assumption, namely the  $\mathcal{U}_2$ -MDDH-assumption of Escala et al. [18]. In the latter paper, this PRF is extended to any MDDH-assumption, which particularly encompasses DDH and DLin. These instantiations differ by the size of the matrices and their distribution. Except for constructions using multilinear maps and lattices [5, 16] or trivial variants, we are not aware of any other construction.

**Commutation Challenge.** From a high level point of view, these PRFs are very similar to the one considered in our algebraic framework in Sect. 3, except elements of keys are now matrices. Unfortunately, matrices do not commute in general, and this lack of commutativity makes everything more complex.

One naive solution would be to extend the LIP theorem by considering non-commutative polynomials, or in other words elements of the free algebra  $\mathbb{Z}_p\langle T_1, \dots, T_n \rangle$ . In this algebra, for example,  $T_1T_2$  and  $T_2T_1$  are distinct and linearly independent elements. The problem is that, as proven by Amitsur and Levitzki [4], for any matrices  $A_1, \dots, A_4 \in \mathbb{Z}_p^{2 \times 2}$ ,  $\sum_{\sigma \in \mathcal{S}_4} \text{sgn}(\sigma) \cdot A_{\sigma(1)} \cdot A_{\sigma(2)} \cdot A_{\sigma(3)} \cdot A_{\sigma(4)} = 0$ , with  $\text{sgn}(\sigma)$  being the parity of the permutation  $\sigma$ . Thus, while the family of non-commutative polynomials  $(P_\sigma = T_{\sigma(1)}T_{\sigma(2)}T_{\sigma(3)}T_{\sigma(4)})_{\sigma \in \mathcal{S}_4}$  is linearly independent in the free algebra, the PRF of domain  $\mathcal{D} = \mathcal{S}_4$ , the PRF defined by  $F(\vec{A}, \sigma) = [A_{\sigma(1)}A_{\sigma(2)}A_{\sigma(3)}A_{\sigma(4)}A']$  would clearly be insecure.

**Assumption Challenge and Generic Symmetric Bilinear Group.** The second challenge is to prove the hardness of the  $\mathcal{E}_{2,d}$ -MDDH assumption in the generic bilinear group, which is done in the full version, using a non-trivial technical lemma. Notably, contrary to the cyclic group case, it is not straightforward to check whether a PRF defined by  $F(\vec{A}, x) = [P_x(\vec{A}) \cdot A']$  is secure in the generic bilinear group model, where  $(P_x)_{x \in \mathcal{D}}$  is a family of non-commutative polynomials,  $\vec{A}$  is a vector of matrices from  $\mathbb{Z}_p^{2 \times 2}$ , and  $A'$  is a matrix from  $\mathbb{Z}_p^{2 \times m}$ , for some  $m \geq 1$ .

### 6.2 Generalized Polynomial Framework

Let us show how we address these challenges.

**Generalized Polynomial (GP) Security.** Let us introduce the  $(k, n, d)$ -GP security of a cyclic group  $\mathbb{G} = \langle g \rangle$  as a generalization of the  $(n, d)$ -LIP security in Sect. 3.2, where the secret scalar  $a' \xleftarrow{\$} \mathbb{Z}_p$  and the secret vector of scalars  $\vec{a} \xleftarrow{\$} \mathbb{Z}_p^n$  are replaced by a secret matrix  $\mathbf{A}' \xleftarrow{\$} \mathbb{Z}_p^{k \times m}$  (for some  $m \geq 1$ ; for the sake of simplicity, in the sequel, we choose  $k = m$ ) and a secret vector of matrices  $\vec{\mathbf{A}} \xleftarrow{\$} (\mathbb{Z}_p^{k \times k})^n$ , respectively.

**Result under  $\mathcal{E}_{2,d}$ -MDDH.** To extend Theorem 1 to symmetric bilinear groups and avoid the commutativity problem, we suppose that all indeterminates appear “in the same order when multiplied together” in each subexpression of the representation of the non-commutative polynomials  $P_j$  (e.g.,  $P_1 = T_1T_3 + T_3T_2$  and  $P_2 = T_3 + T_1T_2$ , where  $T_1$  appears before  $T_3$  which appears before  $T_2$ ). The condition is quite natural and is formally defined in the full version. That makes these non-commutative polynomials behave very similarly to commutative polynomial, and we get the following theorem.

**Theorem 5.** *Let  $\mathbb{G} = \langle g \rangle$  be a group of prime order  $p$ . Let  $\mathcal{A}$  be an adversary against the  $(2, n, d)$ -GP security of  $\mathbb{G}$  that makes  $q$  oracle queries  $P_1, \dots, P_q$ . We suppose that all indeterminates appear in the same order in each monomial of each non-commutative polynomials  $P_j$ . Then we can build an adversary  $\mathcal{B}$  against the  $\mathcal{E}_{2,d}$ -MDDH problem in  $\mathbb{G}$ , such that  $\text{Adv}_{\mathbb{G}}^{(2,n,d)\text{-gp}}(\mathcal{A}) \leq n \cdot d \cdot \text{Adv}_{\mathbb{G}}^{\mathcal{E}_{2,d}\text{-mddh}}(\mathcal{B}) + O(ndq/p)$ . The running time of  $\mathcal{B}$  is that of  $\mathcal{A}$  plus the time to perform a polynomial number (in  $q, n,$  and  $d$ ) of operations in  $\mathbb{Z}_p$  and  $\mathbb{G}$ .*

The proof is similar to the proof of the LIP theorem (with some additional care when partially evaluating polynomials to avoid having polynomials with matrix coefficients) and is given in the full version. Actually, this theorem can trivially be extended to the  $(k, n, d)$ -GP security and the  $\mathcal{E}_{k,d}$ -MDDH assumption. But for  $k \geq 3$  and  $n \geq 2$ , it is not known if the latter assumption is secure in the generic  $k$ -linear group model.

**Results in the Generic Bilinear Group Model.** We may wonder whether the  $(2, k, d)$ -GP security still holds in the generic bilinear group model, when indeterminates do not necessarily appear in the same order in each polynomial  $P_j$ . As seen before, it is not sufficient to suppose that  $(P_j)_{j=1,\dots,q}$  is a linearly independent family. But we show here that under a relatively natural condition, the *DLM* (distinct leading monomial) condition, the  $(2, k, d)$ -GP security still holds.

To formally state our result, we need to introduce some notions, which are formally defined in the full version and which are informally described here. We consider a monomial order for  $\mathbb{Z}_p[T_1, \dots, T_n]$ , which is a total order on monomials  $T_1^{i_1} \dots T_n^{i_n}$  compatible with multiplications and where 1 is the smallest monomial. We then define the commutative leading monomials of a non-commutative



**Table 3.** Summary of our results related to generalized polynomial security

Cyclic Group $\mathbb{G}$	Symmetric Bilinear Group (pairing $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ )
$P_j \in \mathbb{Z}_p[T_1, \dots, T_n]$ commutative polynomial $(a', a_1, \dots, a_n) \stackrel{\$}{\leftarrow} \mathcal{K} = \mathbb{Z}_p^{n+1}$	$P_j \in \mathbb{Z}_p\langle T_1, \dots, T_n \rangle$ non-commutative polynomial $(a', a_1, \dots, a_n) \stackrel{\$}{\leftarrow} \mathcal{K} = (\mathbb{Z}_p^{2 \times 2})^{n+1}$
In generic cyclic group: $(1, n, d)$ -GP security $\Leftrightarrow (P_j)_j$ satisfies the DLM condition $\Leftrightarrow (P_j)_j$ is linearly independent <div style="text-align: right;">(easy)</div>	In generic bilinear group: $(2, n, d)$ -GP security $\Leftrightarrow (P_j)_j$ satisfies the DLM condition <div style="text-align: right;">(Theorem 6)</div>
Under $\mathcal{E}_{1,d}$ -MDDH: $(1, n, d)$ -GP security $\Leftrightarrow$ same condition as above <div style="text-align: right;">(Theorem 1, the LIP theorem)</div>	Under $\mathcal{E}_{2,d}$ -MDDH: $(2, n, d)$ -GP security $\Leftrightarrow$ same condition as above + same order for indeterminates or equivalently, $(P_j)_j$ is linearly independent + same order for indeterminates <div style="text-align: right;">(Theorem 5)</div>

polynomial as the monomials which are the highest for our monomial order, when considered as commutative monomials. There may be many commutative leading monomials for a given polynomial (for example  $T_1T_2^2 + 5T_2T_1T_2$  has two commutative leading monomials:  $T_1T_2^2$  and  $T_2T_1T_2$ ). We say a polynomial has a unique commutative leading monomial if there is only one such monomial.

Finally, we say that a family of polynomials  $(P_j)_j$  satisfies the DLM condition, if there exists a monomial order and an invertible matrix  $M \in \mathbb{Z}_p^{q \times q}$  such that  $M \cdot (P_j)_j$  is a vector of non-commutative polynomials with unique and distinct commutative leading monomials, where  $(P_j)_j$  is the column vector of polynomials  $P_j$ .

**Theorem 6.** *Let  $\mathbb{G} = \langle g \rangle$  be a group of prime order  $p$ . Let  $\mathcal{A}$  be an adversary against the  $(2, n, d)$ -GP security of  $\mathbb{G}$  that makes  $q$  oracle queries  $P_1, \dots, P_q$ . We suppose that  $(P_j)_j$  satisfies the DLM condition. Then, the advantage  $\text{Adv}_{\mathbb{G}}^{(2,n,d)\text{-gp}}(\mathcal{A})$  is negligible in the generic bilinear group model.*

The proof of Theorem 6 is given in the full version. We remark that, in the case of commutative polynomials (i.e., LIP theorem), the DLM condition is exactly the same as saying that the polynomials  $P_j$  are linearly independent (using the Gauss reduction). However, this is not the case with non-commutative polynomials (e.g., consider  $P_1 = T_1T_2$  and  $P_2 = T_2T_1$  which are linearly independent but which have the same leading monomial).

**Summary.** Table 3 provides a summary of all our results about GP security.

### 6.3 Applications

**RKA-PRFs in Generic Bilinear Groups.** The RKA-PRF for permutation of univariate polynomial functions based on WNR (Sect. 5.2) can easily be transformed into an RKA-secure PRF for symmetric bilinear groups for the same set of RKD functions. It is sufficient to change keys from  $\vec{a} \stackrel{s}{\leftarrow} \mathbb{Z}_p^n$  to  $\vec{A} \stackrel{s}{\leftarrow} (\mathbb{Z}_p^{2 \times 2})^n$ . Indeed, the RKA framework extends to this case easily, and the polynomials family we considered verifies the DLM condition as non-commutative polynomials. Actually, our proof of their linear independence can be seen as exhibiting a monomial order (namely the graded lexicographic order) for which these polynomials have distinct leading monomials. In addition, their leading monomials are always unique even as non-commutative polynomials.

**RKA-PRFs under  $\mathcal{E}_{2,d}$ -MDDH.** Unfortunately, Theorem 5 does not apply to RKA-PRF for permutation, as permutation change the order of the indeterminates. However, it still easily enables to construct the first RKA-PRF for univariate polynomial functions, secure in symmetric bilinear groups, using the construction of Sect. 5.2 (or a slightly more efficient variant thereof in the full version). Again, the construction is straightforward and so is the proof.

**Acknowledgments.** This work was supported by the French ANR-10-SEGI-015 PRINCE Project, the *Direction Générale de l'Armement* (DGA), the CFM Foundation, and the European Research Council under the European Union's Seventh Framework Program (FP7/2007–2013 Grant Agreement 339563 – CryptoCloud).

### References

1. Abdalla, M., Benhamouda, F., Passelègue, A., Paterson, K.G.: Related-key security for pseudorandom functions beyond the linear barrier. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 77–94. Springer, Heidelberg (2014)
2. Abdalla, M., Catalano, D., Fiore, D.: Verifiable random functions from identity-based key encapsulation. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 554–571. Springer, Heidelberg (2009)
3. Advanced Encryption Standard (AES). National Institute of Standards and Technology (NIST), FIPS PUB 197, U.S. Department of Commerce, November 2001
4. Amitsur, A.S., Levitzki, J.: Minimal identities for algebras. Proc. Am. Math. Soc. **1**(4), 449–463 (1950)
5. Banerjee, A., Peikert, C.: New and improved key-homomorphic pseudorandom functions. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 353–370. Springer, Heidelberg (2014)
6. Banerjee, A., Peikert, C., Rosen, A.: Pseudorandom functions and lattices. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 719–737. Springer, Heidelberg (2012)
7. Bellare, M., Canetti, R., Krawczyk, H.: Pseudorandom functions revisited: The cascade construction and its concrete security. In: 37th FOCS, October 1996, pp. 514–523. IEEE Computer Society Press (1996)

8. Bellare, M., Cash, D.: Pseudorandom functions and permutations provably secure against related-key attacks. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 666–684. Springer, Heidelberg (2010)
9. Bellare, M., Kohno, T.: A theoretical treatment of related-key attacks: RKA-PRPs, RKA-PRFs, and applications. In: Biham, Eli (ed.) EUROCRYPT 2003. LNCS, vol. 2656. Springer, Heidelberg (2003)
10. 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)
11. Biham, E.: New types of cryptanalytic attacks using related keys. In: Helleseht, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 398–409. Springer, Heidelberg (1994)
12. 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)
13. 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)
14. 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)
15. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)
16. Boneh, D., Lewi, K., Montgomery, H., Raghunathan, A.: Key homomorphic PRFs and their applications. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 410–428. Springer, Heidelberg (2013)
17. Boneh, D., Montgomery, H.W., Raghunathan, A.: Algebraic pseudorandom functions with improved efficiency from the augmented cascade. In: Al-Shaer, E., Keromytis, A.D., Shmatikov, V. (eds.) ACM CCS 10, October 2010, pp. 131–140. ACM Press (2010)
18. Escala, A., Herold, G., Kiltz, E., Ràfols, C., Villar, J.: An algebraic framework for diffie-hellman assumptions. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 129–147. Springer, Heidelberg (2013)
19. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. *J. ACM* **33**(4), 792–807 (1986)
20. Goyal, V., O’Neill, A., Rao, V.: Correlated-input secure hash functions. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 182–200. Springer, Heidelberg (2011)
21. Hofheinz, D., Kiltz, E.: Secure hybrid encryption from weakened key encapsulation. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 553–571. Springer, Heidelberg (2007)
22. Hohenberger, S., Waters, B.: Constructing Verifiable Random Functions with Large Input Spaces. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 656–672. Springer, Heidelberg (2010)
23. Kim, J.-S., Hong, S.H., 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)
24. Knudsen, L.R.: Cryptanalysis of LOKI91. In: Zheng, Yuliang, Seberry, Jennifer (eds.) AUSCRYPT 1992. LNCS, vol. 718. Springer, Heidelberg (1993)
25. Lewi, K., Montgomery, H., Raghunathan, A.: Improved constructions of PRFs secure against related-key attacks. In: Boureau, I., Owesarski, P., Vaudenay, S. (eds.) ACNS 2014. LNCS, vol. 8479, pp. 44–61. Springer, Heidelberg (2014)

26. Lewko, A.B., Waters, B.: Efficient pseudorandom functions from the decisional linear assumption and weaker variants. In: Al-Shaer, E., Jha, S., Keromytis, A.D. (eds.) ACM CCS 09, November 2009, pp. 112–120. ACM Press (2009)
27. Naor, M., Reingold, O.: Number-theoretic constructions of efficient pseudo-random functions. In: 38th FOCS, October 1997, pp. 458–467. IEEE Computer Society Press (1997)
28. Naor, M., Reingold, O.: Synthesizers and their application to the parallel construction of pseudo-random functions. *J. Comput. Syst. Sci.* **58**(2), 336–375 (1999)
29. Shacham, H.: A cramer-shoup encryption scheme from the linear assumption and from progressively weaker linear variants. *Cryptology ePrint Archive*, Report 2007/074 (2007). <http://eprint.iacr.org/2007/074>
30. 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)

# **Block Cipher Cryptanalysis**

# Integral Cryptanalysis on Full MISTY1

Yosuke Todo<sup>(✉)</sup>

NTT Secure Platform Laboratories, Tokyo, Japan  
todo.yosuke@lab.ntt.co.jp

**Abstract.** MISTY1 is a block cipher designed by Matsui in 1997. It was well evaluated and standardized by projects, such as CRYPTREC, ISO/IEC, and NESSIE. In this paper, we propose a key recovery attack on the full MISTY1, i.e., we show that 8-round MISTY1 with 5 FL layers does not have 128-bit security. Many attacks against MISTY1 have been proposed, but there is no attack against the full MISTY1. Therefore, our attack is the first cryptanalysis against the full MISTY1. We construct a new integral characteristic by using the propagation characteristic of the division property, which was proposed in 2015. We first improve the division property by optimizing a public S-box and then construct a 6-round integral characteristic on MISTY1. Finally, we recover the secret key of the full MISTY1 with  $2^{63.58}$  chosen plaintexts and  $2^{121}$  time complexity. Moreover, if we can use  $2^{63.994}$  chosen plaintexts, the time complexity for our attack is reduced to  $2^{107.9}$ . Note that our cryptanalysis is a theoretical attack. Therefore, the practical use of MISTY1 will not be affected by our attack.

**Keywords:** MISTY1 · Integral attack · Division property

## 1 Introduction

MISTY [17] is a block cipher designed by Matsui in 1997 and is based on the theory of provable security [19, 20] against differential attack [3] and linear attack [15]. MISTY has a recursive structure, and the component function has a unique structure, the so-called MISTY structure [16]. There are two types of MISTY, MISTY1 and MISTY2. MISTY1 adopts the Feistel structure whose F-function is designed by the recursive MISTY structure. MISTY2 does not adopt the Feistel structure and uses only the MISTY structure. Both ciphers achieve provable security against differential and linear attacks. MISTY1 is designed for practical use, and MISTY2 is designed for experimental use.

MISTY1 is a 64-bit block cipher with 128-bit security, and it has a Feistel structure with FL layers, where the *FO* function is used in the F-function of the Feistel structure. The *FO* function is constructed by using the 3-round MISTY structure, where the *FI* function is used as the F-function of the MISTY structure. Moreover, the *FI* function is constructed by using the 3-round MISTY structure, where a 9-bit S-box  $S_9$  and 7-bit S-box  $S_7$  are used in the F-function. MISTY1 is the candidate recommended ciphers list of CRYPTREC [6], and it is

**Table 1.** Summary of single secret-key attacks against MISTY1

Rounds	#FL layers	Attack algorithm	Data	Time	Reference
5	0	higher order differential	$11 \times 2^7$ CP	$2^{17}$	[23]
5	3	SQUARE	$2^{34}$ CP	$2^{48}$	[13]
5	4	higher order differential	$2^{22}$ CP	$2^{28}$	[10]
5	4	impossible differential	$2^{38}$ CP	$2^{46.45}$	[8]
6	4	higher order differential	$2^{53.7}$ CP	$2^{53.7}$	[25]
6	4	impossible differential	$2^{51}$ CP	$2^{123.4}$	[8]
7	0	impossible differential	$2^{50.2}$ KP	$2^{114.1}$	[8]
7	4	higher order differential	$2^{54.1}$ CP	$2^{120.7}$	[25]
7	4	higher order differential	$2^{50.1}$ CP	$2^{100.4}$	[2]
7	5	higher order differential	$2^{51.4}$ CP	$2^{121}$	[2]
8	5	integral by division property	$2^{63.58}$ CP	$2^{121}$	This paper
8	5	integral by division property	$2^{63.994}$ CP	$2^{107.9}$	This paper

standardized by ISO/IEC 18033-3 [11]. Moreover, it is a NESSIE-recommended cipher [18] and is described in RFC 2994 [21]. There are many existing attacks against MISTY1, and we summarize these attacks in Table 1. A higher-order differential attack is the most powerful attack against MISTY1, and this type of cryptanalysis was recently improved in [2]. However, there is no attack against the full MISTY1, i.e., 8-round MISTY1 with 5 FL layers.

**Integral Attack.** The integral attack [13] was first proposed by Daemen et al. to evaluate the security of SQUARE [7] and was then formalized by Knudsen and Wagner. There are two major techniques to construct an integral characteristic; one uses the propagation characteristic of integral properties [13], and the other estimates the algebraic degree [12, 14]. We often call the second *s* technique a “higher-order differential attack.” A new technique to construct integral characteristics was proposed in 2015 [24], and it introduced a new property, the so-called “division property,” by generalizing the integral property [13]. It showed the propagation characteristic of the division property for any secret function restricted by an algebraic degree. As a result, several improved results were reported on the structural evaluation of the Feistel network and SPN.

**Our Contribution.** In [24], the focus is only on the secret S-box restricted by an algebraic degree. However, many realistic block ciphers use more efficient structures, e.g., a public S-box and a key addition. In this paper, we show that the division property becomes more useful if an S-box is a public function. Then, we apply our technique to the cryptanalysis on MISTY1. We first evaluate the propagation characteristic of the division property for public S-boxes  $S_7$  and  $S_9$  and show that  $S_7$  has a vulnerable property. We next evaluate the propagation characteristic of the division property for the *FI* function and then evaluate that for the *FO* function. Moreover, we evaluate that for the FL layer. Finally, we create an algorithm to search for integral characteristics on MISTY1 by

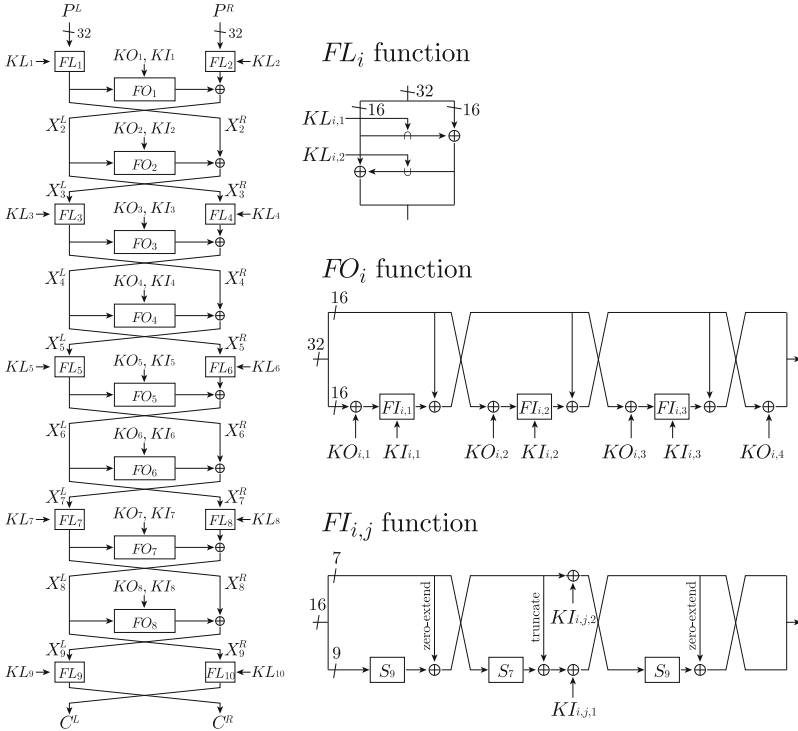


Fig. 1. Specification of MISTY1

assembling these propagation characteristics. As a result, we can construct a new 6-round integral characteristic, where the left 7-bit value of the output is balanced. We recover the round key by using the partial-sum technique [9]. As a result, the secret key of the full MISTY1 can be recovered with  $2^{63.58}$  chosen plaintexts and  $2^{121}$  time complexity. Moreover, if we can use  $2^{63.994}$  chosen plaintexts, the time complexity is reduced to  $2^{107.9}$ . Unfortunately, we have to use almost all chosen plaintexts, and recovering the secret key by using fewer chosen plaintexts is left as an open problem.

## 2 MISTY1

MISTY1 is a Feistel cipher whose F-function has the MISTY structure, and the recommended parameter is 8 rounds with 5 FL layers. Figure 1 shows the structure of MISTY1. Let  $X_i^L$  (resp.  $X_i^R$ ) be the left half (resp. the right half) of an  $i$ -round input. Moreover,  $X_i^L[j]$  (resp.  $X_i^R[j]$ ) denotes the  $j$ th bit of  $X_i^L$  (resp.  $X_i^R$ ) from the left. MISTY1 is a 64-bit block cipher, and the key-bit length is 128 bits. The component function  $FO_i$  consists of  $FL_{i,1}$ ,  $FL_{i,2}$ , and  $FL_{i,3}$ , and the four 16-bit round keys  $KO_{i,1}$ ,  $KO_{i,2}$ ,  $KO_{i,3}$ , and  $KO_{i,4}$  are used. The function  $FL_{i,j}$  consists of  $S_9$  and  $S_7$ , and a 16-bit round key  $KI_{i,j}$  is used. Here,  $S_9$  and  $S_7$  are defined in Appendix A. The component function  $FL_i$  uses two 16-bit round



keys,  $KL_{i,1}$  and  $KL_{i,2}$ . These round keys are calculated from the secret key  $(K_1, K_2, \dots, K_8)$  as

Symbol	$KO_{i,1}$	$KO_{i,2}$	$KO_{i,3}$	$KO_{i,4}$	$KI_{i,1}$	$KI_{i,2}$	$KI_{i,3}$	$KL_{i,1}$	$KL_{i,2}$
Key	$K_i$	$K_{i+2}$	$K_{i+7}$	$K_{i+4}$	$K'_{i+5}$	$K'_{i+1}$	$K'_{i+3}$	$K_{\frac{i+1}{2}}$ (odd $i$ ) $K'_{\frac{i}{2}+2}$ (even $i$ )	$K'_{\frac{i+1}{2}+6}$ (odd $i$ ) $K_{\frac{i}{2}+4}$ (even $i$ )

Here,  $K'_i$  is the output of  $FI_{i,j}$  where the input is  $K_i$  and the key is  $K_{i+1}$ .

### 3 Integral Characteristic by Division Property

#### 3.1 Notations

We make the distinction between the addition of  $\mathbb{F}_2^n$  and addition of  $\mathbb{Z}$ , and we use  $\oplus$  and  $+$  as the addition of  $\mathbb{F}_2^n$  and addition of  $\mathbb{Z}$ , respectively. For any  $a \in \mathbb{F}_2^n$ , the  $i$ th element is expressed in  $a[i]$ , and the Hamming weight  $w(a)$  is calculated as  $w(a) = \sum_{i=1}^n a[i]$ . Moreover,  $a[i, \dots, j]$  denotes a bit string whose elements are values described into square brackets. Let  $1^n \in \mathbb{F}_2^n$  be a value whose all elements are 1. Moreover, let  $0^n \in \mathbb{F}_2^n$  be a value whose all elements are 0.

For any  $\mathbf{a} \in (\mathbb{F}_2^{n_1} \times \mathbb{F}_2^{n_2} \times \dots \times \mathbb{F}_2^{n_m})$ , the vectorial Hamming weight of  $\mathbf{a}$  is defined as  $W(\mathbf{a}) = (w(a_1), w(a_2), \dots, w(a_m)) \in \mathbb{Z}^m$ . Moreover, for any  $\mathbf{k} \in \mathbb{Z}^m$  and  $\mathbf{k}' \in \mathbb{Z}^m$ , we define  $\mathbf{k} \succeq \mathbf{k}'$  if  $k_i \geq k'_i$  for all  $i$ . Otherwise,  $\vec{k} \not\prec \vec{k}'$ .

**Boolean Function.** A Boolean function is a function from  $\mathbb{F}_2^n$  to  $\mathbb{F}_2$ . Let  $\deg(f)$  be the algebraic degree of a Boolean function  $f$ . Algebraic Normal Form (ANF) is often used as representations of the Boolean function. Let  $f$  be any Boolean function from  $\mathbb{F}_2^n$  to  $\mathbb{F}_2$ , and it can be represented as

$$f(x) = \bigoplus_{u \in \mathbb{F}_2^n} a_u^f \left( \prod_{i=1}^n x[i]^{u[i]} \right),$$

where  $a_u^f \in \mathbb{F}_2$  is a constant value depending on  $f$  and  $u$ . If  $\deg(f)$  is at most  $d$ , all  $a_u^f$  satisfying  $d < w(u)$  are 0. An  $n$ -bit S-box can be regarded as the collection of  $n$  Boolean functions. If algebraic degrees of  $n$  Boolean functions are at most  $d$ , we say the algebraic degree of the S-box is at most  $d$ .

#### 3.2 Integral Attack

An integral attack is one of the most powerful cryptanalyses against block ciphers. Attackers prepare  $N$  chosen plaintexts and get the corresponding ciphertexts. If the XOR of all corresponding ciphertexts becomes 0, we say that the block cipher has an integral characteristic with  $N$  chosen plaintexts. In an integral attack, attackers first create an integral characteristic against a reduced-round block cipher. Then, they guess the round keys that are used in the last several rounds and calculate the XOR of the ciphertexts of the reduced-round block cipher. Finally, they evaluate whether or not the XOR becomes 0. If the XOR does not become 0, they can discard the guessed round keys from the candidates of the correct key.

### 3.3 Division Property

A division property, which was proposed in [24], is used to search for integral characteristics. We first prepare a set of plaintexts and evaluate the division property of the set. Then, we propagate the division property and evaluate the division property of the set of texts encrypted over one round. By repeating the propagation, we show the division property of the set of texts encrypted over some rounds. Finally, we can easily determine the existence of the integral characteristic from the propagated division property.

**Bit Product Function.** We first define two bit product functions  $\pi_u$  and  $\pi_{\mathbf{u}}$ , which are used to evaluate the division property of a multiset. Let  $\pi_u : \mathbb{F}_2^n \rightarrow \mathbb{F}_2$  be a function for any  $u \in \mathbb{F}_2^n$ . Let  $x \in \mathbb{F}_2^n$  be the input, and  $\pi_u(x)$  is the AND of  $x[i]$  satisfying  $u[i] = 1$ , i.e., it is defined as

$$\pi_u(x) := \prod_{i=1}^n x[i]^{u[i]}.$$

Let  $\pi_{\mathbf{u}} : (\mathbb{F}_2^{n_1} \times \mathbb{F}_2^{n_2} \times \cdots \times \mathbb{F}_2^{n_m}) \rightarrow \mathbb{F}_2$  be a function for any  $\mathbf{u} \in (\mathbb{F}_2^{n_1} \times \mathbb{F}_2^{n_2} \times \cdots \times \mathbb{F}_2^{n_m})$ . Let  $\mathbf{x} \in (\mathbb{F}_2^{n_1} \times \mathbb{F}_2^{n_2} \times \cdots \times \mathbb{F}_2^{n_m})$  be the input, and  $\pi_{\mathbf{u}}(\mathbf{x})$  is defined as

$$\pi_{\mathbf{u}}(\mathbf{x}) := \prod_{i=1}^m \pi_{u_i}(x_i).$$

**Definition of Division Property.** The division property is given against a multiset, and it is calculated by using the bit product function. Let  $\mathbb{X}$  be an input multiset whose elements take a value of  $(\mathbb{F}_2^{n_1} \times \mathbb{F}_2^{n_2} \times \cdots \times \mathbb{F}_2^{n_m})$ . In the division property, we first evaluate a value of  $\bigoplus_{\mathbf{x} \in \mathbb{X}} \pi_{\mathbf{u}}(\mathbf{x})$  for all  $\mathbf{u} \in (\mathbb{F}_2^{n_1} \times \mathbb{F}_2^{n_2} \times \cdots \times \mathbb{F}_2^{n_m})$ . Then, we divide the set of  $\mathbf{u}$  into a subset whose evaluated value becomes 0 and a subset whose evaluated value becomes unknown<sup>1</sup>. In [24], the focus was on using the Hamming weight of elements of  $\mathbf{u}$  to divide the set.

**Definition 1 (Division Property).** Let  $\mathbb{X}$  be a multiset whose elements take a value of  $(\mathbb{F}_2^{n_1} \times \mathbb{F}_2^{n_2} \times \cdots \times \mathbb{F}_2^{n_m})$ , and  $\mathbf{k}$  is an  $m$ -dimensional vector whose  $i$ th element takes a value between 0 and  $n_i$ . When the multiset  $\mathbb{X}$  has the division property  $\mathcal{D}_{\mathbf{k}^{(1)}, \mathbf{k}^{(2)}, \dots, \mathbf{k}^{(q)}}^{n_1, n_2, \dots, n_m}$ , it fulfils the following conditions: The parity of  $\pi_{\mathbf{u}}(\mathbf{x})$  over all  $\mathbf{x} \in \mathbb{X}$  is always even when

$$\mathbf{u} \in \left\{ (u_1, \dots, u_m) \in (\mathbb{F}_2^{n_1} \times \cdots \times \mathbb{F}_2^{n_m}) \mid W(\mathbf{u}) \not\geq \mathbf{k}^{(1)}, \dots, W(\mathbf{u}) \not\geq \mathbf{k}^{(q)} \right\}.$$

Moreover, the parity becomes unknown when  $\mathbf{u}$  is used such that there exists an  $i$  ( $1 \leq i \leq q$ ) satisfying  $W(\mathbf{u}) \geq \mathbf{k}^{(i)}$ .

<sup>1</sup> If we know all accurate values in a multiset, we can divide the set of  $\mathbf{u}$  into subsets whose evaluated value becomes 0 or 1. However, in the application to cryptanalysis, we evaluate the values in the multiset whose elements are texts encrypted for several rounds. Such elements change depending on the sub keys and the constant bit of plaintexts. Therefore, we consider the subset whose evaluated value becomes 0 or unknown.

Assume that the multiset  $\mathbb{X}$  has the division property  $\mathcal{D}_{\mathbf{k}^{(1)}, \mathbf{k}^{(2)}, \dots, \mathbf{k}^{(q)}}^{n_1, n_2, \dots, n_m}$ . If there exist  $\mathbf{k}^{(i)}$  such that  $k_j^{(i)}$  is greater than 1,  $\bigoplus_{x \in \mathbb{X}} x_j$  becomes 0. See [24] to better understand the concept in detail. Moreover, [22] shows an example, and it helps us understand the division property.

**Propagation Rules of Division Property.** Some propagation rules for the division property are proven in [24]. We summarize them as follows.

**Rule 1 (Substitution).** Let  $F$  be a function that consists of  $m$  S-boxes, where the bit length and the algebraic degree of the  $i$ th S-box is  $n_i$  bits and  $d_i$ , respectively. The input and the output take a value of  $(\mathbb{F}_2^{n_1} \times \mathbb{F}_2^{n_2} \times \dots \times \mathbb{F}_2^{n_m})$ , and  $\mathbb{X}$  and  $\mathbb{Y}$  denote the input multiset and the output multiset, respectively. Assuming that the multiset  $\mathbb{X}$  has the division property  $\mathcal{D}_{\mathbf{k}^{(1)}, \mathbf{k}^{(2)}, \dots, \mathbf{k}^{(q)}}^{n_1, n_2, \dots, n_m}$ , the division property of the multiset  $\mathbb{Y}$  is  $\mathcal{D}_{\mathbf{k}'^{(1)}, \mathbf{k}'^{(2)}, \dots, \mathbf{k}'^{(q)}}^{n_1, n_2, \dots, n_m}$  as

$$k_i^{(j)} = \left\lceil \frac{k_i^{(j)}}{d_i} \right\rceil \quad \text{for } 1 \leq i \leq m, \quad 1 \leq j \leq q.$$

**Rule 2 (Copy).** Let  $F$  be a copy function, where the input  $x$  takes a value of  $\mathbb{F}_2^n$  and the output is calculated as  $(y_1, y_2) = (x, x)$ . Let  $\mathbb{X}$  and  $\mathbb{Y}$  be the input multiset and output multiset, respectively. Assuming that the multiset  $\mathbb{X}$  has the division property  $\mathcal{D}_{\mathbf{k}}^n$ , the division property of the multiset  $\mathbb{Y}$  is  $\mathcal{D}_{\mathbf{k}'^{(1)}, \mathbf{k}'^{(2)}, \dots, \mathbf{k}'^{(k+1)}}^{n, n}$  as

$$k'^{(i+1)} = (k - i, i) \quad \text{for } 0 \leq i \leq k.$$

**Rule 3 (Compression by XOR).** Let  $F$  be a function compressed by an XOR, where the input  $(x_1, x_2)$  takes a value of  $(\mathbb{F}_2^n \times \mathbb{F}_2^n)$  and the output is calculated as  $y = x_1 \oplus x_2$ . Let  $\mathbb{X}$  and  $\mathbb{Y}$  be the input multiset and output multiset, respectively. Assuming that the multiset  $\mathbb{X}$  has the division property  $\mathcal{D}_{\mathbf{k}^{(1)}, \mathbf{k}^{(2)}, \dots, \mathbf{k}^{(q)}}^{n, n}$ , the division property of the multiset  $\mathbb{Y}$  is  $\mathcal{D}_{\mathbf{k}'}^n$  as

$$k' = \min\{k_1^{(1)} + k_2^{(1)}, k_1^{(2)} + k_2^{(2)}, \dots, k_1^{(q)} + k_2^{(q)}\}.$$

Here, if the minimum value of  $k'$  is larger than  $n$ , the propagation characteristic of the division property is aborted. Namely, a value of  $\bigoplus_{y \in \mathbb{Y}} \pi_v(y)$  is 0 for all  $v \in \mathbb{F}_2^n$ .

**Rule 4 (Split).** Let  $F$  be a split function, where the input  $x$  takes a value of  $\mathbb{F}_2^n$  and the output is calculated as  $x = y_1 \| y_2$ , where  $(y_1, y_2)$  takes a value of  $(\mathbb{F}_2^{n_1} \times \mathbb{F}_2^{n-n_1})$ . Let  $\mathbb{X}$  and  $\mathbb{Y}$  be the input multiset and output multiset, respectively. Assuming that the multiset  $\mathbb{X}$  has the division property  $\mathcal{D}_{\mathbf{k}}^n$ , the division property of the multiset  $\mathbb{Y}$  is  $\mathcal{D}_{\mathbf{k}'^{(1)}, \mathbf{k}'^{(2)}, \dots, \mathbf{k}'^{(q)}}^{n_1, n-n_1}$  as

$$\mathbf{k}'^{(i+1)} = (k - i, i) \quad \text{for } 0 \leq i \leq k.$$

Here,  $(k - i)$  is less than or equal to  $n_1$ , and  $i$  is less than or equal to  $n - n_1$ .



**Fig. 2.** The difference between [24] and us. The left figure is an assumption used in [24]. The right one is a new assumption used in this paper.

**Rule 5 (Concatenation).** Let  $F$  be a concatenation function, where the input  $(x_1, x_2)$  takes a value of  $(\mathbb{F}_2^{n_1} \times \mathbb{F}_2^{n_2})$  and the output is calculated as  $y = x_1 || x_2$ . Let  $\mathbb{X}$  and  $\mathbb{Y}$  be the input multiset and output multiset, respectively. Assuming that the multiset  $\mathbb{X}$  has the division property  $\mathcal{D}_{\mathbf{k}^{(1), \mathbf{k}^{(2)}, \dots, \mathbf{k}^{(q)}}}^{n_1, n_2}$ , the division property of the multiset  $\mathbb{Y}$  is  $\mathcal{D}_{k'}^{n_1 + n_2}$  as

$$k' = \min\{k_1^{(1)} + k_2^{(1)}, k_1^{(2)} + k_2^{(2)}, \dots, k_1^{(q)} + k_2^{(q)}\}.$$

### 4 Division Property for Public Function

In an assumption of [24], attackers cannot know the specification of an S-box and only know the algebraic degree of the S-box. However, many specific block ciphers usually use a public S-box and an addition of secret sub keys, where an XOR is especially used for the addition. In this paper, we show that the propagation characteristic of the division property can be improved if an S-box is a public function. The difference between [24] and us is shown in Fig. 2.

We consider the propagation characteristic of the division property against the function shown in the right figure in Fig. 2. The key XORing first be applied, but it does not affect the division property because it is a linear function. Therefore, when we evaluate the propagation characteristic of the division property, we can remove the key XORing. Next, a public S-box is applied, and we can determine the ANF of the S-box. Assuming that an S-box is a function from  $n$  bits to  $m$  bits, the ANF is represented as

$$\begin{aligned} y[1] &= f_1(x[1], x[2], \dots, x[n]), \\ y[2] &= f_2(x[1], x[2], \dots, x[n]), \\ &\vdots \\ y[m] &= f_m(x[1], x[2], \dots, x[n]), \end{aligned}$$

where  $x[i]$  ( $1 \leq i \leq n$ ) is an input,  $y[j]$  ( $1 \leq j \leq m$ ) is an output, and  $f_j$  ( $1 \leq j \leq m$ ) is a Boolean function. The division property evaluates the input multiset and output one by using the bit product function  $\pi_u$ , and we then divide the set of  $u$  into a subset whose evaluated value becomes 0 and a subset whose evaluated value becomes unknown. Namely, we evaluate the equation

$$F_u(x[1], x[2], \dots, x[n]) = \prod_{i=1}^m f_i(x[1], x[2], \dots, x[n])^{u[i]}$$

and divide the set of  $u$ . In [24], a fundamental property of the product of some functions is used, i.e., the algebraic degree of  $F_u$  is at most  $w(u) \times d$  if the algebraic degree of functions  $f_i$  is at most  $d$ . However, since we now know the ANF of functions  $f_1, f_2, \dots, f_m$ , we can calculate the accurate algebraic degree of  $F_u$  for all  $u \in \mathbb{F}_2^n$ . In this case, if the algebraic degree of  $F_u$  is less than  $w(u) \times d$  for all  $u$  for which  $w(u)$  is constant, we can improve the propagation characteristic.

### 4.1 Application to MISTY S-boxes

**Evaluation of  $S_7$ .** The  $S_7$  of MISTY is a 7-bit S-box with degree 3. We show the ANF of  $S_7$  in Appendix A. We evaluate the property of  $(\pi_v \circ S_7)$  to get the propagation characteristic of the division property. The algebraic degree of  $(\pi_v \circ S_7)$  increases in accordance with the Hamming weight of  $v$ , and it is summarized as follows.

$w(v)$	0	1	2	3	4	5	6	7
degree	0	3	5	5	6	6	6	7

If we replace the  $S_7$  with a modified S-box, which is randomly chosen from all 7-bit S-boxes with degree 3, the algebraic degree of  $(\pi_v \circ S)$  is at least 6 with  $w(v) \geq 2$ . However, for the  $S_7$ , the increment of the algebraic degree is bounded by 5 with  $w(v) = 2$  or  $w(v) = 3$  holds<sup>2</sup>. Thus, the propagation characteristic is represented as the following.

$\mathcal{D}_k^7$ for input set $\mathbb{X}$	$\mathcal{D}_0^7$	$\mathcal{D}_1^7$	$\mathcal{D}_2^7$	$\mathcal{D}_3^7$	$\mathcal{D}_4^7$	$\mathcal{D}_5^7$	$\mathcal{D}_6^7$	$\mathcal{D}_7^7$
$\mathcal{D}_k^7$ for output set $\mathbb{Y}$	$\mathcal{D}_0^7$	$\mathcal{D}_1^7$	$\mathcal{D}_1^7$	$\mathcal{D}_1^7$	$\mathcal{D}_2^7$	$\mathcal{D}_2^7$	$\mathcal{D}_4^7$	$\mathcal{D}_7^7$

Notice that the division property  $\mathcal{D}_4^7$  is propagated from the division property  $\mathcal{D}_6^7$ . Assuming that the modified S-box is applied, the division property  $\mathcal{D}_2^7$  is propagated from the division property  $\mathcal{D}_6^7$  [24]. Therefore, the deterioration of the division property for the  $S_7$  is smaller than that for any 7-bit S-box.

**Evaluation of  $S_9$ .** The  $S_9$  of MISTY is a 9-bit S-box with degree 2. We show the ANF of  $S_7$  in Appendix A. We evaluate the property of  $(\pi_v \circ S_9)$  to get the propagation characteristic of the division property. The algebraic degree of  $(\pi_v \circ S_9)$  increases in accordance with the Hamming weight of  $v$ , and it is summarized as follows.

$w(v)$	0	1	2	3	4	5	6	7	8	9
degree	0	2	4	6	8	8	8	8	8	9

<sup>2</sup> This observation was also provided by Theorem 3.1 in [4].

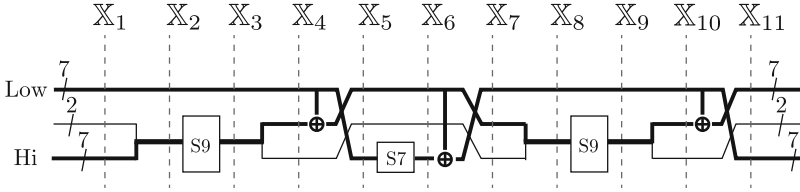


Fig. 3. Structure of *FI* function

Thus, the propagation characteristic is represented as

$\mathcal{D}_k^9$ for input set $\mathbb{X}$	$\mathcal{D}_0^9$	$\mathcal{D}_1^9$	$\mathcal{D}_2^9$	$\mathcal{D}_3^9$	$\mathcal{D}_4^9$	$\mathcal{D}_5^9$	$\mathcal{D}_6^9$	$\mathcal{D}_7^9$	$\mathcal{D}_8^9$	$\mathcal{D}_9^9$
$\mathcal{D}_k^9$ for output set $\mathbb{Y}$	$\mathcal{D}_0^9$	$\mathcal{D}_1^9$	$\mathcal{D}_1^9$	$\mathcal{D}_2^9$	$\mathcal{D}_2^9$	$\mathcal{D}_3^9$	$\mathcal{D}_3^9$	$\mathcal{D}_4^9$	$\mathcal{D}_4^9$	$\mathcal{D}_9^9$

Unlike the propagation characteristic of the division property for  $S_7$ , that for  $S_9$  is the same as that for any 9-bit S-box with degree 2.

### 5 New Integral Characteristic

This section shows how to create integral characteristics on MISTY1 by using the propagation characteristic of the division property. We first evaluate the propagation characteristic for the component functions of MISTY1, i.e., the *FI* function, the *FO* function, and the FL layer. Finally, by assembling these characteristics, we create an algorithm to search for integral characteristics on MISTY1.

#### 5.1 Division Property for *FI* Function

We evaluate the propagation characteristic of the division property for the *FI* function by using those for MISTY S-boxes shown in Sect. 4.1. Since there are a zero-extended XOR and a truncated XOR in the *FI* function, we use a new representation, in which the internal state is expressed in two 7-bit values and one 2-bit value. Figure 3 shows the structure of the *FI* function with our representation, where we remove the XOR of sub keys because it does not affect the division property.

Let  $\mathbb{X}_1$  be the input multiset of the *FI* function. We define every multiset  $\mathbb{X}_2, \mathbb{X}_3, \dots, \mathbb{X}_{11}$  in Fig. 3. Here, elements of the multiset  $\mathbb{X}_1, \mathbb{X}_5, \mathbb{X}_6,$  and  $\mathbb{X}_{11}$  take a value of  $(\mathbb{F}_2^7 \times \mathbb{F}_2^2 \times \mathbb{F}_2^7)$ . Elements of the multiset  $\mathbb{X}_2, \mathbb{X}_3, \mathbb{X}_8,$  and  $\mathbb{X}_9$  take a value of  $(\mathbb{F}_2^9 \times \mathbb{F}_2^7)$ . Elements of the multiset  $\mathbb{X}_4, \mathbb{X}_7,$  and  $\mathbb{X}_{10}$  take a value of  $(\mathbb{F}_2^2 \times \mathbb{F}_2^7 \times \mathbb{F}_2^7)$ . Since elements of  $\mathbb{X}_1$  and  $\mathbb{X}_{11}$  take a value of  $(\mathbb{F}_2^7 \times \mathbb{F}_2^2 \times \mathbb{F}_2^7)$ , the propagation for the *FI* function is calculated on  $\mathcal{D}_{\mathbf{k}^{(1), \mathbf{k}^{(2)}, \dots, \mathbf{k}^{(q)}}}^{7, 2, 7}$ . Here, the propagation is calculated with the following steps.

**From  $\mathbb{X}_1$  to  $\mathbb{X}_2$ :** A 9-bit value is created by concatenating the first 7-bit value with the second 2-bit value. The propagation characteristic can be evaluated by using Rule 5.

**From  $\mathbb{X}_2$  to  $\mathbb{X}_3$ :** The 9-bit S-box  $S_9$  is applied to the first 9-bit value. The propagation characteristic can be evaluated by using Rule 1.

**Algorithm 1.** Propagation for *FI* function

---

```

1: procedure FIEval( $k_1, k_2, k_3$ )
2:    $\mathbf{k}^{(1)}, \mathbf{k}^{(2)}, \dots, \mathbf{k}^{(q)} \leftarrow \text{S9Eval}(\mathbf{k})$   $\triangleright \mathbb{X}_1 \rightarrow \mathbb{X}_5$ 
3:    $\mathbf{k}^{(1)}, \mathbf{k}^{(2)}, \dots, \mathbf{k}^{(q)} \leftarrow \text{S7Eval}(\mathbf{k}^{(1)}, \mathbf{k}^{(2)}, \dots, \mathbf{k}^{(q)})$   $\triangleright \mathbb{X}_5 \rightarrow \mathbb{X}_7$ 
4:    $\mathbf{k}^{(1)}, \mathbf{k}^{(2)}, \dots, \mathbf{k}^{(q)} \leftarrow \text{S9Eval}(\mathbf{k}^{(1)}, \mathbf{k}^{(2)}, \dots, \mathbf{k}^{(q)})$   $\triangleright \mathbb{X}_7 \rightarrow \mathbb{X}_{11}$ 
5:   return SizeReduce( $\mathbf{k}^{(1)}, \mathbf{k}^{(2)}, \dots, \mathbf{k}^{(q)}$ )
6: end procedure

1: procedure S9Eval( $\mathbf{k}^{(1)}, \dots, \mathbf{k}^{(q)}$ )
2:    $q' \leftarrow 0$ 
3:   for  $i \leftarrow 1$  to  $q$  do
4:      $(\ell, c, r) \leftarrow (k_1^{(i)}, k_2^{(i)}, k_3^{(i)})$ 
5:      $k \leftarrow \ell + c$ 
6:     if  $k < 9$  then
7:        $k \leftarrow \lceil k/2 \rceil$ 
8:     end if
9:     for  $c' \leftarrow 0$  to  $\min(2, k)$  do
10:      for  $x \leftarrow 0$  to  $r$  do
11:         $\ell' \leftarrow r - x$ 
12:         $r' \leftarrow k - c' + x$ 
13:        if  $r' \leq 7$  then
14:           $q' \leftarrow q' + 1$ 
15:           $\mathbf{k}'^{(q')} \leftarrow (\ell', c', r')$ 
16:        end if
17:      end for
18:    end for
19:  end for
20:  return  $\mathbf{k}'^{(1)}, \mathbf{k}'^{(2)}, \dots, \mathbf{k}'^{(q')}$ 
21: end procedure

22: procedure S7Eval( $\mathbf{k}^{(1)}, \dots, \mathbf{k}^{(q)}$ )
23:    $q' \leftarrow 0$ 
24:   for  $i \leftarrow 1$  to  $q$  do
25:      $(\ell, c, r) \leftarrow (k_1^{(i)}, k_2^{(i)}, k_3^{(i)})$ 
26:      $k \leftarrow \ell$ 
27:     if  $k = 6$  then
28:        $k \leftarrow 4$ 
29:     else if  $k < 6$  then
30:        $k \leftarrow \lceil k/3 \rceil$ 
31:     end if
32:     for  $x \leftarrow 0$  to  $r$  do
33:        $\ell' \leftarrow c$ 
34:        $c' \leftarrow r - x$ 
35:        $r' \leftarrow k + x$ 
36:       if  $r' \leq 7$  then
37:          $q' \leftarrow q' + 1$ 
38:          $\mathbf{k}'^{(q')} \leftarrow (\ell', c', r')$ 
39:       end if
40:     end for
41:   end for
42:   return  $\mathbf{k}'^{(1)}, \mathbf{k}'^{(2)}, \dots, \mathbf{k}'^{(q')}$ 
43: end procedure

```

---

**From  $\mathbb{X}_3$  to  $\mathbb{X}_4$ :** The 9-bit output value is split into a 2-bit value and a 7-bit value. The propagation characteristic can be evaluated by using Rule 4.

**From  $\mathbb{X}_4$  to  $\mathbb{X}_5$ :** The second 7-bit value is XORed with the last 7-bit value, and then, the order is rotated. The propagation characteristic can be evaluated by using Rule 2 and Rule 3.

**From  $\mathbb{X}_5$  to  $\mathbb{X}_6$ :** The 7-bit S-box  $S_7$  is applied to the first 7-bit value. The propagation characteristic can be evaluated by using Rule 1.

**From  $\mathbb{X}_6$  to  $\mathbb{X}_7$ :** The first 7-bit value is XORed with the last 7-bit value, and then, the order is rotated. The propagation characteristic can be evaluated by using Rule 2 and Rule 3.

**From  $\mathbb{X}_7$  to  $\mathbb{X}_8$ :** A 9-bit value is created by concatenating the first 2-bit value with the second 7-bit value. The propagation characteristic can be evaluated by using Rule 5.

**From  $\mathbb{X}_8$  to  $\mathbb{X}_{11}$ :** The propagation characteristic is the same as that from  $\mathbb{X}_2$  to  $\mathbb{X}_5$ .

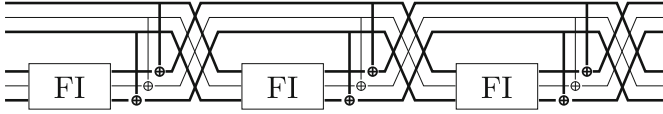


Fig. 4. Structure of  $FO$  function

Algorithm 1 creates the propagation characteristic table for the  $FI$  function. It calls `SizeReduce`, where redundant elements are eliminated, i.e., it eliminates  $\mathbf{k}^{(i)}$  if there exists  $j$  satisfying  $\mathbf{k}^{(i)} \succeq \mathbf{k}^{(j)}$ . Algorithm 1 only creates the propagation characteristic table for which the input property is represented by  $\mathcal{D}_{\mathbf{k}}^{7,2,7}$ . If any input multiset is evaluated, we need to know the propagation characteristic of  $\mathcal{D}_{\mathbf{k}^{(1)}, \mathbf{k}^{(2)}, \dots, \mathbf{k}^{(q)}}^{7,2,7}$ . However, we do not evaluate such propagation in advance because it can easily be evaluated by the table for which the input property is represented by  $\mathcal{D}_{\mathbf{k}}^{7,2,7}$ . We create all propagation characteristic tables by implementing Algorithm 1 and experimentally confirm that Algorithm 1 creates the correct tables.

## 5.2 Division Property for $FO$ Function

We next evaluate the propagation characteristic of the division property for the  $FO$  function by using the propagation characteristic table of the  $FI$  function. Figure 4 shows the structure of the  $FO$  function, where we remove the XOR of sub keys because it does not affect the division property. The input and output of the  $FO$  function take the value of  $(\mathbb{F}_2^7 \times \mathbb{F}_2^2 \times \mathbb{F}_2^2 \times \mathbb{F}_2^7 \times \mathbb{F}_2^2 \times \mathbb{F}_2^7)$ . Therefore, the propagation for the  $FO$  function is calculated on  $\mathcal{D}_{\mathbf{k}^{(1)}, \mathbf{k}^{(2)}, \dots, \mathbf{k}^{(q)}}^{7,2,7,7,2,7}$ .

Similar to that for the  $FI$  function, we create the propagation characteristic table for the  $FO$  function (see Algorithm 2). We create only a table for which the input property is represented by  $\mathcal{D}_{\mathbf{k}}^{7,2,7,7,2,7}$  and the output property is represented by  $\mathcal{D}_{\mathbf{k}^{(1)}, \mathbf{k}^{(2)}, \dots, \mathbf{k}^{(q)}}^{7,2,7,7,2,7}$ .

## 5.3 Division Property for $FL$ Layer

MISTY1 has the  $FL$  layer, which consists of two  $FL$  functions and is applied once every two rounds. In the  $FL$  function, the right half of the input is XORed with the AND between the left half and a sub key  $KL_{i,1}$ . Then, the left half of the input is XORed with the OR between the right half and a sub key  $KL_{i,2}$ .

Since the input and the output of the  $FL$  function take the value of  $\mathbb{F}_2^7 \times \mathbb{F}_2^2 \times \mathbb{F}_2^7 \times \mathbb{F}_2^7 \times \mathbb{F}_2^2 \times \mathbb{F}_2^7$ , the propagation for the  $FL$  function is calculated on  $\mathcal{D}_{\mathbf{k}^{(1)}, \mathbf{k}^{(2)}, \dots, \mathbf{k}^{(q)}}^{7,2,7,7,2,7}$ . `FlEval` in Algorithm 3 calculates the propagation characteristic table for the  $FL$  function, where `SizeReduce` eliminates  $\mathbf{k}^{(i)}$  if there exists  $j$  satisfying  $\mathbf{k}^{(i)} \succeq \mathbf{k}^{(j)}$ . Moreover, the  $FL$  layer consists of two  $FL$  functions. Therefore, we have to consider the propagation characteristic of the division property  $\mathcal{D}_{\mathbf{k}}^{7,2,7,7,2,7,7,2,7,7,2,7}$ , where each  $FL$  function is applied to the left half



**Algorithm 2.** Propagation for *FO* function

---

```

1: procedure FOEval( $k_1, k_2, k_3, k_4, k_5, k_6$ )
2:    $\mathbf{k}^{(1)}, \mathbf{k}^{(2)}, \dots, \mathbf{k}^{(q)} \leftarrow \text{FORound}(\mathbf{k})$ 
3:    $\mathbf{k}^{(1)}, \mathbf{k}^{(2)}, \dots, \mathbf{k}^{(q)} \leftarrow \text{FORound}(\mathbf{k}^{(1)}, \mathbf{k}^{(2)}, \dots, \mathbf{k}^{(q)})$ 
4:    $\mathbf{k}^{(1)}, \mathbf{k}^{(2)}, \dots, \mathbf{k}^{(q)} \leftarrow \text{FORound}(\mathbf{k}^{(1)}, \mathbf{k}^{(2)}, \dots, \mathbf{k}^{(q)})$ 
5:   return SizeReduce( $\mathbf{k}^{(1)}, \mathbf{k}^{(2)}, \dots, \mathbf{k}^{(q)}$ )
6: end procedure
1: procedure FORound( $\mathbf{k}^{(1)}, \mathbf{k}^{(2)}, \dots, \mathbf{k}^{(q)}$ )
2:    $q' \leftarrow 0$ 
3:   for  $i = 1$  to  $q$  do
4:      $\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \dots, \mathbf{y}^{(q_y)} \leftarrow \text{FIEval}(k_1^{(i)}, k_2^{(i)}, k_3^{(i)})$ 
5:     for  $j = 1$  to  $q_y$  do
6:       for all  $\mathbf{x}$  s.t.  $(x_1 \leq k_4^{(i)}) \wedge (x_2 \leq k_5^{(i)}) \wedge (x_3 \leq k_6^{(i)})$  do
7:          $\mathbf{k}' \leftarrow (k_4^{(i)} - x_1, k_5^{(i)} - x_2, k_6^{(i)} - x_3, y_1^{(j)} + x_1, y_2^{(j)} + x_2, y_3^{(j)} + x_3)$ 
8:         if  $(k_4' \leq 7) \wedge (k_5' \leq 2) \wedge (k_6' \leq 7)$  then
9:            $q' \leftarrow q' + 1$ 
10:           $\mathbf{k}'^{(q')} \leftarrow \mathbf{k}'$ 
11:         end if
12:       end for
13:     end for
14:   end for
15:   return  $\mathbf{k}'^{(1)}, \mathbf{k}'^{(2)}, \dots, \mathbf{k}'^{(q')}$ 
16: end procedure

```

---

and the right one. `FLayerEval` in Algorithm 3 calculates the propagation characteristic of the division property for the FL layer.

#### 5.4 Path Search for Integral Characteristic on MISTY1

We created the propagation characteristic table for the *FI* and *FO* functions in Sects. 5.1 and 5.2, respectively. Moreover, we showed the propagation characteristic for the FL layer in Sect. 5.3. By assembling these propagation characteristics, we create an algorithm to search for integral characteristics on MISTY1. Since the input and the output are represented as eight 7-bit values and four 2-bit values, the propagation is calculated on  $\mathcal{D}_{\mathbf{k}^{(1)}, \mathbf{k}^{(2)}, \dots, \mathbf{k}^{(q)}}^{7,2,7,7,2,7,7,2,7,2,7}$ .

The FL layer is first applied to plaintexts, and it deteriorates the propagation of the division property. Therefore, we first remove only the first FL layer and search for integral characteristics on MISTY1 without the first FL layer. The method for passing through the first FL layer is shown in the next paragraph. Algorithm 4 shows the search algorithm for integral characteristics on MISTY1 without the first FL layer.

As a result, we can construct 6-round integral characteristics without the first and last *FL* layers. Each characteristic uses  $2^{63}$  chosen plaintexts, where any one bit of the first seven bits is constant and the others take all values. Figure 5 shows the 6-round integral characteristic, where the bit strings labeled *B*, i.e., the first 7 bits and last 32 bits, are balanced. Notice that the 6-round characteristic

**Algorithm 3.** Propagation for FL layer

---

```

1: procedure FLEval( $k_1, k_2, \dots, k_6$ )
2:    $q' \leftarrow 0$ 
3:    $(\ell, c, r) \leftarrow (k_1 + k_4, k_2 + k_5, k_3 + k_6)$ 
4:   for  $k'_1 \leftarrow 0$  to  $\min(7, \ell)$  do
5:     for  $k'_2 \leftarrow 0$  to  $\min(2, c)$  do
6:       for  $k'_3 \leftarrow 0$  to  $\min(7, r)$  do
7:          $(k'_4, k'_5, k'_6) \leftarrow (\ell - k'_1, c - k'_2, r - k'_3)$ 
8:         if  $(k'_4 \leq 7) \wedge (k'_5 \leq 2) \wedge (k'_6 \leq 7)$  then
9:            $q' \leftarrow q' + 1$ 
10:           $\mathbf{k}'^{(q')} \leftarrow (k'_1, k'_2, k'_3, k'_4, k'_5, k'_6)$ 
11:         end if
12:       end for
13:     end for
14:   end for
15:   return SizeReduce( $\mathbf{k}^{(1)}, \mathbf{k}^{(2)}, \dots, \mathbf{k}^{(q')}$ )
16: end procedure
17: procedure FLlayerEval( $\mathbf{k}^{(1)}, \mathbf{k}^{(2)}, \dots, \mathbf{k}^{(q)}$ )
18:    $q' \leftarrow 0$ 
19:   for  $i \leftarrow 1$  to  $q$  do
20:      $\ell^{(1)}, \ell^{(2)}, \dots, \ell^{(q_\ell)} \leftarrow \text{FLEval}(k_1^{(i)}, k_2^{(i)}, \dots, k_6^{(i)})$ 
21:      $r^{(1)}, r^{(2)}, \dots, r^{(q_r)} \leftarrow \text{FLEval}(k_7^{(i)}, k_8^{(i)}, \dots, k_{12}^{(i)})$ 
22:     for  $j \leftarrow 1$  to  $q_\ell$  do
23:       for  $j' \leftarrow 1$  to  $q_r$  do
24:          $q' \leftarrow q' + 1$ 
25:          $\mathbf{k}'^{(q')} \leftarrow (\ell_1^{(j)}, \ell_2^{(j)}, \ell_3^{(j)}, \ell_4^{(j)}, \ell_5^{(j)}, \ell_6^{(j)}, r_1^{(j')}, r_2^{(j')}, r_3^{(j')}, r_4^{(j')}, r_5^{(j')}, r_6^{(j')})$ 
26:       end for
27:     end for
28:   end for
29:   return ( $\mathbf{k}'^{(1)}, \mathbf{k}'^{(2)}, \dots, \mathbf{k}'^{(q')}$ )
30: end procedure

```

---

becomes a 7-round characteristic if the FL layer that is inserted after the 6th round is removed. Compared with the previous 4-round characteristic [10, 25], our characteristic is improved by two rounds.

As shown in Sect. 4, the  $S_7$  of MISTY1 has the vulnerable property that  $\mathcal{D}_4^7$  is provided from  $\mathcal{D}_6^7$ . Interestingly, assuming that  $S_7$  does not have this property (change lines 27–31 in `S7Eval`), our algorithm cannot construct the 6-round characteristic.

We already know that MISTY1 has the 14th order differential characteristic, which is shown in [23], and the principle was also discussed in [1, 5]. We also evaluate the principle of the characteristic by using the propagation characteristic of the division property. As a result, we confirm that the characteristic always exists if each algebraic degree  $S_9$  and  $S_7$  is 2 and 3, respectively. This result implies that the existence of the 14th order differential characteristic is only derived from the algebraic degree of S-boxes. Namely, even if different S-boxes

**Algorithm 4.** Path search for  $r$ -round characteristics without first FL layer

---

```

1: procedure RoundFuncEval( $\mathbf{k}^{(1)}, \mathbf{k}^{(2)}, \dots, \mathbf{k}^{(q)}$ )
2:    $q' = 0$ 
3:   for  $i \leftarrow 1$  to  $q$  do
4:     for all  $\mathbf{x}$  s.t.  $x_j \leq k_j^{(i)}$  for all  $j = 1, 2, \dots, 6$  do
5:        $(r_1, r_2, r_3) \leftarrow (k_1^{(i)} - x_1, k_2^{(i)} - x_2, k_3^{(i)} - x_3)$ 
6:        $(r_4, r_5, r_6) \leftarrow (k_4^{(i)} - x_4, k_5^{(i)} - x_5, k_6^{(i)} - x_6)$ 
7:        $\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \dots, \mathbf{y}^{(q_y)} \leftarrow \text{FOEval}(x_1, x_2, x_3, x_4, x_5, x_6)$ 
8:       for  $i' \leftarrow 1$  to  $q_y$  do
9:          $(\ell_1, \ell_2, \ell_3) \leftarrow (k_7^{(i)} + y_1^{(i')}, k_8^{(i)} + y_2^{(i')}, k_9^{(i)} + y_3^{(i')})$ 
10:         $(\ell_4, \ell_5, \ell_6) \leftarrow (k_{10}^{(i)} + y_4^{(i')}, k_{11}^{(i)} + y_5^{(i')}, k_{12}^{(i)} + y_6^{(i')})$ 
11:        if  $\ell_{j'} \leq 7$  for  $j' \in \{1, 3, 4, 6\}$  and  $\ell_{j'} \leq 2$  for  $j' \in \{2, 5\}$  then
12:           $q' \leftarrow q' + 1$ 
13:           $\mathbf{k}'^{(q')} \leftarrow (\ell_1, \ell_2, \ell_3, \ell_4, \ell_5, \ell_6, r_1, r_2, r_3, r_4, r_5, r_6)$ 
14:        end if
15:      end for
16:    end for
17:  end for
18:  return SizeReduce( $\mathbf{k}'^{(1)}, \mathbf{k}'^{(2)}, \dots, \mathbf{k}'^{(q')}$ )
19: end procedure

1: procedure Misty1Eval( $k_1, k_2, \dots, k_{12}, r$ )
2:    $\mathbf{k}^{(1)}, \mathbf{k}^{(2)}, \dots, \mathbf{k}^{(q)} \leftarrow \text{RoundFuncEval}(\mathbf{k})$  ▷ 1st round
3:   for  $i = 1$  to  $r$  do
4:     if  $i$  is even then
5:        $\mathbf{k}^{(1)}, \mathbf{k}^{(2)}, \dots, \mathbf{k}^{(q)} \leftarrow \text{FLayerEval}(\mathbf{k}^{(1)}, \mathbf{k}^{(2)}, \dots, \mathbf{k}^{(q)})$  ▷ FL Layer
6:     end if
7:      $\mathbf{k}^{(1)}, \mathbf{k}^{(2)}, \dots, \mathbf{k}^{(q)} \leftarrow \text{RoundFuncEval}(\mathbf{k}^{(1)}, \mathbf{k}^{(2)}, \dots, \mathbf{k}^{(q)})$  ▷ (i+1)th round
8:   end for
9: end procedure

```

---

are chosen in  $S_7$  and  $S_9$ , the 14th order differential characteristic exists unless the algebraic degree increases.

**Passage of First FL Layer.** Our new characteristic removes the first FL layer. Therefore, we have to create a set of chosen plaintexts to construct integral characteristics by using guessed round keys  $KL_{1,1}$  and  $KL_{1,2}$ . Here, we have to carefully choose the set of chosen plaintexts to avoid the use of the full code book (see Figs. 6, 7, and 8). In every figure,  $A_i$  denotes for which we prepare an input set that  $i$  bits are active. As an example, we consider an integral characteristic for which the first one bit is constant and the remaining 63 bits are active. Since all bits of the right half are active, we focus only on the left half. We first guess that  $KL_{1,2}[1] = 1$ , and we then prepare the set of plaintexts like in Fig. 6. We next guess that  $(KL_{1,1}[1], KL_{1,2}[1]) = (0, 0)$ , and we then prepare the set of plaintexts like in Fig. 7. Moreover, we guess that  $(KL_{1,1}[1], KL_{1,2}[1]) = (1, 0)$ , and we then prepare the set of plaintexts like in Fig. 8. Their chosen plaintexts construct 6-round integral characteristics if the guessed key bits are correct.

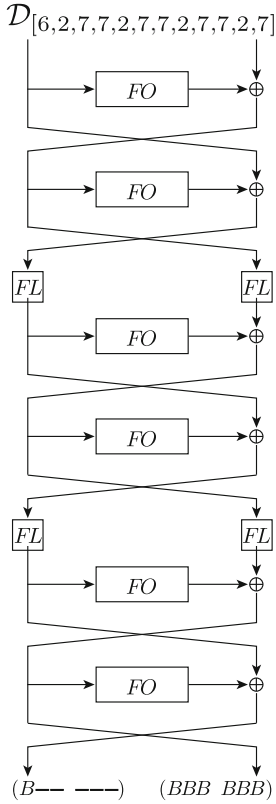


Fig. 5. New 6-round integral characteristic

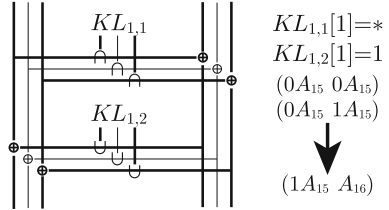


Fig. 6.  $KL_{1,2} = 1$

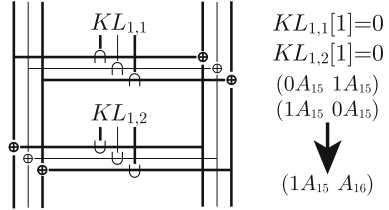


Fig. 7.  $KL_{1,1} = 0, KL_{1,2} = 0$

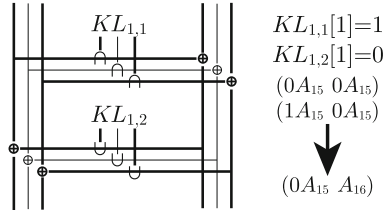


Fig. 8.  $KL_{1,1} = 1, KL_{1,2} = 0$

Notice that we do not use  $2^{62}$  chosen plaintexts as  $(1A_{15} 1A_{15} A_{16} A_{16})$ . Thus, our integral characteristics use  $2^{64} - 2^{62} \approx 2^{63.58}$  chosen plaintexts.

## 6 Key Recovery Using New Integral Characteristic

This section shows the key recovery step of our cryptanalysis, which uses the 6-round integral characteristic shown in Sect. 5. In the characteristic, the left 7-bit value of  $X_7^I$  is balanced. To evaluate this balanced seven bits, we have to calculate two FL layers and one FO function by using the guessed round keys. Figure 9 shows the structure of our key recovery step.

### 6.1 Sub Key Recovery Using Partial-Sum Technique

We guess  $KL_{1,1}[i](= K_1[i])$  and  $KL_{1,2}[i](= K'_7[i])$  and then prepare a set of chosen plaintexts to construct an integral characteristic. In the characteristic,

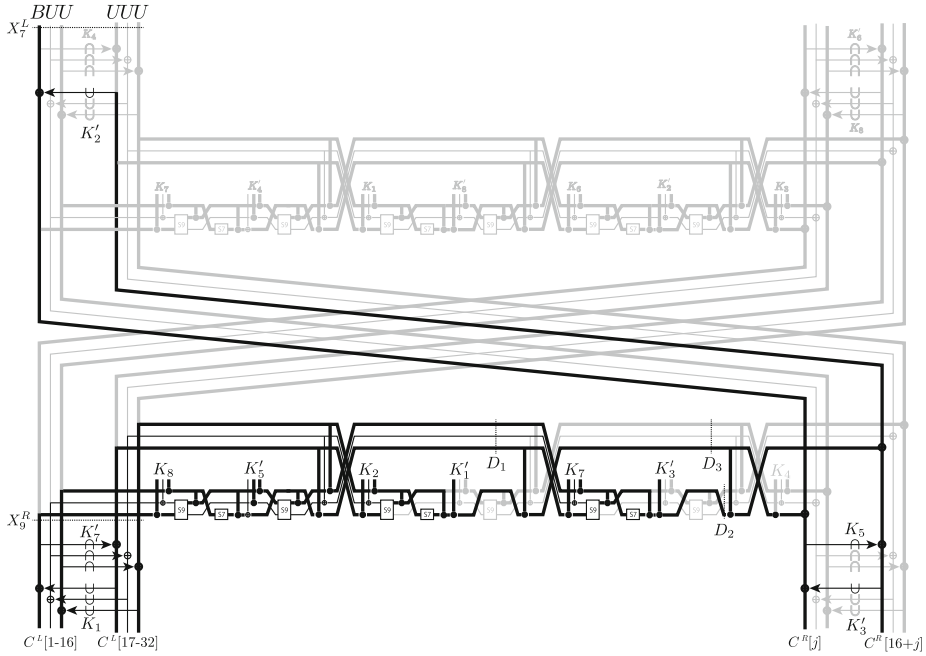


Fig. 9. Key recovery step

Table 2. Procedure of key recovery step

Step	Guessed key	#guessed total bits	New value	Discarded values	#texts	Values in set	Complexity
1		0			$2^{34}$	$C^L, C^R[j, 16 + j]$	
2	$K_1, K_7^L$	32	$X_9^R$	$C^L$	$2^{34}$	$X_9^R, C^R[j, 16 + j]$	$2^{34+32} = 2^{66}$
3	$K_8, K_5^L$	64	$D_1$	$X_9^R[1, \dots, 16]$	$2^{34}$	$D_1, X_9^R[17, \dots, 32], C^R[j, 16 + j]$	$2^{34+64} = 2^{98}$
4	$K_3^L[j], (K_7)$	65	$D_2[j]$	$D_1$ w/o $D_1[j]$	$2^{20}$	$D_1[j], D_2[j], X_9^R[17, \dots, 32], C^R[j, 16 + j]$	$2^{34+65} = 2^{99}$
5	$K_2, (K_1^L[j])$	81	$D_3[j]$	$X_9^R[17, \dots, 32], D_1[j]$	$2^4$	$D_2[j], D_3[j], C^R[j, 16 + j]$	$2^{20+81} = 2^{101}$
6	$K_5[j], K_2^L[j]$	83	$X_7^L[j]$	$D_2[j], D_3[j], C^R[j, 16 + j]$	$2^1$	$X_7^L[j]$	$2^{4+83} = 2^{87}$

seven bits  $X_7^L[1, \dots, 7]$  are balanced. Therefore, we evaluate whether or not  $X_7^L[j]$  is balanced for  $j \in \{1, 2, \dots, 7\}$  by using a partial-sum technique [9].

In the first step, we store the frequency of 34 bits ( $C^L, C^R[j, 16 + j]$ ) into a voting table for  $j \in \{1, 2, \dots, 7\}$ . Then, we partially guess round keys, discard the size of the voting table, and calculate the XOR of  $X_7^L[j]$ . Table 2 summarizes the procedure of the key recovery step, where every value is defined in Fig. 9. Since the time complexity is the sum of all steps, the time complexity is about  $2^{101.5}$ .

When we repeat the procedure for seven balanced bits, the time complexity becomes  $7 \times 2^{101.5} = 2^{104.3}$ .

The key recovery step has to guess the 124-bit key

$$K_1, K_2, K_5[1, \dots, 7], K_7, K_8, \\ K'_1[1, \dots, 7], K'_2[1, \dots, 7], K'_3[1, \dots, 7], K'_5, K'_7.$$

Here,  $K'_7$  and  $K'_1[1, \dots, 7]$  are uniquely determined by guessing  $K_7, K_8$  and  $K_1, K_2$ , respectively. Thus, the guessed key bit size is reduced to

$$K_1, K_2, K_5[1, \dots, 7], K_7, K_8, \\ K'_2[1, \dots, 7], K'_3[1, \dots, 7], K'_5,$$

and it becomes 101 bits. Moreover, since we already guessed 2 bits, i.e.,  $K_1[i]$  and  $K'_7[i]$ , to construct integral characteristics, the guessed key bit size is reduced to 99 bits. For wrong keys, the probability that  $X_7^L[1, \dots, 7]$  is balanced is  $2^{-7}$ . Therefore, the number of the candidates of round keys is reduced to  $2^{92}$ . Finally, we guess the 27 bits:

$$K_5[8, \dots, 16], K'_2[8, \dots, 16], K'_3[8, \dots, 16].$$

Notice that  $K_3, K_4$ , and  $K_6$  are uniquely determined from  $(K_2, K'_2)$ ,  $(K_3, K'_3)$ , and  $(K_5, K'_5)$ , respectively. Therefore, the total time complexity is  $2^{92+27} = 2^{119}$ . We guess the correct key from  $2^{119}$  candidates by using two plaintext-ciphertext pairs, and the time complexity is  $2^{119} + 2^{119-64} \approx 2^{119}$ . We have to execute the above procedure against  $(K_1[i], K'_7[i]) = (0, 0), (0, 1), (1, 0), (1, 1)$ , and the time complexity becomes  $4 \times 2^{119} = 2^{121}$ .

## 6.2 Trade-Off Between Time and Data Complexity

In Sect. 6.1, we use only one set of chosen plaintexts, where  $(2^{64} - 2^{62})$  chosen plaintexts are required. Since the probability that wrong keys are not discarded is  $2^{-7}$ , a brute-force search is required with a time complexity of  $2^{128-7} = 2^{119}$ , and it is larger than the time complexity of the partial-sum technique. Therefore, if we have a higher number of characteristics, the total time complexity can be reduced.

To prepare several characteristics, we choose some constant bits from seven bits ( $i \in \{1, 2, \dots, 7\}$ ). If we use a characteristic with  $i = 1$ , we use chosen plaintexts for which plaintext  $P^L$  takes the following values

$$(00A_{14} \ 00A_{14}), (00A_{14} \ 01A_{14}), (01A_{14} \ 00A_{14}), (01A_{14} \ 01A_{14}), \\ (00A_{14} \ 10A_{14}), (00A_{14} \ 11A_{14}), (01A_{14} \ 10A_{14}), (01A_{14} \ 11A_{14}), \\ (10A_{14} \ 00A_{14}), (10A_{14} \ 01A_{14}), (11A_{14} \ 00A_{14}), (11A_{14} \ 01A_{14}),$$

where  $A_{14}$  denotes that all values appear the same number independent of other bits, e.g.,  $(00A_{14} \ 00A_{14})$  uses  $2^{60}$  chosen plaintexts because  $P^R$  also takes all

**Table 3.** Trade-off between time and data complexity

#characteristics	Complexity for partial-sum	Complexity for brute-force	Total
1	$1 \times 3 \times 2^{104.3}$	$2^{121}$	$2^{121}$
2	$2 \times 3 \times 2^{104.3}$	$2^{114}$	$2^{114}$
3	$3 \times 3 \times 2^{104.3}$	$2^{107}$	$2^{108.3}$
4	$4 \times 3 \times 2^{104.3}$	$2^{100}$	$2^{107.9}$
5	$5 \times 3 \times 2^{104.3}$	$2^{93}$	$2^{108.2}$

values. Moreover, if we use a characteristic with  $i = 2$ , we use chosen plaintexts for which  $P^L$  takes the following values

$$\begin{aligned} & (00A_{14} \ 00A_{14}), (00A_{14} \ 10A_{14}), (10A_{14} \ 00A_{14}), (10A_{14} \ 10A_{14}), \\ & (00A_{14} \ 01A_{14}), (00A_{14} \ 11A_{14}), (10A_{14} \ 01A_{14}), (10A_{14} \ 11A_{14}), \\ & (01A_{14} \ 00A_{14}), (01A_{14} \ 10A_{14}), (11A_{14} \ 00A_{14}), (11A_{14} \ 10A_{14}). \end{aligned}$$

When both characteristics are used, they do not require choosing plaintexts for which  $P^L$  takes  $(11A_{14} \ 11A_{14})$ . Therefore,  $(2^{64} - 2^{60})$  chosen plaintexts are required, and the probability that wrong keys are not discarded becomes  $2^{-14}$ . Similarly, when three characteristics, which require  $(2^{64} - 2^{58})$  chosen plaintexts, are used, the probability that wrong keys are not discarded becomes  $2^{-21}$ .

Table 3 summarizes the trade-off between time and data complexity, and it shows that the use of four characteristics is optimized from the perspective of time complexity. Namely, when  $(2^{64} - 2^{56}) \approx 2^{63.994}$  chosen plaintexts are required, the time complexity to recovery the secret key is  $2^{107.3}$ .

## 7 Conclusions

In this paper, we showed a cryptanalysis of the full MISTY1. MISTY1 was well evaluated and standardized by several projects, such as CRYPTREC, ISO/IEC, and NESSIE. We constructed a new integral characteristic by using the propagation characteristic of the division property. Here, we improved the division property by optimizing a public S-box. As a result, a new 6-round integral characteristic is constructed, and we can recover the secret key of the full MISTY1 with  $2^{63.58}$  chosen plaintexts and  $2^{121}$  time complexity. If we can use  $2^{63.994}$  chosen plaintexts, our attack can recover the secret key with a time complexity of  $2^{107.9}$ .

## A MISTY S-boxes

The ANF of  $S_7$  is represented as

$$\begin{aligned} y[0] = & x[0] \oplus x[1]x[3] \oplus x[0]x[3]x[4] \oplus x[1]x[5] \oplus x[0]x[2]x[5] \oplus x[4]x[5] \\ & \oplus x[0]x[1]x[6] \oplus x[2]x[6] \oplus x[0]x[5]x[6] \oplus x[3]x[5]x[6] \oplus 1, \end{aligned}$$

$$\begin{aligned}
y[1] &= x[0]x[2] \oplus x[0]x[4] \oplus x[3]x[4] \oplus x[1]x[5] \oplus x[2]x[4]x[5] \oplus x[6] \oplus x[0]x[6] \\
&\quad \oplus x[3]x[6] \oplus x[2]x[3]x[6] \oplus x[1]x[4]x[6] \oplus x[0]x[5]x[6] \oplus 1, \\
y[2] &= x[1]x[2] \oplus x[0]x[2]x[3] \oplus x[4] \oplus x[1]x[4] \oplus x[0]x[1]x[4] \oplus x[0]x[5] \oplus x[0]x[4]x[5] \\
&\quad \oplus x[3]x[4]x[5] \oplus x[1]x[6] \oplus x[3]x[6] \oplus x[0]x[3]x[6] \oplus x[4]x[6] \oplus x[2]x[4]x[6], \\
y[3] &= x[0] \oplus x[1] \oplus x[0]x[1]x[2] \oplus x[0]x[3] \oplus x[2]x[4] \oplus x[1]x[4]x[5] \oplus x[2]x[6] \\
&\quad \oplus x[1]x[3]x[6] \oplus x[0]x[4]x[6] \oplus x[5]x[6] \oplus 1, \\
y[4] &= x[2]x[3] \oplus x[0]x[4] \oplus x[1]x[3]x[4] \oplus x[5] \oplus x[2]x[5] \oplus x[1]x[2]x[5] \oplus x[0]x[3]x[5] \\
&\quad \oplus x[1]x[6] \oplus x[1]x[5]x[6] \oplus x[4]x[5]x[6] \oplus 1, \\
y[5] &= x[0] \oplus x[1] \oplus x[2] \oplus x[0]x[1]x[2] \oplus x[0]x[3] \oplus x[1]x[2]x[3] \oplus x[1]x[4] \\
&\quad \oplus x[0]x[2]x[4] \oplus x[0]x[5] \oplus x[0]x[1]x[5] \oplus x[3]x[5] \oplus x[0]x[6] \oplus x[2]x[5]x[6], \\
y[6] &= x[0]x[1] \oplus x[3] \oplus x[0]x[3] \oplus x[2]x[3]x[4] \oplus x[0]x[5] \oplus x[2]x[5] \oplus x[3]x[5] \\
&\quad \oplus x[1]x[3]x[5] \oplus x[1]x[6] \oplus x[1]x[2]x[6] \oplus x[0]x[3]x[6] \oplus x[4]x[6] \oplus x[2]x[5]x[6].
\end{aligned}$$

Moreover, the ANF of  $S_9$  is represented as

$$\begin{aligned}
y[0] &= x[0]x[4] \oplus x[0]x[5] \oplus x[1]x[5] \oplus x[1]x[6] \oplus x[2]x[6] \oplus x[2]x[7] \oplus x[3]x[7] \oplus x[3]x[8] \\
&\quad \oplus x[4]x[8] \oplus 1, \\
y[1] &= x[0]x[2] \oplus x[3] \oplus x[1]x[3] \oplus x[2]x[3] \oplus x[3]x[4] \oplus x[4]x[5] \oplus x[0]x[6] \oplus x[2]x[6] \\
&\quad \oplus x[7] \oplus x[0]x[8] \oplus x[3]x[8] \oplus x[5]x[8] \oplus 1, \\
y[2] &= x[0]x[1] \oplus x[1]x[3] \oplus x[4] \oplus x[0]x[4] \oplus x[2]x[4] \oplus x[3]x[4] \oplus x[4]x[5] \oplus x[0]x[6] \\
&\quad \oplus x[5]x[6] \oplus x[1]x[7] \oplus x[3]x[7] \oplus x[8], \\
y[3] &= x[0] \oplus x[1]x[2] \oplus x[2]x[4] \oplus x[5] \oplus x[1]x[5] \oplus x[3]x[5] \oplus x[4]x[5] \oplus x[5]x[6] \\
&\quad \oplus x[1]x[7] \oplus x[6]x[7] \oplus x[2]x[8] \oplus x[4]x[8], \\
y[4] &= x[1] \oplus x[0]x[3] \oplus x[2]x[3] \oplus x[0]x[5] \oplus x[3]x[5] \oplus x[6] \oplus x[2]x[6] \oplus x[4]x[6] \\
&\quad \oplus x[5]x[6] \oplus x[6]x[7] \oplus x[2]x[8] \oplus x[7]x[8], \\
y[5] &= x[2] \oplus x[0]x[3] \oplus x[1]x[4] \oplus x[3]x[4] \oplus x[1]x[6] \oplus x[4]x[6] \oplus x[7] \oplus x[3]x[7] \\
&\quad \oplus x[5]x[7] \oplus x[6]x[7] \oplus x[0]x[8] \oplus x[7]x[8], \\
y[6] &= x[0]x[1] \oplus x[3] \oplus x[1]x[4] \oplus x[2]x[5] \oplus x[4]x[5] \oplus x[2]x[7] \oplus x[5]x[7] \oplus x[8] \\
&\quad \oplus x[0]x[8] \oplus x[4]x[8] \oplus x[6]x[8] \oplus x[7]x[8] \oplus 1, \\
y[7] &= x[1] \oplus x[0]x[1] \oplus x[1]x[2] \oplus x[2]x[3] \oplus x[0]x[4] \oplus x[5] \oplus x[1]x[6] \oplus x[3]x[6] \\
&\quad \oplus x[0]x[7] \oplus x[4]x[7] \oplus x[6]x[7] \oplus x[1]x[8] \oplus 1, \\
y[8] &= x[0] \oplus x[0]x[1] \oplus x[1]x[2] \oplus x[4] \oplus x[0]x[5] \oplus x[2]x[5] \oplus x[3]x[6] \oplus x[5]x[6] \\
&\quad \oplus x[0]x[7] \oplus x[0]x[8] \oplus x[3]x[8] \oplus x[6]x[8] \oplus 1.
\end{aligned}$$

## References

1. Babbage, S., Frisch, L.: On MISTY1 higher order differential cryptanalysis. In: Won, D. (ed.) ICISC 2000. LNCS, vol. 2015, pp. 22–36. Springer, Heidelberg (2001)
2. Bar-On, A.: Improved higher-order differential attacks on MISTY1. In: FSE (2015)
3. 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)
4. Boura, C., Canteaut, A.: On the influence of the algebraic degree of  $f^1$  on the algebraic degree of  $G \circ F$ . IEEE Trans. Inf. Theor. **59**(1), 691–702 (2013)
5. Canteaut, A., Videau, M.: Degree of composition of highly nonlinear functions and applications to higher order differential cryptanalysis. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 518–533. Springer, Heidelberg (2002)



6. CRYPTREC: Specifications of e-government recommended ciphers (2013). <http://www.cryptrec.go.jp/english/method.html>
7. Daemen, J., Knudsen, L.R., Rijmen, V.: The block cipher SQUARE. In: Biham, E. (ed.) FSE 1997. LNCS, vol. 1267, pp. 149–165. Springer, Heidelberg (1997)
8. Dunkelman, O., Keller, N.: An improved impossible differential attack on MISTY1. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 441–454. Springer, Heidelberg (2008)
9. Ferguson, N., Kelsey, J., Lucks, S., Schneier, B., Stay, M., Wagner, D., Whiting, D.L.: Improved cryptanalysis of Rijndael. In: Schneier, B. (ed.) FSE 2000. LNCS, vol. 1978, pp. 213–230. Springer, Heidelberg (2001)
10. Hatano, Y., Tanaka, H., Kaneko, T.: Optimization for the algebraic method and its application to an attack of MISTY1. IEICE Trans. **87–A**(1), 18–27 (2004)
11. ISO/IEC: JTC1: ISO/IEC 18033: Security techniques – encryption algorithms – part 3: Block ciphers (2005)
12. Knudsen, L.R.: Truncated and higher order differentials. In: Preneel, B. (ed.) FSE 1994. LNCS, vol. 1008, pp. 196–211. Springer, Heidelberg (1995)
13. Knudsen, L.R., Wagner, D.: Integral cryptanalysis. In: Daemen, J., Rijmen, V. (eds.) FSE 2002. LNCS, vol. 2365, pp. 112–127. Springer, Heidelberg (2002)
14. Lai, X.: Higher order derivatives and differential cryptanalysis. In: Blahut, R.E., Costello Jr., D.J., Maurer, U., Mittelholzer, T. (eds.) Communications and Cryptography. The Springer International Series in Engineering and Computer Science, vol. 276, pp. 227–233. Springer, USA (1994)
15. Matsui, M.: Linear cryptanalysis method for DES cipher. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 386–397. Springer, Heidelberg (1994)
16. Matsui, M.: New structure of block ciphers with provable security against differential and linear cryptanalysis. In: Gollmann, D. (ed.) FSE 1996. LNCS, vol. 1039, pp. 205–218. Springer, Heidelberg (1996)
17. Matsui, M.: New block encryption algorithm MISTY. In: Biham, E. (ed.) FSE 1997. LNCS, vol. 1267, pp. 54–68. Springer, Heidelberg (1997)
18. NESSIE: New european schemes for signatures, integrity, and encryption (2004). <https://www.cosic.esat.kuleuven.be/nessie/>
19. Nyberg, K.: Linear approximation of block ciphers. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 439–444. Springer, Heidelberg (1995)
20. Nyberg, K., Knudsen, L.R.: Provable security against a differential attack. J. Cryptology **8**(1), 27–37 (1995)
21. Ohta, H., Matsui, M.: A description of the MISTY1 encryption algorithm (2000). <https://tools.ietf.org/html/rfc2994>
22. Sun, B., Hai, X., Zhang, W., Cheng, L., Yang, Z.: New observation on division property. IACR Cryptology ePrint Archive 2015, 459 (2015). <http://eprint.iacr.org/2015/459>
23. Tanaka, H., Hisamatsu, K., Kaneko, T.: Strength of MISTY1 without FL function for higher order differential attack. In: Fossorier, M.P.C., Imai, H., Lin, S., Poli, A. (eds.) AAEC 1999. LNCS, vol. 1719, pp. 221–230. Springer, Heidelberg (1999)
24. Todo, Y.: Structural evaluation by generalized integral property. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 287–314. Springer, Heidelberg (2015)
25. Tsunoo, Y., Saito, T., Shigeri, M., Kawabata, T.: Higher order differential attacks on reduced-round MISTY1. In: Lee, P.J., Cheon, J.H. (eds.) ICISC 2008. LNCS, vol. 5461, pp. 415–431. Springer, Heidelberg (2009)

# New Attacks on Feistel Structures with Improved Memory Complexities

Itai Dinur<sup>1</sup>, Orr Dunkelman<sup>2,4(✉)</sup>, Nathan Keller<sup>3,4</sup>, and Adi Shamir<sup>4</sup>

<sup>1</sup> Département d'Informatique, École Normale Supérieure, Paris, France

<sup>2</sup> Computer Science Department, University of Haifa, Haifa, Israel

orrd@cs.haifa.ac.il

<sup>3</sup> Department of Mathematics, Bar-Ilan University, Ramat Gan, Israel

<sup>4</sup> Computer Science Department, The Weizmann Institute, Rehovot, Israel

**Abstract.** Feistel structures are an extremely important and extensively researched type of cryptographic schemes. In this paper we describe improved attacks on Feistel structures with more than 4 rounds. We achieve this by a new attack that combines the main benefits of meet-in-the-middle attacks (which can reduce the time complexity by comparing only half blocks in the middle) and dissection attacks (which can reduce the memory complexity but have to guess full blocks in the middle in order to perform independent attacks above and below it). For example, for a 7-round Feistel structure on  $n$ -bit inputs with seven independent round keys of  $n/2$  bits each, a MITM attack can use  $(2^{1.5n}, 2^{1.5n})$  time and memory, while dissection requires  $(2^{2n}, 2^n)$  time and memory. Our new attack requires only  $(2^{1.5n}, 2^n)$  time and memory, using a few known plaintext/ciphertext pairs. When we are allowed to use more known plaintexts, we develop new techniques which rely on the existence of multicollisions and differential properties deep in the structure in order to further reduce the memory complexity.

Our new attacks are not just theoretical generic constructions — in fact, we can use them to improve the best known attacks on several concrete cryptosystems such as round-reduced CAST-128 (where we reduce the memory complexity from  $2^{111}$  to  $2^{64}$ ) and full DEAL-256 (where we reduce the memory complexity from  $2^{200}$  to  $2^{144}$ ), without affecting their time and data complexities. An extension of our techniques applies even to some non-Feistel structures — for example, in the case of FOX, we reduce the memory complexity of all the best known attacks by a factor of  $2^{16}$ .

**Keywords:** Cryptanalysis · Block cipher · Feistel structure · Dissection · Meet-in-the-middle · Splice-and-cut · CAST-128 · DEAL

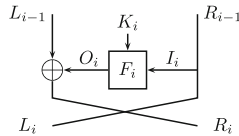
---

O. Dunkelman—The second author was supported in part by the Israeli Science Foundation through grant No. 827/12.

N. Keller—The third author was supported by the Alon Fellowship.

## 1 Introduction

Feistel structures were first used in the design of DES [18], and had a major influence on the development of both the theory and the practice of cryptography (e.g., in the Luby-Rackoff [17] construction of pseudo random permutations and in the design of numerous block cipher proposals). In this paper we will primarily consider generic Feistel structures whose  $i$ -th round is depicted in Fig. 1. They divide their  $n$ -bit blocks into two equal parts ( $L_i, R_i$ ), use independent  $n/2$ -bit subkeys in their  $\ell$  rounds, and have round functions  $F_i$  over  $n/2$ -bit inputs, outputs and subkeys which are perfect in the sense that they cannot be broken with attacks that are faster than exhaustive search. This choice of parameters allows us to consider any two consecutive rounds in a Feistel structure as a single round in a regular (non-Feistel) structure that has  $n$ -bit inputs outputs and subkeys. However, when we describe attacks on concrete schemes which have a Feistel structure, we will consider the relevant key and block sizes, and exploit some of the weaknesses of the actual round functions.



**Fig. 1.** The  $i$ 'th round of a Feistel structure

A major type of low-data attacks which can be applied to multi-round constructions is the Meet-In-The-Middle (abbreviated as MITM) attack, which was proposed by Diffie and Hellman [8] in 1977 as a method for cryptanalyzing double encryption schemes. It gained additional fame in 1985, when Chaum and Evertse [7] applied it to reduced-round variants of DES [18], and it is now considered as an essential part in any course in cryptanalysis. In the last few years, research of MITM techniques had expanded in diverse directions and numerous new extensions of the basic MITM appeared, including partial matching [4], probabilistic matching [15, 20], bicliques [3], sieve-in-the-middle [6], and many others. One of the recent approaches is the dissection attack, which was introduced by Dinur et al. [9] at CRYPTO 2012. Dissection can solve a wide variety of combinatorial search problems with improved combinations of time and memory complexities. In its cryptanalytic application, dissection significantly improved the time/memory tradeoff achievable by MITM attacks on multiple encryption schemes with more than 3 rounds.

The main difference between these two types of low-data attacks can be described in the following way: In basic MITM the adversary starts from the known plaintexts and ciphertexts at the endpoints, and works from both endpoints towards the middle by guessing some keys and building appropriate lookup tables. The equality of the pairs of values in the middle is used just as a filtering condition to identify the correct keys, and there is no need to know them in order to start the attack. In dissection attacks the adversary starts by

guessing the relevant values in the middle, and works from the middle towards the endpoints. In fact, the knowledge of the middle values enables the adversary to break the cryptanalytic problem into two independent smaller problems with new known plaintext and ciphertext pairs at their endpoints, which can be solved recursively by another dissection, or with MITM at the leaves of the recursion tree.

When we compare the two types of attacks on a Feistel structure with an odd number of  $\ell = 2r + 1$  rounds, we notice that each one of them offers different advantages and disadvantages. The MITM attack can ignore the middle round by comparing only the  $n/2$ -bit half blocks which are not affected by this round in the Feistel structure. This enables the MITM attack to be more time-efficient in the case of Feistel structures, since it does not have to guess the middle subkey. Even though dissection is naturally more efficient than MITM, it loses in its time complexity in this case since it has to guess the full  $n$ -bit middle value in order to be able to encrypt and decrypt this guessed value through multiple rounds.

In this paper we present new techniques which enable us to combine the MITM and dissection approaches, along with additional ingredients, such as iterating over values that are not used later in the MITM attack, and using multi-collisions and differential properties in the middle of the Feistel structure. We first consider the case of Feistel structures with an odd number of rounds, and try to reduce the memory complexity of the most time-efficient attacks on them. We show that the memory complexity of the MITM attack on  $\ell = 2r + 1$  rounds for  $r \geq 3$  can be reduced all the way from  $2^{0.5rn}$  to about  $2^{\lceil r/2 \rceil 0.5n}$  (like in dissection) without increasing the time complexity, at the expense of increasing the data complexity to about  $2^{\lceil \frac{r-3}{r+1} \rceil 0.5n}$  known plaintexts. If no additional plaintexts are allowed, we are still able to reduce the memory, but only to about  $2^{\lceil 2r/3 \rceil 0.5n}$ . In particular, we can reduce the memory complexity of the standard MITM attack on 7-round Feistel from  $2^{1.5n}$  to  $2^n$  with no effect on the data and time complexities.<sup>1</sup>

A different goal is to reduce the time complexity of the most memory-efficient nontrivial attacks (in which the memory available to the adversary is restricted to  $2^{0.5n}$ , which makes it possible to store in memory all the possible values of a single half-block or a single subkey, but not more). Here, we assume that the round functions can be inverted efficiently when their subkey is given. In [9], Dinur et al. considered low-memory dissection attacks on general (non-Feistel) structures, in which the available memory is restricted to  $2^n$  (where  $n$  is the length of a single block or a single subkey). They defined the *gain* of a dissection attack of time complexity  $T$  over a standard MITM attack with  $2^n$  memory (whose time complexity is  $2^{(\ell-1)n}$  for  $\ell$  rounds) by  $(\ell - 1) - \log_2(T)/n$ . Then, they computed the sequence of round numbers  $\ell$  for which the gain of the best dissection attack increases by one —  $\{4, 7, 11, 16, 22, 29, 37, \dots\}$ . We use our techniques to compute a similar sequence of round numbers  $\ell$  of Feistel structures for which the gain

---

<sup>1</sup> We alert the reader that similarly to [9], we concentrate on asymptotic complexity and ignore small polynomial factors in  $n$ . Moreover, we treat  $\ell$  as a (possibly big) constant, and therefore disregard all multiplicative factors related to  $\ell$  (and  $r$ ).

(over MITM, whose time complexity is  $2^{(\ell-2)0.5n}$ ) increases by one — it turns out to be  $\{5,10,15,22,29,38,47,\dots\}$ , and the asymptotic complexity of an attack on  $\ell$ -round Feistel using a minimal amount of  $2^{0.5n}$  memory turns out to be  $2^{0.5n(\ell-2-\sqrt{2\ell+o(\sqrt{\ell})})}$ . In particular, we present an attack on 5-round Feistel with time complexity of  $2^n$  and memory complexity of  $2^{0.5n}$  (compared to  $(2^n, 2^n)$  or  $(2^{1.5n}, 2^{0.5n})$  which are the best that can be obtained by previous attacks).

To deal with an even number of rounds without having to guess the extra key, we show that our algorithms can be combined with a recent algorithm presented by Isobe and Shibutani [12] at Asiacrypt 2013. This algorithm extends MITM attacks on Feistel structures by one round, at the expense of increasing the time complexity by  $2^{0.25n}$  and using  $2^{0.25n}$  chosen plaintexts. As a result, we obtain an attack on a Feistel structures with  $\ell = 2r$  rounds (for  $r \geq 4$ ) that requires  $2^{(0.5r-0.25)n}$  time, about  $2^{(0.5\lceil r/2 \rceil - 0.25)n}$  memory, and  $\max\{2^{0.25n}, 2^{\lceil \frac{r-4}{r} \rceil 0.5n}\}$  chosen plaintexts. Alternatively, we can use only  $2^{0.25n}$  chosen plaintexts like in [12], with  $2^{(0.5r-0.25)n}$  time and about  $2^{(\lceil 2(r-1)/3 \rceil 0.5+0.25)n}$  memory. In particular, we reduce the memory complexity of Isobe and Shibutani’s attack [12] on 8-round Feistel structures from  $2^{1.75n}$  to  $2^{1.25n}$ , with no effect on the data and time complexities.

In Table 1 we compare the complexity of our attacks with previous results for certain numbers of rounds which have “clean” exponents.

While all the techniques described so far are completely generic, they allow us to significantly improve the best known attacks on several concrete block ciphers. In particular, we reduce the memory complexity of the best known attack on 8-round CAST-128 [1] from  $2^{111}$  to  $2^{64}$ , and the memory complexity of the best known attack on 8-round DEAL [16] with 256-bit keys from  $2^{200}$  to  $2^{144}$ , both without affecting the time and data complexities. It is interesting to note that an extension of our techniques can even be applied to certain non-Feistel cryptosystems, such as FOX [14], in which the best MITM attack uses the *partial matching* technique.

This paper is organized as follows: In Sect. 2 we describe the improved memory complexities we obtain when we consider the most time-efficient attacks, and in Sect. 3 we describe the improved time complexities which can be obtained when we consider the most memory-efficient attacks. In Sect. 4 we sketch the application of our results to concrete block ciphers. We conclude the paper in Sect. 5.

## 2 Improving the Memory Complexity of the Most Time-Efficient Attacks on Feistel Structures

Consider a standard Feistel structure with an odd number  $\ell = 2r + 1$  of rounds. The generic dissection attack on this construction (that does not exploit the Feistel structure) requires  $2^{0.5(r+1)n}$  time and roughly  $2^{0.25(r+1)n}$  memory.<sup>2</sup> The

<sup>2</sup> The precise generic formula for the memory complexity of dissection is less relevant here. Note that such an attack has to treat every two consecutive Feistel rounds as a single round with an  $n$ -bit block and an  $n$ -bit key, and it is the last “half-round” which makes it suboptimal.

**Table 1.** Comparing and summarizing some of our results

Rounds	Complexity			Attack
	Time	Memory	Data	
5	$2^n$	$2^n$	3 KP	Meet in the middle <sup>a</sup>
	$2^{1.5n}$	$2^{0.5n}$	3 KP	Meet in the middle
	$2^n$	$2^{0.5n}$	3 KP	<b>New</b> (Sect. 3.1)
7	$2^{1.5n}$	$2^{1.5n}$	4 KP	Meet in the middle
	$2^{2n}$	$2^n$	4 KP	Dissection
	$2^{1.5n}$	$2^n$	4 KP	<b>New</b> (Sect. 2.1)
8	$2^{2n}$	$2^{1.5n}$	4 KP	Meet in the middle
	$2^{2n}$	$2^n$	4 KP	Dissection
	$2^{1.75n}$	$2^{1.75n}$	$2^{0.25n}$ CP	Splice-and-cut [12]
	$2^{1.75n}$	$2^{1.25n}$	$2^{0.25n}$ CP	<b>New</b> (Sect. 2.3)
15	$2^{3.5n}$	$2^{3.5n}$	8 KP	Meet in the middle
	$2^{6.5n}$	$2^{0.5n}$	8 KP	Meet in the middle
	$2^{4n}$	$2^{2n}$	8 KP	Dissection
	$2^{5n}$	$2^{0.5n}$	8 KP	<b>New</b> (Sect. 3.2)
	$2^{3.5n}$	$2^{2n}$	$2^{0.25n}$ KP	<b>New</b> (Sect. 2.2)
31	$2^{7.5n}$	$2^{7.5n}$	16 KP	Meet in the middle
	$2^{14.5n}$	$2^{0.5n}$	16 KP	Meet in the middle
	$2^{8n}$	$2^{4n}$	16 KP	Dissection
	$2^{12n}$	$2^{0.5n}$	16 KP	<b>New</b> (Sect. 3.2)
	$2^{7.5n}$	$2^{5n}$	16 KP	<b>New</b> (Sect. 2.1)
	$2^{7.5n}$	$2^{4n}$	$2^{0.375n}$ KP	<b>New</b> (Sect. 2.2)
32	$2^{8n}$	$2^{7.5n}$	16 KP	Meet in the middle
	$2^{15n}$	$2^{0.5n}$	16 KP	Meet in the middle
	$2^{8n}$	$2^{4n}$	16 KP	Dissection
	$2^{7.75n}$	$2^{7.75n}$	$2^{0.25n}$ CP	Splice-and-cut [12]
	$2^{12.5n}$	$2^{0.5n}$	16 KP	<b>New</b> (Sect. 3.2)
	$2^{7.75n}$	$2^{7.25n}$	$2^{0.25n}$ CP	<b>New</b> (Sect. 2.3)

KP — Known plaintext, CP — Chosen plaintext

<sup>a</sup> In the case of 5-round Feistel, the dissection attack is not better than the meet in the middle attack

standard MITM attack can exploit the Feistel structure to reduce the time complexity to  $2^{0.5rn}$ , at the expense of enlarging the memory complexity to  $2^{0.5rn}$ . No attacks faster than  $2^{0.5rn}$  are known (unless additional assumptions are made on the round functions or on the key schedule) and thus we concentrate in this section on attacks which have this time complexity. Our goal is to combine the benefits of both MITM and dissection attacks in order to reduce the memory

complexity to  $2^{0.25(r+1)n}$ . We show that this is indeed possible, but at the expense of somewhat enlarging the data complexity of the attack.

First, we present a basic 7-round attack<sup>3</sup> that requires  $2^{1.5n}$  time and  $2^n$  memory (compared to  $(2^{1.5n}, 2^{1.5n})$  and  $(2^{2n}, 2^n)$  in generic MITM and dissection, respectively), and extend it to an attack on  $2r + 1$  rounds that requires  $2^{0.5rn}$  time and about  $2^{0.33rn}$  memory. Then, we present a more sophisticated attack that requires  $2^{0.5rn}$  time and about  $2^{0.25rn}$  memory as desired, but at the expense of enlarging the data complexity to  $2^{\lceil (r-4)/r \rceil \cdot 0.5n}$  known plaintexts. Finally, we show that our attacks can be combined with a technique of Isobe and Shibutani [12] that allows extending MITM attacks on Feistel structures by one round using a splice-and-cut technique [2]. We obtain an attack on  $\ell = 2r$ -round Feistel that requires  $2^{(0.5r-0.25)n}$  time, about  $2^{0.25(r+1)n}$  memory, and  $2^{\max(\lceil \frac{r-4}{r} \rceil, 0.5) \cdot 0.5n}$  chosen plaintexts.

### 2.1 Attacks with a Low Data Complexity

In this section we present attacks that are time-efficient (i.e., have a time complexity of  $2^{0.5rn}$  for  $2r + 1$  rounds) and also data-efficient (i.e., require only a few known plaintexts, like the standard MITM attack).

**A Standard MITM Attack.** In order to put our attacks in context, we begin by describing a standard MITM attack on a 7-round Feistel structure (see left side of Fig. 2).

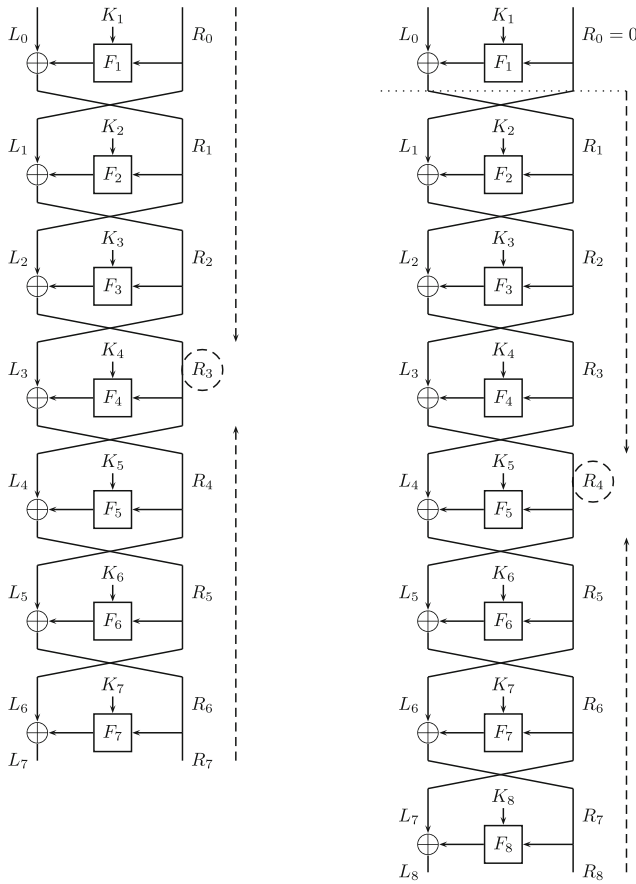
#### A 7-Round MITM Attack

1. Obtain 4 plaintext-ciphertext pairs  $(P^i, C^i)$  ( $i = 1, 2, 3, 4$ ).
2. For each value of  $K_1, K_2, K_3$ :
  - (a) Partially encrypt  $P^i$  for  $i \in \{1, 2, 3, 4\}$  through the first three rounds and obtain suggestions for  $R_3^i$ . Store the suggestions in a list *List* sorted by the  $R_3^i$  values.
3. For each value of  $K_5, K_6, K_7$ :
  - (a) Partially decrypt  $C^i$  for  $i \in \{1, 2, 3, 4\}$  through the last three rounds, obtain suggestions for  $R_3^i$  and search the suggestions in *List*. For each match, retrieve  $K_1, K_2, K_3$ , guess<sup>4</sup>  $K_4$ , and test the full key using trial encryptions.

The time complexity of Step 2 is about  $2^{1.5n}$ , which is also the size of *List*. In order to calculate the time complexity of Step 3, we note that we have a total of  $2^{1.5n}$  key suggestions from each side of the encryption, each associated

<sup>3</sup> We consider attacks on less than 7 rounds in Sect. 3.

<sup>4</sup> We note that  $K_4$  can also be found by a precomputed table instead of guessing, but this will not make a big difference as will be explained in the sequel.



**Fig. 2.** On the left: 7-round MITM attack. On the right: 8-round splice-and-cut attack.

with 4 values of  $R_3^i$  (filtering conditions). Thus, the expected total number of key suggestions that remain after the  $2n$ -bit match in Step 3 is  $2^{1.5n+1.5n-2n} = 2^n$ . For each such suggestion, we guess  $K_4$ , and thus we expect to perform about  $2^{1.5n}$  trial encryptions. Consequently, the time complexity of Step 3 is also about  $2^{1.5n}$ , which is the time complexity of the full attack.

**A 7-Round Attack.** The basic idea of our reduced memory attack is to guess the  $n/2$ -bit value  $R_3^1$  and to iterate over all the possible guesses as an outer loop. Each guess imposes an  $n/2$ -bit constraint on the key suggestions for  $K_1, K_2, K_3$  and  $K_5, K_6, K_7$ , and thus, allows reducing the expected size of *List* to  $2^n$ . In order to compute the reduced lists efficiently, we prepare auxiliary tables  $T_{upper}, T_{lower}$  that allow retrieving the subkey  $K_3$  (resp.,  $K_5$ ) instantly given the input  $(L_2, R_2)$  of round 3 (resp., the output  $(L_5, R_5)$  of round 5).



The table  $T_{upper}$  is computed as follows. We guess the intermediate value  $R_2^1$  and the subkey  $K_3$ . Since  $(R_2^1, R_3^1) = (L_3^1, R_3^1)$  form the full state after the 3'rd round in the encryption process of  $P^1$ , the guesses enable us to partially decrypt through round 3 and obtain  $(R_1^1, R_2^1) = (L_2^1, R_2^1)$ . We store the triple  $(L_2^1, R_2^1, K_3)$  in  $T_{upper}$ , sorted by  $(L_2^1, R_2^1)$ . The table  $T_{lower}$  is constructed similarly.

**7-Round Attack with Reduced Memory Complexity**

1. Obtain 4 plaintext-ciphertext pairs  $(P^i, C^i)$ .
2. For each value of  $R_3^1$ :
  - (a) For each value of  $K_3$  and  $I_3^1 = R_2^1$ , compute  $L_2^1 = F_3(K_3, I_3^1) \oplus R_3^1$ , and store the triplet  $(L_2^1, R_2^1, K_3)$  in a table  $T_{upper}$ , sorted according to  $(L_2^1, R_2^1)$ .
  - (b) For each value of  $K_5$  and  $I_5^1 = R_4^1$ , compute  $R_5^1 = F_5(K_5, I_5^1) \oplus R_3^1$ , and store the triplet  $(L_5^1, R_5^1, K_5)$  in a table  $T_{lower}$ , sorted according to  $(L_5^1, R_5^1)$ .
  - (c) For each value of  $K_1, K_2$ :
    - i. Partially encrypt  $P^1$  through the first two rounds to obtain suggestions for  $R_2^1$  and  $L_2^1$ .
    - ii. Search for the pair  $(L_2^1, R_2^1)$  in  $T_{upper}$  and obtain suggestions for  $K_3$ . For each suggestion, given  $K_1, K_2, K_3$ , partially encrypt  $P^i$  for  $i \in \{2, 3, 4\}$  through the first three rounds and obtain suggestions for  $R_3^i$ . Store the suggestions in a list  $List1$ , sorted by the values  $R_3^2, R_3^3, R_3^4$ .
  - (d) For each value of  $K_6, K_7$ :
    - i. Partially decrypt  $C^1$  through the last two rounds to obtain suggestions for  $R_5^1$  and  $L_5^1 = R_4^1$ .
    - ii. Search for the pair  $(L_5^1, R_5^1)$  in  $T_{lower}$  and obtain suggestions for  $K_5$ . For each suggestion, given  $K_5, K_6, K_7$ , partially decrypt  $C^i$  for  $i \in \{2, 3, 4\}$  through the last three rounds, obtain suggestions for  $R_3^i$  and search the suggestions in  $List1$ . For each match, retrieve  $K_1, K_2, K_3$ , guess  $K_4$ , and test the full key using trial encryptions.

In Steps 2a and 2b, a single round function (either  $F_3$  or  $F_5$ ) is called once for each guess of  $(R_3^1, I_3, K_3)$  (or  $(R_3^1, I_5, K_5)$ , respectively), and thus, their time complexity is  $2^{1.5n}$ . The memory complexity of the tables  $T_{upper}$  and  $T_{lower}$  is  $2^n$ . In Steps 2(c)ii and 2(d)ii there is an average of one match in  $T_{upper}$  and  $T_{lower}$ , respectively. Thus, on average, we perform a constant number of partial encryption and decryption operations per guess of  $K_1, K_2$  and  $K_6, K_7$  in Steps 2c and 2d, respectively. The expected number of matches in Step 2(d)ii is  $2^{n+n-1.5n} = 2^{0.5n}$ , and the expected number of trial encryptions (after guessing  $K_4$ ) is  $2^{0.5n+0.5n} = 2^n$ . Therefore, the time complexity of each of Steps 2c and 2d is about  $2^n$  each, and the total time complexity of the attack is about  $2^{1.5n}$ , as in the standard MITM attack. On the other hand, the memory complexity of the attack is reduced from  $2^{1.5n}$  to about  $2^n$ , which is the expected number of

elements in *List1* (based on standard randomness assumptions on the round functions).

We note that the time complexity of the attack can be slightly reduced by precomputing a table for  $F_4$ , which allows to avoid guessing  $K_4$  in Step 2(d)ii. However, this requires an additional table of size  $2^n$ , i.e., maintaining the  $2^n$  total memory complexity.

**Extension to  $6r+1$  Rounds.** The 7-round attack presented above can be extended to an attack on a  $6r+1$ -round Feistel structure, with time complexity of  $2^{1.5rn}$  (as in standard MITM) and memory complexity of  $2^{rn}$  (instead of  $2^{1.5rn}$  in standard MITM). As the attack is similar to the 7-round attack, we describe it briefly. The reader can follow this attack by verifying that the case  $r = 1$  reduces exactly to the attack described above.

First, we obtain  $3r + 1$  plaintext/ciphertext pairs  $(P^i, C^i)$ . Then, the outer loop is performed for all guesses of the  $r$  intermediate values  $R_{3r}^1, R_{3r}^2, \dots, R_{3r}^r$ . In the inner loop, we guess the  $2r$  values  $R_{3r-1}^1, R_{3r-1}^2, \dots, R_{3r-1}^r, K_{3r}, \dots, K_{2r+1}$ . Since for each  $i$ ,  $(R_{3r-1}^i = L_{3r}^i, R_{3r}^i)$  forms the full state after the  $3r$ 'th round in the encryption process of  $P^i$ , we can partially decrypt this state through rounds  $3r, \dots, 2r + 1$  to obtain the corresponding values  $(R_{2r-1}^i = L_{2r}^i, R_{2r}^i)$ . This allows us to prepare a table  $T_{upper}$  of size  $2^{rn}$  of the values  $((L_{2r}^i, R_{2r}^i)_{i=1}^r, K_{2r+1}, \dots, K_{3r})$ , sorted by  $((L_{2r}^i, R_{2r}^i)_{i=1}^r)$ . The table  $T_{lower}$  is prepared similarly. Note that the  $2r$  values guessed from each side are used only for preparing the tables and not in the rest of the attack.

After preparing the tables, we guess the subkeys  $K_1, K_2, \dots, K_{2r}$ , obtain the intermediate values  $(L_{2r}^i, R_{2r}^i)_{i=1}^r$  and access the table  $T_{upper}$  to obtain a suggestion for the subkeys  $K_{2r+1}, \dots, K_{3r}$ . For each suggestion, given  $K_1, K_2, \dots, K_{3r}$ , we partially encrypt  $P^i$  for  $i \in \{r + 1, \dots, 3r + 1\}$  through the first  $3r$  rounds and obtain suggestions for  $R_{3r}^i$ . We store the suggestions in a list *List1*, sorted by the values  $R_{3r}^{r+1}, \dots, R_{3r}^{3r+1}$ . Then, we guess the subkeys  $K_{6r+1}, \dots, K_{4r+2}$ , access the table  $T_{lower}$  to obtain a suggestion for  $K_{4r+1}, \dots, K_{3r+2}$ , partially decrypt the ciphertexts to get suggestions for  $R_{3r}^i$  ( $i = r + 1, \dots, 3r + 1$ ), and search them in *List1*. For each match, we retrieve  $K_1, \dots, K_{3r}$ , guess  $K_{3r+1}$ , and test the full key using trial encryptions.

The analysis of the attack is similar to that of the 7-round attack described above, and yields time complexity of  $2^{1.5rn}$  and memory complexity of  $2^{rn}$ . The same attack applies for a general odd number  $2r' + 1$  of rounds. The time complexity is  $2^{r'n}$  (like in MITM), but the memory complexity has to be rounded up to  $2^{\lceil 2r'/3 \rceil \cdot 0.5n}$ , due to lack of balance between the part of table creation and the rest of the attack.

## 2.2 Using Multi-Collisions to Further Reduce the Memory Complexity

We now present a more sophisticated variant of the attacks described above, that allows to reduce the memory complexity to the “desired”  $2^{0.25(r+1)n}$  (similarly to dissection on multiple encryption schemes), with no increase in the time complexity, but at the expense of some increase in the data complexity.

Consider the  $6r + 1$ -round attack described above. In the course of preparing the table  $T_{upper}$ , we make an auxiliary guess of the values  $R_{3r-1}^1, \dots, R_{3r-1}^r, K_{3r}, \dots, K_{2r+1}$ , and in the course of preparing the table  $T_{lower}$ , we guess  $R_{3r+1}^1, \dots, R_{3r+1}^r, K_{3r+2}, \dots, K_{4r+1}$ . If there was some relation between the guessed values, we could have used this relation to enumerate over some “common relative” in the outer loop of the attack, and thus reduce the memory complexity. We cannot hope for such a relation between the subkeys, as they are assumed to be independent.<sup>5</sup> However, some relation between  $R_{3r-1}^i$  and  $R_{3r+1}^i$  may exist.

We observe that such a relation can be “created”, using multi-collisions. Assume that the partial encryption of the plaintexts  $P^1, P^2, \dots, P^r$  considered in the attack results in an  $r$ -multi-collision at the state  $R_{3r}$ , i.e., that  $R_{3r}^1 = R_{3r}^2 = \dots = R_{3r}^r$ . In such a case, the  $r$  values  $R_{3r-1}^i \oplus R_{3r+1}^i = O_{3r+1}^i$  ( $i = 1, \dots, r$ ) are all equal! This allows us to enumerate over the  $r - 1$  values  $R_{3r-1}^i \oplus R_{3r-1}^j$  ( $i = 2, \dots, r$ ) in the outer loop, such that in the inner loop, a single guess of  $R_{3r-1}^1$  provides all the values  $\{R_{3r-1}^i\}_{i=2, \dots, r}$ , while a single guess of  $R_{3r+1}^1$  provides all the values  $\{R_{3r+1}^i\}_{i=2, \dots, r}$ . In order to obtain the multi-collision, we consider  $2^{((r-1)/r) \cdot 0.5n}$  known plaintexts (which guarantees that an  $r$ -multi-collision exists in the data with a constant probability<sup>6</sup>), and repeat the attack for all  $r$ -tuples of plaintext/ciphertext pairs in the data set.

In the description of the algorithm below, we switch from  $6r + 1$  rounds to  $8r - 1$  rounds, in order to balance the complexities of all the steps of the attack. Hence, the external guesses are performed at state  $R_{4r-1}$ , instead of  $R_{3r}$ . Note that for  $r = 1$ , the algorithm reduces to the 7-round attack presented above.

**An  $8r-1$ -Round Attack Using Multi-Collisions**

1. Obtain  $2^{((r-1)/r) \cdot 0.5n}$  plaintext-ciphertext pairs  $(P^i, C^i)$ .
2. For each  $r$ -tuple  $(P_{i_1}, C_{i_1}), (P_{i_2}, C_{i_2}), \dots, (P_{i_r}, C_{i_r})$  of plaintext-ciphertext pairs in the data set (hereinafter denoted for simplicity by  $(P^1, C^1), \dots, (P^r, C^r)$ ), for each possible value of  $R_{4r-1}^1$ , and for all possible differences  $R_{4r-2}^i \oplus R_{4r-2}^j$  ( $i = 2, 3, \dots, r$ ):
  - (a) For each  $I_{4r-1}^1 = R_{4r-2}^1$  and the subkeys  $K_{4r-1}, K_{4r-2}, \dots, K_{2r+1}$ , compute<sup>7</sup>  $(L_{2r}^i, R_{2r}^i)$  for all  $i = 1, \dots, r$ , and store the vector  $((L_{2r}^i, R_{2r}^i)_{i=1, \dots, r}, K_{2r+1}, \dots, K_{4r-1})$  in a table  $T_{upper}$ , sorted according to  $(L_{2r}^i, R_{2r}^i)_{i=1, \dots, r}$ .
  - (b) For each  $I_{4r+1}^1 = R_{4r}^1$  and the subkeys  $K_{4r+1}, \dots, K_{6r-1}$ , compute  $(L_{6r-1}^i, R_{6r-1}^i)$  for all  $i = 1, \dots, r$ , and store the vector  $((L_{6r-1}^i, R_{6r-1}^i)_{i=1, \dots, r}, K_{4r+1}, \dots, K_{6r-1})$  in a table  $T_{lower}$ , sorted according to  $(L_{6r-1}^i, R_{6r-1}^i)_{i=1, \dots, r}$ .
  - (c) For each value of  $K_1, K_2, \dots, K_{2r}$ :

<sup>5</sup> Some useful relations between subkeys may exist if they are derived from a master key using a simple key schedule. However, in this paper, we focus on the most general Feistel constructions and do not assume the existence of such relations.

<sup>6</sup> Roughly speaking, given  $D$  data, we have about  $D^r$  different  $r$ -tuples of the  $0.5n$ -bit value  $R_{3r}$ . For each  $r$ -tuple, the probability that all the values of  $R_{3r}$  are equal is  $2^{-0.5n \cdot (r-1)}$ . Therefore, we require  $D^r = 2^{0.5n \cdot (r-1)}$  or  $D = 2^{((r-1)/r) \cdot 0.5n}$ .

- i. Partially encrypt  $P^1, \dots, P^r$  through the first  $2r$  rounds to obtain suggestions for  $(L_{2r}^i, R_{2r}^i)_{i=1, \dots, r}$ .
- ii. Search for  $(L_{2r}^i, R_{2r}^i)_{i=1, \dots, r}$  in  $T_{upper}$  and obtain suggestions for  $K_{2r+1}, \dots, K_{4r-1}$ . For each suggestion, given  $K_1, \dots, K_{4r-1}$ , partially encrypt  $2r+1$  additional plaintexts  $P^j$  for  $j \in \{1, \dots, 2r+1\}$  through the first  $4r-1$  rounds and obtain suggestions for  $R_{4r-1}^j$ . Store the suggestions in a list *List1*, sorted by the values  $\{R_{4r-1}^j\}_{j=1, \dots, 2r+1}$ .
- (d) For each value of  $K_{8r-1}, \dots, K_{6r}$ :
  - i. Partially decrypt  $C^1, \dots, C^r$  through the last  $2r$  rounds to obtain suggestions for  $(L_{6r-1}^i, R_{6r-1}^i)_{i=1, \dots, r}$ .
  - ii. Search for  $(L_{6r-1}^i, R_{6r-1}^i)_{i=1, \dots, r}$  in  $T_{lower}$  and obtain suggestions for  $K_{4r+1}, \dots, K_{6r-1}$ . For each suggestion, given  $K_{4r+1}, \dots, K_{8r-1}$ , partially decrypt the additional ciphertexts  $C^j$  for  $j \in \{1, \dots, 2r+1\}$  through the last  $4r-1$  rounds, obtain suggestions for  $\{R_{4r-1}^j\}_{j=1, \dots, 2r+1}$ , and search the suggestions in *List1*. For each match, retrieve  $K_1, \dots, K_{4r-1}$ , guess  $K_{4r}$ , and test the full key using trial encryptions.

The inner loop of the algorithm is repeated for each of the  $2^{(r-0.5)n}$  values of the external guess (there are  $2^{(r-1) \cdot 0.5n}$   $r$ -tuples in the data set,  $2^{0.5n}$  possible values of  $R_{4r-1}^1$ , and  $2^{(r-1) \cdot 0.5n}$  possible differences  $R_{4r-2}^1 \oplus R_{4r-2}^i$ ). In each of Steps 2(a) and 2(b), we perform  $2^{rn}$  partial encryptions/decryptions and construct a table of size  $2^{rn}$ . In Steps 2.(c).ii. and 2.(d).ii. there is an average of one match in  $T_{upper}$  and  $T_{lower}$ , respectively. Thus, on average, we perform a constant number of partial encryption and decryption operations per guess of  $K_1, \dots, K_{2r}$  and  $K_{6n}, \dots, K_{8n-1}$  in Steps 2.(c) and 2.(d), respectively. The expected number of matches in Step 2.(d).ii is  $2^{rn+rn-(r+0.5)n} = 2^{(r-0.5)n}$ , and the expected number of trial encryptions (after guessing  $K_4$ ) is  $2^{(r-0.5)n+0.5n} = 2^{rn}$ . Therefore, the time complexity of Steps 2.(c) and 2.(d) is about  $2^{rn}$  and the total time complexity of the attack is about  $2^{(2r-0.5)n}$ , as in the standard MITM attack. On the other hand, the memory complexity of the attack is reduced from  $2^{(2r-0.5)n}$  to about  $2^{rn}$ , which is the expected number of elements in *List1*.

The same attack applies for a general odd number  $2r'+1$  of rounds. The time complexity is  $2^{r'n}$  (like in MITM), the memory complexity has to be rounded up to  $2^{\lceil 0.5(r'+1) \rceil \cdot 0.5n}$ , due to lack of balance between the part of table creation and the rest of the attack, and the data complexity is  $2^{\lceil \frac{r'-3}{r'+1} \rceil \cdot 0.5n}$  known plaintexts.

---

<sup>7</sup> The computation of  $(L_{2r}^i, R_{2r}^i)$  for all  $i = 1, \dots, r$  is feasible, since by the assumption that  $(P^1, C^1), \dots, (P^r, C^r)$  is an  $r$ -multi-collision, the value  $R_{4r-2}^1$  along with the externally guessed values are sufficient for obtaining the values  $R_{4r-2}^2, \dots, R_{4r-2}^r$ .

### 2.3 Attacks on Feistel Structures with an Even Number of Rounds

In this section we show that all the attacks presented above can be combined with the recent technique of Isobe and Shibutani [12] that allows to extend MITM attacks on Feistel structures by one round, at the expense of a relatively small increase in the time complexity and of using  $2^{0.25n}$  chosen plaintexts. The generic attack of [12] on a  $2r$ -round Feistel structures requires  $2^{(0.5r-0.25)n}$  time,  $2^{(0.5r-0.25)n}$  memory, and  $2^{0.25n}$  chosen plaintexts. Our attacks allow to either reduce the memory complexity to about  $2^{0.33(r-1)n+0.25n}$  with no effect on the time and data complexities or to reduce the memory complexity all the way to about  $2^{0.25(r+1)n}$ , while increasing the data complexity to  $2^{\max(\lceil \frac{r-4}{r} \rceil, 0.5) \cdot 0.5n}$  chosen plaintexts, with no effect on the time complexity. In the specific case of the 8-round Feistel structures considered in [12], our attack reduces the memory complexity significantly from  $2^{1.75n}$  to  $2^{1.25n}$ , without affecting the other complexities.

Consider a MITM attack on a  $2r$ -round Feistel structure. In a standard application (skipping the guess of the middle subkey), the attack is not balanced, as  $r$  subkeys are guessed on one side of the MITM, while  $r-1$  subkeys are guessed on the other side. The attack of [12] aims to rebalance the attack, by “splitting” the guess of one subkey between the two sides. The basic idea behind the attack is as follows. If in all plaintexts used in the attack, the right half is equal to a fixed value  $R_0$  (e.g.,  $R_0 = 0$ ), then in all encryptions, we have  $R_1 = Const \oplus L_0$ , where  $Const$  is an unknown constant that depends on  $K_1$ . This allows to replace the  $2r$ -round Feistel with an equivalent construction that consists of a  $2r-1$ -round Feistel, prepended by an addition of  $Const$  to the right half of the plaintext that can be treated as a subkey addition (see right side of Fig. 2 for a sketch of an 8-round attack). This, in turn, allows to use the splice-and-cut technique [2] to “split” the guess of the  $n/2$ -bit value  $Const$  between the two sides of the MITM, at the price of using  $2^{n/4}$  chosen plaintexts (each associated with an  $n/4$ -bit value of  $Const$ , while the remaining  $n/4$  bits of  $Const$  are guessed from the other side of the attack). As a result, the attack becomes balanced and the time complexity is reduced from  $2^{0.5rn}$  to  $2^{(0.5r-0.25)n}$ . For a full description of the attack, see [12].

In order to incorporate the splice-and-cut procedure of [12] into our attacks, we consider the equivalent  $2r-1$ -round variant, perform one of our attacks against it, and insert the splice-and-cut procedure into the “key guessing” part of the attack (i.e., Steps 2(c) and 2(d)), without changing the “table construction” part (Steps 2(a) and 2(b)). As a result, the time complexity of our  $2r-1$ -round attack is increased by a factor of  $2^{0.25n}$  to  $2^{(0.5r-0.25)n}$  (just like the complexity of the attack of [12]), and the memory complexity is increased by a factor of  $2^{0.25n}$  to either  $2^{0.33(r-1)n+0.25n}$  (in the low data complexity attack) or to about  $2^{0.25(r+1)n}$  (in the attack using multi-collisions). As for the data complexity, in our low data complexity attack the data complexity increases to  $2^{0.25n}$  chosen plaintexts (required for the splice-and-cut procedure), and in the multi-collision based attack the data complexity increases to  $2^{\max(\lceil \frac{r-4}{r} \rceil, 0.5) \cdot 0.5n}$ , as the plaintexts required for the multi-collision can be chosen in such a way that they will contain the structures required for the splice-and-cut attack.

### 3 Memory-Restricted Attacks on Feistel Structures

After analyzing the most time-efficient attacks, a natural question to explore is what are the most memory-efficient attacks one can devise against an  $r$ -round Feistel structure. Specifically, we shall concentrate on the problem of devising attacks with  $2^{0.5n}$  memory complexity, since it is the smallest amount of memory that enables us to list all the values of a single subkey or of half a block.

With such a restriction, a standard meet in the middle attack takes  $2^{(\ell-2)\cdot 0.5n}$  on an  $\ell$ -round Feistel, and one can trade time for memory. One can also try to consider the original dissection attack of [9]. However, as noted before, dissection takes at least  $2^n$  memory to store all the possible values of a full block, which implies that it cannot be used in this context, even though we adopt several concepts from it.

Section 3.1 presents our new attack on 5-round Feistel structures that uses  $2^n$  time and  $2^{0.5n}$  memory. This is to be compared with meet in the middle attacks that use time of  $2^{1.5n}$  with  $2^{0.5n}$  memory or time of  $2^n$  with  $2^n$  memory. We then generalize the attack to more rounds, and show in Sect. 3.2 how to increase the gain over meet in the middle attacks as the number of rounds increases. The attacks in this section assume that the round function is efficiently invertible given the round's subkey. We postpone the discussion of this assumption to Appendix A, but note that it holds for almost any Feistel block cipher we are aware of.

#### 3.1 A Memory-Restricted Attack Against 5-Round Feistel Constructions

The algorithm of our basic 5-round attack is as follows.

##### A 5-Round Attack with $2^{0.5n}$ Memory ( $DF_2(5, 1)$ )

1. Obtain 3 plaintext-ciphertext pairs  $(P^i, C^i)_{i=1,2,3}$ .
2. For each value of  $R_2^1 = I_3^1$ :
  - (a) Compute  $O_2^1 = I_3^1 \oplus R_0^1$  and  $O_4^1 = I_3^1 \oplus R_4^1$ .
  - (b) For each value of  $K_1$ , compute  $R_1^1 = F_1(K_1, I_1^1) \oplus L_0^1$  and store the pair  $(R_1^1, K_1)$  in a table  $T_{upper}$  sorted according to  $R_1^1$ .
  - (c) For each value of  $K_2$ , compute  $R_1^1 = F_2^{-1}(K_2, O_2^1)$ , search for the value  $R_1^1$  in  $T_{upper}$  and obtain suggestions for  $K_1$ . For each suggestion, given  $K_1, K_2$ , compute  $R_2^2, R_2^3$  for  $P^2, P^3$ . Store the suggestions  $(R_2^2, R_2^3, K_1, K_2)$  in a list  $List1$  sorted by the value of  $(R_2^2, R_2^3)$ .
  - (d) For each value of  $K_5$ , compute  $R_3^1 = F_5(K_5, I_5^1) \oplus R_5^1$  and store the pair  $(R_3^1, K_5)$  in a table  $T_{lower}$  sorted according to  $R_3^1$ .
  - (e) For each value of  $K_4$ , compute  $R_3^1 = F_4^{-1}(K_4, O_4^1)$ , search for the value  $R_3^1$  in  $T_{lower}$  and obtain suggestions for  $K_5$ . For each suggestion, given  $K_4, K_5$ , compute  $R_2^2, R_2^3$  from  $C^2, C^3$ , and search the suggestion in  $List1$ . For each match, retrieve  $K_1, K_2$ , guess  $K_3$ , and test the full key using trial encryptions.

For reasons which will become apparent later, we call the above attack  $DF_2(5, 1)$ .

As before,  $T_{upper}, T_{lower}$ , are each of size  $2^{0.5n}$ . The memory complexity of  $List1$  depends on the number of  $(K_1, K_2)$  pairs that satisfy the meet in the middle condition on the value of  $I_2$  in Step 2c. For sufficiently random round functions, we expect about  $2^{0.5n}$  such  $(K_1, K_2)$  pairs.

The time complexity of the algorithm is  $2^n$ , as it iterates over  $2^{n/2}$  values for  $R_2^1$ , and each step of the loop takes  $2^{0.5n}$  operations (besides the XOR of Step 2a, which takes less).

We note that the time complexity of the attack can be slightly reduced by pre-computing a table for  $F_3$  (given its input value  $I_3^1$ ), which allows to avoid guessing  $K_3$ . However, this requires an additional table of size  $2^{0.5n}$ , which increases the memory complexity by a small constant factor.

Finally, it is important to note that given only two plaintext-ciphertext pairs, the above attack finds all possible  $(K_1, K_2, K_3, K_4, K_5)$  in time  $2^n$  and memory of  $2^{0.5n}$ . The expected number of candidates is about  $2^{0.5n}$ . This observation will be used in the subsequent attacks.

### 3.2 Extension to More Rounds

As the time complexity of a MITM attack with  $2^{0.5n}$  memory on an  $\ell$ -round Feistel structure is  $2^{(\ell-2)0.5n}$ , we define the *gain* over MITM of an attack on  $\ell$ -round Feistel that requires  $T$  time and  $2^{0.5n}$  memory by  $(\ell - 2) - \log_2(T)/0.5n$ . Thus, the 5-round attack presented above has gain of 1. We denote by  $Gain(\ell)$  the maximal gain achieved by an  $\ell$ -round attack with  $2^{0.5n}$  memory. In this section, we extend the 5-round attack to a sequence of attacks which show that asymptotically,  $Gain(\ell) = \Omega(\sqrt{\ell})$ , and compute the sequence of round numbers for which the gain is strictly increased.

Obviously, it is possible to attack 6-round Feistel by guessing  $K_6$ , and applying the 5-round attack for each guess. The result is an attack of  $2^{1.5n}$  time and  $2^{0.5n}$  memory on 6-round Feistel structure. This approach can obviously be extended, but maintains a gain of 1.

**Attacking 10-Round Feistel Constructions.** To increase the gain to 2, we consider the case of 10-round Feistel, and develop the following attack:

**10-Round Dissection Attack with  $2^{n/2}$  Memory ( $DF_5(10, 4)$ )**

1. Obtain 5 plaintext-ciphertext pairs  $(P^i, C^i)_{i=1, \dots, 5}$ .
2. For each value of  $(L_5^1, R_5^1)$  and  $(L_5^2, R_5^2)$ :
  - (a) Run  $DF_2(5, 1)$  on the first 5 rounds, and obtain a list of  $2^{0.5n}$  candidates for  $(K_1, K_2, K_3, K_4, K_5)$ .
  - (b) For each candidate for the subkeys  $(K_1, K_2, K_3, K_4, K_5)$ , partially encrypt a third plaintext  $P^3$  through the first 5 rounds, and store the suggestions (with the keys)  $(L_5^3, R_5^3, K_1, \dots, K_5)$  in a list  $List1$  sorted by the values of  $(L_5^3, R_5^3)$ .

- (c) Run  $DF_2(5, 1)$  on the last 5 rounds, and obtain a list of  $2^{0.5n}$  candidates for  $(K_6, K_7, K_8, K_9, K_{10})$ .
- (d) For each candidate for the subkeys  $(K_6, K_7, K_8, K_9, K_{10})$ , partially decrypt  $C^3$  through the last 5 rounds, and obtain suggestions for  $(L_5^3, R_5^3)$  and search the suggestion in *List1*. For each match, retrieve  $K_1, \dots, K_5$ , and test the full key using trial encryptions.

It is easy to see that the 10-round attack calls  $2^{2n}$  times two independent 5-round attacks, each running in time  $2^n$ . Hence, the time complexity of the 10-round attack is  $2^{3n}$ , and the memory complexity is  $2^{0.5n}$ . Hence, the gain of the 10-round attack is 2.

We use the following notations,  $DF_2(5, 1)$  denotes the 5-round attack presented earlier, as it is a generalized Dissection attack on Feistel structures with 5 rounds, which guesses one  $n/2$ -bit value after two rounds of encryption. Similarly,  $DF_5(10, 4)$  attacks 10 rounds by guessing 4  $n/2$ -bit values after 5 rounds of encryption (i.e., two full intermediate encryption values). As in [9], we now explore how to extend the attack to more rounds.

**Attacking 15-Round Feistel Constructions.** We can increase the gain to 3, when attacking 15-round Feistel: Guess two complete intermediate encryption values after 5 and after 10 rounds (a total of four internal values), and run the 5-round attack three times subsequently (on rounds 1–5, 6–10, and 11–15), resulting in  $2^{0.5n}$  candidates for each set of corresponding subkeys. Then, the correct value can be found by an additional standard MITM. If the memory is kept at  $2^{0.5n}$ , this means that the last layer of the MITM takes  $2^n$  time. Hence, the total time complexity of the 15-round attack is  $2^{5n}$ , and thus, its gain is 3.

In other words,  $DF_5(15, 4)$  is based on guessing four  $n/2$ -bit internal state words after 5 rounds, and recursively running  $DF_2(5, 1)$  on the first rounds, and  $DF_5(10, 4)$  on the last rounds. This contrasts with the works of [9], where each new layer in the dissection was of a different size.

The different “expanding” rule is due to two inherent differences between the dissection attacks presented in [9] and our new attacks. First, in our attacks, guessing a full intermediate state adds two “units” of time complexity (as each full state contains two  $n/2$ -bit values) compared with one in the case of regular dissection attacks. The second difference is more subtle, but has a larger effect on the way the attack scales up: In our attacks we can enjoy the “Feistel” advantage (meeting in the middle only on  $n/2$  bits) only once in the internal recursion step (e.g., in the 5-round attack), as the external steps must rely on guessing a full internal state. The second difference is already apparent in the transition from 5-round to 10-round (comparing  $DF_2(5, 1)$  and  $DF_5(10, 4)$ ): whereas the 5-round attack guesses a single  $n/2$ -bit value, the 10-round attack starts by guessing 4 such values.



**Attacking 22-Round Feistel Structures.** We now turn our attention to 22-round Feistel structures. Due to the differences between regular dissection attacks and attacking Feistels, the extension of the 15-round attack into the 22-round attack follows a slightly different path than the extension from the 10-round to the 15-round:

**22-Round Dissection Attack with  $2^{n/2}$  Memory ( $DF_7(22, 6)$ )**

1. Obtain 11 plaintext-ciphertext pairs  $(P^i, C^i)_{i=1, \dots, 11}$ .
2. For each possible value of  $(L_7^1, R_7^1)$ ,  $(L_7^2, R_7^2)$ , and  $(L_7^3, R_7^3)$ :
  - (a) Run  $DF_2(7, 3)$  on the first 7 rounds, and obtain a list of  $2^{0.5n}$  candidates for  $(K_1, K_2, \dots, K_7)$ .
  - (b) For each candidate for the subkeys  $(K_1, K_2, \dots, K_7)$ , partially encrypt a fourth plaintext  $P^4$  through the first 7 rounds, and store the suggestions (with the keys)  $(L_7^4, R_7^4, K_1, \dots, K_7)$  in a list *List1* sorted by the values of  $(L_7^4, R_7^4)$ .
  - (c) Run  $DF_5(15, 4)$  on the last 15 rounds, and obtain a list of  $2^{4.5n}$  candidates<sup>8</sup> for  $(K_8, K_9, \dots, K_{22})$ .
  - (d) For each candidate for the subkeys  $(K_8, \dots, K_{22})$ , partially decrypt  $C^4$  through the last 15 rounds, and obtain suggestions for  $(L_7^4, R_7^4)$  and search the suggestion in *List1*. For each match, retrieve  $K_1, \dots, K_7$ , and test the full key using trial encryptions.

It is easy to see that the memory complexity of the attack is  $2^{0.5n}$ . The 7-round attack  $DF_2(7, 3)$  is actually  $DF_2(5, 1)$ , run when  $K_6, K_7$  are guessed, i.e., takes  $2^{2n}$  time for  $2^{0.5n}$ . Both the 7-round attack and the 15-round attack are called  $2^{3n}$  times, suggesting a total running time of  $2^{8n}$ , i.e., the attack offers a gain of 4.

**Generalization to More Rounds.** In the second generalization, we prepend 5 rounds to the 10-round attack (obtaining 15 rounds in total), and again, guess two full internal states in order to run two independent attacks — one on 5 rounds, and the other on 10 rounds. The 22-round attack is based on guessing three additional full internal states, and prepending 7 rounds before the 15-round attack. Similarly, a 29-round attack with a gain of 5 can be obtained by prepending 7 rounds before the 22-round attack and guessing three additional full internal states. It is now apparent that the sequence of round numbers for which the gain increases is  $\{5, 10, 15, 22, 29, 38, 47, 58, 69, \dots\}$ ,<sup>9</sup> which shows that for all  $k$ ,  $Gain(2k^2 + 6k + 2) \geq 2k$  and  $Gain(2k^2 + 8k + 5) \geq 2k + 1$ . It follows that asymptotically,  $Gain(\ell)$  grows as  $\sqrt{2\ell}$ .

The general form of the recursion is described in Fig. 3. We note that the recursion yields only a lower bound on  $Gain(\ell)$ .

<sup>8</sup> We guess  $3n$  bits in Step 2, giving rise to  $3n$  constraints on  $7.5n$  key bits.

<sup>9</sup> A gain of  $j$  is first achieved when attacking  $5j + 2 \sum_{i=1}^{j-1} \lfloor i/2 \rfloor$  rounds.



has only 56 bits rather than 64, yields time complexity of  $2^{64+56+56+32} = 2^{208}$ . However, the time complexity can be reduced to  $2^{200}$  by performing the external enumeration over 56 out of the 64 bits of the intermediate value  $R_4^1$ , rather than over the full value. In the phase of table preparation, we guess the remaining 8 bits of  $R_4^1$ , along with the auxiliary guess of  $R_3^1, K_4$ , and thus, the complexity of this step is increased to  $2^{8+64+56} = 2^{128}$ . However, this complexity is still dominated by the  $2^{32+56+56} = 2^{144}$  complexity of the key guessing step. As a result, the overall time complexity remains  $2^{56+56+56+32} = 2^{200}$ , the memory complexity is reduced to  $2^{56+56+32} = 2^{144}$ , and the data complexity remains  $2^{32}$  chosen plaintexts.

In a similar way we can reduce the memory complexity of the improved MITM attack on the full 6-round DEAL with 192-bit keys from  $2^{56+56+32} = 2^{144}$  to  $2^{56+32} = 2^{88}$  (while keeping the  $2^{144}$  time complexity and  $2^{32}$  data complexity unchanged), using a modification of the generic attack on 6-round Feistel structures presented in Sect. 3. The resulting attack in this case is the best known attack which uses  $2^{32}$  data, but is outperformed (in terms of time complexity) by the impossible differential attack presented by Knudsen [16] that requires  $2^{121}$  time but uses  $2^{70}$  chosen plaintexts. Table 2 compares the complexities of attacks against the variants of DEAL.

**Table 2.** Comparison of results against DEAL

Key size	Rounds	Complexity			Attack
		Time	Memory	Data	
192	6	$2^{121}$	$2^{64}$	$2^{70}$ CP	Impossible differential [16]
	6	$2^{144}$	$2^{144}$	$2^{32}$ CP	Splice-and-cut <sup>a</sup> [12]
	6	$2^{144}$	$2^{88}$	$2^{32}$ CP	<b>New</b>
256	8	$2^{224}$	$2^{168}$	4 KP	Meet in the middle
	8	$2^{200}$	$2^{200}$	$2^{32}$ CP	Splice-and-cut <sup>a</sup> [12]
	8	$2^{200}$	$2^{144}$	$2^{32}$ CP	<b>New</b>

KP — Known plaintext, CP — Chosen plaintext

<sup>a</sup> This attack was not really suggested in [12], but can be derived from the paper

## 4.2 A Lower Memory Attack on CAST-128

CAST-128 [1] is a 16-round Feistel structure that uses 64-bit inputs and 128-bit keys, which was designed in 1996 by Adams. It is used in several real-life products, such as GPG, PGP, and SSH2. The currently best known attack on the cipher (excluding weak-key attacks such as [21]) is the MITM attack of Isobe and Shibutani [12], breaking 8 out of the 16 rounds in time complexity of about  $2^{118}$ , using 8 chosen ciphertexts and a memory complexity of  $2^{111}$  words. Using our techniques, the memory complexity can be reduced significantly to  $2^{64}$ , while maintaining the same data and time complexities.

In the round function  $F_i$  of CAST-128, the 32 LSBs of the 37-bit round key  $K_i$  (denoted as  $K_{m_i}$ ) are first either XORed, added (modulo  $2^{32}$ ), or subtracted (modulo  $2^{32}$ ) from  $R_{i-1}$  (depending on the round). Then, the result is rotated to the left by 0–31 bits, according to the value of the 5 MSBs of  $K_i$  (denoted as  $K_{r_i}$ ). Finally, a key-less function  $f_i$  is applied to the result. The first round of CAST-128 (that uses modular addition) is shown in Fig. 4.

Since each round key of CAST-128 is of 37 bits, the time complexity of a basic MITM attack on a 8-round variant is  $2^{4 \cdot 37} = 2^{148}$ . Using the generic attack of [12] on 8-round Feistel structures (described in Sect. 2.3), the time complexity can be reduced to  $2^{3 \cdot 37 + 16} = 2^{127}$ , which is only slightly faster than exhaustive key search. Isobe and Shibutani [12] showed that the specific structure of the round function of CAST-128 can be used to further reduce the time complexity to  $2^{118}$ . The main idea of [12] is that by fixing most of the ciphertext bits (in all ciphertexts) to a constant value and exploiting the specific round function structure, the amount of key material required for partial decryption can be reduced (and not only divided between the upper and lower halves of the MITM, like in the generic attack). To achieve this, [12] consider an equivalent 7-round Feistel structure, with different round functions  $F'_5, F'_6, F'_7$  that imitate the four round functions  $F_5, F_6, F_7, F_8$  for the specifically chosen ciphertexts. See the full version of this paper [10] for details of the attack.

As in case of the generic attack of Isobe and Shibutani discussed in Sect. 2.3, we can incorporate the advanced attack procedure in the “key guessing” part of our generic memory-efficient attack on 7-round Feistel structures. As a result, the memory complexity of the attack is reduced from  $2^{111}$  to  $2^{79}$ , without increasing the time and data complexities. The memory complexity can be further reduced using a refined attack that exploits the relatively simple round function of CAST-128. As we show in the full version of this paper [10], it is possible to guess two intermediate values  $R_3^1, R_3^2$  (instead of a single value in the generic 7-round attack) and to structures separate tables  $T_{upper1}, T_{upper2}$  for rounds 2,3 (and similarly, separate tables  $T_{lower1}, T_{lower2}$  for rounds 5,6). These tables make use of complex differential properties of  $F_2$  and  $F_6$  that simultaneously combine different operations over  $GF(2)$  and over  $GF(2^{32})$ . As a result, the memory complexity is reduced to  $2^{64}$  with no effect on the time complexity. The details of this (rather involved) attack are given in the full version of this paper [10].

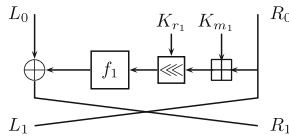


Fig. 4. The first round of CAST-128

### 4.3 Lower Memory Attacks on Other Cryptosystems

We conclude this section by mentioning briefly applications of our generic techniques to several other specific and generic structures.

1. **FOX.** Fox is a non-Feistel block cipher. The memory complexity of all the attacks of [13] on round-reduced variants of the block cipher FOX (namely, on 6 and 7-round FOX-64 and FOX-128), which are currently the best known attacks on FOX, can be reduced by a factor of  $2^{16}$  (with no change in the data or time complexities). We note that in these attacks, the 16 bits of filtering on which the attack iterates in the outer loop of the attack are not actual state bits, but rather linear combinations of state bits. However, our techniques are still applicable in this case, as in the inner loop, for each side of the attack, we simply complement these linear combinations to obtain an intermediate encryption state of FOX, and invert its round function as done in our generic attacks.
2. **Camellia.** The memory complexity of the attacks of [12] on reduced variants of Camellia can be reduced by a factor of at least  $2^{16}$  (depending on the attack) with no effect on the data or time complexities. We note however that MITM attacks are not the best known attacks on Camellia (in terms of the number of rounds).<sup>10</sup>
3. **Feistel-2 Schemes.** The memory complexity of the attacks of [12] on the 8 and 9-round Feistel-2 scheme (which is a more specific Feistel implementation compared to the generic Feistel-1) with a  $2n$ -bit key can be reduced from about  $2^{1.5n}$  to  $2^n$  (with no change in the data or time complexities). We note that these MITM attacks are the best known attacks on this specific Feistel-2 only when the data complexity is limited. With large data complexity in the chosen plaintext model, the attacks of [11] have superior time complexities.

## 5 Conclusions

In this paper we introduced some new cryptanalytic techniques, and combined the known techniques of MITM and dissection in new ways which enabled us to merge their advantages and avoid their disadvantages. Taken together, these techniques allowed us to develop improved attacks on Feistel structures with more than four rounds, and to improve the best known concrete attacks on several well known block ciphers.

## A On the Invertibility of the Round Function

We first note that our 5-round only needs two round functions to be efficiently invertible — namely,  $F_2(\cdot)$  and  $F_4(\cdot)$ . As noted before, we are not aware of any Feistel cipher which does not possess this property. Even DEAL [16], whose

<sup>10</sup> Although they reach fewer rounds, the advantage of MITM attacks over other attacks on Camellia (namely, impossible differential attacks [5]) is their low data complexity.

round functions are full DES encryptions, has invertible round functions given the subkey, as our attack described in Sect. 4.1 shows.

Moreover, we can relax a bit the requirement over the invertibility of the round functions  $F_2(\cdot)$  and  $F_4(\cdot)$ . We remind the reader that we are allowed  $2^{0.5n}$  memory, which can help in inverting the round functions. For example, if the cipher is a Feistel-2 structure (i.e., the round function is  $F_i(K_i, I_i) = G_i(K_i \oplus I_i)$ , for some completely one-way function  $G_i(\cdot)$ ), a simple enumeration of all input/output pairs of  $G_i(\cdot)$  is sufficient to invert the round function.

Finally, we note that when we discuss the general Feistel-2 structure, the memory complexity can be slightly reduced, as no memory is needed for the meet in the middle step in itself. For example, instead of structure  $T_1$ , given  $O_2^1$ , we invert  $G_2$ , to obtain  $I_2^1 \oplus K_2$ . Hence, for any  $K_1$  value, it is possible to immediately obtain the corresponding  $K_2$ .

## References

1. Adams, C.: The CAST-128 Encryption Algorithm. RFC 2144 (1997). <https://tools.ietf.org/html/rfc2144>
2. Aoki, K., Sasaki, Y.: Preimage attacks on one-block MD4, 63-step MD5 and more. In: Avanzi, R.M., Kelihher, L., Sica, F. (eds.) SAC 2008. LNCS, vol. 5381, pp. 103–119. Springer, Heidelberg (2009)
3. Bogdanov, A., Khovratovich, D., Rechberger, C.: Biclique cryptanalysis of the full AES. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 344–371. Springer, Heidelberg (2011)
4. Bogdanov, A., Rechberger, C.: A 3-subset meet-in-the-middle attack: cryptanalysis of the lightweight block cipher KTANTAN. In: Biryukov, A., Gong, G., Stinson, D.R. (eds.) SAC 2010. LNCS, vol. 6544, pp. 229–240. Springer, Heidelberg (2011)
5. Boura, C., Naya-Plasencia, M., Suder, V.: Scrutinizing and improving impossible differential attacks: applications to CLEFIA, Camellia, LBlock and SIMON. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8873, pp. 179–199. Springer, Heidelberg (2014)
6. Canteaut, A., Naya-Plasencia, M., Vayssière, B.: Sieve-in-the-middle: improved MITM attacks. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 222–240. Springer, Heidelberg (2013)
7. Chaum, D., Evertse, J.-H.: Cryptanalysis of DES with a reduced number of rounds: sequences of linear factors in block ciphers. In: Williams, H.C. (ed.) CRYPTO 1985. LNCS, vol. 218, pp. 192–211. Springer, Heidelberg (1986)
8. Diffie, W., Hellman, M.E.: Cryptanalysis of the NBS data encryption standard. *Computer* **10**(6), 74–84 (1977)
9. Dinur, I., Dunkelman, O., Keller, N., Shamir, A.: Efficient dissection of composite problems, with applications to cryptanalysis, knapsacks, and combinatorial search problems. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 719–740. Springer, Heidelberg (2012)
10. Dinur, I., Dunkelman, O., Keller, N., Shamir, A.: New attacks on Feistel structures with improved memory complexities. *IACR Cryptology ePrint Arch.* **2015**, 146 (2015)
11. Guo, J., Jean, J., Nikolić, I., Sasaki, Y.: Meet-in-the-middle attacks on generic Feistel constructions. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8873, pp. 458–477. Springer, Heidelberg (2014)

12. Isobe, T., Shibutani, K.: Generic key recovery attack on Feistel scheme. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part I. LNCS, vol. 8269, pp. 464–485. Springer, Heidelberg (2013)
13. Isobe, T., Shibutani, K.: Improved All-subkeys recovery attacks on FOX, KATAN and SHACAL-2 block ciphers. In: Cid, C., Rechberger, C. (eds.) FSE 2014. LNCS, vol. 8540, pp. 104–126. Springer, Heidelberg (2015)
14. Junod, P., Vaudenay, S.: FOX : a new family of block ciphers. In: Handschuh, H., Hasan, M.A. (eds.) SAC 2004. LNCS, vol. 3357, pp. 114–129. Springer, Heidelberg (2004)
15. Khovratovich, D., Rechberger, C., Savelieva, A.: Bieliques for preimages: attacks on skein-512 and the SHA-2 family. In: Canteaut, A. (ed.) FSE 2012. LNCS, vol. 7549, pp. 244–263. Springer, Heidelberg (2012)
16. Knudsen, L.: DEAL - A 128-bit Block Cipher. NIST AES Proposal (1998)
17. Luby, M., Rackoff, C.: How to construct pseudorandom permutations from pseudorandom functions. *SIAM J. Comput.* **17**(2), 373–386 (1988)
18. National Bureau of Standards. Data encryption standard. Federal Information Processing Standards Publications (FIPS) 46 (1977)
19. Sarkar, P., Iwata, T. (eds.): Advances in Cryptology - ASIACRYPT 2014–20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7–11, 2014. Proceedings, Part I. Lecture Notes in Computer Science, vol. 8873. Springer, Heidelberg (2014)
20. Wang, L., Sasaki, Y.: Finding preimages of tiger up to 23 steps. In: Hong, S., Iwata, T. (eds.) FSE 2010. LNCS, vol. 6147, pp. 116–133. Springer, Heidelberg (2010)
21. Wang, M., Wang, X., Chow, K., Hui, L.C.K.: New Differential Cryptanalytic Results for Reduced-Round CAST-128. *IEICE Trans.* **93**(12), 2744–2754 (2010)

# Known-Key Distinguisher on Full PRESENT

Céline Blondeau<sup>1</sup>, Thomas Peyrin<sup>2</sup>, and Lei Wang<sup>2,3</sup> (✉)

<sup>1</sup> Department of Computer Science, School of Science, Aalto University,  
Aalto, Finland

`celine.blondeau@aalto.fi`

<sup>2</sup> Nanyang Technological University, Singapore, Singapore

`thomas.peyrin@ntu.edu.sg`

<sup>3</sup> Shanghai Jiao Tong University, Shanghai, China

`wangleihb83@gmail.com`

**Abstract.** In this article, we analyse the known-key security of the standardized PRESENT lightweight block cipher. Namely, we propose a known-key distinguisher on the full PRESENT, both 80- and 128-bit key versions. We first leverage the very latest advances in differential cryptanalysis on PRESENT, which are as strong as the best linear cryptanalysis in terms of number of attacked rounds. Differential properties are much easier to handle for a known-key distinguisher than linear properties, and we use a bias on the number of collisions on some predetermined input/output bits as distinguishing property. In order to reach the full PRESENT, we eventually introduce a new meet-in-the-middle layer to propagate the differential properties as far as possible. Our techniques have been implemented and verified on the small scale variant of PRESENT. While the known-key security model is very generous with the attacker, it makes sense in practice since PRESENT has been proposed as basic building block to design lightweight hash functions, where no secret is manipulated. Our distinguisher can for example apply to the compression function obtained by placing PRESENT in a Davies-Meyer mode. We emphasize that this is the very first attack that can reach the full number of rounds of the PRESENT block cipher.

**Keywords:** PRESENT · Known-key model · Distinguisher · Differential cryptanalysis · Linear cryptanalysis

## 1 Introduction

The pervasive deployment of tiny computational devices brings with it many interesting, and potentially difficult, security issues. Recently, lightweight cryptography has naturally attracted a lot of attention from the symmetric-key cryptography community and many lightweight block ciphers [8, 10, 18] and hash functions [2, 7, 17] have been proposed in the past few years. Among these primitives, PRESENT [8] is probably the one which has been the most scrutinized. It remains unbroken, even though many lightweight block ciphers have been successfully attacked. As such, it has become an ISO/IEC standard [19] and is now expected to be deployed in many industrial applications.



It is well known that block ciphers and hash functions are very close cryptographic primitives, as the latter can be built from the former and vice versa. For example, the Davies-Meyer construction or the Miyaguchi-Preneel construction can transform a secure block cipher into a secure compression function (which can in turn be used to build a secure hash function by plugging it into some domain extension algorithm). However, while the security is usually guaranteed with a security proof that considers the internal block cipher as a black-box, it is very important that this internal primitive presents no flaw whatsoever. A classical example is the devastating effect on the compression function security of weak keys for a block cipher [32], which are usually considered as a minor flaw for a block cipher if the set of these weak-keys is small.

Therefore, the security notions to consider for a block cipher will vary depending if this block cipher will be used in a hash function setting or not. In a hash setting, block cipher security models such as the known-key [22] (the attacker can know the key) or the chosen-key model (the attacker can choose the key) make sense since in practice the attacker has full access and control over the internal computations. Moreover, an attack in these models depicts a structural flaw of the cipher, while it should be desired to work with a primitive that does not have any flaw, even in the most generous security model for the attacker. Several known-key or chosen-key attacks on reduced-round AES-128 were published [14, 16, 21, 22], and Gilbert [15] eventually exhibited a known-key attack on the full 10 rounds with  $2^{64}$  computations.

PRESENT is a natural candidate to build lightweight compression functions and hash functions, and such constructions were proposed in [9]. It is therefore meaningful to study PRESENT even in security models very generous for the attacker. Thus far, the best secret-key attacks on PRESENT [5, 12] can reach 26 rounds over the 31 total. Related-key attacks (where the key is secret, but the attacker is allowed to ask queries for some keys related to the original one) are thus far not more powerful, probably due to the impossibility of the attacker to properly control linear/differential propagation in the PRESENT key schedule. Regarding known or chosen-key model, the best attack could only reach 18 rounds [23] using rebound attacks and multi-differential cryptanalysis (their distinguisher worked not only for the internal block cipher, but also for the DM-PRESENT compression function). Only very recently Lauridsen et al. [25] managed to reach 26 rounds of 80-bit key version of PRESENT and 27 rounds of 128-bit key version of PRESENT by combining rebound attacks with linear cryptanalysis in the known-key model (it works for only a small portion of keys, *e.g.*  $2^{71.8}$  out of  $2^{128}$ ). It is noticeable that even though the security margin of PRESENT is rather small, the best attacks in the classical (secret)-single-key model and in the known or chosen-key models are almost reaching the same number of rounds. This is quite surprising as one would expect many more rounds to be broken when the attacker is given so much power. As analogy, one can remark that the best attacks on AES-128 can break 7 rounds in the single-key or related-key model, but the full 10-round cipher can be broken when the attacker knows the key [15]. It seems that, in the case of PRESENT, leveraging the degrees of freedom obtained by knowing or choosing the key will not greatly help the attacker to improve linear attacks [25].

**Our Contribution.** In this article, we exhibit the very first known-key attack on the full PRESENT cipher. More precisely, using the framework from [5], we avoid the issues when trying to improve linear attacks with more freedom degrees. We start from some of the best differential distinguishers from [5] and we managed to extend them by several rounds by adding a meet-in-the-middle layer. Overall, storing  $2^{35.58}$  bytes, we can distinguish the full 31-round PRESENT in the known-key model in a time corresponding to  $2^{56}$  encryptions. The success of our known-key distinguisher on the full PRESENT is 50.5% and is equal to 100% when considering a version reduced to 27 rounds. More details are provided in Table 3. The distinguishing attacks presented in this paper are independent of the key-size and are valid for both PRESENT-80 and PRESENT-128.

In order to validate our results, we have implemented and verified our distinguisher on a small scale variant of PRESENT proposed in [26]. Our findings indicate that one should avoid using PRESENT as building block to create compression functions and hash functions, as it was proposed in [9]. Actually, our distinguisher can also apply to DM-PRESENT (both 80 and 128-bit versions) and to H-PRESENT. We emphasize that this cryptanalysis is the very first non-random property found for the full PRESENT.

In Sect. 2 we first describe the PRESENT block cipher and then we introduce our attack model in Sect. 3. Then, we explain the method to build our distinguisher in Sect. 4 and finally provide experiments and summarize our results in Sect. 5. Section 6 concludes this paper.

## 2 The PRESENT Block Cipher

### 2.1 Description of PRESENT

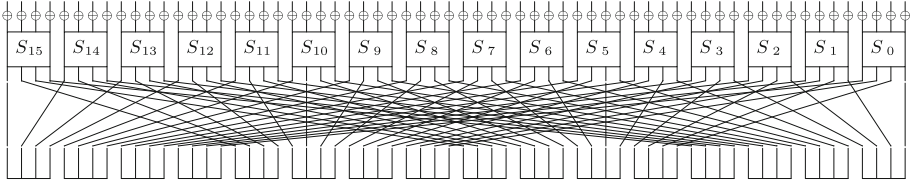
PRESENT [8] is a 64-bit lightweight block cipher proposed at CHES 2007. It is composed of 31 rounds and the 64-bit 32 round-keys are derived from a 80-bit or a 128-bit master key (we will respectively refer to PRESENT-80 or PRESENT-128). The round function is composed of a round-key XOR, an Sbox layer and a simple linear bit permutation layer, as depicted in Fig. 1.

The permutation layer operates linearly on the 64 bits as follows: the bit  $i$  of the state is moved to the bit position  $P(i)$  where

$$P(i) = \begin{cases} 16 \times i \bmod (63) & \text{for } 0 \leq i < 63, \\ 63 & \text{for } i = 63. \end{cases}$$

Even though the same Sbox is applied for all nibbles at each round, we numbered the Sboxes from 0 to 15 (see Fig. 1) to simplify the description of our attacks. Note that, as in the original PRESENT paper, the least significant bit and the least significant Sbox are on the right. In particular the input of the Sbox  $S_i$ ,  $0 \leq i \leq 15$ , corresponds to the bits  $4i, 4i + 1, 4i + 2, 4i + 3$  denoted by  $[4i, 4i + 3]$ . The 4-bit PRESENT Sbox can be described with the following table in hexadecimal display:

$$S[] = \{0xC, 0x5, 0x6, 0xB, 0x9, 0x0, 0xA, 0xD, 0x3, 0xE, 0xF, 0x8, 0x4, 0x7, 0x1, 0x2\}.$$



**Fig. 1.** One round of PRESENT.

We do not describe the key schedule of PRESENT as it has no impact on our attack, yet we refer to [8] for a more complete description of the cipher.

## 2.2 Previous Results on PRESENT

In the last couple of years, various analyses [1, 3–5, 12, 13, 27, 30, 31] on reduced versions of PRESENT in the (secret)-single-key model have been proposed. Among these analyses, the most important one remains the multidimensional linear attack from Cho [12], which takes advantage of the easy-to-trace linear trails with large correlations, and eventually threatens the security of PRESENT up to 26 rounds. Until 2014, as shown in Table 1, linear cryptanalysis-based attacks were much more powerful against PRESENT, as only 19 rounds were reachable using differential cryptanalysis-based attacks. Some of these key-recovery attacks take advantage of the key-schedule and their time complexity have often been computed for a single version.

However, based on a link between differential probability and linear correlation, it has recently been shown in [5] that one can convert a multidimensional linear distinguisher into a truncated differential one. In Sect. 4.2 we will provide more details on this technique, which permitted to push truncated differential attacks up to 26 rounds of PRESENT.

Regarding known-key or chosen-key settings, in [23] the authors presented an analysis of DM-PRESENT, i.e. the compression function built by placing PRESENT in a Davis-Meyer mode. Based on a combination of differential distinguishers and rebound techniques, they manage to obtain collision and second-preimage attacks on 12 rounds of DM-PRESENT-80, but also a distinguisher up to 18 rounds (this distinguisher can also be applied to the cipher itself). Nevertheless, this approach does not seem to be the most promising one since using classical methods, simple differential-based attacks on PRESENT have been much less powerful than linear-based attacks, as illustrated by Table 1. In the next sections, we will use the model provided in [5] to take advantage of much longer truncated differential distinguishers.

Very recently, Lauridsen et al. [25] combined linear cryptanalysis and rebound attacks to obtain known-key distinguishers on the PRESENT cipher, for both 80-bit and 128-bit versions. Eventually, they managed to reach 27 rounds, that is one more round than the best (secret)-single-key model attack on PRESENT. Their distinguisher on 27 rounds of PRESENT-128 requires  $2^{10}$  computations and  $2^{61.67}$

**Table 1.** Relevant attacks on PRESENT in the (secret)-single-key model

#Rounds	Version	Attack	Data	Time	Mem	Ref	Year
16	80	differential	$2^{64.0}$	$2^{64.0}$	$2^{32.0}$	[30]	2008
19	128	algebraic differential	$2^{62.0}$	$2^{113.0}$	n/r	[1]	2009
19	128	multiple differential	$2^{62}$	$2^{120}$	$2^{60}$	[4]	2013
25	128	linear	$2^{64.0}$	$2^{96.7}$	$2^{40.0}$	[28]	2009
26	80	multidimensional linear	$2^{64.0}$	$2^{72.0}$	$2^{32.0}$	[12]	2010
26	80	truncated differential	$2^{63.16}$	$2^{76.0}$	$2^{29.0}$	[5]	2014

steps of verification, but works only with probability  $2^{-56.2}$  since the distinguisher is considered valid for  $2^{71.8}$  keys among the  $2^{128}$  possible. The issue with this method is that it seems not very well fit for known-key or chosen-key scenarios, as only one extra round is reached compared to the best attack in the (secret)-single-key model.

In the next sections, we will describe a meet-in-the-middle approach that fits very well with differential-based attacks, and that will allow us to reach 5 more rounds compared to the best attack in the (secret)-single-key model. Moreover, our distinguishers can apply to DM-PRESENT and H-PRESENT [9] as well.

### 3 Known-Key Distinguisher

The known-key model has been introduced by Knudsen and Rijmen [22] to analyse the security of AES-128 and some Feistel-based ciphers. The goal of this model was to get a better estimation of the security margin of a cipher, but also to encompass the scenario of block cipher-based hashing, where the key is known and even chosen by the attacker. The property exhibited for their distinguisher was an integral structure on the input and output of a set of plaintext/ciphertext pairs, for a given known key. Several other types of known-key distinguishers were subsequently proposed, such as the subspace distinguisher [24], the limited-birthday distinguisher [16, 20], and more recently a quite complex property related to an integral structure was described by Gilbert [15] to reach the full AES-128.

When one proposes a new known-key distinguisher, it is important to prove or at least give very strong arguments that there is no generic attack that can obtain the same property with an equal or lower complexity than the distinguisher. In other words, an attacker having only blackbox access to encryption and decryption oracles of the cipher should not be able to obtain the same property with equal or lower complexity than for the distinguisher. In our case, the property we will exhibit is quite trivial: we will observe a bias on the number of collisions on some predetermined input and output bits for a set of many plaintext/ciphertext pairs. More precisely, let the cipher block size to be  $n$  bits and let  $s$  (respectively  $q$ ) denote the number of bits from the input (respectively the

output) on which we will observe these collisions. We will generate  $N$  messages, such that they all have the same value on the  $s$  input bits and such that there is a bias on the number of collisions observed on the  $q$  output bits.

When having blackbox access to encryption/decryption oracles, an attacker would maximise his success rate by asking only encryption queries. Indeed it is much harder for him to ensure that he will get exactly the required value on the  $s$  input bits (which is basically the strongest bias possible) when asking decryption queries, than trying to obtain a weak bias on the  $q$  output bits when asking encryption queries. In other words, the best strategy for him is to ensure that the strongest bias is pre-verified when building its queries, and then hoping to observe the weakest bias on the outputs of the oracle. Moreover, to further maximize his success rate, all its encryption queries should have the same value on the  $s$  input bits. Indeed, since the encryption oracle is a blackbox to him, all the queries which have different value on the  $s$  input bits can be considered completely independent, and therefore will not help him (this is similar to the reasoning given in the limited-birthday problem proof [20]). To summarize, in order to validate our distinguisher, we must compare with the generic attack that consists in simply picking  $N$  random inputs (all having the same value on the  $s$  predefined bits), querying them to the encryption oracle, and counting the number of collisions obtained on the  $q$  predefined bits of the output. In Sect. 4.2, we will explain the details regarding the computation of the distinguisher's success probability against this type of generic attacker.

Moreover, it is important that this exhibited property can be checked efficiently, and one should count this cost in the overall complexity of the distinguisher. Our distinguishing property can be very easily checked by simply verifying that all the  $N$  plaintext/ciphertext pairs have indeed the same value on the  $s$  predefined input bits, and by maintaining counters for each possible value taken by the  $q$  predefined ciphertext bits. Then, according to these counters, the distinguisher will compute a simple scoring function and decide if he believes to be communicating with PRESENT or with a random permutation (see Sect. 4.2). We note that our known-key distinguishers will work for any key choice. Thus, one can actually have the key value used as challenge for the attacker, which further confirms the validity of our model.

## 4 Distinguishing Full PRESENT

### 4.1 Distinguisher Overview

While previous known-key distinguishers on other ciphers benefit much from a start-from-the-middle approach, it cannot effectively be applied to PRESENT (at least in a straightforward way). Typically, those distinguishers are built upon a differential characteristic with desired input/output differences such that extra short differential characteristics with high probability can be pre- and post-added in order to attack as many rounds as possible. The start-from-the-middle approach is then to firstly find solutions of the intermediate differential characteristic. Although such a characteristic usually has a very low probability, thanks

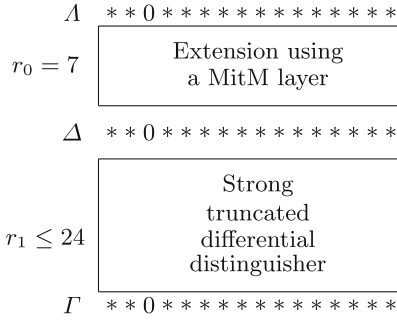
to the degrees of freedom obtained from knowing or choosing the key and the internal state values, the distinguisher is able to efficiently compute its solutions. After that, the distinguisher propagates these solutions backwards and forwards to probabilistically satisfy the pre- and post-added extra characteristics. As demonstrated in [23], a rather straightforward application of the start-from-the-middle approach works on very limited number of rounds of PRESENT. The difficulty comes from the impossibility to find an intermediate differential characteristic with a large number of rounds, while maintaining an affordable time complexity to find its solutions even leveraging the degree of freedom from knowing or choosing the key and the internal state values. We refer the interested readers to [23, 30].

Instead of differential characteristic, our distinguisher on PRESENT is built based on the truncated differential of [5]. This is motivated by the fact that truncated differential attack reaches the maximum number of attacked rounds so far as shown in Table 1. Moreover, as far as we know, it is much easier to handle than multidimensional linear attack. Hence, the distinguishing property is a statistical bias of the number of collisions on a few predetermined output bits, where the inputs collide on a few predetermined input bits. We note that such a bias is a very small value and thus cannot be observed with a non-negligible success probability unless a very large set of input/output pairs are provided. Moreover, the necessary number of input/output pairs increases with the number of attacked rounds of truncated differential in order to have a non-negligible success probability. Therefore, there are two issues when adding extra rounds to the truncated differential to extend the number of attacked rounds of the distinguisher. The first one is that we cannot post-add extra rounds, because the predetermined colliding output bits of truncated differential will be input to different Sboxes in the next round and as a result the bias cannot be observed any more from the final outputs. The second one is that if we pre-add a differential characteristic, it sets extra constraints on the inputs of the truncated differential, i.e. the inputs must satisfy the extra differential characteristic, which consequently reduces the total number of available inputs to the truncated differential and, a fortiori, lowers the success probability of the distinguisher. On one hand, one surely prefers to use a longer extra characteristic in order to attack more rounds. On the other hand, a longer extra characteristic sets more constraints on the inputs of the truncated differential path. Particularly, if the total number of the available inputs of truncated differential is lower than the necessary number to observe the statistical bias, the overall distinguisher fails.

Thus instead of pre-adding extra differential characteristics, we propose a new layer called meet-in-the-middle (MitM) layer in order to pre-add extra rounds to the truncated differential. It sets constraints only at its input bits and its output bits, but not at any of its internal state bits. More precisely, the constraints on its input bits is trivially due to defining the distinguishing property.<sup>1</sup>

---

<sup>1</sup> When attacking DM-PRESENT or H-PRESENT, the input bits with constraints of the MitM layer must be located in the same bit positions with the output bits with constraints of the truncated differential, due to the feed-forward operation.



**Fig. 2.** Distinguisher overview. Each symbol represents 4 bits.

The constraints on its output bits are coming from the truncated differential, i.e. the output difference of the MitM layer must satisfy the input constraints of the truncated differential.

Before providing the details of our known-key distinguisher on PRESENT, we give a general overview. As illustrated in Fig. 2, the main idea is to take advantage of a strong truncated differential distinguisher ( $\Delta \rightarrow \Gamma$ ) over the  $r_1 \leq 24$  last rounds. We denote by  $p$  the probability of this distinguisher. From the knowledge of the key, using a MitM approach, we are able to generate a large number of plaintexts which fulfill the following property: for all plaintexts with input difference in the set  $A$ , their differences after 7 rounds is in the set  $\Delta$ . The truncated differential distinguisher is described in Sect. 4.2, the MitM layer in Sect. 4.3.

### 4.2 A Statistical Bias on Reduced-Round PRESENT

Given an  $n$ -bit permutation  $F$ , splitting the input space into  $s + t$  bits and the output space into  $q + r$  bits, we have the following results which link the probability of a truncated differential with the capacity of a multidimensional linear approximation.

**Theorem 1** [5]. *Let  $\mathbb{F}_2^n = \mathbb{F}_2^s \times \mathbb{F}_2^t = \mathbb{F}_2^q \times \mathbb{F}_2^r$  and*

$$F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n, x = (x_s, x_t) \mapsto (y_q, y_r).$$

*Given a multidimensional approximation  $[(a_s, 0), (b_q, 0)]_{a_s \in \mathbb{F}_2^s, b_q \in \mathbb{F}_2^q}$  with capacity*

$$C = \sum_{(a_s, b_q) \neq (0,0)} \mathbf{cor}^2(a_s \cdot x_s \oplus b_q \cdot y_q),$$

*and a truncated differential composed of  $2^t$  input differences  $(0, \delta_t) \in \{0\} \times \mathbb{F}_2^t$ , and  $2^r$  output differences  $(0, \gamma_r) \in \{0\} \times \mathbb{F}_2^r$  with probability*

$$p = \frac{1}{2^q} \sum_{\delta_t, \gamma_r \in \mathbb{F}_2^t \times \mathbb{F}_2^r} \mathbf{P}[(0, \delta_t) \rightarrow (0, \gamma_r)],$$

where  $\mathbf{P}[(0, \delta_t) \xrightarrow{F} (0, \Delta_r)] = 2^{-n} \#\{x \in \mathbb{F}_2^n \mid F(x) \oplus F(x \oplus (0, \delta_t)) = (0, \gamma_r)\}$ . We have

$$p = 2^{-q}(C + 1). \quad (1)$$

**Truncated Differential with Strong Bias.** While the previous attack on DM-PRESENT [23] is derived from a differential with high probability, in this paper, we take advantage of the strong relation between differential probability and capacity to derive a large truncated differential over up to 24 rounds with large bias.

Throughout this paper, we make a distinction on the set of output differences depending if we want to distinguish PRESENT or DM-PRESENT. For our distinguisher on the PRESENT cipher, the sets of differences  $\Delta$  and  $\Gamma$  do not need to be similar and the last permutation can be omitted which is not the case if we want to distinguish DM-PRESENT or H-PRESENT. Depending of the context, the set of considered output differences will be denoted  $\Gamma$  if it is placed after the last permutation layer and  $\Gamma'$  if it is placed after the last Sbox layer.

We denote by  $I$  (resp.  $J$ ) the number of Sboxes in the first round (resp. last round) with no difference in their input. In the context of the distinguishing attack on PRESENT, we express Theorem 1 as follows.

**Corollary 1.** *Given a multidimensional linear approximation involving the input bits  $\cup_{i \in \{i_1, \dots, i_I\}} [4i, 4i + 3]$  and the output bits  $\cup_{j \in \{j_1, \dots, j_J\}} [4j, 4j + 3]$  with capacity  $C$ , we have a truncated differential with input difference*

$$\Delta = \{\delta = (\delta_0, \dots, \delta_{63}) \in \mathbb{F}_2^{64} \mid \delta_b = 0 \text{ for } b \in \cup_{i \in \{i_1, \dots, i_I\}} [4i, 4i + 3]\},$$

and output difference

$$\Gamma = \{\gamma = (\gamma_0, \dots, \gamma_{63}), \in \mathbb{F}_2^{64} \mid \gamma_b = 0 \text{ for } b \in \cup_{j \in \{j_1, \dots, j_J\}} [4j, 4j + 3]\}.$$

The probability of this differential is given as  $p = 2^{-4J}(C + 1) = 2^{-4J} + 2^{-4J}C$ . Following the notation of Theorem 1, we have  $s = 4I$  and  $q = 4J$ . We call  $2^{-4J}C$  the bias of this truncated differential approximation.

While in classical truncated differential attacks only few differentials are involved, for this distinguisher derived from a multidimensional linear distinguisher the number of involved differentials is  $2^{128-4I-4J}$  (as PRESENT is a 64-bit cipher).

Part of the analysis consists at selecting a truncated differential with high relative bias. To understand the meaning of high relative bias we first study the success of a distinguishing truncated differential attack.

**Success of the Distinguishing Attack.** Given a truncated differential with probability  $p = 2^{-q}(C + 1)$ , we use the following method with data complexity  $N$ , time complexity  $N$  encryptions and negligible memory complexity to distinguish the cipher from a random permutation.

1. Set a table  $T$  of size  $2^q$  to 0



2. For all  $N$  messages  $x$  with same value on the bits  $\cup_{i \in \{i_1, \dots, i_I\}} [4i, 4i + 3]$ 
  - (a) Compute  $y = E_K(x)$
  - (b) Given  $y_q$  the truncation of  $y$  reduced to  $q$  bits, increment  $T[y_q]$
3. Compute  $D = \sum_{0 \leq \ell \leq 2^q - 1} T[\ell](T[\ell] - 1)/2$
4. If  $D > \tau$ , consider that this is the cipher

Without comparing the pairs directly, the scoring function  $D$  gives us number of pairs which fulfill the differential [5]. From  $N$  messages with same values on the bits  $\cup_{i \in \{i_1, \dots, i_I\}} [4i, 4i + 3]$  we can generate  $N_S = N^2/2$  pairs of message with no difference on these bits, meaning that for a random permutation the expected number of pairs fulfilling the truncated differential should be  $\mu_W = N_S \cdot 2^{-q} = N_S \cdot 2^{-4J}$ . We can show [6] that the random variable  $\mathcal{D}_R$  corresponding to this scoring function for the given permutation follows a normal distribution with mean  $\mu_R = N_S \cdot 2^{-q}(1 + C)$  and variance  $\sigma_R^2 \approx N_S \cdot 2^{-q}(1 + C) \approx N_S \cdot 2^{-q}$ . On the other hand we have  $\mu_W = N_S \cdot 2^{-q}$  and  $\sigma_W^2 \approx N_S \cdot 2^{-q}$ . We can show that when using  $N_S$  pairs, the success probability  $P_S$  of the distinguishing attack is given by,

$$P_S(N_S) = \Phi \left( \frac{\mu_R - \mu_W}{\sigma_R + \sigma_W} \right) \approx \Phi \left( \frac{\sqrt{2^{-q} N_S} \cdot C}{2} \right) \tag{2}$$

where  $\Phi$  the cumulative distribution function of the central normal distribution. This success probability corresponds to a threshold  $\tau = \mu_R - \sigma_R \cdot \Phi^{-1}(P_S) = \mu_W + \sigma_W \cdot \Phi^{-1}(P_S)$ .

**Strong Truncated Differential on PRESENT.** For a fixed  $N_S$  number of pairs, we derive from (2) that the best truncated differentials are the ones which maximize  $2^{-q/2}C$ . As explained in the previous section the number  $N_S$  of available pairs is fixed by the MitM part and the size of  $\Lambda, \Delta$ . For the purpose of this attack, we computed the capacity  $C$  of different set of linear approximations. From this analysis it turns out that in combination with the MitM phase, if we want to be able to transform this known distinguisher to a distinguisher on DM-PRESENT, the best choice is achieved for  $I = 1$  and  $J = 1$ .

As explained in [12], the capacity of a multidimensional linear approximation can be obtained from the 1-bit linear trails. Given the multidimensional linear input space involving the bits  $[4i, 4i + 3]$ ,  $0 \leq i \leq 15$  and an output space after the Sbox layer involving the bits  $[4j, 4j + 3]$ ,  $0 \leq j \leq 15$ , we denote by  $U$  the set  $\{P(4i), P(4i+1), P(4i+2), P(4i+3)\}$  and  $V$  the set  $\{4j, 4j+1, 4j+2, 4j+3\}$ . We can show (see the explanation in [12]) that an estimate of the capacity  $C'_{r_1}$  over  $r_1$  rounds without the last linear layer is obtained from the following formula

$$C'_{r_1} = \sum_{u \in U, v \in V} M^{r_1-2}[u, v], \tag{3}$$

where  $M$  denotes the  $64 \times 64$  matrix with coefficients the square correlation of the 1-bit linear approximations over one round in rest of the paper. On the other hand, when the last linear layer is included, since the linear trails activate

different Sboxes in the last round, we can estimate the capacity  $C_{r_1}$  over  $r_1$  rounds as follows

$$C_{r_1} = \sum_{u \in U, v \in V} M^{r_1-1}[u, v]. \tag{4}$$

From our computation we found that when selecting  $\Delta = \{\delta | \delta_b = 0 \text{ for } b \in [52, 55]\}$ , the best truncated differentials are obtained for  $\Gamma'_i = \{\gamma' | \gamma'_b = 0 \text{ for } b \in [4i, 4i + 3]\}$  and  $i = 5, 7, 13, 15$ . For instance such truncated differential distinguisher on 24 rounds has a probability of  $2^{-4}(1 + 2^{-58.77}) = 2^{-4} + 2^{-62.77}$  to be fulfilled. By using  $2^{56}$  messages (the reason of this number is due to the MitM layer explained in Sect. 4.3), we can distinguish 24-round of PRESENT from a random permutation in 50.5% of the cases.

In the next section we explain how in the known-key model we can extend this distinguisher to reach more rounds. More explicitly we explain how we can ensure that all the generated messages have a fixed value over the bits [52, 55] after 7 rounds.

### 4.3 The Meet-in-the-Middle Layer

This section illustrates the meet-in-the-middle (MitM) layer, which is prepended to the truncated differential in order to extend the number of attacked rounds of the known-key distinguisher. It consists of several rounds, and sets constraints on the differences of input/output bits of these rounds. Moreover, the constraints on the output bits of the MitM layer must be exactly the same with those set on the input bits of the truncated differential layer. Then next is to identify a set of plaintexts which can satisfy the constraints on both input and output of the MitM layer. Namely, if these plaintexts are input to PRESENT, their internal state after several rounds as the output of the MitM layer can satisfy the input constraints of the truncated differential. Thus these plaintexts can be used to launch a distinguisher on the (reduced) PRESENT consisting of both the MitM layer and the truncated differential layer. To efficiently identify such a set of plaintexts, we adopt a meet-in-the-middle approach, which benefits from the small Sbox and the bit-permutation linear layer of PRESENT. More precisely, for two rounds of computations, an input bit (or four input bits of an Sbox) interacts with only few other bits, and with those bits together can determine partial output bits. Thus we carry out a forward computation to get partial internal state bits for the first two rounds of the MitM layer by guessing just few bits. Similarly we carry out an independent backward computation to get partial internal state bits for the last one and half round of the MitM layer. Finally we carry out a gradually matching process to link and meanwhile fully determine the internal states obtained from the forward and from the backward computations, which can work up to 3 rounds.

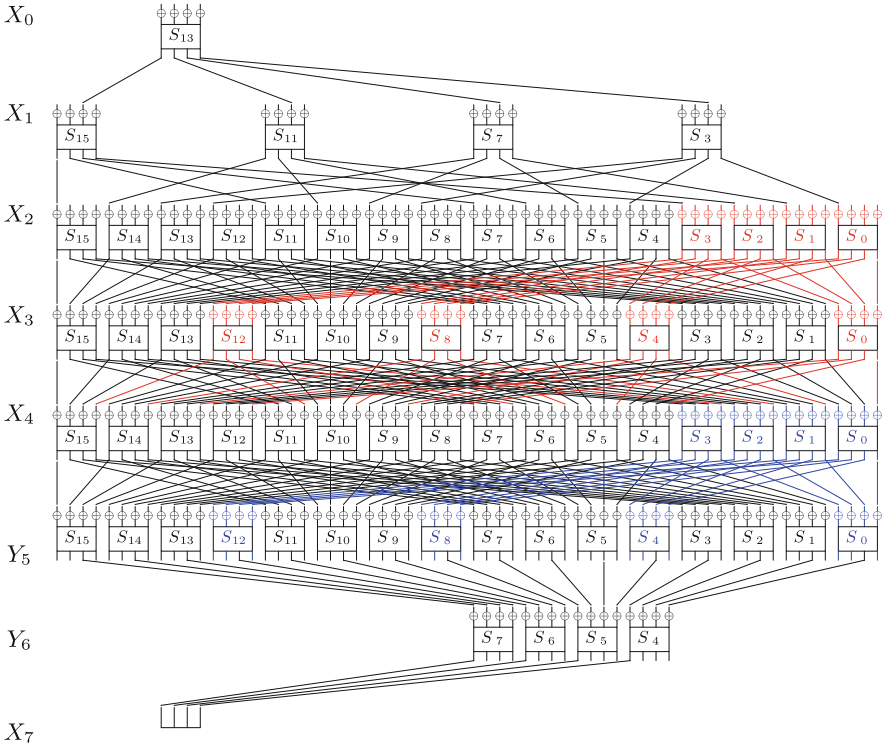
We describe in detail the concrete MitM procedure that is used in our attack on full PRESENT. It consists of 7 rounds. The constraints on the inputs are that they share the same values at bits [52, 55], i.e. the input bits to  $S_{13}$ . The constraints on the outputs are that they share the same values at bits [52, 55], i.e.

the input bits to  $S_{13}$  for next round. In this section, we denote by  $X_i$  the internal state after  $i$ -th round of PRESENT, and by  $Y_i$  the internal state after applying Sbox layer to  $X_i$ .

Firstly, we set the bits [52, 55] of plaintext to a randomly chosen 4-bit value, and compute bits 13, 29, 45 and 61 of  $X_1$  in the forward direction. These bits are input to Sboxes  $S_3$ ,  $S_7$ ,  $S_{11}$  and  $S_{15}$  in the second round. Then we exhaustively guess the other 12 bits input to these Sboxes, that include bits 12, 14, 15, 28, 30, 31, 44, 46, 47, 60, 62 and 63 of  $X_1$ , and continue to compute in the forward direction to get 16 bits of  $X_2$ , i.e. bits  $4i + 3$  for  $0 \leq i \leq 15$ . It is also depicted as the first two rounds in Fig. 3. In total we get a set of  $2^{12}$  such values of  $X_2$  and each value has 16 bits determined.

Secondly, we set the bits [52, 55] of  $X_7$  to a randomly chosen 4-bit value, and compute bits 19, 23, 27 and 31 of  $Y_6$  in the backward direction. These bits are input to compute the inversion of Sboxes  $S_4$ ,  $S_5$ ,  $S_6$  and  $S_7$  in sixth round. We guess the other 12 bits input to the inversion of these Sboxes, that include bits [16, 18, 20, 22, 24, 26, 28, 30], and continue to compute in the backward direction to get 16 bits of  $Y_5$ , i.e. bits  $4i + 1$  for  $0 \leq i \leq 15$ . It is also depicted as the last two rounds in Fig. 3. In total we get a set of  $2^{12}$  such values of  $Y_5$  and each value has 16 bits determined.

Finally, we carry out a gradually-matching algorithm for each pair of  $X_2$  and  $Y_5$  obtained from the forward and the backward computations respectively. Recall that each of  $X_2$  and  $Y_5$  has 16 bits fixed, which will be named *fixed bits* in the following description. The algorithm is to find a set of internal state values of  $X_4$ , whose corresponding values of  $X_2$  and  $Y_5$  can satisfy all the fixed bits, and in turn the corresponding plaintexts can satisfy the constraints on the input and output of the MitM layer. In details, at the third round of the MitM layer, we re-group the bits of  $X_2$  into 4 groups; the  $i$ -th group contains bits  $[16i, 16i + 15]$  for  $0 \leq i \leq 3$ . Hence each group contains input bits to 4 consecutive Sboxes, and has 4 bits fixed, i.e. bits  $16i + 3$ ,  $16i + 7$ ,  $16i + 11$  and  $16i + 15$  for the  $i$ -th group. Then for each group independently, we exhaustively guess its 12 unfixed bits, and compute in the forward direction to get 16 bits of  $X_4$ , that is bits  $4j + i$ ,  $0 \leq j \leq 15$ , for the  $i$ -th group. We store the values of partially determined  $X_4$  computed from the  $i$ -th group in a table  $TF_i$ . See Fig. 3 for an example group in red color. Independently and similarly, at the sixth round of the MitM layer, we also re-group the bits of  $Y_5$  to 4 groups; the  $i$ -th group contains bits  $[4i, 4i + 3] \cup [4i + 16, 4i + 19] \cup [4i + 32, 4i + 35] \cup [4i + 48, 4i + 51]$  for  $0 \leq i \leq 3$ . Then for each group independently, we exhaustively guess the unfixed 12 bits, and compute in the backward direction to get 16 bits of  $X_4$ , that is bits  $[16i, 16i + 15]$ , for the  $i$ -th group. We store the values of partially determined  $X_4$  computed from the  $i$ -th group in a table  $TB_i$ . See Fig. 3 for an example group in blue color. After that, we merge those tables to find a set of fully-determined values of  $X_4$ . To begin with, we merge  $TF_i$  and  $TB_i$ , and the merged table is denoted as  $T_i$ , independently for each  $0 \leq i \leq 3$ . By merging these two tables, we mean to merge every two partially-determined values of  $X_4$ , each from a table and sharing the same bit values at the common determined bit positions, into a



**Fig. 3.** MitM over the 7 first rounds of PRESENT

new (partially-determined) value of  $X_4$  with all their determined bits, and then to include this new value of  $X_4$  in table  $T_i$ . Note that each value of  $TF_i$  and each value of  $TB_i$  share 4 determined bit positions. Hence table  $T_i$  has on average  $2^{20}$  values. Then, we merge  $T_0$  and  $T_1$  and merge  $T_2$  and  $T_3$  independently, and store the results in two tables  $T_{0,1}$  and  $T_{2,3}$  respectively. As  $T_0$  (respectively  $T_2$ ) shares 8 common bits with  $T_1$  (respectively  $T_3$ ), we get that each of resulted tables has on average  $2^{32}$  values. In the end, we merge  $T_{0,1}$  and  $T_{2,3}$ , which gives on average  $2^{32}$  values of fully-determined  $X_4$  since they share 32 common bits.

Overall, there are  $2^{24}$  pairs of partially-determined  $X_2$  and  $Y_5$  obtained from the forward and the backward computations respectively, and each pair results on average  $2^{32}$  fully-determined values of  $X_4$ . Thus in total we can get on average  $2^{56(=24+32)}$  plaintexts by inversely computing from the fully-determined values of  $X_4$ , and these plaintexts can satisfy the constraints on the input and output of the MitM layer.

It is important to note that by running over all pairs of  $X_2$  and  $Y_5$ , we have filtered out all the plaintexts that can satisfy the constraints on both input and

output of the MitM layer. In fact it is trivial to evaluate the expected number of such plaintexts. Since there are 4 bit-constraints at bits [52, 55] of plaintext and 4 bit-constraints at bits [52, 55] of  $X_7$ , the expected number of desired plaintexts should be  $2^{56(=64-4-4)}$ . This means that on average (at most)  $2^{56}$  values can be input to the truncated differential, which contributes to  $2^{111}$  pairs, to observe the bias. It has an impact to the success probability of overall distinguisher. More details are given in Sect. 5.

**Complexity.** The complexities of both the forward computation and the backward computations are  $2^{12}$  computations of 2 PRESENT-rounds. For the gradually-matching phase, the algorithm is executed  $2^{24}$  times since there are  $2^{12}$   $X_2$  from the forward computation and  $2^{12}$   $Y_5$  from the backward computations. The complexity of each execution is obviously dominated by merging  $T_{0,1}$  and  $T_{2,3}$ , which needs  $2^{32}$  table lookups. Hence in total the complexity of the gradually-matching phase is  $2^{56}$  table lookups.

Once a match of the MitM layer has been found, we can encrypt this value  $X_4$  over the  $r_1 + 3$  rounds and increment the counter  $D$  given in the previous section. Therefore the memory complexity of this attack is dominated by the storage of the table  $T_{0,1}$  and  $T_{2,3}$  which is  $2 \cdot 2^{32} \cdot 6$  bytes. Overall the total time complexity of the distinguisher is  $2^{56}$  table lookups and  $2^{56}$  encryptions.

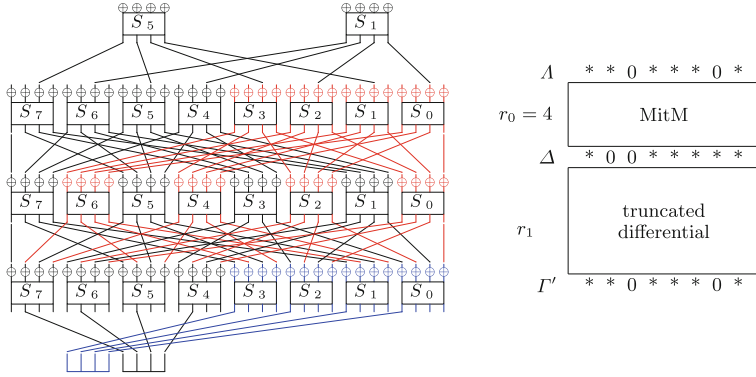
## 5 Results

### 5.1 Experiments

To confirm the validity of the distinguisher presented in this paper, we implemented a similar known-key distinguisher on SMALLPRESENT-[8], a 32-bit scaled-version of PRESENT [26]. A general overview of the cipher as well as a description of the parameters for this known-key distinguishing attack are provided in Fig. 4.

For this experimental attack with  $I = 2$  and  $J = 2$ , the expected number of messages obtained from the MitM layer should be  $2^{32-4I-4J} = 2^{16}$ . We repeated the experiments with different keys by 100000 times, and computed that the average number of generated messages was  $2^{16.0009}$ . We also computed the standard deviation of these experiments, which was  $2^{12.73}$ . Based on this deviation, we got that for more than 99.9% of the experiments, the number of messages generated by the MitM phase was greater than  $2^{15}$ . Therefore we take the value  $N = 2^{15}$  ( $N_G = N^2/2 = 2^{29}$ ) into consideration to compute a conservative success probability of the attack. A resume of the obtained results for  $5 \leq r_1 \leq 8$  and a comparison with the theoretical success probability obtained by formula (2) are given in Table 2.

As expected, these results confirm the validity of the known-key distinguishing model presented in this paper.



**Fig. 4.** Left: The MitM part of our experiments on SMALLPRESENT-[8]. Right: Description of the differential involved in our experimental attacks.

**Table 2.** Experimental attacks on SMALLPRESENT-[8]

#Rounds	$C$ over $r - 4$ rounds	$P_S(2^{29})$	Exp. $P_S$
9	$2^{-5.42}$	100 %	100 %
10	$2^{-8.46}$	98.0 %	96.6 %
11	$2^{-10.30}$	75.6 %	78.9 %
12	$2^{-16.17}$	54.3 %	54.6 %

### 5.2 Results

**Distinguisher on PRESENT.** The results of our known-key distinguishing attack on PRESENT are given in Table 3. The input difference set is  $\Lambda = \{\lambda \mid \lambda_b = 0 \text{ for } b \in [52, 55]\}$ . For this distinguishing attack we selected the output difference set after the last Sbox layer to be  $\Gamma' = \{\gamma' \mid \gamma'_b = 0 \text{ for } b \in [52, 55]\}$ . From this set  $\Gamma'$  we derive the following set of output differences after the last linear layer  $\{\gamma \mid \gamma_b = 0 \text{ for } b \in \{13, 29, 45, 61\}\}$

As from the MitM phase we can extend the truncated differential distinguisher over 7 rounds, its probability has been computed over  $r - 7$  rounds. As explained in Sect. 4.3, from the MitM phase we can generate in average  $2^{111}$  plaintext pairs with difference in the set  $\Lambda$  leading after 7 rounds to  $2^{111}$  pairs with difference in  $\Delta$ . As from the MitM phase the number of generated pairs is not the always same, we computed a conservative success probability assuming that from the MitM phase only  $2^{109}$  pairs are generated. Note that we have conducted experiments on the 7 first rounds of PRESENT showing that when merging  $TF_i$  and  $TB_i$ ,  $0 \leq i \leq 3$ , the expected number of matches was, as expected,  $2^{20}$  and the standard deviation was dependent of the group and lower than  $2^{14.5}$ . These experiments support the fact that assuming that only  $2^{109}$  pairs are generated should gives us an underestimate of the success of the attack. The memory complexity of this distinguishing attack is dominated by the storage of the tables

**Table 3.** Success probability of the known-key distinguisher ( $A \rightarrow I'$ ) on PRESENT. The probability of the truncated differential over  $r - 7$  rounds is obtained from the formula  $p = 2^{-4}(C'_{r-7} + 1)$  with  $C'_{r-7}$  computed from (3).

#Rounds	$C'_{r-7}$	$P_S(2^{111})$	$P_S(2^{109})$
27	$2^{-48.33}$	100 %	100 %
28	$2^{-50.94}$	99.8 %	93.0 %
29	$2^{-53.55}$	68.6 %	59.5 %
30	$2^{-56.16}$	53.2 %	51.5 %
31	$2^{-58.77}$	50.5 %	50.3 %

in the MitM phase and corresponds to the storage of  $2^{35.58}$  bytes. The time complexity of this known-key distinguisher on the full PRESENT is  $2^{56}$  table lookups plus  $2^{56}$  encryptions.

**Using Multiple Truncated Differentials.** In contrary to the distinguishers on DM-PRESENT and H-PRESENT, for the distinguisher on PRESENT the input and output differences can be selected independently of each other, in particular it is possible to consider different sets of output differences simultaneously. For instance we can simultaneously check that the output pairs have difference in the set  $I'_5$  or  $I'_7$  or  $I'_{13}$  or  $I'_{15}$  (see Sect. 4.2 for the notation). Meaning that we can simultaneously check that the output pairs have no-difference on the bits [20, 23] or [28, 31] or [52, 55] or [60, 63]. Given the four multidimensional linear approximations with input masks involving the bits [52, 55] and output masks involving the bits of one of the previous set with same capacity  $C$  we derive that the probability of the union of these four events is  $p \approx 4 \cdot 2^{-4}(1 + C)$ . The success of such multiple truncated differential distinguisher is

$$P_S \approx \Phi \left( \frac{N_S \cdot 4 \cdot 2^{-4}C}{2\sqrt{N_S \cdot 4 \cdot 2^{-4}}} \right) = \Phi \left( 2^{-2}\sqrt{N_S}C \right).$$

Using this multiple truncated differential distinguisher, having  $2^{109}$  plaintext pairs the success of the attack on 31 rounds is 50.5 %. Using different distribution table  $T_i$  and different counter values  $D_i$  for each set of differentials  $1 \leq i \leq 4$ , the time complexity of this attack remains the same than that of the simple distinguisher.

**Distinguisher on DM-PRESENT and H-PRESENT.** For these two compression functions DM-PRESENT and H-PRESENT, the last linear layer has to be considered. In particular, as explained in Sect. 4.2 the probability of a truncated differential distinguisher with output difference equal to 0 on the bits [52, 55] after the last linear layer can be computed from the capacity of the related multidimensional linear approximation using (4). From the linear properties of the Sbox

**Table 4.** Success probability of the known-key distinguisher ( $\Delta \rightarrow \Gamma'$ ) on DM-PRESENT and H-PRESENT. The probability of the truncated differential over  $r - 7$  rounds is obtained from the formula  $p = 2^{-3}(C_{r-7} + 1)$  with  $C_{r-7}$  computed from (4).  $N_S = 2^{111}$  has been chosen in a conservative way to have a good estimate of the success probability.

#Rounds	$C_{r-7}$	$P_S(2^{113})$	$P_S(2^{111})$
27	$2^{-50.94}$	100 %	100 %
28	$2^{-53.55}$	100 %	97.3 %
29	$2^{-56.16}$	73.7 %	62.4 %
30	$2^{-58.77}$	54.1 %	52.1 %
31	$2^{-61.39}$	50.7 %	50.3 %

of PRESENT, we derive the particularity that for all  $v \in \{4j\}_{0 \leq j < 16}$  we have  $M[u, v] = 0$ . Meaning that (4) is equivalent to

$$C_{r_1} = \sum_{u \in U, v \in V} M^{r_1-1}[u, v],$$

where  $V = \{4j + 1, 4j + 2, 4j + 3\}$ . Reducing the output multidimensional linear space from  $2^4$  values to  $2^3$  values we can increase the success of our distinguishing attack on PRESENT. In this case we define the set  $\tilde{\Gamma} = \{\gamma | \gamma_b = 0 \text{ for } b \in [53, 55]\}$ . Our truncated differential distinguisher ( $\Delta \rightarrow \tilde{\Gamma}$ ) has a probability  $p = 2^{-3}(C_{r_1} + 1)$ .

In this case the value of  $X_0[52]$  does not have to be fixed and the MitM presented in Sect. 4.3 can be repeated for the two values of  $X_0[52]$ , meaning that in average  $2^{57}$  messages ( $2^{113}$  pairs) with fixed  $X_0[53, 55]$  and  $X_7[52, 55]$  can be generated. The set  $\Lambda$  is now equal to  $\{\lambda | \lambda_b = 0 \text{ for } b \in [53, 55]\}$ . Meaning that an attack on 31-round DM-PRESENT and H-PRESENT can be performed in time  $2^{57}$  table lookups and  $2^{57}$  encryptions with success probability 50.3%. This distinguishing attack requires the storage of  $2^{35.58}$  bytes. The success of this distinguishing attack on reduced-round DM-PRESENT and H-PRESENT is given in Table 4.

The same attack with same success probability could also be performed on PRESENT. However, the time complexity of this new known-key distinguishing attack on PRESENT is twice the one of the previous attack.

## 6 Conclusion

In this article, we proposed a known-key distinguisher on the full PRESENT block cipher, in both 80- and 128-bit versions. This is the very first non-random property exhibited for the full number of rounds of this standardized cipher. It seems an interesting future work to analyse what an attacker would be able to do when not only knowing the key, but when he can actually choose it (chosen-key



model). Similarly to the previous strange situation that no attack improvement could be obtained when switching from the secret-key model to the known-key model, it would be surprising that no further improvement could be obtained in the chosen-key model.

**Acknowledgements.** The authors would like to thank the anonymous referees for their helpful comments. The second and third authors are supported by the Singapore National Research Foundation Fellowship 2012 (NRF-NRFF2012-06). Lei Wang is also supported by Major State Basic Research Development Program (973 Plan) (2013CB338004), National Natural Science Foundation of China (61472250), and Innovation Plan of Science and Technology of Shanghai (14511100300).

## References

1. Albrecht, M., Cid, C.: Algebraic techniques in differential cryptanalysis. In: Dunkelman, O. (ed.) FSE 2009. LNCS, vol. 5665, pp. 193–208. Springer, Heidelberg (2009)
2. Aumasson, J.-P., Henzen, L., Meier, W., Naya-Plasencia, M.: QUARK: a lightweight hash. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 1–15. Springer, Heidelberg (2010)
3. Blondeau, C., Gérard, B.: Multiple differential cryptanalysis: theory and practice. In: Joux, A. (ed.) FSE 2011. LNCS, vol. 6733, pp. 35–54. Springer, Heidelberg (2011)
4. Blondeau, C., Nyberg, K.: New links between differential and linear cryptanalysis. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 388–404. Springer, Heidelberg (2013)
5. Blondeau, C., Nyberg, K.: Links between truncated differential and multidimensional linear properties of block ciphers and underlying attack complexities. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 165–182. Springer, Heidelberg (2014)
6. Blondeau, C., Nyberg, K.: Links between truncated differential and multidimensional linear properties of block ciphers and underlying attack complexities. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 165–182. Springer, Heidelberg (2014)
7. Bogdanov, A., Knežević, M., Leander, G., Toz, D., Varıcı, K., Verbauwhede, I.: SPONGENT: a lightweight hash function. In: Preneel, B., Takagi, T. (eds.) CHES 2011 [29]. LNCS, vol. 6917, pp. 312–325. Springer, Heidelberg (2011)
8. Bogdanov, A.A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M., Seurin, Y., Vikkelsoe, C.: PRESENT: an ultra-lightweight block cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 450–466. Springer, Heidelberg (2007)
9. Bogdanov, A., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y.: Hash functions and RFID tags: mind the gap. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 283–299. Springer, Heidelberg (2008)
10. De Cannière, C., Dunkelman, O., Knežević, M.: KATAN and KTANTAN — a family of small and efficient hardware-oriented block ciphers. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 272–288. Springer, Heidelberg (2009)
11. Canteaut, A. (ed.): FSE 2012. LNCS, vol. 7549. Springer, Heidelberg (2012)

12. Cho, J.Y.: Linear cryptanalysis of reduced-round PRESENT. In: Pieprzyk, J. (ed.) CT-RSA 2010. LNCS, vol. 5985, pp. 302–317. Springer, Heidelberg (2010)
13. Collard, B., Standaert, F.-X.: A statistical saturation attack against the block cipher PRESENT. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 195–210. Springer, Heidelberg (2009)
14. Fouque, P.-A., Jean, J., Peyrin, T.: Structural evaluation of AES and chosen-key distinguisher of 9-round AES-128. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 183–203. Springer, Heidelberg (2013)
15. Gilbert, H.: A simplified representation of AES. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8873, pp. 200–222. Springer, Heidelberg (2014)
16. Gilbert, H., Peyrin, T.: Super-Sbox cryptanalysis: improved attacks for AES-Like permutations. In: Hong, S., Iwata, T. (eds.) FSE 2010. LNCS, vol. 6147, pp. 365–383. Springer, Heidelberg (2010)
17. Guo, J., Peyrin, T., Poschmann, A.: The PHOTON family of lightweight hash. In: Rogaway, P. (ed.) CRYPTO 2011 [30]. LNCS, vol. 6841, pp. 222–239. Springer, Heidelberg (2011)
18. Guo, J., Peyrin, T., Poschmann, A., Robshaw, M.: The LED block cipher. In: Preneel, B., Takagi, T. (eds.) CHES 2011 [29]. LNCS, vol. 6917, pp. 326–341. Springer, Heidelberg (2011)
19. ISO/IEC: information technology - security techniques - lightweight cryptography - part 2: block ciphers. ISO/IEC 29192-2:2012 (2012)
20. Iwamoto, M., Peyrin, T., Sasaki, Y.: Limited-birthday distinguishers for hash functions. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part II. LNCS, vol. 8270, pp. 504–523. Springer, Heidelberg (2013)
21. Jean, J., Naya-Plasencia, M., Peyrin, T.: Multiple limited-birthday distinguishers and applications. In: Lange, T., Lauter, K., Lisoněk, P. (eds.) SAC 2013. LNCS, vol. 8282, pp. 533–550. Springer, Heidelberg (2014)
22. Knudsen, L.R., Rijmen, V.: Known-Key distinguishers for some block ciphers. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 315–324. Springer, Heidelberg (2007)
23. Koyama, T., Sasaki, Y., Kunihiro, N.: Multi-differential cryptanalysis on reduced DM-PRESENT-80: collisions and other differential properties. In: Kwon, T., Lee, M.-K., Kwon, D. (eds.) ICISC 2012. LNCS, vol. 7839, pp. 352–367. Springer, Heidelberg (2013)
24. 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)
25. Lauridsen, M.M., Rechberger, C.: Linear distinguishers in the key-less setting: application to PRESENT. In: Leander, G. (ed.) Fast Software Encryption - FSE 2015. Lecture Notes in Computer Science. Springer (2015, to appear)
26. Leander, G.: Small scale variants of the block cipher PRESENT. Cryptology ePrint Archive, Report 2010/143 (2010). <https://eprint.iacr.org/2010/143>
27. Leander, G.: On linear hulls, statistical saturation attacks, PRESENT and a cryptanalysis of PUFFIN. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 303–322. Springer, Heidelberg (2011)
28. Nakahara Jr., J., Sepehrdad, P., Zhang, B., Wang, M.: Linear (Hull) and algebraic cryptanalysis of the block cipher PRESENT. In: Garay, J.A., Miyaji, A., Otsuka, A. (eds.) CANS 2009. LNCS, vol. 5888, pp. 58–75. Springer, Heidelberg (2009)
29. Preneel, B., Takagi, T. (eds.): Cryptographic Hardware and Embedded Systems – CHES 2011. LNCS, vol. 6917. Springer, Heidelberg (2011)

30. Wang, M.: Differential cryptanalysis of reduced-round PRESENT. In: Vaudenay, S. (ed.) AFRICACRYPT 2008. LNCS, vol. 5023, pp. 40–49. Springer, Heidelberg (2008)
31. Wang, M., Sun, Y., Tischhauser, E., Preneel, B.: A model for structure attacks, with applications to PRESENT and serpent. In: Canteaut, A. (ed.) FSE 2012 [11]. LNCS, vol. 7549, pp. 49–68. Springer, Heidelberg (2012)
32. Wei, L., Peyrin, T., Sokołowski, P., Ling, S., Pieprzyk, J., Wang, H.: On the (In)Security of IDEA in various hashing modes. In: Canteaut, A. (ed.) FSE 2012 [11]. LNCS, vol. 7549, pp. 163–179. Springer, Heidelberg (2012)

# Key-Recovery Attack on the ASASA Cryptosystem with Expanding S-Boxes

Henri Gilbert<sup>(✉)</sup>, Jérôme Plût, and Joana Treger

ANSSI, Paris, France  
henri.gilbert@ssi.gouv.fr

**Abstract.** We present a cryptanalysis of the ASASA public key cipher introduced at ASIACRYPT 2014 [3]. This scheme alternates three layers of affine transformations  $A$  with two layers of quadratic substitutions  $S$ . We show that the partial derivatives of the public key polynomials contain information about the intermediate layer. This enables us to present a very simple distinguisher between an ASASA public key and random polynomials. We then expand upon the ideas of the distinguisher to achieve a full secret key recovery. This method uses only linear algebra and has a complexity dominated by the cost of computing the kernels of  $2^{26}$  small matrices with entries in  $\mathbb{F}_{16}$ .

## Introduction

A long-standing challenge in asymmetric cryptography is to bring asymmetric cryptography closer to symmetric cryptography by designing public key schemes whose overall structure and elementary operations are similar to those used in mainstream block ciphers such as AES. Solving this appealing but difficult challenge would not only increase the diversity in asymmetric cryptography, but might also help reducing the considerable performance gap between asymmetric cryptography and symmetric cryptography (the latter currently being more efficient by several orders of magnitude). This might as well allow the emergence of symmetric algorithms with some extra features, as for instance symmetric encryption schemes with a secure white-box implementation. Until 2014 however, as far as we know, all attempts of public key scheme designs with block cipher features, *e.g.* [10, 15, 16], eventually turned out to be weak [1, 7, 8, 13, 18].

**Asymmetric ASASA Schemes.** Some new candidate solutions to the above challenge were proposed in a paper published at ASIACRYPT 2014 by Biryukov et al. [3]. One conducting idea for the new designs stems from the observations that: (1) Traditional SPN block ciphers such as AES can be viewed as an alternance of (at least partly secret) affine transformations  $A$  and S-box layers  $S$ , and generally comprise a substantial number of  $A$  rounds (essentially 10 in the

---

This work was partially supported by the French National Research Agency through the BRUTUS project (contract ANR-14-CE28-0015).

case of AES-128); as shown by Biryukov and Shamir [4], some efficient generic attacks exist for the ASASA structure with secret S and A layers and small bijective S-boxes. (2) The efforts to design public key schemes with an alternance of A and S layers mainly focused so far on multivariate schemes with an ASA structure, with one single large S-box described by low degree equations over a finite field.<sup>1</sup> Based on the former considerations, the authors of [3] advocate for use of public multivariate schemes with an ASASA structure, *i.e.* with the simplest possible structure for which no generic attack is known in the case of small bijective S-boxes—or more generally of injective S-boxes whose non-zero linear combinations of outputs are not too strongly biased.

More precisely, the authors of [3] proposed a public-key encryption scheme named the *asymmetric ASASA scheme with expanding S-boxes*, conjecturing that it offers a comfortable security margin with respect to the potential lines of attack identified in their security analysis. This scheme uses small input-expanding injective quadratic S-boxes. Since these S-boxes have a length expansion factor of 2, the whole scheme has a length expansion factor of 4. The standard plaintext and ciphertext length for this scheme are respectively 128 and 512 bits.

While this work focuses exclusively on the ASASA scheme with expanding S-boxes, the same authors also proposed in [3] a second public-key scheme based on the ASASA structure, named the  $\chi$ -scheme. Indeed, this alternative construction makes use of Daemen's bijective quadratic S-box  $\chi$  based on cellular automata [5] and also used in various recent hash functions. The standard plaintext and ciphertext length is 128 bits. In this  $\chi$ -scheme, one single large S-box is used at each S layer. In their security analysis though, the authors of [3] consider many attacks on weakened versions of the  $\chi$ -scheme and conclude that the security margin of the  $\chi$ -scheme must be lower than that of the expanding scheme. They therefore express some caveats on its security and only "offer it as a cryptanalytic challenge, but not for practical use", unlike the expanding scheme.

In both ASASA public key encryption schemes, quadratic S-boxes are being used. The public key consists of the quartic equations of the encryption function and the private key consists of the specification of the A and S layers and of some *perturbation polynomials*, which are added to a few components of the vector representing the output of the second S layer.<sup>2</sup> Another property of these public key encryption schemes is that they can also be viewed as symmetric ciphers with a decent encryption and decryption speed and the following extra feature: as an alternative to using the secret key to efficiently encrypt, the public key provides

<sup>1</sup> While a noticeable exception is the multivariate scheme *R2* [14] that contains two S layers with small S-boxes, one weakness highlighted by attacks on *R2* or its variant *R2<sup>-</sup>* that were eventually discovered is the fact that the *R2* S-boxes are not injective.

<sup>2</sup> While the role of perturbations is essential in the case of the  $\chi$ -scheme since its variants without perturbations are reported in [3] to be vulnerable to efficient Gröbner basis attacks, in the case of the expanding scheme, perturbations are mostly introduced to provide some extra resistance against decomposition attacks that could potentially reduce the ASASA structure to the functional composition of two ASA structures.

a slower *strong white-box* implementation of the encryption function — i.e. an obfuscated implementation that is conjectured to prevent that the decryption function be derivable by an adversary who has full access to it.

**Our Contribution.** In this paper, we present an efficient attack on the ASASA scheme with expanding S-boxes. The starting point for our attack is the analysis of the homogeneous cubic part of the derivatives of the (quartic) polynomials of the public key. We first show that this analysis provides a distinguisher that allows to tell apart the public key of the scheme from a vector of random quartic polynomials. We then describe how to leverage the first information about the secret key provided by the distinguisher to retrieve the intermediate values that lie between the two S-layers, for an equivalent representation of the scheme. At this point, we are essentially left with the problem of solving two quadratic ASA layers. Though generic techniques to solve this problem exist, we give our own algorithm, that is well-adapted to the scheme considered. The overall complexity of the attack is equivalent to at most  $2^{26}$  computations of kernels of matrices of size  $64 \times 96$  over the finite field  $\mathbb{F}_{16}$ . We estimate the corresponding computational time to a few CPU-hours, which places this cryptanalysis well within practical limits.

This paper is organised as follows. Section 1 provides a description of the expanding ASASA scheme and presents some useful preliminary results. Section 2 introduces a distinguisher for this scheme that can be used to derive some first information on the secret key. Finally, Sect. 3 shows how to efficiently derive an equivalent secret key from the public key.

## 1 The ASASA Cryptosystem

### 1.1 Definition and First Notations

The two asymmetric ASASA schemes of [3] are composed of polynomial transformations over the base field  $k = \mathbb{F}_{16}$ ; they are obtained by alternating three  $k$ -affine layers and two non-linear polynomial-based S layers. The ASASA scheme with expanding S-boxes, on which we are focusing, involves S-boxes whose output is twice as big as their input; 32 perturbation polynomials of degree four are also applied just before the last affine transform. More precisely, each S-box maps a 4-tuple of  $k$ -values onto an 8-tuple of  $k$ -values, defined as degree 2 polynomials over  $k$  in the inputs. The resulting scheme, which we simply call the ASASA cryptosystem in the remaining of this paper for simplification purposes, has then degree 4 over  $k$ . Going into details, the private key of the ASASA cryptosystem consists of:

- Three uniformly random invertible affine transformations  $\mathcal{A}^x$  of  $k^{32}$ ,  $\mathcal{A}^y$  of  $k^{64}$ , and  $\mathcal{A}^z$  of  $k^{128}$ ;
- Two sets of uniformly random quadratic functions from  $k^4$  to  $k^8$  corresponding to the first and second S-box layer  $S^x = \{S_{0,0}^x, \dots, S_{0,7}^x, \dots, S_{7,0}^x, \dots, S_{7,7}^x\}$ , that determines 8 S-boxes  $S_0^x, \dots, S_7^x$ , and  $S^y = \{S_{0,0}^y, \dots, S_{7,0}^y, \dots, S_{15,0}^y, \dots, S_{15,7}^y\}$ , that determines 16 S-boxes  $S_0^y, \dots, S_{15}^y$ ;

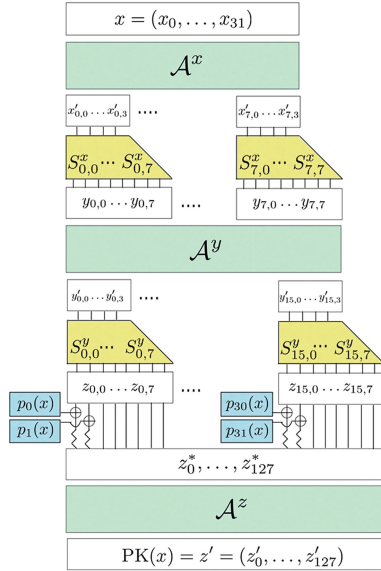


Fig. 1. The ASASA cryptosystem with expanding S-boxes.

- Thirty-two uniformly random quartic perturbation polynomials  $p_0, \dots, p_{31}$  on  $k^{32}$ .

The public key and the associated public encryption function are derived from the secret key as illustrated at Fig. 1, following those successive steps:

- (i) The plaintext state is the tuple of variables  $x = (x_0, \dots, x_{31}) \in k^{32}$ ;
- (ii) Let  $x' = \mathcal{A}^x \cdot x \in k^{32}$ ;
- (iii) Define  $y = (y_{0,0}, \dots, y_{7,7}) \in k^{64}$  as  $y_{r,i} = S^x_{r,i}(x'_{r,0}, \dots, x'_{r,3})$ , for  $r, i = 0 \dots 7$ ;
- (iv) Let  $y' = \mathcal{A}^y \cdot y \in k^{64}$ ;
- (v) Define  $z = (z_{0,0}, \dots, z_{15,7}) \in k^{128}$  as  $z_{r,i} = S^y_{r,i}(y'_{r,0}, \dots, y'_{r,3})$ , for  $i = 0 \dots 7$  and  $r = 0 \dots 15$ ;
- (vi) For  $r \in \{0, 15\}$ , do  $z^*_{8r} \leftarrow z_{r,0} + p_{2r}(x)$ ,  $z^*_{8r+1} \leftarrow z_{r,1} + p_{2r+1}(x)$ , and  $z^*_{8r+i} \leftarrow z_{r,i}$  for  $i = 2 \dots 7$  (the two first components out of 8 contiguous components of  $z$  are modified);
- (vii) The public key  $\text{PK}(x)$  is the vector of 128 polynomials over  $k$ ,  $z' = \mathcal{A}^z \cdot z^* \in (k[x_0, \dots, x_{31}])^{128}$ ; the public encryption function is the associated function from  $k^{32}$  to  $k^{128}$ .

**Dimension of the Secret and Public Key Spaces.** Since the dimensions of various vector spaces are central in our analysis, we compute the size of the secret and public key spaces. We do not find the exact same numbers as the original authors [3, 2.5], although the order of magnitude is the same. An

affine transform on  $n$  variables is representable by a matrix of size  $n \times (n + 1)$ ; therefore, the three  $A$  layers have a key size of  $32 \times 33 + 64 \times 65 + 128 \times 129 = 21\,728$  elements of  $\mathbb{F}_{16}$ . An S-box output is an (inhomogeneous) quadratic polynomial in four variables, and is therefore described by  $\binom{6}{2} = 15$  coefficients. (The dimensions of spaces of homogeneous polynomials are as given below in Sect. 1.3 of this paper; inhomogeneous polynomials in  $n$  variables correspond bijectively to homogeneous polynomials of the same degree in  $n + 1$  variables). Therefore, the two  $S$  layers have a key size of  $24 \times 8 \times 15 = 2880$  elements of  $\mathbb{F}_{16}$ . In total, the secret key size is  $24\,608$  elements of  $\mathbb{F}_{16}$ , or approximately  $2^{13.6}$  bytes of data. This does not, however, count the perturbation polynomials, which occupy a space of  $32 \times \binom{36}{4}$  elements, or  $2^{19.8}$  bytes of data. The public key is a set of  $128 \times \binom{36}{4}$  elements of  $\mathbb{F}_{16}$ , or  $2^{21.8}$  bytes of data.

## 1.2 Equivalent Simple Keys

As for most multivariate cryptosystems, there are multiple private keys that correspond to a given ASASA public key, and we show here that each secret key is equivalent to a simpler one, that we describe. We also redefine the ASASA system in terms of those “simple” keys, which make our attack easier to explain; we point out that this simplification is purely cosmetic though and that our attack does apply to the ASASA system as described in [3].

First, since for each  $r = 0, \dots, 15$ , the two outputs  $z_{r,0}, z_{r,1}$  of the S-box  $S_r^y$  are added to arbitrary perturbation polynomials  $p_i$ , we obtain the same public key when replacing  $p_{2r}, p_{2r+1}$  by  $p_{2r} + z_{r,0}, p_{2r+1} + z_{r,1}$  and  $S_{r,0}^y, S_{r,1}^y$  by zero.

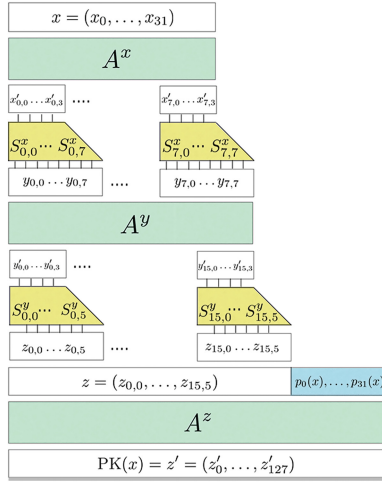
Let  $\mathcal{A}^x = A^x + a^x$  and  $\mathcal{A}^y = A^y + a^y$  be the decomposition of  $\mathcal{A}^x$  and  $\mathcal{A}^y$  as their linear part plus their constant. We can actually assume that  $\mathcal{A}^x = A^x$  and  $\mathcal{A}^y = A^y$ , as it is always possible to consider a modified  $S^y$  S-box layer where the first addition by  $a^x$  and the second addition by  $a^y$  are absorbed by the polynomials of  $S^y$ . The same goes for  $\mathcal{A}^z$ , where the addition of the 32 components with index  $8r$  and  $8r + 1$  of  $a^z$  can be viewed as part of the perturbation polynomials, so that  $S_{r,0}^y$  and  $S_{r,1}^y$  are still zero. This shows that we can assume  $a^x = a^y = a^z = 0$  and consider from now on  $A^x, A^y$  and  $A^z$  instead of  $\mathcal{A}^x, \mathcal{A}^y$  and  $\mathcal{A}^z$ .

Finally, notice that it is always possible to compose the output of the quadratic map  $S_i^x$  by a linear transform, and the corresponding input block of the linear map  $A^y$  by its inverse. The same applies to  $S_i^y$  and  $A^z$ , for  $i \neq 8r, 8r + 1$ , *i.e.* as long as the corresponding 32 zero polynomials of  $S^y$  are not affected.

To sum up, the description of the ASASA private key of Sect. 1.1 is equivalent to the following:

- Three uniformly random invertible linear transformations  $A^x$  of  $k^{32}$ ,  $A^y$  of  $k^{64}$  and  $A^z$  of  $k^{128}$ ;
- Two sets of  $8 \times 8$  and  $16 \times 6$  uniformly random quadratic functions on  $k^4$ ,  $S^x = \{S_{0,0}^x, \dots, S_{7,7}^x\}$  and  $S^y = \{S_{0,0}^y, \dots, S_{15,5}^y\}$ ;
- A set of 32 uniformly random quartic polynomials  $p_0, \dots, p_{31}$  on  $k^{32}$ .





**Fig. 2.** The ASASA cryptosystem with expanding S-boxes, equivalent representation.

In the remaining of this paper, we shall always consider such *simple private keys*; the corresponding encryption mechanism is illustrated at Fig. 2.

**1.3 Notations and Preliminaries**

We write  $k = \mathbb{F}_{16}$  for the finite field with 16 elements. Throughout this work, we let  $q = |k| = 16$ .

**Homogeneous Polynomials.** We write  $H_{d,n}$  for the space of homogeneous polynomials of degree  $d$  over  $n$  variables (over the base field  $k = \mathbb{F}_{16}$ ); it is a vector space of dimension  $\binom{n+d-1}{d}$  [17, 2.2.1]. Throughout this paper, we shall usually work with the vector space  $H_{d,32}$  of homogeneous polynomials of degree  $d$  on the 32 input variables  $x_i$ . We shall write  $H_d$  instead of  $H_{d,32}$ .

Let  $f(x_1, \dots, x_n)$  be a polynomial. For any integer  $d$ , we write  $f_{(d)}$  for the degree  $d$  homogeneous part of  $f$ .

**Vector Spaces.** Given subspaces  $E \subset H_{d,n}$  and  $E' \subset H_{d',n}$ , we define  $E \cdot E'$  as the subspace of  $H_{d+d',n}$  generated by all the products  $u \cdot u'$  for  $u \in E$  and  $u' \in E'$ . We also define  $D$  as the vector space generated by the 32 derivations  $\partial_i = \partial/\partial x_i$ . For any vector space  $E \subset H_{d,n}$ , we define  $DE \subset H_{d-1,n}$  as the vector space generated by all  $\delta(u)$  for  $\delta \in D$  and  $u \in E$ .

**Counting Matrices of a Given Rank.** We introduce the following notations, used in the computation of the number of matrices with given size and rank. For any integers  $n \geq d$ , we define

$$[n, d] = \prod_{i=0}^{d-1} q^n - q^i \quad \text{and} \quad \begin{bmatrix} n \\ d \end{bmatrix} = \frac{[n, d]}{[d, d]}. \tag{1}$$

We omit the value  $q$  from these notations, since we shall always use  $q = 16$ . We use the following classical result (see [12, VII.19]).

**Proposition 1.** *Let  $m, n, d$  be three integers.*

- (i) *There are exactly  $\begin{bmatrix} n, d \end{bmatrix}$  injective maps from  $k^d$  to  $k^n$ .*
- (ii) *There are exactly  $\begin{bmatrix} n \\ d \end{bmatrix}$  subspaces of dimension  $d$  of  $k^n$ .*
- (iii) *There are exactly  $\frac{\begin{bmatrix} n, d \end{bmatrix} \cdot \begin{bmatrix} m, d \end{bmatrix}}{\begin{bmatrix} d, d \end{bmatrix}}$  matrices of size  $m \times n$  with rank exactly  $d$ .  $\square$*

**Proposition 2.** *Let  $V$  be a vector space of dimension  $d$  over  $k = \mathbb{F}_{16}$ . A set of  $n$  uniformly random, independent vectors  $v_i \in V$  generates  $V$  with overwhelming probability if  $n \geq d + 32$ .*

*Proof.* Let  $\pi(n, d)$  be the probability that a random matrix of size  $n \times d$  has maximal rank  $d$ . We see by Proposition 1 that

$$\pi(n, d) = \frac{[n, d]}{q^{nd}} = \prod_{i=0}^{d-1} 1 - q^{-(n-i)}. \tag{2}$$

For  $n > d \gg 0$ , since  $q^{-n} \ll 1$ , we have the asymptotic expansion

$$\pi(n, d) = \exp\left(\sum_{i=0}^{d-1} \log(1 - q^{-(n-i)})\right) \simeq \exp\left(-q^{-n} \frac{q^d - 1}{q - 1}\right). \tag{3}$$

For any  $\varepsilon > 0$ , we find that

$$\pi(n, d) > 1 - \varepsilon \quad \text{iff} \quad n > \log_q \frac{q^d - 1}{q - 1} - \log_q \log \frac{1}{1 - \varepsilon}. \tag{4}$$

Using  $\varepsilon = 2^{-128}$  as our definition of “overwhelming probability”, the above condition (4) becomes  $n \geq d + 32$ .

**Expected Behaviour of Derivatives and Product of Random Polynomials.** We shall use throughout this work the following heuristics about the derivatives and products of random polynomials.

*For  $f_1, \dots, f_r$  uniformly random, independent elements of  $H_{d,n}$ :*

- (i) *if  $r < \frac{1}{n} \dim H_{d-1,n}$ , then the  $nr$  derivatives  $\partial_i f_j$  behave like uniformly random, independent elements of  $H_{d-1,n}$ ;*
- (ii) *if  $r < \frac{1}{n} \dim H_{d+1,n}$ , then the  $nr$  products  $x_i f_j$  behave like uniformly random, independent elements of  $H_{d+1,n}$ .*

In particular, according to Proposition 2, we expect that, with overwhelming probability, if  $nr \leq \dim H_{d-1,n} - 32$ , then the  $\partial_i f_j$  are free in  $H_{d-1,n}$ ; if  $nr \leq \dim H_{d+1,n} - 32$ , then the  $x_i f_j$  are free in  $H_{d+1,n}$ . Although giving a detailed proof of these facts would be out of the scope of this work, we obtained an empiric confirmation in the cases of interest to us (namely  $n = 32$ , and either  $d = 4, r = 128$  for derivation, or  $d = 2, r = 64$  for multiplication), as well as of the bounds of validity.

## 2 A Simple Distinguisher

We shall see that the ASASA cryptosystem presents the same flaw as several multivariate cryptosystems, that is, it is possible to distinguish the equations of the public key for the ASASA scheme from random polynomials of the same degree over  $k$ . The distinguisher we present here is extremely simple: namely, computing the rank of the matrix of partial derivatives of the polynomials is enough. However, by elaborating on the structure of this distinguisher, we shall explain in Sect. 3 how it is possible to fully recover the secret key.

### 2.1 Considerations on the Dimension of Vector Spaces Derived from the Public Key

The key observation underlying the distinguisher is that, while the space of homogeneous cubic polynomials  $H_3 = H_{3,32}$  has dimension  $\binom{34}{3} = 5984$ , the homogeneous cubic parts of the derivatives of the public key  $(\partial \text{PK}_i / \partial x_j)_{(3)}$  actually belong to a much smaller subspace of  $H_3$  which happens to have dimension at most 3072.

**The Case of the ASASA Cryptosystem with No Perturbation Polynomials.** As a warm-up, we first consider the encryption of the message  $(x_0, \dots, x_{31})$  under the “non-perturbed” ASASA scheme, *i.e.* the ASASA scheme with no perturbation polynomials. We recall that all intermediate values  $\{y_{r,i}\}, \{y'_{r,i}\}, \{z_{r,i}\}$  and  $\{z'_i\}$  introduced at Sect. 1 can be seen as polynomials in the 32 input variables  $x_i$  with coefficients in  $k = \mathbb{F}_{16}$  (see Fig. 1).

To see that the  $(\partial \text{PK}_i / \partial x_j)_{(3)}$ , for  $i = 0 \dots 127, j = 0 \dots 31$ , belong to a restricted subspace, we consider the second S-box layer  $S^y$ ; recall that  $z_{r,i}$  denotes the  $i$ -th output of the S-box  $S^y_r$  with input  $y'_{r,0}, \dots, y'_{r,3}$ . The quadratic polynomials  $y'_{r,i}$  may be written  $y'_{r,i} = (y'_{r,i})_{(2)} + (y'_{r,i})_{(1)} + (y'_{r,i})_{(0)}$ , as the sum of a homogeneous degree 2 part, a linear part and a constant. We therefore see that the homogeneous parts of degree 4 of the polynomials  $z_{r,i}$  output by the S-box  $S^y_r$  are linear combinations of the terms  $(y'_{r,m})_{(2)} \cdot (y'_{r,n})_{(2)}$ . Let us write  $\partial z_{r,i} / \partial x_j$  for the derivative of the output  $z_{r,i}$  along the input variable  $x_j$ . There exists coefficients  $a_{m,n} \in k$  such that

$$(\partial z_{r,i} / \partial x_j)_{(3)} = \sum_{m,n} a_{m,n} (y'_{r,m})_{(2)} (\partial y'_{r,n} / \partial x_i)_{(1)}. \tag{5}$$

Let  $Y'_{(2)} \subset H_{32,2}$  be the vector space spanned by the 64 homogeneous quadratic parts of the polynomials  $y'_{r,m}$ ; it has dimension at most 64. Let also  $(DZ)_{(3)}$  be the vector space spanned by the  $128 \times 32 = 4096$  homogeneous cubic parts of the derivatives of the polynomials  $z_{r,i}$ . The expression (5) above implies

$$(DZ)_{(3)} \subset Y'_{(2)} \cdot H_1. \quad (6)$$

The vector space  $(DZ')_{(3)} = \langle (\partial \text{PK}_i / \partial x_j)_{(3)} \rangle$  being a linear image of  $(DZ)_{(3)}$  by  $A^z$ , we also have

$$(DZ')_{(3)} \subset Y'_{(2)} \cdot H_1, \quad (7)$$

and

$$\dim(DZ')_{(3)} \leq \dim Y'_{(2)} \cdot H_1 \leq 64 \times 32 = 2048. \quad (8)$$

**The General ASASA Cryptosystem.** For the general ASASA scheme, we have to slightly adapt the result (7) to take into account the perturbation polynomials. We refer the reader to Fig. 2 for the description of ASASA used in this paragraph.

We established in our analysis of the unperturbed scheme that the homogeneous cubic parts of the derivatives of the polynomials  $\{z_{r,i}\}$  belong to the vector space  $Y'_{(2)} \cdot H_1$ , which has dimension 2048. This still holds for the general ASASA scheme, since up to generation  $z$ , the perturbation polynomials do not appear. The next step in the algorithm is the linear transform  $A^z$ ; its input is the concatenation  $z || (p_0, \dots, p_{31})$ , where the first 96 elements are the polynomials of  $z$  and the last 32 are the perturbation polynomials. This means that the polynomials of the public key  $z'$  are linear combinations of the polynomials of  $z$  and the perturbation polynomials  $p_0, \dots, p_{15}$ . Let  $DP$  be the vector space spanned by the homogeneous cubic parts  $(\partial p_i / \partial x_j)_{(3)}$  of the derivatives of the perturbation polynomials; it has dimension at most  $32 \times 32 = 1024$ . We necessarily have

$$(DZ')_{(3)} \subset Y'_{(2)} \cdot H_1 + DP. \quad (9)$$

In terms of dimensions, Eq. (9) implies

$$\dim(DZ')_{(3)} \leq \dim(Y'_{(2)}) \cdot H_1 + 1024 \leq 3072, \quad (10)$$

as claimed.

## 2.2 The Distinguisher

We now turn the observation of Sect. 2.1 into a working distinguisher.

**Proposition 3.** *It is possible to distinguish the public key polynomials of the ASASA scheme from uniformly random quartic polynomials by computing the rank of a matrix of size  $5984 \times 4096$  with coefficients in  $\mathbb{F}_{16}$ .*

*Proof.* Let  $T = (T_0, \dots, T_{127})$  be a vector of 128 polynomials that are either uniformly random quartic polynomials, or a (perturbed) ASASA public key PK. We consider the matrix  $M$  of size  $5984 \times 4096$  whose columns are the vectors  $(\partial T_i / \partial x_j)_{(3)} \in H_{3,32}$ , with the usual notations. If the  $T_i$  are uniformly random polynomials, then the rank of  $M$  is 32 times the rank of the family  $(T_i)$ ; since  $128 \times 32 < \dim H_{3,32}$ , according to our heuristic, this is usually  $32 \times 128 = 4096$ . If on the contrary  $T$  is an ASASA public key, then by (10), the rank of  $M$  is at most 3072.

The distinguisher that returns ASASA if the rank of  $M$  is  $\leq 3072$ , and random otherwise, succeeds with overwhelming probability.<sup>3</sup>  $\square$

### 3 Key Recovery

We now present a secret key recovery attack on an ASASA scheme. The ideas used here are based on the properties already identified in Sect. 2, namely, the space of derivatives of PK contains information about the intermediate values between the two quadratic layers.

To attack the system, we first identify the vector space of quadratic forms manipulated in the middle of the algorithm (as output of the first S layer, and input to the second one). This is the crucial point of the cryptanalysis. It enables us to reduce the problem to two much simpler ASA problems. We then solve each ASA instance in turn. (Note that although we present a specific way to solve to these two ASA instances, it is well-known that ASA instances are weak, and techniques to solve such systems can be found in the literature [2, 6, 9, 11]).

This key-recovery only relies on linear algebra in various spaces of homogeneous polynomials. We refer the reader to Sect. 1.3 for some useful general results in algebra used in the attack; a few other results will be introduced on the fly when needed. For simplicity in this whole Sect. 3, we write  $Y$ , instead of  $Y'_{(2)}$ , for the space generated by the homogeneous quadratic parts of the polynomials of  $y$ .

The overall complexity of the attack is about  $2^{26}$  times the computation of the rank of a square matrix of size  $64 \times 96$  with coefficients in  $\mathbb{F}_{16}$ . We point out that our method only uses the quadratic terms of the secret quadratic layers; it is therefore also applicable to homogeneous instances of the ASASA cryptosystem.

#### 3.1 Computing the Middle Layer

As already mentioned, our attack uses the same data as the distinguisher. More precisely, the key result was given at Eq. (9): the vector space  $DZ'$  of derivatives of PK contains information about the space  $Y \cdot H_1$ , *i.e.* about the vector space  $Y$  of homogeneous quadratic functions produced by the first S layer. However, the

<sup>3</sup> We may also investigate the case of a reinforced ASASA scheme with more perturbation polynomials, *i.e.* with 96 “legitimate” outputs and  $p \geq 32$  perturbations. We easily find that our distinguisher works at least for  $p \leq 90$ . The same bound applies to the key recovery attack of Sect. 3 below.

observed vector space deduced from the public key also contains some unwanted vectors originating in the perturbation polynomials.

To access the space  $Y \cdot H_1$  and see beyond the perturbation polynomials, the first step is to construct several subspaces of  $DZ'$  including  $Y \cdot H_1$ . We are then able to recover  $Y \cdot H_1$  as the intersection of all those subspaces. In a second step we show how, from this recovered vector space, we compute the space  $Y$  itself.

**Eliminating the Perturbations.** This first steps aims at computing the vector space  $Y \cdot H_1$  from the public key.

Recall that a public key PK of the ASASA cryptosystem is given as a vector of 128 polynomials  $PK_i$  in the 32 input variables. We define  $F_i = (PK_i)_{(4)}$  as the homogeneous quartic part of the public key, and  $F$  as the vector space generated by all  $F_i$ .

For any derivation  $\delta \in D$  and for any  $f \in F$ , we saw when describing the distinguisher that

$$\delta f \in Y \cdot H_1 + \delta P, \tag{11}$$

where  $Y$  is the vector space generated by the 64 quadratic polynomials  $(y_{r,i})_{(2)}$ , and  $P$  is the vector space generated by the 32 perturbation polynomials  $(p_i)_{(4)}$ .

Let  $\Delta \subset D$  be a vector space of dimension  $d$ . By (11), we then have

$$\Delta F \subset Y \cdot H_1 + \Delta P, \tag{12}$$

where the right-hand space has dimension at most  $64 \times 32 + 32 \times d = 2048 + 32d$ . The space  $\Delta F$  is generated by  $128d$  elements  $\delta f_i$ , for  $\delta \in \Delta$  and  $i = 0, \dots, 127$ . By Proposition 2, these  $128d$  elements generate the whole space  $Y \cdot H_1 + \Delta P$  as long as  $128d \geq 2048 + 32d + 32$ , or equivalently,  $d \geq 22$ .

We now consider a family of  $m$  vector spaces  $\Delta_1, \dots, \Delta_m \subset D$ , each space  $\Delta_i$  being of dimension  $d = 22$ . We know by what precedes that for each one of them,  $\Delta_i F = Y \cdot H_1 + \Delta_i P$ . This implies that

$$\bigcap_{i=1}^m \Delta_i F = Y \cdot H_1 + \bigcap_{i=1}^m \Delta_i P. \tag{13}$$

Since  $\dim D = 32$  and  $\dim \Delta_i = 22$  for each  $i$ , the intersection of two spaces  $\Delta_i$  generally has dimension 12 (this is always the case if we choose the  $\Delta_i$  correctly), and likewise the intersection of three such spaces has dimension 2, and for  $m \geq 4$ , we easily find  $\Delta_1, \dots, \Delta_m$  such that  $\Delta_1 \cap \dots \cap \Delta_m = 0$ . This implies that

$$\bigcap_{i=1}^m \Delta_i P = 0 \text{ for } m \geq 4. \tag{14}$$

Formula (14) means that the intersection  $\bigcap_{i=1}^m \Delta_i F$  is then exactly the space  $Y \cdot H_1$ :

$$\bigcap_{i=1}^m \Delta_i F = Y \cdot H_1 \text{ for } m \geq 4. \tag{15}$$

**Computing the Middle Terms.** This part explains how we recover the 64-dimensional space  $Y$  from the space  $Y \cdot H_1 \subset H_3$  obtained during the previous step.

We first prove a short lemma. Let  $V \subset H_2$  be a vector space of dimension  $d$  and basis  $(v_j)$ . The vector space  $V \cdot H_1 \subset H_3$  is generated by the  $32d$  elements  $x_i v_j$ . If  $d \leq 186$ , then  $32d \leq \dim H_3 - 32$ ; by Proposition 2, we therefore expect these  $32d$  elements to be linearly independent in  $H_3$ . This implies that  $\dim(V \cdot H_1) = 32d$ . In particular, this means that when  $V$  has dimension  $\leq 186$ , the correspondence between  $(\dim V)$  and  $(\dim V \cdot H_1)$  behaves, with very high probability, as a strictly increasing function.

We now use this lemma to characterize the space  $Y$ . Let  $\bar{Y} \subset H_2$  be the vector space of all functions  $g$  such that, for all  $i$ ,  $gx_i \in Y \cdot H_1$ . Trivially,  $Y \subset \bar{Y}$  and  $\bar{Y} \cdot H_1 = Y \cdot H_1$ . Since  $\dim Y = 64 \leq 186$ , we are in the conditions of the previous lemma, which implies that  $Y = \bar{Y}$  with overwhelming probability.

We may easily compute the space  $\bar{Y}$  from  $Y \cdot H_1$  as follows. For each  $i = 0, \dots, 31$ , we define  $G_i$  as the subspace of multiples of  $x_i$  in  $Y \cdot H_1$ ; by definition,  $\bar{Y}$  is the intersection of all spaces  $x_i^{-1}G_i$ ,

$$\bar{Y} = \bigcap_{i=0}^{31} x_i^{-1}G_i. \tag{16}$$

### 3.2 Solving a Quadratic ASA Layer

As already mentioned, there exists generic techniques [2, 6, 9, 11] for inverting a public key in the ASA form. We give our own solution here, as it is simple and works well in the particular case of an ASA layer based on quadratic S-boxes. We shall use this technique twice, once for the inner ASA layer, and then once more for the outer one.

This section and the next one (Sect. 3.3) are mutually independent. We present the inner layer ASA first since it is easier to explain.

**Notations.** The inner ASA layer is represented as the (known) vector space  $Y$  generated by the 64 (unknown) quadratic forms  $y_{r,i}$  in the 32 input variables  $x_i$ . We restrict ourselves here to the homogeneous part of the  $y_{r,i}$ , since this case is more difficult to solve, but easier to present.

For each fixed  $r$ , the eight functions  $y_{r,0}, \dots, y_{r,7}$  are quadratic forms in the same 4 fixed linear combinations  $x'_{r,0}, \dots, x'_{r,3}$  of the input variables  $x_i$ . We write  $X_r$  for the vector space generated by the  $x'_{r,i}$ . We may also decompose the space of differentials  $D = \langle \partial / \partial x_i \rangle$  according to the S-boxes; namely, for each  $r$ , we define  $D^r$  as the set of all elements  $\delta \in D$  whose restriction to  $X_r$  is zero.

We note that  $\dim X_r = 4$  and  $\dim D^r = 28$ . Therefore, for a given  $r$ , an uniformly random element of  $D$  belongs to  $D^r$  with probability  $q^{-4} = 2^{-16}$ .

**Separating the Inputs of the S-Boxes.** We show that we are able to identify the elements of  $D^r$ , *i.e.* the differentials that vanish on the inputs of a particular S-box.<sup>4</sup>

For any quadratic form  $f \in Y$  and any  $\delta \in D$ , the function  $\delta f$  is a linear form of  $x \in X$ ; this means that  $(\delta f)(x)$  is a trilinear function of  $(\delta, f, x)$ . Therefore, for a fixed value of  $\delta$ , it is a bilinear function of  $(f, x)$ . We write  $F_\delta$  for this bilinear form. It is represented by a matrix of size  $64 \times 32$  whose coefficients are the  $(\delta f_i)(x_j)$ , where  $(f_i)$  is a basis of  $Y$  and  $(x_j)$  is the standard basis of  $X$ .

Let  $f \in Y$ ; we can write  $f$  as a sum  $\sum f_s$  for  $s = 0, \dots, 7$ , where  $f_s$  is a quadratic form on  $X_s$ . For any  $\delta \in D^r$ , we have  $\delta f_r = 0$ , so that  $\delta f$  is the sum of the  $\delta f_s$  for  $s \neq r$ . Since each one of these terms uses only the variables from  $X_s$ , this means that  $(\delta f)(x) = 0$  for  $x \in X_r$ . Put differently, the kernel of  $F_\delta$  (here seen as a linear map from  $X$  to the dual of  $Y$ ) contains  $X_r$ .

Now let  $\delta$  be an element of  $D$  not belonging to any of the  $D^r$ : since the maps  $(\delta f_i)$  are 64 random linear forms on the 32-dimensional space  $X$ ; by Proposition 2, with overwhelming probability, the intersection of their kernels is zero.

As a result, we see that the rank of the matrix  $F_\delta$  is always  $\leq 28$  if  $\delta$  belongs to at least one of the  $D^r$  and 32 with overwhelming probability otherwise. This also provides a test, given two elements  $\delta$  and  $\varepsilon$  of  $D^r$  and  $D^s$ , for the equality  $r = s$ : since the kernels of the matrices  $F_\delta, F_\varepsilon$  contain respectively  $X_r$  and  $X_s$ , their intersection is non-trivial when  $r = s$ .

**The Algorithm.** We compute the spaces  $X_r$  and  $D^r$  at the same time, and up to a permutation of  $\{0, \dots, 7\}$ , since we do not know the labeling of the S-boxes. For each  $r = 0, \dots, 7$ , we keep candidates  $U_r$  and  $V_r$  for the spaces  $X_r$  and  $D^r$ ; initially,  $U_r$  is the whole space  $X$  while the space  $V_r$  is empty. During the whole algorithm, we have  $U_r \supset X_r$  and  $V_r \subset D^r$ , with both inclusions being equalities at the end. We also note that, at every step of the algorithm,  $U_r$  is the intersection of the kernels of all elements of  $V_r$ .

We now describe the algorithm. We randomly generate elements  $\delta$  of  $D$  and compute the kernel  $K \subset X$  of the matrix  $F_\delta$ . If this kernel has dimension at least 4, then it intersects non-trivially one of the spaces  $U_r$  and the intersection also has dimension at least 4. We then update  $V_r$  to  $V_r \oplus \langle \delta \rangle$  and  $U_r$  to  $U_r \cap K$ . The algorithm ends when each space  $V_r$  has the required dimension 28, as then  $V_r = D^r$  as required, and therefore  $U_r = X_r$ .

**Recovering the S-Boxes.** Once the spaces  $X_r$  are known, computing the S-boxes is easy. We write  $Y_r$  for the vector space generated by the 8 outputs of the S-box  $S_r^x$ , and  $X^r$  for the direct sum of all  $X_s$  for  $s \neq r$ . Then  $Y_r$  is exactly the

<sup>4</sup> Since the elements of  $Y$  are quadratic forms, their differentials are exactly the associated polar forms. This means that we may represent the derivations as elements of  $X$ , using the relation  $(\delta f)(x) = f(x + \delta) - f(x) - f(\delta)$ ; in this view, the space  $D^r$  is the direct sum of all the  $X_s$  for  $s \neq r$ . However, we chose to use an explanation based on differentials, since this is both closer to the differential cryptanalysis point of view, and easier to generalize to polynomials of higher degree.



set of elements of  $Y$  that vanish on all points of  $X^r$ . We may therefore compute  $Y_r$  with linear algebra. (Another possibility is to use the derivations spaces  $D^r$ , also computed in the previous step, since  $Y_r$  is the set of elements  $f \in Y$  such that, for all  $\delta \in D^r$ ,  $\delta f = 0$ ). Once bases of both  $X_r$  and  $Y_r$  are known, we recover the secret functions  $S_r^x$  by interpolation.

**Complexity.**  $q^4 \times 28 \times 8 \approx 15 \cdot 10^6$  elements of  $X$ , the expected cost for the execution of this algorithm is approximately  $2^{23.8}$  times that of the computation of the kernel of a matrix of size  $32 \times 64$  with entries in  $\mathbb{F}_{16}$ .

### 3.3 Solving the Outer ASA Layer

We again use the representation of the middle layer as a 64-dimensional vector space  $Y$  of quadratic forms computed in Sect. 3.1. We now determine the output functions  $F_i$  as linear combinations of quadratic forms in the elements of  $Y$  and the 32 perturbation polynomials  $p_i$ .

**Computing the Outputs of the S-Boxes.** We recall that  $F$  is the vector space generated by the homogeneous quartic part of the ASASA public key. This vector space is the direct sum of the 32-dimensional space  $P$  generated by the homogeneous quartic parts of the perturbations, and the 96-dimensional vector space  $Z$  generated by the outputs of the 16 S-boxes  $S_r^y$ . Since the middle layer  $Y$  is known as a result of Sect. 3.1, we may compute  $Z$  as the intersection  $Z = (Y \cdot Y) \cap F$ .

**Reducing to a Quadratic ASA Layer.** We now study the 16 S-boxes  $S_r^y$ . We already know the 64-dimensional vector space  $Y$  of their (quadratic) inputs and the 96-dimensional vector space  $Z$  of their (quartic) outputs. Our goal for now is to rewrite each element of  $Z$  as an explicit quadratic function of the elements of  $Y$ , so as to be able to apply the techniques of Sect. 3.2.<sup>5</sup>

We represent  $Y$  by its Hermite normal basis relative to a particular basis of  $H_2$ : the first 32 elements of  $Y$  have the form  $u_i = x_i^2 + \dots$  for  $i = 1, \dots, 32$ , the next 31 elements have the form  $v_i = x_1 x_i + \dots$  for  $i = 2, \dots, 32$ , and the last one is  $w = x_2 x_3 + \dots$ , where none of the omitted “...” expressions contain either squares or terms  $x_1 x_i$  or  $x_2 x_3$ . In the (unlikely) case where the Hermite normal form of  $Y$  does not contain the monomials  $x_1^2, \dots, x_2 x_3$ , we may always replace the public key  $F$  by its composition  $F \circ \sigma$  by a random invertible linear

---

<sup>5</sup> We note that in the (very unlikely) case where the computation of the space  $Y$  performed in Sect. 3.1 returned a space  $Y' \supsetneq Y$ , the computation performed here will allow us to remove the few extra elements of  $Y'$ : namely, since the elements of  $Z$  are quadratic forms in the elements of  $Y$ , the unneeded elements of  $Y'$  will not appear in these expressions. This means that, in practice, this step (Sect. 3.3) should be performed before the inner step (Sect. 3.2).

transformation of the input variables, such that  $Y \circ \sigma$  has the suitable Hermite normal form.

We now consider a basis of the space  $Z$ . Any term of the form  $\lambda x_i^4$  appearing in a basis element of  $Z$  comes, in its expression as a quadratic forms over  $Y$ , from a term  $\lambda u_i^2$ . Likewise, any term of the form  $\mu x_1^3 x_i$  comes from a term  $\mu u_1 v_i$ , and so on.

In this way, we identify the second ASA layer as a quadratic map from  $Y$  to  $Z$ .

**Solving the ASA Problem.** The problem we have to solve is now almost identical to the one we solved in Sect. 3.2, except that we now have 16 instead of 8 S-boxes, and 96 instead of 64 quadratic forms.

Applying the previous algorithm to this case thus has a global complexity of approximately  $q^4 \times 60 \times 16 \approx 2^{25.9}$  times the cost of computing the kernel of a matrix of size  $64 \times 96$  with entries in  $\mathbb{F}_{16}$ . This is the dominant step in the key recovery procedure. We estimate the corresponding computational cost to a few CPU-hours.

### 3.4 Computing the Inhomogeneous Terms

We just presented an algorithm computing the *homogeneous* part (of degree two) of the quadratic S layers of the ASASA public key. These homogeneous terms represent the largest part of the secret key. Once they are computed, recovering the inhomogeneous terms (of degree one) is quite simple.

Each output S-box has one such linear term, represented by four coefficients; in total, there are therefore  $(64+96) \times 4 = 640$  unknown coefficients. We consider the homogeneous parts of degree one and three of the public key  $\text{PK}_i$ . These functions are linear in the unknown inhomogeneous terms. Since there are exactly  $(\dim H_1 + \dim H_3) \times 32 = 192\,512$  such functions, we have enough linear equations to recover all inhomogeneous terms.

## Conclusion

We presented a very efficient distinguisher on the main ASASA scheme proposition of [3], that evolved into a full key-recovery algorithm with very reasonable complexity. The complexity of the attack can be approximated by the cost of computing the kernels of  $2^{26}$  matrices of size  $64 \times 96$  with entries in  $\mathbb{F}_{16}$ . This cost is well within practical limits. A classical venture to “repairing” a multivariate cryptosystem is to consider the homogeneous variant of the broken scheme. We point out that our cryptanalysis works by considering the homogenous quadratic parts of the polynomials of the public key, thus defeats any such attempt. Another possibility would be to reinforce the scheme by adding more perturbation polynomials. However, our attack still works without any modification even for a larger number of perturbations.

## References

1. Biham, E.: Cryptanalysis of patarin's 2-round public key system with S boxes (2R). In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 408–416. Springer, Heidelberg (2000)
2. Billet, O., Gilbert, H., Ech-Chatbi, C.: Cryptanalysis of a white box AES implementation. In: Handschuh, H., Hasan, M.A. (eds.) SAC 2004. LNCS, vol. 3357, pp. 227–240. Springer, Heidelberg (2004)
3. Biryukov, A., Bouillaguet, C., Khovratovich, D.: Cryptographic Schemes based on the ASASA structure: black-box, white-box, and public-key (extended abstract). In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8873, pp. 63–84. Springer, Heidelberg (2014)
4. Biryukov, A., Shamir, A.: Structural cryptanalysis of ASASA. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 395–405. Springer, Heidelberg (2001)
5. Daemen, J.: Cipher and hash function design strategies based on linear and differential cryptanalysis. Ph.D. thesis, Doctoral Dissertation, KU Leuven, March 1995
6. De Mulder, Y., Roelse, P., Preneel, B.: Cryptanalysis of the Xiao – Lai white-box AES implementation. In: Knudsen, L.R., Wu, H. (eds.) SAC 2012. LNCS, vol. 7707, pp. 34–49. Springer, Heidelberg (2013)
7. Ding-Feng, Y., Kwok-Yan, L., Zong-Duo, D.: Cryptanalysis of 2R schemes. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 315–325. Springer, Heidelberg (1999)
8. Faugère, J.-C., Perret, L.: Cryptanalysis of  $2R^-$  schemes. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 357–372. Springer, Heidelberg (2006)
9. Goubin, L., Masereel, J.-M., Quisquater, M.: Cryptanalysis of white box DES implementations. In: Adams, C., Miri, A., Wiener, M. (eds.) SAC 2007. LNCS, vol. 4876, pp. 278–295. Springer, Heidelberg (2007)
10. 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)
11. Michiels, W., Gorissen, P., Hollmann, H.D.L.: Cryptanalysis of a generic class of white-box implementations. In: Avanzi, R.M., Keliher, L., Sica, F. (eds.) SAC 2008. LNCS, vol. 5381, pp. 414–428. Springer, Heidelberg (2009)
12. Newman, M.: Integral Matrices. Academic Press, New York (1972)
13. Patarin, J.: Cryptanalysis of the matsumoto and imai public key scheme of euro-crypt '88. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 248–261. Springer, Heidelberg (1995)
14. Patarin, J.: Asymmetric cryptography with a hidden monomial. In: Kobitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 45–60. Springer, Heidelberg (1996)
15. Patarin, J., Goubin, L.: Asymmetric cryptography with S-Boxes is it easier than expected to design efficient asymmetric cryptosystems? In: Information and Communications, Security, pp. 369–380 (1997)
16. Rijmen, V., Preneel, B.: A family of trapdoor ciphers. In: Biham, E. (ed.) FSE 1997. LNCS, vol. 1267, pp. 139–148. Springer, Heidelberg (1997)
17. Sturmfels, B.: Algorithms in Invariant Theory. Springer Science & Business Media, New York (2008)
18. Wu, H., Bao, F., Deng, R.H., Ye, Q.-Z.: Cryptanalysis of Rijmen-Preneel trapdoor ciphers. In: Ohta, K., Pei, D. (eds.) ASIACRYPT 1998. LNCS, vol. 1514, pp. 126–132. Springer, Heidelberg (1998)

**Integrity**

# Online Authenticated-Encryption and its Nonce-Reuse Misuse-Resistance

Viet Tung Hoang<sup>1,2(✉)</sup>, Reza Reyhanitabar<sup>3</sup>, Phillip Rogaway<sup>4</sup>,  
and Damian Vizár<sup>3</sup>

<sup>1</sup> Department of Computer Science, Georgetown University, Washington, D.C, USA  
vth005@eng.ucsd.edu

<sup>2</sup> Department of Computer Science, University of Maryland, College Park, USA  
<sup>3</sup> EPFL, Lausanne, Switzerland

<sup>4</sup> Department of Computer Science, University of California, Davis, USA

**Abstract.** A definition of *online authenticated-encryption* (OAE), call it OAE1, was given by Fleischmann, Forler, and Lucks (2012). It has become a popular definitional target because, despite allowing encryption to be online, security is supposed to be maintained even if nonces get reused. We argue that this expectation is effectively wrong. OAE1 security has also been claimed to capture best-possible security for any online-AE scheme. We claim that this understanding is wrong, too. So motivated, we redefine OAE-security, providing a radically different formulation, OAE2. The new notion effectively *does* capture best-possible security for a user’s choice of plaintext segmentation and ciphertext expansion. It is achievable by simple techniques from standard tools. Yet even for OAE2, nonce-reuse can still be devastating. The picture to emerge is that no OAE definition can meaningfully tolerate nonce-reuse, but, at the same time, OAE security ought never have been understood to turn on this question.

**Keywords:** Authenticated encryption · CAESAR competition · Misuse resistance · Nonce reuse · Online AE · Symmetric encryption

## 1 Introduction

BETWEEN NAE & MRAE. With typical nonce-based authenticated-encryption (nAE) schemes [52, 54], nonces must never repeat when encrypting a series of messages; if they do, it is possible—and routine—that all security will be forfeit. To create some breathing room around this rigid requirement, Rogaway and Shrimpton defined a stronger authenticated-encryption (AE) notion, which they called *misuse-resistant AE* (MRAE) [55]. In a scheme achieving this, repeating a nonce has no adverse impact on authenticity, while privacy is damaged only to the extent that an adversary can detect repetitions of  $(N, A, M)$  triples, these variables representing the nonce, associated data (AD), and plaintext.

While it’s easy to construct MRAE schemes [55], any such scheme must share a particular inefficiency: *encryption can’t be online*. When we speak of

encryption being *online* we mean that it can be realized with constant memory while making a single left-to-right pass over the plaintext  $M$ , writing out the ciphertext  $C$ , also left-to-right, during that pass. The reason an MRAE scheme can't have online encryption is simple: the definition entails that every bit of ciphertext depends on every bit of the plaintext, so one can't output the first bit of a ciphertext before reading the last bit of plaintext. Coupled with the constant-memory requirement, single-pass MRAE becomes impossible.

Given this efficiency/security tension, Fleischmann, Forler, and Lucks (FFL) put forward a security notion [28] that slots between nAE and MRAE. We call it OAE1. Its definition builds on the idea of an *online cipher* due to Bellare, Boldyreva, Knudsen, and Namprempre (BBKN) [15]. Both definitions depend on a constant  $n$ , the *blocksize*. Let  $\mathbf{B}_n = \{0, 1\}^n$  denote the set of  $n$ -bit strings, or *blocks*. An *online cipher* is a blockcipher  $\mathcal{E}: \mathcal{K} \times \mathbf{B}_n^* \rightarrow \mathbf{B}_n^*$  (meaning each  $\mathcal{E}(K, \cdot)$  is a length-preserving permutation) where the  $i$ th block of ciphertext depends only on the key and the first  $i$  blocks of plaintext. An OAE1-secure AE scheme is an AE scheme where encryption behaves like an  $(N, A)$ -tweaked [43] online cipher of blocksize  $n$  followed by a random,  $(N, A, M)$ -dependent tag.

PROBLEMS WITH OAE1. FFL assert that OAE1 supports online-AE and nonce-reuse security. We disagree with the second claim, and even the first.

To begin, observe that as the blocksize  $n$  decreases, OAE1 becomes weaker, in the sense that the ability to perform a chosen-plaintext attack (CPA) implies the ability to decrypt the ciphertext of an  $m$ -block plaintext with  $(2^n - 1)m$  encryption queries. Fix a ciphertext  $C = C_1 \cdots C_m T$  with  $C_i \in \mathbf{B}_n$ , a nonce  $N$ , and an AD  $A$ . Using just an encryption oracle  $\text{Enc}$ , we want to recover  $C$ 's plaintext  $M = M_1 \cdots M_m$  with  $M_i \in \mathbf{B}_n$ . Here's an attack for  $n = 1$ . If  $\text{Enc}(N, A, 0) = C_1$  set  $M_1 = 0$ ; otherwise, set  $M_1 = 1$ . Next, if  $\text{Enc}(N, A, M_1 0) = C_1 C_2$  set  $M_2 = 0$ ; otherwise, set  $M_2 = 1$ . Next, if  $\text{Enc}(N, A, M_1 M_2 0) = C_1 C_2 C_3$  set  $M_3 = 0$ ; otherwise, set  $M_3 = 1$ . And so on, until, after  $m$  queries, one recovers  $M$ . For  $n > 1$  generalize this by encrypting  $M_1 \cdots M_{i-1} M_i$  (instead of  $M_1 \cdots M_{i-1} 0$ ) with  $M_i$  taking on values in  $\mathbf{B}_n$  until one matches  $C_1 \cdots C_i$  or there's only a single possibility remaining. The worst-case number of  $\text{Enc}$  queries becomes  $(2^n - 1)m$ . We call this the *trivial* attack.

The trivial attack might suggest hope for OAE1 security as long as the blocksize is fairly large, like  $n = 128$ . We dash this hope by describing an attack, what we call a *chosen-prefix / secret-suffix* (CPSS) attack, that breaks any OAE1-secure scheme, for any  $n$ , in the sense of recovering  $S$  from given an oracle for  $\mathcal{E}_K^{N,A}(L \parallel \cdot \parallel S)$ , for an arbitrary, known  $L$ . See Sect. 3. The idea was introduced, in a different setting, with the BEAST attack [27].

While many real-world settings won't enable a CPSS attack, our own take is that, for a general-purpose tool, such a weakness effectively refutes any claim of misuse resistance. If the phrase is to mean anything, it should entail that the basic characteristics of nAE are maintained in the presence of nonce-reuse. An AE scheme satisfying nAE (employing non-repeating nonces) or MRAE (without that restriction) would certainly be immune to such an attack.

	OAE1 (from FFL [28])	OAE2 (this paper)
Definitional idea	Online cipher then a tag	Aencrypt each segment
Segmentation	Fixed-size blocks of scheme-determined lengths	Variable-size segments of user-determined lengths
Typical block/seg size	5–16 bytes	1–10000 bytes?
Ciphertext expansion	$\tau$ bits per message	$\tau$ bits per segment
Message space	$M \in \mathbb{B}_n^*$ for blocksize $n$	$M \in \{0, 1\}^*$ or $M \in \{0, 1\}^{**}$
Decryption online?	No, not in general	Yes, automatically
Can aencrypt infinite streams?	No, msgs must end	Yes, msgs can be infinite
OK to repeat nonces?	No, attacks always possible	No, attacks always possible

**Fig. 1. Approaches to formulating online-AE.** It is a thesis of this paper that OAE1 misformulates the desired goal and wrongly promises nonce-reuse misuse-resistance.

We next pull back and take a more philosophical view. We argue that the definition of OAE1 fails in quite basic ways to capture the intuition for what secure online-AE (OAE) ought to do. First, schemes targeting OAE1 conflate the blocksize of the tool being used to construct the scheme and the memory restrictions or latency requirements that motivate OAE in the first place [61]. These two things are unrelated and ought to be disentangled. Second, OAE1 fails to define security for plaintexts that aren’t a multiple of the blocksize. But constructions do just that, encrypting arbitrary bit strings or byte strings. Third, OAE1 measures privacy against an idealized object that’s an online cipher followed by a tag. But having such a structure is not only unnecessary for achieving online encryption, but also undesirable for achieving good security. Finally, while OAE1 aims to ensure that encryption is online, it ignores decryption. The elision has engendered an additional set of definitions for RUP security, “releasing unverified plaintext” [7]. We question the utility of online encryption when one still needs to buffer the entire ciphertext before any portion of the (speculative) plaintext may be disclosed, the implicit assumption behind OAE1.

AN ALTERNATIVE: OAE2. There *are* environments where online encryption is needed. The designer of an FPGA or ASIC encryption/decryption engine might be unable to buffer more than a kilobyte of message. An application like SSH needs to send across a character interactively typed at the keyboard. Netflix needs to stream a film [46] that is “played” as it is received, never buffering an excessive amount or incurring excessive delays. A software library might want to support an incremental encryption and decryption API. Whatever the setting, we think of the plaintext and ciphertext as having been *segmented* into a sequence of *segments*. We don’t control the size of segments—that’s a user’s purview—and different segments can have different lengths.

Thus the basic problem that OAE2 formalizes involves a (potentially long, even infinite) plaintext  $M$  that gets segmented by the user to  $(M_1, \dots, M_m)$ . We must encrypt each segment  $M_i$  as soon as it arrives, carrying forward only a constant-size state. Thus  $M$  gets transformed into a segmented ciphertext  $(C_1, \dots, C_m)$ . Each  $C_i$  must enable immediate recovery of  $M_i$  (the receiver can

no more wait for  $C$ 's completion than the sender can wait for  $M$ 's). We don't insist that  $|C_i| = |M_i|$ ; in fact, the user will do better to grow each segment,  $|C_i| > |M_i|$ , to support expedient verification of what has come so far. See Fig. 1 for a brief comparison of OAE1 and OAE2.

After formulating OAE2, which we do in three approximately-equivalent ways, we describe simple means to achieve it. We don't view OAE2 as a goal for which one should design a fundamentally new AE scheme; the preferred approach is to use a conventional AE scheme and wrap it in a higher-level protocol. We describe two such protocols. The first, **CHAIN**, can be used to turn an MRAE scheme (e.g., SIV) into an OAE2 scheme. The second, **STREAM**, can be used to turn an nAE scheme (e.g., OCB) into a nonce-based OAE scheme. That aim, nOAE, is identical to OAE2 except for insisting that, on the encryption side, nonces don't repeat. Finally, we consider a weakening of OAE2, we call it dOAE, stronger than nOAE and achievable with online processing of each segment.

For reasons of length, the treatment of nOAE, dOAE, and **STREAM** appear only in the full version of this paper [33]. Also see the full version for proofs and a more complete discussion of related work.

We emphasize that moving from OAE1 to OAE2 does not enable one to safely repeat nonces; an OAE2-secure scheme will still be susceptible to CPSS attack, for example. In that light, we would not term an OAE2 scheme *misuse resistant*. What makes OAE2 "better" than OAE1 is not added robustness to nonce-reuse (at least none that we know how to convincingly formalize) but a better modeling of the problem at hand, and a more faithful delivery on the promise of achieving best-possible security for an online-AE scheme. In view of the fact that, with OAE2, one must still deprecate nonce reuse, we would view nOAE as the base-level aim for online-AE.

**RELATED WORK.** A crucial idea for moving beyond BBKN's and FFL's conceptions of online encryption is to sever the association of the blocksize of some underlying tool and the quantum of text a user is ready to operate on. A 2009 technical report of Tsang, Solomakhin, and Smith (TSS) [61] expressed this insight and provided a definition based on it. TSS explain that AE à la Boldyreva and Taesombut [23] (or BBKN or FFL, for that matter) "processes and outputs ... blocks as soon as the next input block is received" [61, p. 4], whence they ask, "what if the input is smaller than a block?", even a bit, or what "if the input is a [segment] ... of arbitrary length?" TSS maintain that such situations occur in practice, and they give examples [61, Sect. 8].

There are major difference in how TSS proceed and how we do. They insist on schemes in which there is ciphertext expansion only at the beginning and end, and their definition is oriented towards that assumption. They do not authenticate the segmented plaintext but the string that is their concatenation. Our formalization of OAE2 lets the adversary run multiple, concurrent sessions of online encryption and decryption, another novum. In the end, the main commonality is some motivation and syntax.

Bertoni, Daemen, Peeters, and Van Assche (BDPV) present a mechanism, the duplex construction, to turn a cryptographic permutation  $f$  into an object very



much like what we are calling an OAE2 scheme [19]. BDPV consider encrypting and authenticating a sequence of messages  $(B_1, B_2, \dots)$  having corresponding headers  $(A_1, A_2, \dots)$ . Asserting that it is “interesting to authenticate and encrypt a sequence of messages in such a way that the authenticity is guaranteed not only on each  $(A, B)$  but also on the sequence received so far” [19, p. 323], they encrypt each  $(A_i, B_i)$  to a ciphertext  $C_i$  and a tag  $T_i$ , the process depending on the prior  $(A_j, B_j)$  values. The authors explain that “Intermediate tags can also be useful in practice to be able to catch fraudulent transactions early” [19, p. 323]. BDPV provide a definition for the kind of AE they speak of [19, Sect. 2]. It resembles both OAE2 and nOAE, and inspired dOAE.

A REAL-WORLD NEED. Netflix recently described a protocol of theirs, MSL, for streaming video [46]. The movie is broken into variable-length segments and each segment is independently encrypted and authenticated, with the ordering of the segments itself authenticated. MSL is based on Encrypt-then-MAC composition, where the encryption is AES-CBC with PKCS#5 padding and the MAC is HMAC-SHA256. The choice suggests that even in real-time applications, use of a two-pass AE scheme for each segment can be fine, as long as segments are of appropriate length. MSL resembles an instantiation of **STREAM**. The current paper provides foundations for the problem that Netflix faced, offering definitions and generic solutions with good provable security.

## 2 OAE1 Definition

All OAE definitions of widespread use spring from FFL [28], who married the definition of an online cipher from Bellare, Boldyreva, Knudsen, and Namprempre [15] with the definition of authenticity of ciphertexts (also called integrity of ciphertexts) [16, 41, 54]. In this section we recall the FFL definition, staying true to the original exposition as much as possible, but necessarily deviating to correct an error. We call the (corrected) definition OAE1.

SYNTAX. For any  $n \geq 1$  let  $B_n = \{0, 1\}^n$  denote the set of  $n$ -bit blocks. A *block-based AE scheme* is a triple  $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$  where the *key space*  $\mathcal{K}$  is a nonempty set with an associated distribution and where the *encryption algorithm*  $\mathcal{E}$  and *decryption algorithm*  $\mathcal{D}$  are deterministic algorithms with signatures  $\mathcal{E}: \mathcal{K} \times \mathcal{H} \times B_n^* \rightarrow \{0, 1\}^*$  and  $\mathcal{D}: \mathcal{K} \times \mathcal{H} \times \{0, 1\}^* \rightarrow B_n^* \cup \{\perp\}$ . The set  $\mathcal{H}$  associated to  $\Pi$  is the *header space*. FFL assumes that it is  $\mathcal{H} = B_n^+ = \mathcal{N} \times \mathcal{A}$  with  $\mathcal{N} = B_n$  and  $\mathcal{A} = B_n^*$  the *nonce space* and *AD space*. The value  $n$  associated to  $\Pi$  is its *blocksize*. Note that the *message space*  $\mathcal{M}$  of  $\Pi$  must be  $\mathcal{M} = B_n^*$  and the *blocksize*  $n$  will play a central role in the security definition. We demand that  $\mathcal{D}(K, N, A, \mathcal{E}(K, N, A, M)) = M$  for all  $K \in \mathcal{K}$ ,  $N \in \mathcal{N}$ ,  $A \in \mathcal{A}$ , and  $M \in B_n^*$ .

To eliminate degeneracies it is important to demand that  $|\mathcal{E}(K, H, M)| \geq |M|$  for all  $K, H, M$  and that  $|\mathcal{E}(K, H, M)|$  depends on at most  $H$  and  $|M|$ . To keep things simple, we assume that the ciphertext expansion  $|\mathcal{E}(K, H, M)| - |M|$  is a constant  $\tau \geq 0$  rather than an arbitrary function of  $H$  and  $|M|$ .

<pre> <b>proc initialize</b> <math>K \leftarrow \mathcal{K}</math>  <b>proc Enc</b>(<math>H, M</math>) <b>if</b> <math>H \notin \mathcal{H}</math> <b>or</b> <math>M \notin \mathbb{B}_n^*</math> <b>then</b>   <b>ret</b> <math>\perp</math> <b>ret</b> <math>\mathcal{E}(K, H, M)</math>  <b>proc Dec</b>(<math>H, C</math>) <b>if</b> <math>H \notin \mathcal{H}</math> <b>then ret</b> <math>\perp</math> <b>ret</b> <math>\mathcal{D}(K, H, C)</math>         </pre>	<pre> <b>proc initialize</b> <b>for</b> <math>H \in \mathcal{H}</math> <b>do</b> <math>\pi_H \leftarrow \text{OPerm}[n]</math> <b>for</b> <math>(H, M) \in \mathcal{H} \times \mathbb{B}_n^*</math> <b>do</b> <math>R_{H,M} \leftarrow \{0, 1\}^\tau</math>  <b>proc Enc</b>(<math>H, M</math>) <b>if</b> <math>H \notin \mathcal{H}</math> <b>or</b> <math>M \notin \mathbb{B}_n^*</math> <b>then ret</b> <math>\perp</math> <b>ret</b> <math>\pi_H(M) \parallel R_{H,M}</math>  <b>proc Dec</b>(<math>H, C</math>) <b>ret</b> <math>\perp</math>         </pre>
---	---

**Fig. 2. OAE1 security.** Defining security for a block-based AE scheme  $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$  with header space  $\mathcal{H}$ , blocksize  $n$ , and ciphertext expansion  $\tau$ .

SECURITY. Let  $\text{OPerm}[n]$  be the set of all length-preserving permutations  $\pi$  on  $\mathbb{B}_n^*$  where  $i$ th block of  $\pi(M)$  depends only on the first  $i$ -blocks of  $M$ ; more formally, a length-preserving permutation  $\pi: \mathbb{B}_n^* \rightarrow \mathbb{B}_n^*$  is in  $\text{OPerm}[n]$  if the first  $|X|$  bits of  $\pi(XY)$  and  $\pi(XY')$  coincide for all  $X, Y, Y' \in \mathbb{B}_n^*$ . Despite its being infinite, one can endow  $\text{OPerm}[n]$  with the uniform distribution in the natural way. To sample from this we write  $\pi \leftarrow \text{OPerm}[n]$ .

Fix a block-based AE scheme  $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$  with  $\mathcal{E}: \mathcal{K} \times \mathcal{H} \times \mathbb{B}_n^* \rightarrow \{0, 1\}^*$ . Then we associate to  $\Pi$  and an adversary  $\mathcal{A}$  the real number  $\text{Adv}_{\Pi}^{\text{OAE1}}(\mathcal{A}) = \Pr[\mathcal{A}^{\text{Real1}} \Rightarrow 1] - \Pr[\mathcal{A}^{\text{Ideal1}} \Rightarrow 1]$  where games **Real1** and **Ideal1** are defined in Fig. 2. Adversary  $\mathcal{A}$  may not ask a Dec query  $(H, C)$  after an Enc query  $(H, M)$  returned  $C$ . Informally,  $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$  is OAE1 secure if  $\text{Adv}_{\Pi}^{\text{OAE1}}(\mathcal{A})$  is small for any reasonable  $\mathcal{A}$ . Alternatively, we can speak of OAE1[ $n$ ] security to emphasize the central role in defining security of the scheme’s blocksize  $n$ .

DISCUSSION. The OAE1 definition effectively says that, with respect to privacy, a ciphertext must resemble the image of a plaintext under a random online permutation (tweaked by the nonce and AD) followed by a  $\tau$ -bit random string (the authentication tag). But the original definition from FFL somehow omitted the second part [28, Definition 3]. The lapse results in a definition that makes no sense, as  $\mathcal{E}$  must be length-increasing to provide authenticity. The problem was large enough that it wasn’t clear to us what was intended. Follow-on work mostly replicated this [2, 29]. After discussions among ourselves and checking with one of the FFL authors [44], we concluded that the intended definition is the one we have given.

LCP LEAKAGE. Say that a block-based AE scheme  $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$  with blocksize  $n$  is LCP[ $n$ ] (for “longest common prefix”) if for all  $K, H, M$ , and  $i \leq \lfloor |M|/n \rfloor$ , the first  $i$  blocks of  $\mathcal{E}_K^H(M)$  depend only on the first  $i$  blocks of  $M$ . While all schemes we know claiming to be OAE1[ $n$ ] are also LCP[ $n$ ], an OAE1[ $n$ ]-secure scheme isn’t necessarily LCP[ $n$ ]. This is because the requirement for OAE1[ $n$ ] security is to be computationally close to an object that is LCP[ $n$ ], and

something being computationally close to something with a property  $P$  doesn't mean it has property  $P$ . Indeed it is easy to construct an artificial counterexample; for example, starting with a OAE1[ $n$ ]-secure scheme that is LCP[ $n$ ], augment the key with  $n$  extra bits,  $K'$ , and modify encryption so that when the first block of plaintext coincides with  $K'$ , then reverse the bits of the remainder of the plaintext before proceeding. OAE1 security is only slightly degraded but the scheme is no longer LCP[ $n$ ]. Still, despite such counterexamples, an OAE1[ $n$ ]-secure scheme must be *close* to being LCP[ $n$ ]. Fix  $\Pi$  as above and consider an adversary  $\mathcal{A}$  that is given an oracle  $\mathcal{E}_K(\cdot, \cdot)$  for  $K \leftarrow \mathcal{K}$ . Consider  $\mathcal{A}$  to be *successful* if it outputs  $H \in \mathcal{H}$  and  $X, Y, Y' \in \mathbb{B}_n^*$  such that the first  $|X|/n$  blocks of  $\mathcal{E}_K^H(XY)$  and  $\mathcal{E}_K^H(XY')$  are different (i.e., the adversary found non-LCP behavior). Let  $\text{Adv}_{\Pi}^{\text{lcp}}(\mathcal{A})$  be the probability that  $\mathcal{A}$  is successful. Then it's easy to transform  $\mathcal{A}$  into an equally efficient adversary  $\mathcal{B}$  for which  $\text{Adv}_{\Pi}^{\text{OAE1}}(\mathcal{B}) = \text{Adv}_{\Pi}^{\text{lcp}}(\mathcal{A})$ . Because of this, there is no real loss of generality, when discussing OAE1[ $n$ ] schemes, to assume them LCP[ $n$ ]. In the next section we will do so.

### 3 CPSS Attack

Section 1 described the *trivial* attack to break OAE1-secure schemes with too small a blocksize. We now describe a different fixed-header CPA attack, this one working for any blocksize. We call the attack a *chosen-prefix, secret-suffix* (CPSS) attack. The attack is simple, yet devastating. It is inspired by the well-known BEAST (Browser Exploit Against SSL/TLS) attack [27].

Let  $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$  be a block-based AE scheme with blocksize  $n$  satisfying LCP[ $n$ ]. We consider a setting where messages  $M = P \parallel S$  that get encrypted can be logically divided into a prefix  $P$  that is controlled by an adversary, then a suffix  $S$  that is secret, fixed, and not under the adversary's control. The adversary wants to learn  $S$ . We provide it the ability to obtain an encryption of  $\mathcal{E}_K^H(P \parallel S)$  for any  $P$  it wants—except, to be realistic, we insist that  $P$  be a multiple of  $b$  bits. This is assumed for  $S$  too. Typically  $P$  and  $S$  must be byte strings, whence  $b = 8$ ; for concreteness, let us assume this. Also for concreteness, assume a blocksize of  $n = 128$  bits. Assume that  $\mathcal{E}$  can in fact operate on arbitrary byte-length strings, but suffers LCP leakage on block-aligned prefixes (this is what happens if one pads and then applies an OAE1-secure scheme). Finally, assume  $|S|$  is a multiple of the blocksize.

To recover  $S$ , the adversary proceeds as follows. First it selects an arbitrary string  $P_1$  whose byte length is one byte shorter than  $p$  blocks, for an arbitrary  $p \geq 1$ . (For example, it would be fine to have  $P_1 = 0^{120}$ .) The adversary requests ciphertext  $C_1 = \mathcal{E}_K^H(P_1 \parallel S)$ . This will be used to learn  $S_1$ , the first byte of  $S$ . To do so, the adversary requests ciphertexts  $C_{1,B} = \mathcal{E}_K^H(P_1 \parallel B \parallel S)$  for all 256 one-byte values  $B$ . Due to LCP leakage, exactly one of these values, the one with  $B = S_1$ , will agree with  $C_1$  on the first  $p$  blocks. At this point the adversary knows the first byte of  $S$ , and has spent 257 queries to get it. There is an obvious strategy to reduce this to 256 queries: omit one of the 256-possible byte values for  $B$  and use this for  $S_1$  if no other match is found.

Now the adversary wants to learn  $S_2$ , the second byte of  $S$ . It selects an arbitrary string  $P_2$  that is *two* bytes short of  $p$  blocks, for any  $p \geq 1$ . The adversary requests the ciphertext  $C_2 = \mathcal{E}_K^H(P_2 \parallel S)$ ; and it requests ciphertexts  $C_{2,B} = \mathcal{E}_K^H(P_2 \parallel S_1 \parallel B \parallel S)$  for all 256 one-byte values  $B$ . Due to LCP leakage and the fact that we have matched the first byte  $S_1$  of  $S$  already, exactly one of these 256 values, call it  $S_2$ , will agree with  $C_2$  on the first  $p$  blocks. At this point the adversary knows  $S_2$ , the second byte of  $S$ . It has used 257 more queries to get this. This can be reduced to 256 as before.

Continuing in this way, the adversary recovers all of  $S$  in  $256|S|/8$  queries. In general, we need  $2^b|S|/b$  queries to recover  $S$ . Note that the adversary has considerable flexibility in selecting the values that prefix  $S$ : rather than this being completely chosen by the adversary, it is enough that it be a known, fixed value, followed by the byte string that the adversary can fiddle with. That is, the CPSS attack applies when the adversary can manipulate a portion  $R$  of values  $L \parallel R \parallel S$  that get encrypted, where  $L$  is known and  $S$  is not.

HOW PRACTICAL? It is not uncommon to have protocols where there is a predictable portion  $L$  of a message, followed by an adversarially mutable portion  $R$  specifying details, followed by further information  $S$ , some or all of which is sensitive. This happens in HTTP, for example, where the first portion of the request specifies a method, such as `GET`, the second specifies a resource, such as `/img/scheme.gif/`, and the final portion encodes information such as the HTTP version number, an end-of-line character, and a session cookie. If an LCP-leaking encryption scheme is used in such a setting, one is asking for trouble.

Of course we do not suggest that LCP leakage will always foreshadow a real-world break. But the whole point of having general-purpose notions and provable-security guarantees is to avoid relying on application-specific characteristics of a protocol to enable security. If misuse comes as easily as giving adversaries the ability to manipulate a middle portion  $L \parallel R \parallel S$  of plaintexts, one has strayed very far indeed from genuine misuse-resistance.

MRAE AND CPSS. In the full version [33] we evidence that MRAE provides a modicum of misuse resistance that OAE1 lacks by establishing the rather obvious result that any MRAE-secure scheme resists CPSS attack.

## 4 Broader OAE1 Critique

The CPSS attack suggests that the OAE1 definition is “wrong” in the sense that it promises nonce-reuse security but compliant protocols are susceptible to realistic fixed-nonce attacks. In this section we suggest that OAE1’s defects are more fundamental—that the definition fails to capture the intuition about what something called “online-AE” ought do. Our complaints are thus philosophical, but only in the sense that assessing the worth of a cryptographic definition always includes assessing the extent to which it delivers on some underlying intuition.

*The Blocksize Should not be a Scheme-Dependent Constant.* A reasonable syntactic requirement for online-AE would say that the  $i$ th bit of ciphertext should depend only on the first  $i$  bits of plaintext (and, of course, the key, nonce, and AD). This would make online-AE something akin to a stream cipher. But the requirement above is not what OAE1 demands—it demands that the  $i$ th *block* depends only on the first  $i$  *blocks* of plaintext. Each of these blocks has a fixed *blocksize*, some number  $n$  associated to the scheme *and* its security definition. Thus implicit in the OAE1 notion is the idea that there is going to be *some* buffering of the bits of an incoming message before one can output the next block of bits. It is not clear if this fixed amount of buffering is done as a matter of efficiency, simplicity, or security. In schemes targeting OAE1-security, the blocksize is usually small, like 128 bits, the value depending on the width of some underlying blockcipher or permutation used in the scheme’s construction.

That there’s a blocksize parameter at all implies that, to the definition’s architects, it is desirable, or at least acceptable, to buffer *some* bits of plaintext before acting on them—just not too many. But the number of bits that are reasonable to buffer is application-environment specific. One application might need to limit the blocksize to 128 KB, so as to fit comfortably within the L2 cache of some CPU. Another application might need to limit the blocksize to 1 KB, to fit compactly on some ASIC or FPGA. Another application might need to limit the blocksize to a single byte, to ensure bounded latency despite bytes arriving at indeterminate times. The problem is that the designer of a cryptographic scheme is in no position to know the implementation-environment’s constraint that motivates the selection of a blocksize in the first place. By choosing some fixed blocksize, a scheme’s designer simultaneously forecloses on an implementation’s potential need to buffer less *and* an implementation’s potential ability to buffer more. *Any* choice of a blocksize replaces a user’s environment-specific constraint by a hardwired choice from a primitive’s designer.

(Before moving on let us point out that, if it *is* the amount of memory available to an implementation that is an issue, the right constraint is not the blocksize  $n$ , where block  $C_i$  depends only on prior blocks, but the requirement that an implementation be achievable in one pass and  $n$  bits of memory. These are not the same thing [56, p. 241]. And the former is a poor substitute for the latter since context sizes vary substantially from scheme to scheme. While one *could* build an OAE notion by parameterizing its online memory requirement, we find it more appealing to eliminate any such parameter.)

*Security must be Defined for all Plaintexts.* The OAE1[ $n$ ] notion only defines security when messages are a multiple of  $n$  bits. What should security mean when the message space is larger, like  $\mathcal{M} = \{0, 1\}^*$ ? Saying “we pad first, so needn’t deal with strings that aren’t multiples of the blocksize” is a complete non-answer, as it leaves unspecified what the goal *is* one is aiming to achieve by padding on the message space of interest—the one before padding is applied.

There are natural ways to try to extend OAE1[ $n$ ] security to a larger message space; see, for example, the approach used for online ciphers on  $\{0, 1\}^{\geq n}$  [56]. This can be extended to OAE1. But it is not the only approach, and there will

still be issues for dealing with strings of fewer than  $n$  bits. In general, we think that an online-AE definition is not really meaningful, in practice, until one has specified what security means on the message space  $\mathcal{M} = \{0, 1\}^*$ .

*Decryption too must be Online.* If one is able to produce ciphertext blocks in an online fashion one had better be able to process them as they arrive. Perhaps the message was too long to store on the encrypting side. Then the same will likely hold on the decrypting side. Or perhaps there are timeliness constraints that one needs to act on a message fragment *now*, before the remainder of it arrives. Think back to the Netflix scenario. It would be pointless to encrypt the film in an online fashion only to have to buffer the entire thing at the receiver before it could play.

But online decryption is not required by OAE1 security, and it is routine that online decryption of each provided block would be fatal. We conjecture that it is an unusual scenario where it is important for encryption be computable online but irrelevant if decryption can be online as well.

*The OAE1 Reference Object is not Ideal.* The reference object for OAE1[ $n$ ] security pre-supposes that encryption resembles an online-cipher followed by a random-looking tag. But it is wrong to think of this as capturing ideal behavior. First, it implicitly assumes that all authenticity is taken care of at the very end. But if a plaintext is long and one is interested in encryption being online to ensure timeliness, then waiting until the end of a ciphertext to check authenticity make no sense. If one is going to act on a prefix of a plaintext when it's recovered, it better be authenticated. Second, it is simply irrelevant, from a security point of view, if, prior to receipt of an authentication tag, encryption amounts to length-preserving permutation. Doing this may minimize ciphertext length, but that is an efficiency issue, not a basic goal. And achieving this particular form of efficiency is at odds with possible authenticity aims.

## 5 OAE2: Reformalizing Online-AE

We provide a new notion for online-AE. We will call it OAE2. To accurately model the underlying goal, not only must the security definition depart from that used by nAE and MRAE, but so too must a scheme's basic syntax. In particular, we adopt an API-motivated view in which the segmentation of a plaintext is determined by the caller.

After defining the syntax we offer three ways to quantify the advantage an adversary gets in attacking an OAE2 scheme. We term these advantage measures OAE2a, OAE2b, OAE2c. The notions are essentially equivalent. We provide quantitative results to make this *essentially* precise.

Why describe three different advantage measures of OAE2 security? We think it helps clarify just what OAE2 really *is*. The measures have different characteristics. The first, OAE2a, is a vector-oriented formulation. It employs a fairly easy-to-understand reference object. The second advantage measure, OAE2b, is

a string-oriented formulation. It employs a tighter and more realistic accounting of the adversary’s actual resource expenditure. The third advantage measure, OAE2c, is more aspirational in character. Yet it is the easiest notion to work with, at least for proving schemes OAE2-secure. The OAE2c measure only makes sense if the segment-expansion  $\tau$  is fairly large.

We begin with a bit of notation.

**SEGMENTED STRINGS.** Denote by  $\{0, 1\}^{**} = (\{0, 1\}^*)^*$  the set of *segmented-strings*: a segmented string  $\mathbf{X} \in \{0, 1\}^{**}$  is a vector (or list) of strings. Each of its components, which we call a segment, is a string. The segmented-string with zero components is the empty list  $\Lambda$ . This is different from the empty string  $\varepsilon$ . The number of components in a segmented-string  $\mathbf{X}$  is denoted  $|\mathbf{X}|$ , while the  $i$ th component of  $\mathbf{X}$ ,  $i \in [1..|\mathbf{X}|]$ , is denoted  $\mathbf{X}[i]$ . Note that indexing begins at 1. For  $\mathbf{X} \in \{0, 1\}^{**}$  and  $1 \leq i \leq j \leq |\mathbf{X}|$ , by  $\mathbf{X}[i..j]$  we mean the  $(j - i + 1)$ -vector  $(\mathbf{X}[i], \mathbf{X}[i + 1], \dots, \mathbf{X}[j])$ . If  $\mathbf{X} \in \{0, 1\}^{**}$  and  $X \in \{0, 1\}^*$  then  $\mathbf{X} \parallel X$  is the  $|\mathbf{X}| + 1$  vector consisting of the components of  $\mathbf{X}$ , in order, followed by  $X$ . Keep in mind that this is not concatenation of strings but, instead, appending a string to vector of strings to get a longer vector of strings. We emphasize that a segmented string is not a string.

**SCHEME SYNTAX.** A *segmented-AE scheme* is a tuple  $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$  where the *key space*  $\mathcal{K}$  is a nonempty set with an associated distribution and both encryption  $\mathcal{E} = (\mathcal{E}.init, \mathcal{E}.next, \mathcal{E}.last)$  and decryption  $\mathcal{D} = (\mathcal{D}.init, \mathcal{D}.next, \mathcal{D}.last)$  are specified by triples of deterministic algorithms. Associated to  $\Pi$  are its *nonce space*  $\mathcal{N} \subseteq \{0, 1\}^*$  and its *state space*  $\mathcal{S}$ . For simplicity, a scheme’s *AD space*  $\mathcal{A} = \{0, 1\}^*$ , *message space*  $\mathcal{M} = \{0, 1\}^*$ , and *ciphertext space*  $\mathcal{C} = \{0, 1\}^*$  are all strings. While an AD will be provided with each plaintext segment, a single nonce is provided for the entire sequence of segments. The signature of the components of  $\mathcal{E}$  and  $\mathcal{D}$  are as follows:

$$\begin{array}{ll} \mathcal{E}.init: \mathcal{K} \times \mathcal{N} \rightarrow \mathcal{S} & \mathcal{D}.init: \mathcal{K} \times \mathcal{N} \rightarrow \mathcal{S} \\ \mathcal{E}.next: \mathcal{S} \times \mathcal{A} \times \mathcal{M} \rightarrow \mathcal{C} \times \mathcal{S} & \mathcal{D}.next: \mathcal{S} \times \mathcal{A} \times \mathcal{C} \rightarrow (\mathcal{M} \times \mathcal{S}) \cup \{\perp\} \\ \mathcal{E}.last: \mathcal{S} \times \mathcal{A} \times \mathcal{M} \rightarrow \mathcal{C} & \mathcal{D}.last: \mathcal{S} \times \mathcal{A} \times \mathcal{C} \rightarrow \mathcal{M} \cup \{\perp\} \end{array}$$

When an algorithm takes or produces a point  $S \in \mathcal{S}$  from its state space, it is understood that a fixed encoding of  $S$  is employed.

Given a segmented-AE scheme  $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$  there are *induced* encryption and decryption algorithms  $\mathcal{E}, \mathcal{D}: \mathcal{K} \times \mathcal{N} \times \{0, 1\}^{**} \times \{0, 1\}^{**} \rightarrow \{0, 1\}^{**}$  (note the change to bold font) that operate, all at once, on vectors of plaintext, ciphertext, and AD. These maps are defined in Fig. 3. Observe how  $\text{Dec}(K, N, \mathbf{A}, \mathbf{C})$  returns a longest  $\mathbf{M}$  whose encryption (using  $K$ ,  $N$ , and  $\mathbf{A}$ ) is a prefix of  $\mathbf{C}$ ; in essence, we stop at the first decryption failure, so  $|\mathbf{C}| = |\mathbf{M}|$  if and only if  $\mathbf{C}$  is entirely valid. We require the following validity condition for any segmented-AE scheme  $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$  with induced  $(\mathcal{E}, \mathcal{D})$ : if  $K \in \mathcal{K}$ ,  $N \in \mathcal{N}$ ,  $\mathbf{A} \in \{0, 1\}^{**}$ ,  $\mathbf{M} \in \{0, 1\}^{**}$ , and  $\mathbf{C} = \mathcal{E}(K, N, \mathbf{A}, \mathbf{M})$ , then  $\mathbf{M} = \mathcal{D}(K, N, \mathbf{A}, \mathbf{C})$ .

<pre> <b>algorithm</b> <math>\mathcal{E}(K, N, \mathbf{A}, \mathbf{M})</math> <math>m \leftarrow  \mathbf{M} </math>; <b>if</b> <math>m = 0</math> <b>or</b> <math> \mathbf{A}  \neq  \mathbf{M} </math> <b>then ret</b> <math>\perp</math> <math>(A_1, \dots, A_m) \leftarrow \mathbf{A}</math> <math>(M_1, \dots, M_m) \leftarrow \mathbf{M}</math> <math>S_0 \leftarrow \mathcal{E}.init(K, N)</math> <b>for</b> <math>i \leftarrow 1</math> <b>to</b> <math>m - 1</math> <b>do</b>   <math>(C_i, S_i) \leftarrow \mathcal{E}.next(S_{i-1}, A_i, M_i)</math> <math>C_m \leftarrow \mathcal{E}.last(S_{m-1}, A_m, M_m)</math> <b>ret</b> <math>(C_1, \dots, C_m)</math> </pre>	<pre> <b>algorithm</b> <math>\mathcal{D}(K, N, \mathbf{A}, \mathbf{C})</math> <math>m \leftarrow  \mathbf{C} </math> <b>if</b> <math>m = 0</math> <b>or</b> <math> \mathbf{A}  \neq  \mathbf{M} </math> <b>then ret</b> <math>\perp</math> <math>(A_1, \dots, A_m) \leftarrow \mathbf{A}</math>; <math>(C_1, \dots, C_m) \leftarrow \mathbf{C}</math> <math>S_0 \leftarrow \mathcal{D}.init(K, N)</math> <b>for</b> <math>i \leftarrow 1</math> <b>to</b> <math>m - 1</math> <b>do</b>   <b>if</b> <math>\mathcal{D}.next(S_{i-1}, A_i, C_i) = \perp</math> <b>then</b>     <b>if</b> <math>m = 1</math> <b>ret</b> <math>\perp</math>     <b>else ret</b> <math>(M_1, \dots, M_{i-1})</math>   <b>else</b> <math>(M_i, S_i) \leftarrow \mathcal{D}.next(S_{i-1}, A_i, C_i)</math> <math>M_m \leftarrow \mathcal{D}.last(S_{m-1}, A_m, C_m)</math> <b>if</b> <math>M_m = \perp</math> <b>then ret</b> <math>(M_1, \dots, M_{m-1})</math> <b>else ret</b> <math>(M_1, \dots, M_m)</math> </pre>
--	--

**Fig. 3. Operating on segmented strings.** The figure shows the algorithms  $\mathcal{E}$  and  $\mathcal{D}$  that are *induced* by the segmented encryption scheme  $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ .

CIPHERTEXT EXPANSION. We focus on segmented-AE schemes with constant segment-expansion, defined as follows: associated to  $\Pi$  is a number  $\tau \geq 0$  such that if  $K \in \mathcal{K}$ ,  $N \in \mathcal{N}$ ,  $\mathbf{A} \in \{0, 1\}^{**}$ ,  $\mathbf{M} \in \{0, 1\}^{**}$ ,  $m = |\mathbf{A}| = |\mathbf{M}|$ , and  $\mathbf{C} = \mathcal{E}(K, N, \mathbf{A}, \mathbf{M})$ , then  $|\mathbf{C}[i]| = |\mathbf{M}[i]| + \tau$  for all  $i \in [1..m]$ . Thus each segment grows by exactly  $\tau$  bits, for some constant  $\tau$ . We call  $\tau$  the *segment-expansion* of  $\Pi$ .

We favor constant segment-expansion because we think it runs contrary to the spirit of online-AE to furnish interior segments with an inferior authenticity guarantee than that afforded to the whole message. After all, much of the point of online-AE is to allow a decrypting party to safely act on a ciphertext segment as soon as its available. Still, there is an obvious efficiency cost to expanding every segment. See the heading “Multivalued segment-expansion” for the case where the amount of segment-expansion is position dependent.

ONLINE COMPUTABILITY. We say that a segmented-AE scheme  $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$  has *online-encryption* if its state space  $\mathcal{S}$  is finite and there’s a constant  $w$  such that  $\mathcal{E}.next$  and  $\mathcal{E}.last$  use at most  $w$  bits of working memory. The value  $w$  excludes memory used for storing an algorithm’s inputs or output; we elaborate below. Similarly, scheme  $\Pi$  has *online-decryption* if its state space  $\mathcal{S}$  is finite and there’s a constant  $w$  such that  $\mathcal{D}.next$  and  $\mathcal{D}.last$  use at most  $w$  bits of working memory. A segmented-AE scheme is *online* if it has online-encryption and online-decryption. In accounting for memory above, the model of computation provides input values on a read-only input tape; the input’s length is not a part of the working memory accounted for by  $w$ . Similarly, algorithms produce output by writing to a write-only output tape in a left-to-right fashion. The number of bits written out has nothing to do with the working memory  $w$ .

Our security definitions don’t care if a segmented-AE scheme is online: that’s an efficiency requirement, not a security requirement. Yet a good part of the purpose of the segmented-AE syntax is to properly deal with schemes that have such efficiency constraints.



<pre> <b>proc initialize</b> <math>K \leftarrow \mathcal{K}</math>  <b>proc Enc</b>(<math>N, \mathbf{A}, M</math>) <b>if</b> <math>N \notin \mathcal{N}</math> <b>or</b> <math> \mathbf{A}  \neq  M </math> <b>then ret</b> <math>\perp</math> <b>ret</b> <math>\mathcal{E}(K, N, \mathbf{A}, M)</math>  <b>proc Dec</b>(<math>N, \mathbf{A}, C</math>) <b>if</b> <math>N \notin \mathcal{N}</math> <b>or</b> <math> \mathbf{A}  \neq  M </math> <b>then ret</b> <math>\perp</math> <b>ret</b> <math>\mathcal{D}(K, N, \mathbf{A}, C)</math>                 </pre>	<div style="border: 1px solid black; padding: 2px; display: inline-block;"><b>Real2A<math>_{\Pi}</math></b></div>	<pre> <b>proc initialize</b> <math>F \leftarrow \text{IdealOAE}(\tau)</math>  <b>proc Enc</b>(<math>N, \mathbf{A}, M</math>) <b>if</b> <math>N \notin \mathcal{N}</math> <b>or</b> <math> \mathbf{A}  \neq  M </math> <b>then ret</b> <math>\perp</math> <b>ret</b> <math>F(N, \mathbf{A}, M, 1)</math>  <b>proc Dec</b>(<math>N, \mathbf{A}, C</math>) <b>if</b> <math>N \notin \mathcal{N}</math> <b>or</b> <math> \mathbf{A}  \neq  M </math> <b>then ret</b> <math>\perp</math> <b>if</b> <math>\exists M</math> s.t. <math>F(N, \mathbf{A}, M, 1) = C</math> <b>ret</b> <math>M</math> <math>M \leftarrow</math> the longest vector in     {<math>M: F(N, \mathbf{A}, M, 0)[i] = C[i]</math>      for <math>i \in [1.. M  - 1]</math>} <b>ret</b> <math>M</math>                 </pre>	<div style="border: 1px solid black; padding: 2px; display: inline-block;"><b>Ideal2A<math>_{\Pi}</math></b></div>
---	---	--	--

**Fig. 4. OAE2a security.** The segmented-AE scheme  $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$  has nonce space  $\mathcal{N}$  and segment-expansion  $\tau$ . It induces algorithms  $\mathcal{E}, \mathcal{D}$  as per Fig. 3. The distribution  $\text{IdealOAE}(\tau)$  is described in the text.

FIRST OAE2 DEFINITION: OAE2a. We begin by defining the *ideal* behavior for an OAE scheme. Let  $\text{Inj}(\tau)$  denote the set of all  $\tau$ -expanding injective functions—the set of all functions  $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$  that are injective and satisfy  $|f(x)| = |x| + \tau$ . Endow this set with the uniform distribution in the natural way. We write  $f \leftarrow \text{Inj}(\tau)$  to denote uniformly sampling a random,  $\tau$ -expanding injective function. Now define a distribution on functions  $\text{IdealOAE}(\tau)$  as follows:

```

for  $m \in \mathbb{Z}^+, N \in \{0, 1\}^*, \mathbf{A} \in (\{0, 1\}^*)^m, \mathbf{M} \in (\{0, 1\}^*)^{m-1}$  do
     $f_{N, \mathbf{A}, M, 0} \leftarrow \text{Inj}(\tau); f_{N, \mathbf{A}, M, 1} \leftarrow \text{Inj}(\tau)$ 
for  $m \in \mathbb{Z}^+, \mathbf{A} \in (\{0, 1\}^*)^m, \mathbf{X} \in (\{0, 1\}^*)^m, \delta \in \{0, 1\}$  do
     $F(N, \mathbf{A}, \mathbf{X}, \delta) \leftarrow (f_{N, \mathbf{A}[1..1], \mathbf{A}, 0}(\mathbf{X}[1]), f_{N, \mathbf{A}[1..2], \mathbf{X}[1..1], 0}(\mathbf{X}[2]),$ 
         $f_{N, \mathbf{A}[1..3], \mathbf{X}[1..2], 0}(\mathbf{X}[3]), \dots, f_{N, \mathbf{A}[1..m-1], \mathbf{X}[1..m-2], 0}(\mathbf{X}[m-1]),$ 
         $f_{N, \mathbf{A}[1..m], \mathbf{X}[1..m-1], \delta}(\mathbf{X}[m]))$ 
ret  $F$ 
                
```

Thus  $F \leftarrow \text{IdealOAE}(\tau)$  grows by accretion, the  $i$ th component of  $F(N, \mathbf{A}, \mathbf{X}, 0)$  depending on  $N, \mathbf{A}[1..i]$ , and  $\mathbf{X}[1..i]$ . It must be decryptable (hence the injectivity) and have the mandated length. The final input to  $F$ , the flag  $\delta$ , indicates if the argument  $\mathbf{X}$  is complete: a 1 means it is, a 0 means it's not.

Figure 4 defines games **Real2A $_{\Pi}$**  and **Ideal2A $_{\Pi}$**  for a  $\tau$ -expanding segmented-AE scheme  $\Pi$ . Given an adversary  $\mathcal{A}$  with oracles  $\text{Enc}$  and  $\text{Dec}$  determined by these games, let  $\text{Adv}_{\Pi}^{\text{OAE2a}}(\mathcal{A}) = \Pr[\mathcal{A}^{\text{Real2A}_{\Pi}} \Rightarrow 1] - \Pr[\mathcal{A}^{\text{Ideal2A}_{\Pi}} \Rightarrow 1]$  be the adversary's distinguishing advantage.

DISCUSSION. The security notion may be described as follows. A user wants to encrypt a segmented message  $\mathbf{M} = (M_1, \dots, M_m)$  into a ciphertext  $\mathbf{C} = (C_1, \dots, C_m)$  using  $K, N, \mathbf{A}$ . He wants to do this *as well as possible* subject to the constraint that segments grow by exactly  $\tau$  bits and  $M_1 \cdots M_i$  are recoverable from  $K, N, (A_1, \dots, A_i), (C_1, \dots, C_i)$ . As with robust-AE [34], the phrase “as well

as possible” targets an achievable (instead of aspirational) goal. Specifically, it is formalized by comparing the real object to a random element from  $\text{IdealOAE}(\tau)$  and its inverse, the later understood to invert as many components as possible, stopping at the first point one can't proceed.

The definition of  $\text{IdealOAE}(\tau)$  is complex enough that an example may help. Consider encrypting a segmented plaintext  $M = (A, B, C, D)$  with a fixed key, nonce, and AD. Let  $(U, V, X, Y)$  be the result. Now encrypt  $M' = (A, B, C)$ . We want this to give  $(U, V, Z)$ , not  $(U, V, X)$ , as the final segment is special: processed by  $\mathcal{E}.\text{last}$  instead of  $\mathcal{E}.\text{next}$ , it is as though  $M = (A, B, C, D)$  means  $(A, B, C, D\$)$ , while  $M = (A, B, C)$  means  $(A, B, C\$)$ , where the  $\$$ -symbol is an end-of-message sentinel. Written like this, it is clear that the two segmented ciphertexts should agree on the first two components but not the third. Correspondingly, possession of  $(U, V, X, Y)$  ought not enable a forgery of  $(U, V, X)$ . All of this understanding gets quietly embedded into the definition of  $\text{IdealOAE}(\tau)$ , whose member functions get a final argument  $\delta$  with semantics indicating if the message is *complete*. Thus  $F(N, \mathbf{A}, (A, B, C), 0)$  is what  $\mathbf{M} = (A, B, C)$  should map to if more segments are to come, while  $F(N, \mathbf{A}, (A, B, C), 1)$  is what it should map to if  $C$  is the final segment of  $\mathbf{M}$ .

**SECOND OAE2 DEFINITION:** OAE2b. Figure 5 gives a more fine-grained and string-oriented measure for OAE2 security. The adversary, instead of providing  $N, \mathbf{A}, \mathbf{M}$  and getting a vector  $\mathbf{C} = \text{Enc}(N, \mathbf{A}, \mathbf{M})$ , can adaptively grow  $\mathbf{A}$  and  $\mathbf{M}$  one component at a time. Similarly, instead of providing a segmented ciphertext  $N, \mathbf{A}, \mathbf{C}$  and getting  $\mathbf{M} = \text{Dec}(N, \mathbf{A}, \mathbf{C})$ , it can adaptively grow  $\mathbf{A}, \mathbf{C}$ . As before, we associate to a  $\tau$ -expanding segmented-AE scheme  $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$  and an adversary  $\mathcal{A}$  the real number  $\text{Adv}_{\Pi}^{\text{OAE2b}}(\mathcal{A}) = \Pr[\mathcal{A}^{\text{Real2B}_{\Pi}} \Rightarrow 1] - \Pr[\mathcal{A}^{\text{Ideal2B}_{\Pi}} \Rightarrow 1]$  that is its distinguishing advantage.

The OAE2a and OAE2b measures are essentially equivalent. The *essentially* of this sentence entails a simple result explaining how to convert an adversary for one definition into an adversary for the other. First, given an oae2a-style adversary  $\mathcal{A}$  we can construct an equally effective oae2b-style adversary  $\mathcal{B}$ : it translates each  $\text{Enc}(N, (A_1, \dots, A_m), (M_1, \dots, M_m))$  asked by adversary  $\mathcal{A}$  into an  $\text{Enc}.\text{init}$ , then  $m - 1$   $\text{Enc}.\text{next}$  calls, then an  $\text{Enc}.\text{last}$  call, assembling the answers into a segmented ciphertext  $(C_1, \dots, C_m)$ . Similarly, it translates  $\text{Dec}(N, (A_1, \dots, A_m), (C_1, \dots, C_m))$  calls into  $\text{Dec}.\text{init}$ ,  $\text{Dec}.\text{next}$ ,  $\text{Dec}.\text{last}$  calls. Adversary  $\mathcal{B}$  gets exactly the oae2b-advantage that  $\mathcal{A}$  had as oae2a-advantage. It runs in almost the exact same time.

Simulation in the other direction is less efficient. Given an adversary  $\mathcal{A}$  attacking the oae2b-security of a  $\Pi$ , we construct an adversary  $\mathcal{B}$  for attacking the oae2a-security of the same scheme. Adversary  $\mathcal{B}$  maintains lists  $N_i, \mathbf{A}_i, \mathbf{M}_i$  that are initialized in the natural way with each  $\text{Enc}.\text{init}$  call (incrementing  $i$ , initially zero, with each  $\text{Enc}.\text{init}$ ). Calls of the form  $\text{Enc}.\text{next}(i, A, M)$ , when valid, result in appending  $A$  to  $\mathbf{A}_i$  and  $M$  to  $\mathbf{M}_i$ , making an  $\text{Enc}(N_i, \mathbf{A}_i \parallel \varepsilon, \mathbf{M}_i \parallel \varepsilon)$  call, and returning its  $|\mathbf{M}_i|$ -th component. Calls of the form  $\text{Enc}.\text{last}(i, A, M)$  result in making an  $\text{Enc}(N_i, \mathbf{A}_i \parallel A, \mathbf{M}_i \parallel M)$  call, returning its last component, resetting  $\mathbf{M}_i$  to  $\perp$  before doing so. Calls of the form  $\text{Dec}.\text{init}$ ,  $\text{Dec}.\text{next}$ ,

<pre> <b>proc initialize</b> <span style="border: 1px solid black; padding: 2px;">Real2B<math>\Pi</math></span> <i>I, J</i> <math>\leftarrow</math> 0; <i>K</i> <math>\leftarrow</math> <math>\mathcal{K}</math>  <b>proc Enc.init</b>(<i>N</i>) <b>if</b> <i>N</i> <math>\notin</math> <math>\mathcal{N}</math> <b>then ret</b> <math>\perp</math> <i>I</i> <math>\leftarrow</math> <i>I</i> + 1; <i>S<sub>I</sub></i> <math>\leftarrow</math> <math>\mathcal{E}</math>.init(<i>K, N</i>) <b>ret</b> <i>I</i>  <b>proc Enc.next</b>(<i>i, A, M</i>) <b>if</b> <i>i</i> <math>\notin</math> [1..<i>I</i>] <b>or</b> <i>S<sub>i</sub></i> = <math>\perp</math> <b>then ret</b> <math>\perp</math> (<i>C, S<sub>i</sub></i>) <math>\leftarrow</math> <math>\mathcal{E}</math>.next(<i>S<sub>i</sub>, A, M</i>) <b>ret</b> <i>C</i>  <b>proc Enc.last</b>(<i>i, A, M</i>) <b>if</b> <i>i</i> <math>\notin</math> [1..<i>I</i>] <b>or</b> <i>S<sub>i</sub></i> = <math>\perp</math> <b>then ret</b> <math>\perp</math> <i>C</i> <math>\leftarrow</math> <math>\mathcal{E}</math>.last(<i>S<sub>i</sub>, A, M</i>) <i>S<sub>i</sub></i> <math>\leftarrow</math> <math>\perp</math>; <b>ret</b> <i>C</i>  <b>proc Dec.init</b>(<i>N</i>) <b>if</b> <i>N</i> <math>\notin</math> <math>\mathcal{N}</math> <b>then ret</b> <math>\perp</math> <i>J</i> <math>\leftarrow</math> <i>J</i> + 1; <i>S'<sub>J</sub></i> <math>\leftarrow</math> <math>\mathcal{D}</math>.init(<i>K, N</i>) <b>ret</b> <i>J</i>  <b>proc Dec.next</b>(<i>j, A, C</i>) <b>if</b> <i>j</i> <math>\notin</math> [1..<i>J</i>] <b>or</b> <i>S'<sub>j</sub></i> = <math>\perp</math> <b>then ret</b> <math>\perp</math> (<i>M, S'<sub>j</sub></i>) <math>\leftarrow</math> <math>\mathcal{D}</math>.next(<i>S'<sub>j</sub>, A, C</i>) <b>ret</b> <i>M</i>  <b>proc Dec.last</b>(<i>j, A, C</i>) <b>if</b> <i>j</i> <math>\notin</math> [1..<i>J</i>] <b>or</b> <i>S'<sub>j</sub></i> = <math>\perp</math> <b>then ret</b> <math>\perp</math> <i>M</i> <math>\leftarrow</math> <math>\mathcal{D}</math>.last(<i>S'<sub>j</sub>, A, C</i>) <i>S'<sub>j</sub></i> <math>\leftarrow</math> <math>\perp</math> <b>ret</b> <i>M</i>                 </pre>	<pre> <b>proc initialize</b> <span style="border: 1px solid black; padding: 2px;">Ideal2B<math>\Pi</math></span> <i>I, J</i> <math>\leftarrow</math> 0; <i>F</i> <math>\leftarrow</math> IdealOAE(<math>\tau</math>)  <b>proc Enc.init</b>(<i>N</i>) <b>if</b> <i>N</i> <math>\notin</math> <math>\mathcal{N}</math> <b>then ret</b> <math>\perp</math> <i>I</i> <math>\leftarrow</math> <i>I</i> + 1; <i>N<sub>I</sub></i> <math>\leftarrow</math> <i>N</i>; <i>A<sub>I</sub></i> <math>\leftarrow</math> <math>\Lambda</math>; <i>M<sub>I</sub></i> <math>\leftarrow</math> <math>\Lambda</math> <b>ret</b> <i>I</i>  <b>proc Enc.next</b>(<i>i, A, M</i>) <b>if</b> <i>i</i> <math>\notin</math> [1..<i>I</i>] <b>or</b> <i>M<sub>i</sub></i> = <math>\perp</math> <b>then ret</b> <math>\perp</math> <i>A<sub>i</sub></i> <math>\leftarrow</math> <i>A<sub>i</sub></i> <math>\parallel</math> <i>A</i>; <i>M<sub>i</sub></i> <math>\leftarrow</math> <i>M<sub>i</sub></i> <math>\parallel</math> <i>M</i>; <i>m</i> <math>\leftarrow</math>  <i>M<sub>i</sub></i>  <i>C</i> <math>\leftarrow</math> <i>F</i>(<i>N<sub>i</sub>, A<sub>i</sub>, M<sub>i</sub>, 0</i>); <b>ret</b> <i>C</i>[<i>m</i>]  <b>proc Enc.last</b>(<i>i, A, M</i>) <b>if</b> <i>i</i> <math>\notin</math> [1..<i>I</i>] <b>or</b> <i>M<sub>i</sub></i> = <math>\perp</math> <b>then ret</b> <math>\perp</math> <i>A<sub>i</sub></i> <math>\leftarrow</math> <i>A<sub>i</sub></i> <math>\parallel</math> <i>A</i>; <i>M<sub>i</sub></i> <math>\leftarrow</math> <i>M<sub>i</sub></i> <math>\parallel</math> <i>M</i>; <i>m</i> <math>\leftarrow</math>  <i>M<sub>i</sub></i>  <i>C</i> <math>\leftarrow</math> <i>F</i>(<i>N<sub>i</sub>, A<sub>i</sub>, M<sub>i</sub>, 1</i>); <i>M<sub>i</sub></i> <math>\leftarrow</math> <math>\perp</math>; <b>ret</b> <i>C</i>[<i>m</i>]  <b>proc Dec.init</b>(<i>N</i>) <b>if</b> <i>N</i> <math>\notin</math> <math>\mathcal{N}</math> <b>then ret</b> <math>\perp</math> <i>J</i> <math>\leftarrow</math> <i>J</i> + 1; <i>N'<sub>J</sub></i> <math>\leftarrow</math> <i>N</i>; <i>A'<sub>J</sub></i> <math>\leftarrow</math> <math>\Lambda</math>; <i>C<sub>J</sub></i> <math>\leftarrow</math> <math>\Lambda</math> <b>ret</b> <i>J</i>  <b>proc Dec.next</b>(<i>j, A, C</i>) <b>if</b> <i>j</i> <math>\notin</math> [1..<i>J</i>] <b>or</b> <i>C<sub>j</sub></i> = <math>\perp</math> <b>then ret</b> <math>\perp</math> <i>A'<sub>j</sub></i> <math>\leftarrow</math> <i>A<sub>j</sub></i> <math>\parallel</math> <i>A</i>; <i>C<sub>j</sub></i> <math>\leftarrow</math> <i>C<sub>j</sub></i> <math>\parallel</math> <i>C</i>; <i>m</i> <math>\leftarrow</math>  <i>C<sub>j</sub></i>  <b>if</b> <math>\exists M</math> s.t. <i>F</i>(<i>N'<sub>j</sub>, A'<sub>j</sub>, M, 0</i>) = <i>C<sub>j</sub></i> <b>then ret</b> <i>M</i>[<i>m</i>] <b>else</b> <i>C<sub>j</sub></i> <math>\leftarrow</math> <math>\perp</math>; <b>ret</b> <math>\perp</math>; <b>fi</b>  <b>proc Dec.last</b>(<i>j, A, C</i>) <b>if</b> <i>j</i> <math>\notin</math> [1..<i>J</i>] <b>or</b> <i>C<sub>j</sub></i> = <math>\perp</math> <b>then ret</b> <math>\perp</math> <i>A'<sub>j</sub></i> <math>\leftarrow</math> <i>A</i> <math>\parallel</math> <i>A</i>; <i>C<sub>j</sub></i> <math>\leftarrow</math> <i>C<sub>j</sub></i> <math>\parallel</math> <i>C</i>; <i>m</i> <math>\leftarrow</math>  <i>C<sub>j</sub></i>  <b>if</b> <math>\exists M</math> s.t. <i>F</i>(<i>N'<sub>j</sub>, A'<sub>j</sub>, M, 1</i>) = <i>C<sub>j</sub></i> <b>then</b> <i>C<sub>j</sub></i> <math>\leftarrow</math> <math>\perp</math>; <b>ret</b> <i>M</i>[<i>m</i>] <b>else</b> <i>C<sub>j</sub></i> <math>\leftarrow</math> <math>\perp</math>; <b>ret</b> <math>\perp</math> <b>fi</b>                 </pre>
---	---

**Fig. 5. OAE2b security.** The segmented-AE scheme  $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$  has nonce space  $\mathcal{N}$  and segment-expansion  $\tau$ .

and Dec.last are treated analogously, maintaining  $N'_i, A'_i, C_i$  values. Once again the simulation is perfect, so  $\text{Adv}_{\Pi}^{\text{OAE2a}}(\mathcal{B}) = \text{Adv}_{\Pi}^{\text{OAE2b}}(\mathcal{A})$ . But now there is a quadratic slowdown in running time: the argument lists can grow long, as can return values, only one component of which is used with each call.

While the OAE2a definition is more compact, the improved concision for the adversary's queries in the OAE2b definition ultimately make it preferable, particularly as this concision better models the real-world semantics, where an adversary might be able to incrementally grow a plaintext or ciphertext with the unwitting cooperation of some encrypting or decrypting party. We note that we could achieve greater concision still by introducing a shorthand that would

allow the adversary to grow a tree and not just a chain. But this would not seem to model anything meaningful in the real-world.

There are a couple of further reasons to favor OAE2b. One is that it more directly captures the possibility of “infinite” (non-terminating) plaintexts (an infinite “stream” of messages). This is simply the setting where `Enc.last` and `Dec.last` are never called. Second, the OAE2b definition makes it easier to define nonce-respecting adversaries for the OAE setting. Such adversaries may adaptively grow a plaintext based on a single nonce, but it may grow only *one* plaintext for any given nonce. Building on the OAE2a formulation this is awkward to say, but building on the OAE2b formulation, it is natural.

**THIRD OAE2 DEFINITION: OAE2c.** Let  $\Pi$  be a segmented-AE scheme with segment-expansion  $\tau$  and nonce-space  $\mathcal{N}$ . Our final formulation of OAE2 security uses a two-part definition, separately defining privacy and authenticity requirements. With games defined in Fig. 6, we let  $\mathbf{Adv}_{\Pi}^{\text{oe2-priv}}(\mathcal{A}) = \Pr[\mathcal{A}^{\mathbf{Real2C}_{\Pi}} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathbf{Rand2C}_{\Pi}} \Rightarrow 1]$ . Similarly, define  $\mathbf{Adv}_{\Pi}^{\text{oe2-auth}}(\mathcal{A}) = \Pr[\mathcal{A}^{\mathbf{Forge2C}_{\Pi}}]$ , meaning the probability that  $\mathcal{A}$  returns a value that, when provided as input to the procedure `finalize`, evaluates to `true`. Informally, OAE2c security for a scheme  $\Pi$  means that reasonable adversaries get small oae2-priv advantage *and* small oae2-auth advantage.

Definition OAE2c is simpler than prior games in the sense that, for privacy, no decryption oracles are provided and the reference experiment simply returns the right number of uniformly random bits. For the authenticity portion of the definition, forgeries are defined to allow any  $(N, \mathbf{A}, \mathbf{C})$  that the adversary does not trivially know to be valid, the adversary marking in  $\mathbf{C}$  has terminated ( $b = 1$ ) or not ( $b = 0$ ). Set  $\mathcal{Z}$  records the tuples that the trivially adversary knows by virtue of encryption queries.

The following propositions show that OAE2b and OAE2c are close, assuming that the segment-expansion  $\tau$  is fairly large. The proofs are in the full version [33].

**Proposition 1 (oe2c  $\Rightarrow$  oe2b).** *Let  $\Pi$  be a segmented-AE scheme with ciphertext expansion  $\tau$ . There are explicit given reductions  $R_1$  and  $R_2$  with the following property. For any adversary  $\mathcal{A}$ , adversaries  $\mathcal{B}_1 = R_1(\mathcal{A})$  and  $\mathcal{B}_2 = R_2(\mathcal{A})$  satisfy  $\mathbf{Adv}_{\Pi}^{\text{oe2b}}(\mathcal{A}) \leq \mathbf{Adv}_{\Pi}^{\text{oe2-priv}}(\mathcal{B}_1) + p \cdot \mathbf{Adv}_{\Pi}^{\text{oe2-auth}}(\mathcal{B}_2) + q^2/2^{\tau}$ , where  $p$  and  $q$  are the number of decryption chains and the number of queries of  $\mathcal{A}$ , respectively. For each  $i \in \{1, 2\}$ , adversary  $\mathcal{B}_i$  uses about the same running time as  $\mathcal{A}$ , and the length of its queries is also at most that of  $\mathcal{A}$ 's queries.*

**Proposition 2 (oe2b  $\Rightarrow$  oe2c).** *Let  $\Pi$  be a segmented-AE scheme with ciphertext expansion  $\tau$ . There are explicit given reductions  $R_1$  and  $R_2$  with the following property. For any adversaries  $\mathcal{A}_1$  and  $\mathcal{A}_2$ , adversaries  $\mathcal{B}_1 = R_1(\mathcal{A}_1)$  and  $\mathcal{B}_2 = R_2(\mathcal{A}_2)$  satisfy  $\mathbf{Adv}_{\Pi}^{\text{oe2-priv}}(\mathcal{A}_1) \leq \mathbf{Adv}_{\Pi}^{\text{oe2b}}(\mathcal{B}_1)$  and  $\mathbf{Adv}_{\Pi}^{\text{oe2-auth}}(\mathcal{A}_2) \leq \mathbf{Adv}_{\Pi}^{\text{oe2b}}(\mathcal{B}_2) + \ell/2^{\tau}$ , where  $\ell$  is the number of segments in  $\mathcal{A}_2$ 's output. For each  $i \in \{1, 2\}$ , adversary  $\mathcal{B}_i$  uses about the same running time as  $\mathcal{A}_i$ , and the length of its queries is at most that of  $\mathcal{A}_i$ 's queries.*

<pre> <b>proc initialize</b> <span style="border: 1px solid black; padding: 2px;">Real2C<sub>Π</sub></span> <span style="border: 1px solid black; padding: 2px; color: blue;">Forge2C<sub>Π</sub> ←</span>     I ← 0; K ← K     Z ← ∅  <b>proc</b> Enc.init(N) <b>if</b> N ∉ N <b>then ret</b> ⊥     I ← I + 1; S<sub>I</sub> ← E.init(K, N)     N<sub>I</sub> ← N; A<sub>I</sub> ← M<sub>I</sub> ← C<sub>I</sub> ← A; <b>ret</b> I  <b>proc</b> Enc.next(i, A, M) <b>if</b> i ∉ [1..I] <b>or</b> S<sub>i</sub> = ⊥ <b>then ret</b> ⊥     (C, S<sub>i</sub>) ← E.next(S<sub>i</sub>, A, M)     A<sub>i</sub> ← A<sub>i</sub> ∥ A; M<sub>i</sub> ← M<sub>i</sub> ∥ M; C<sub>i</sub> ← C<sub>i</sub> ∥ C     Z ← Z ∪ {(N<sub>i</sub>, A<sub>i</sub>, C<sub>i</sub>, 0)}; <b>ret</b> C  <b>proc</b> Enc.last(i, A, M) <b>if</b> i ∉ [1..I] <b>or</b> S<sub>i</sub> = ⊥ <b>then ret</b> ⊥     C ← E.last(S<sub>i</sub>, A, M); S<sub>i</sub> ← ⊥     A<sub>i</sub> ← A<sub>i</sub> ∥ A; M<sub>i</sub> ← M<sub>i</sub> ∥ M; C<sub>i</sub> ← C<sub>i</sub> ∥ C     Z ← Z ∪ {(N<sub>i</sub>, A<sub>i</sub>, C<sub>i</sub>, 1)}; <b>ret</b> C  <b>proc finalize</b> (N, A, C, b) ← <b>if</b>  A  ≠  C  <b>or</b>  A  = 0 <b>or</b> (N, A, C, b) ∈ Z ←     <b>then ret</b> false ←     S ← D.init(K, N); m ←  C  ←     <b>for</b> i ← 1 <b>to</b> m - b <b>do</b> ←         (M, S) ← D.next(S, A[i], C[i]) ←         <b>if</b> M = ⊥ <b>then ret</b> false ←     <b>if</b> b = 1 <b>and</b> D.last(S, A[m], C[m]) = ⊥ ←         <b>then ret</b> false ←     <b>ret</b> true ←         </pre>	<pre> <b>proc initialize</b> <span style="border: 1px solid black; padding: 2px;">Rand2C<sub>Π</sub></span>     I ← 0     E(x) ← undef for all x  <b>proc</b> Enc.init(N) <b>if</b> N ∉ N <b>then ret</b> ⊥     I ← I + 1     N<sub>I</sub> ← N; A<sub>i</sub> ← M<sub>i</sub> ← A     <b>ret</b> I  <b>proc</b> Enc.next(i, A, M) <b>if</b> i ∉ [1..I] <b>or</b> N<sub>i</sub> = ⊥ <b>then ret</b> ⊥     A<sub>i</sub> ← A<sub>i</sub> ∥ A; M<sub>i</sub> ← M<sub>i</sub> ∥ M     <b>if</b> E(N<sub>i</sub>, A<sub>i</sub>, M<sub>i</sub>, 0) = undef <b>then</b>         E(N<sub>i</sub>, A<sub>i</sub>, M<sub>i</sub>, 0) ← {0, 1}<sup> M +τ</sup>     C ← E(N<sub>i</sub>, A<sub>i</sub>, M<sub>i</sub>, 0)     <b>ret</b> C  <b>proc</b> Enc.last(i, A, M) <b>if</b> i ∉ [1..I] <b>or</b> N<sub>i</sub> = ⊥ <b>then ret</b> ⊥     A<sub>i</sub> ← A<sub>i</sub> ∥ A; M<sub>i</sub> ← M<sub>i</sub> ∥ M     <b>if</b> E(N<sub>i</sub>, A<sub>i</sub>, M<sub>i</sub>, 1) = undef <b>then</b>         E(N<sub>i</sub>, A<sub>i</sub>, M<sub>i</sub>, 1) ← {0, 1}<sup> M +τ</sup>     C ← E(N<sub>i</sub>, A<sub>i</sub>, M<sub>i</sub>, 1); N<sub>i</sub> ← ⊥     <b>ret</b> C         </pre>
---	---

**Fig. 6. OAE2c security.** Privacy and authenticity are separately defined, the first by comparing games **Real2C** and **Rand2C**, and the second using game **Forge2C**, which includes the **additional lines** indicated.

MULTIVALUED SEGMENT-EXPANSION. It is easy to extend the definitions of this section to schemes for which the segment-expansion varies according to segment position. In particular, one could use one expansion value,  $\sigma$ , for plaintext components other than the last, and a different expansion value,  $\tau$ , at the end. For such a  $(\sigma, \tau)$ -expanding scheme, distribution  $\text{IdealOAE}(\tau)$  would be adjusted to  $\text{IdealOAE}(\sigma, \tau)$  in the natural way.

The main reason for considering multivalued segment-expansion is to clarify how OAE2 security relates to prior notions in the literature. In particular, OAE2 resembles OAE1 where the segment-expansion is  $(0, \tau)$  and where all segments are required to have some fixed length  $n$ . Yet even then the definitions would be very different: the OAE2 version would be stronger, since an online decryption capability is not allowed to compromise OAE2 security, whereas the capability may compromise OAE1 security. It is easy to give a separating example [33].

Another potential reason to consider multivalued segment-expansion is as a way to save on bits; obviously one will use fewer total bits, over a sequence of two or more segments, if only the last is expanded. But we suspect that this benefit is rarely worth its cost. If segments are 1 KByte (which is fairly short) and tags are 128 bits (which is fairly long), the difference (in total number of needed bits) between authenticating every segment and authenticating only the last one will always be less than 2%. This seems a small price to pay to have each and every segment properly authenticated.

WHY VECTOR-VALUED AD? In modeling OAE it is unclear if one ought think of the AD as a fixed string that is known before the plaintext begins to arrive, or if, instead, one should think of the AD as vector-valued, its  $i$ th segment available when the  $i$ th segment of plaintext is. We adopted the second view (switching from the first at the urging of the Keyak team) for closer concordance with prior work [19] and for greater generality: a string-valued AD of  $A$  can be regarded as a vector-valued AD of  $\mathbf{A} = (A, \varepsilon, \varepsilon, \dots)$ . More philosophically, the two conceptions correspond to whether one thinks of breaking up a fixed plaintext  $M$  into a sequence of segments  $M_i$  or one regards the  $M_i$  values as more autonomous, each encrypted when available, each with its own associated context. With plaintexts and AD both vector-valued, one conceptually extends across time a channel that securely transmit pairs of strings, one component with privacy and both with authenticity. All that said, the authors are uncertain of the actual utility of vector-valued over string-valued AD.

## 6 Achieving OAE2

In the special case that each segmented-string has only one component, OAE2 degenerates to the notion of a *pseudorandom injection* (PRI) [55]. The notion is close to MRAE [55], with a gap  $q^2/2^{s+\tau} + q/2^\tau$  where  $q$  is the number of queries and  $s$  is the length of the shortest plaintext queried. Below we construct an OAE2-secure scheme *from* a PRI-secure scheme. The scheme could be SIV [55] if  $\tau$  is large, say  $\tau = 128$ , or AEZ scheme [34], for arbitrary  $\tau$ . We begin by recalling the PRI notion.

PSEUDORANDOM INJECTIONS. Let  $\Pi = (\mathbf{K}, \mathbf{E}, \mathbf{D})$  be a conventional AE scheme, meaning that (i) the key space  $\mathbf{K}$  is a nonempty set with an associated distribution, (ii)  $\mathbf{E}: \mathbf{K} \times \mathcal{N} \times \mathcal{A} \times \{0, 1\}^* \rightarrow \{0, 1\}^*$  is the encryption scheme, and (iii)  $\mathbf{D}: \mathbf{K} \times \mathcal{N} \times \mathcal{A} \times \{0, 1\}^* \rightarrow \{0, 1\}^* \cup \{\perp\}$  is the decryption scheme. Both  $\mathbf{E}$  and  $\mathbf{D}$  are deterministic, and decryption reverses encryption, meaning that for every  $N \in \mathcal{N}$ ,  $A \in \mathcal{A}$ ,  $M \in \{0, 1\}^*$ , and  $K \in \mathbf{K}$ , we have  $\mathbf{D}_K^{N,A}(\mathbf{E}_K^{N,A}(M)) = M$ . We insist there be a constant  $\tau$  associated to  $\Pi$ , its *ciphertext-expansion*, where  $|\mathbf{E}_K^{N,A}(M)| = |M| + \tau$  for all  $N \in \mathcal{N}$ ,  $A \in \mathcal{A}$ ,  $M \in \{0, 1\}^*$ ,  $K \in \mathbf{K}$ . Define  $\text{Adv}_\Pi^{\text{pri}}(\mathcal{A}) = \Pr[\mathcal{A}^{\text{RealPRI}}_\Pi \Rightarrow 1] - \Pr[\mathcal{A}^{\text{IdealPRI}}_\Pi \Rightarrow 1]$  using Fig. 7’s games.

ACHIEVING OAE2 SECURITY. Fix integers  $n \geq \tau \geq 0$ . For a string  $X \in \{0, 1\}^*$  and  $1 \leq i \leq j \leq |X|$ , let  $X[i, j]$  denote the substring of  $X$  from the  $i$ th bit to the  $j$ th bit (inclusive). Let  $\langle \cdot \rangle$  denote an encoding that maps a pair  $(A, d) \in$

<pre> <b>proc initialize</b> <math>K \leftarrow \mathbf{K}</math>  <b>proc Enc</b>(<math>N, A, M</math>) <b>if</b> <math>N \notin \mathcal{N}</math> <b>or</b> <math>A \notin \mathcal{A}</math> <b>then ret</b> <math>\perp</math> <b>ret</b> <math>\mathbf{E}(K, N, A, M)</math>  <b>proc Dec</b>(<math>N, A, C</math>) <b>if</b> <math>N \notin \mathcal{N}</math> <b>or</b> <math>A \notin \mathcal{A}</math> <b>then ret</b> <math>\perp</math> <b>ret</b> <math>\mathbf{D}(K, N, A, C)</math>                 </pre>	<div style="border: 1px solid black; display: inline-block; padding: 2px 5px;"><b>RealPRI<math>_{\Pi}</math></b></div>
<pre> <b>proc initialize</b> <b>for</b> <math>(N, A) \in \mathcal{N} \times \mathcal{A}</math> <b>do</b> <math>\rho_{N,A} \leftarrow \text{Inj}(\tau)</math>  <b>proc Enc</b>(<math>N, A, M</math>) <b>if</b> <math>N \notin \mathcal{N}</math> <b>or</b> <math>A \notin \mathcal{A}</math> <b>then ret</b> <math>\perp</math> <b>ret</b> <math>\rho_{N,A}(M)</math>  <b>proc Dec</b>(<math>N, A, C</math>) <b>if</b> <math>N \notin \mathcal{N}</math> <b>or</b> <math>A \notin \mathcal{A}</math> <b>then ret</b> <math>\perp</math> <b>ret</b> <math>\rho_{N,A}^{-1}(C)</math>                 </pre>	<div style="border: 1px solid black; display: inline-block; padding: 2px 5px;"><b>IdealPRI<math>_{\Pi}</math></b></div>

**Fig. 7. PRI security.** Defining security for an AE scheme  $\Pi = (\mathbf{K}, \mathbf{E}, \mathbf{D})$  with expansion  $\tau$ , nonce space  $\mathcal{N}$ , and AD space  $\mathcal{A}$ . Here  $\text{Inj}(\tau)$  is the set of all injective functions  $f: \{0, 1\}^* \rightarrow \{0, 1\}^*$  such that  $|f(x)| = |x| + \tau$  for all  $x \in \{0, 1\}^*$ . For each  $y \in \{0, 1\}^*$  let  $f^{-1}(y) = x$  if there's an  $x \in \{0, 1\}^*$  such that  $f(x) = y$ , and  $f^{-1}(y) = \perp$  otherwise.

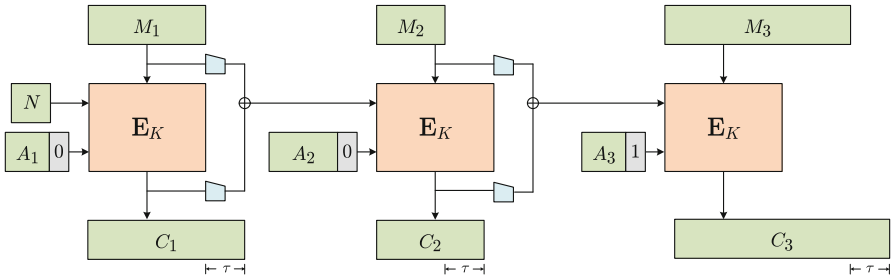
$\{0, 1\}^* \times \{0, 1, 2\}$  to a string  $\langle A, d \rangle \in \{0, 1\}^*$ . For example, one can represent  $d$  by a two-bit string, and append this to  $A$ . Let  $\Pi = (\mathbf{K}, \mathbf{E}, \mathbf{D})$  be a conventional AE scheme of ciphertext-expansion  $\tau$ , nonce space  $\{0, 1\}^n$ , and AD space  $\{0, 1\}^*$ . Figure 8 defines a segmented-AE scheme  $\mathbf{CHAIN}[\Pi, \langle \cdot \rangle, n] = (\mathcal{K}, \mathcal{E}, \mathcal{D})$  with segment expansion  $\tau$ , nonce space  $\{0, 1\}^n$ , AD space  $\{0, 1\}^*$ , and state space  $\mathbf{K} \times \{0, 1\}^n$ . The proof of the following theorem is in the full version [33].

**Theorem 1.** *Let  $\Pi$ ,  $\langle \cdot \rangle, n$ , and  $\mathbf{CHAIN}[\Pi, \langle \cdot \rangle, n]$  be as above. There is an explicit reduction  $R$  with the following property. For any adversary  $\mathcal{A}$ , adversary  $\mathcal{B} = R(\mathcal{A})$  satisfies  $\text{Adv}_{\mathbf{CHAIN}[\Pi, \langle \cdot \rangle, n]}^{\text{OAE2B}}(\mathcal{A}) \leq \text{Adv}_{\Pi}^{\text{PRI}}(\mathcal{B}) + 2q^2/2^n$  where  $q$  is the number of  $\mathcal{A}$ 's queries. Adversary  $\mathcal{B}$  uses about the same running time as  $\mathcal{A}$  and the total length of  $\mathcal{B}$ 's queries is that of  $\mathcal{A}$  plus at most  $5qn$  bits.*

DISCUSSION. In  $\mathcal{E}.\text{next}$  and  $\mathcal{D}.\text{next}$ , the state is computed via  $M[1, n] \oplus C[1, n]$ . One might instead xor the  $n$ -bit suffix of  $M$  and  $C$ ; this makes no difference. On the other hand, suppose one uses just  $C[1, n]$ , eliminating the xor with  $M[1, n]$ . Call this variant  $\mathbf{CHAIN1}[\Pi, \langle \cdot \rangle, n]$ . The method is insecure for small  $\tau$ . Here is an attack for the case  $\tau = 0$ . The adversary makes a single query  $(N, \mathbf{A}, \mathbf{C})$  to the decryption oracle, where  $N$  is arbitrary,  $\mathbf{A} = (\varepsilon, \varepsilon, \varepsilon)$  and  $\mathbf{C} = (0^n, 0^n, 0^n, 0^n)$ . Let the answer be  $\mathbf{M} = (M_1, M_2, M_3, M_4)$ . The adversary will output 1 only if  $M_2 = M_3$ . In the **Ideal2B** game the strings  $M_2$  and  $M_3$  are independent random strings. However, in game **Real2B** we always have  $M_2 = M_3 = \mathbf{D}_K(0^n, \langle \varepsilon, 0 \rangle, 0^n)$ . Hence the adversary can win with advantage  $1 - 2^{-n}$ . In contrast, for large  $\tau$ , scheme  $\mathbf{CHAIN1}[\Pi, \langle \cdot \rangle, n]$  is OAE2 secure.

To achieve OAE2 with multivalued segment-expansion, use an RAE-secure underlying scheme [34], a generalization of PRI that allows one to select an arbitrary ciphertext-expansion for each query. The construction is modified in the natural way.

<pre> <b>proc</b> <math>\mathcal{E}.\text{init}(K, N)</math>           <math>\mathcal{E}</math> algorithms <b>ret</b> <math>(K, N)</math>  <b>proc</b> <math>\mathcal{E}.\text{next}(S, A, M)</math> <math>(K, V) \leftarrow S</math>; <math>C \leftarrow \mathbf{E}_K(V, \langle A, 0 \rangle, M)</math> <b>if</b> <math> M  \geq n</math> <b>then</b> <math>V \leftarrow (C[1, n] \oplus M[1, n])</math> <b>else</b> <math>V \leftarrow (\mathbf{E}_K(V, \langle A, 2 \rangle, M \parallel 0^n))[1, n]</math> <b>ret</b> <math>(C, (K, V))</math>  <b>proc</b> <math>\mathcal{E}.\text{last}(S, A, M)</math> <math>(K, V) \leftarrow S</math>; <b>ret</b> <math>\mathbf{E}_K(V, \langle A, 1 \rangle, M)</math>                 </pre>	<pre> <b>proc</b> <math>\mathcal{D}.\text{init}(K, N)</math>           <math>\mathcal{D}</math> algorithms <b>ret</b> <math>(K, N)</math>  <b>proc</b> <math>\mathcal{D}.\text{next}(S, A, C)</math> <math>(K, V) \leftarrow S</math>; <math>M \leftarrow \mathbf{D}_K(V, \langle A, 0 \rangle, C)</math> <b>if</b> <math>M = \perp</math> <b>then</b> <b>ret</b> <math>(\perp, \perp)</math> <b>if</b> <math> M  \geq n</math> <b>then</b> <math>V \leftarrow C[1, n] \oplus M[1, n]</math> <b>else</b> <math>V \leftarrow (\mathbf{E}_K(V, \langle A, 2 \rangle, M \parallel 0^n))[1, n]</math> <b>ret</b> <math>(M, (K, V))</math>  <b>proc</b> <math>\mathcal{D}.\text{last}(S, A, C)</math> <math>(K, V) \leftarrow S</math>; <b>ret</b> <math>\mathbf{D}_K(V, \langle A, 1 \rangle, C)</math>                 </pre>
--	--



**Fig. 8. The CHAIN construction for OAE2.** **Top:** Encryption scheme  $\Pi = (\mathbf{K}, \mathbf{E}, \mathbf{D})$ , secure as a PRI with expansion  $\tau$ , is turned into a segmented-AE scheme  $\text{CHAIN}[\Pi, \langle \cdot \rangle, n] = (\mathcal{K}, \mathcal{E}, \mathcal{D})$  with  $\mathcal{K} = \mathbf{K}$ . **Bottom:** Illustration of the scheme. Each segment of  $(M_1, M_2, M_3)$  has at least  $n$  bits. Trapezoids represent truncation to  $n$  bits.

## 7 Escalating Claims, Diminishing Guarantees

A survey of the literature shows increasingly strong rhetoric surrounding nonce-reuse security of online schemes. We document this trend. In doing so we identify some of the notions (all quite weak, in our view) that have come to be regarded as nonce-reuse misuse-resistant.

SHIFTING LANGUAGE. The paper defining MRAE [55] never suggested that nonce-reuse was OK; it said that an MRAE scheme must do “as well as possible with whatever IV is provided” [55, p. 1]. Elaborating, the authors “aim for an AE scheme in which if the IV is a nonce then one achieves the usual notion for nonce-based AE; and if the IV does get repeated then authenticity remains and privacy is compromised only to the extent that [one reveals] if this plaintext is equal to a prior one, and even that . . . only if both the message and its header have been used with this particular IV” [55, p. 12–13].

The FFL paper indicates that the authors wish “to achieve both simultaneously: security against nonce-reusing adversaries . . . and support for on-line-encryption” [28, p. 197]. While the authors understood that they were weakening MRAE, they saw the weakening as relatively inconsequential: they say that their scheme, McOE, “because of being on-line, satisfies a *slightly weaker* security



<p><b>OAE1</b> Leaks equality of block-aligned prefixes, formalized by comparing <math>\mathcal{E}_K</math> with: a random <math>n</math>-bit-blocksize online permutation tweaked by the nonce, AD and plaintext; followed by a random <math>\tau</math>-bit function of the nonce, AD, and plaintext. Schemes1: COPA [9], Deoxys [37], Joltik [38], KIASU [39], Marble [32], McOE [28], SHELL [63], POET [1, 2], Prøst-COPA [20] Schemes2: ++AE [51]</p>
<p><b>OAE1a</b> Leaks equality of block-aligned prefixes, formalized by comparing <math>\mathcal{E}_K</math> with: a random <math>n</math>-bit-blocksize online <i>function</i> tweaked by the nonce, AD and plaintext; followed by a random <math>\tau</math>-bit function of the nonce, AD, and plaintext. Schemes1: APE [6], ELMd [25], ELMe [26], Prøst-APE [20]</p>
<p><b>OAE1b</b> Leaks equality of block-aligned prefixes, formalized by comparing <math>\mathcal{E}_K</math> with: a random <math>n</math>-bit-blocksize online <i>function</i> tweaked by the nonce and plaintext (but <i>not</i> the AD); followed by a random <math>\tau</math>-bit function of the nonce, AD, and plaintext. The relaxation enables a compliant scheme to process the plaintext before the AD is presented. However it also renders a compliant scheme vulnerable to CCA, CPSS, and NM attacks even if AD values are unique. Schemes1: COBRA [12]</p>
<p><b>OAE1c</b> Leaks equality of any blocks at the same position. E.g., if ciphertexts <math>C</math> and <math>C'</math> arise from 4-block plaintexts <math>P = A \parallel B \parallel C \parallel D</math> and <math>P' = E \parallel B \parallel F \parallel D</math> then <math>C_2 = C'_2</math> and <math>C_4 = C'_4</math>. Security is formalized by comparing <math>\mathcal{E}_K</math> with: a function from <math>n</math> bits to <math>n</math> bits tweaked by the nonce and an integer, the position; followed by a random tag. Schemes1: Minalpher [59]</p>
<p><b>OAE1d</b> Leaks equality of block-aligned prefixes and the XOR of the block directly following this prefix. E.g., if <math>C, C'</math> arise from 4-block plaintexts <math>P = A \parallel B \parallel C \parallel D</math> and <math>P' = A \parallel B \parallel E \parallel F</math> we always have <math>C_1 = C'_1</math>, <math>C_2 = C'_2</math>, and <math>C_3 \oplus C'_3 = C \oplus E</math>. Ciphertexts <math>C, C'</math> arising from 4-block plaintexts <math>P = A \parallel B \parallel C \parallel D</math> and <math>P' = E \parallel F \parallel G \parallel H</math> will have <math>C_1 \oplus C'_1 = A \oplus E</math>. Schemes2: Artemia [5] CBEAM [57], ICEPOLE [50], iFeed [66], Jambu [64], Keyak [18], MORUS [65], NORX [13], STRIBOB [58]</p>
<p><b>NAE1</b> Retains full security as long as all <math>(N, A)</math> pairs are unique among the encryption queries. If a pair repeats, all privacy is lost, but authenticity remains unchanged. Schemes1: CLOC [35], SILC [36]</p>
<p><b>NAE0</b> Retains full security as long as all <math>(N, A)</math> pairs are unique among the encryption queries. If a pair repeats, all security is forfeit. Schemes1: NORX [13], TriviacK [24] Schemes2: OTR [47]</p>

**Fig. 9. A menagerie of OAE notions and schemes.** All of the schemes are CAE-SAR submissions except ElmE and McOE. Schemes1 lists proposals that claim some flavor of nonce-reuse misuse resistance. Schemes2 lists proposals that didn't, yet are or were marked as such in the AE Zoo [14] or AFL survey [4].

definition against nonce-reusing adversaries” [28, p. 198] (emphasis ours). The paper did not investigate the definitional consequences of this weakening.

An early follow-on to FFL, the COPA paper, asserts that OAE1 schemes are distinguished by “not relying on the non-reuse of a nonce” [9, p. 438]. Andreeva *et al.* classify AE schemes according to the type of initialization vector (IV) one needs: either *random*, *nonce*, or *arbitrary*. A scheme satisfying OAE1 is understood to be an arbitrary-IV scheme, where “no restrictions on the IV are imposed, thus an adversary may choose any IV for encryption” [7, p. 9]. The

authors add that “Often a deterministic AE scheme does not even have an IV input” [7, p. 9]. The linguistic progression reaches its logical conclusion in the rebranding of OAE1-secure schemes as *nonce-free*, as seen, for example, in recent talks of Guo [32, slide 2] and Lauridsen [21, Slides 4, 6].

We have thus seen a transformation in language, played out over eight years, taking us from a strong definition (MRAE) pitched as trying to capture the best one can do when a nonce gets reused to a comparatively weak definition (OAE1) nowadays pitched as being so strong so as to render nonces superfluous. Meanwhile, the best-one-can-do positioning of MRAE was mirrored in the online setting. The COPA authors indicate that their mode achieves “the maximum attainable for single pass schemes” [8, p. 7]. Identical language is found in the COBRA submission [11, p. 7]. In our view, such claims are wrong; there would seem to be a significant gap between OAE1 and OAE2 security.

**WEAKER NOTIONS.** Concurrent with the rhetoric for what OAE1 delivers being ratcheted up, weakened variants of OAE1 have proliferated. We document this trend in Fig. 9, which introduces a variety of OAE notions. They are all weaker than OAE1 except for OAE1a; by standard arguments, OAE1 and OAE1a are quantitatively close if the blocksize is reasonably large. In this race to the bottom, it may seem as though the scheme comes first and whatever properties it provides is branded as *some* form misuse resistance.

The number of different OAE definitions, and their meanings, has never been clear. The evolution of what’s been indicated in the Nonce-MR column of the AE Zoo [14] illustrates the struggle of researchers trying to accurately summarize the extent of nonce-reuse misuse-resistance for tens of AE schemes. Our own attempt at sorting this out, Fig. 9, is not definitive. We do not formalize the notions in this table except for OAE1. (Some of the definitions are obvious, some are not.) The table is based on both author assertions (Schemes1) and assertions of others (Schemes2). The OAE1x notions only consider security for messages that are blocksize multiples.

**Acknowledgments.** The authors appreciate the excellent comments received from the Keyak/Ketje team: Joan Daemen, Guido Bertoni, Michaël Peeters, Gilles Van Assche, and Ronny Van Keer. Their feedback called our attention to the duplexing-the-sponge paper [19] and led to our decision to generalize to vector-valued AD and to remove the key  $K$  from .next and .last calls. We appreciate further comments and corrections from Farzaneh Abed, Nasour Bagheri, Dan Bernstein, Danilo Gligoroski, Stefan Lucks, Samuel Neves, and Kenny Paterson.

Much of the work on this paper was done while Phil Rogaway was visiting Ueli Maurer’s group at ETH Zürich. Many thanks to Ueli for hosting that sabbatical. Rogaway was also supported by NSF grants CNS-1228828 and CNS-1314885. Reyhanitabar and Vizár were partially supported by Microsoft Research under the Swiss Joint Research Centre MRL Contract No. 2014-006 (DP1061305).

## References

1. Abed, F., Fluhrer, S., Foley, J., Forler, C., List, E., Lucks, S., McGrew, D., Wenzel, J.: The POET Family of On-Line Authenticated Encryption Schemes (Version 1.01). CAESAR submission (2014)
2. Abed, F., Fluhrer, S., Forler, C., List, E., Lucks, S., McGrew, D., Wenzel, J.: Pipelineable on-line encryption. In: Cid, C., Rechberger, C. (eds.) FSE 2014. LNCS, vol. 8540, pp. 205–223. Springer, Heidelberg (2015)
3. Abed, F., Forler, C., List, E., Lucks, S., Wenzel, J.: Don't Panic! The Cryptographer's Guide to Robust (On-line) Encryption: Draft, 11 March 2015
4. Abed, F., Forler, C., Lucks, S.: General Overview of the First-Round CAESAR Candidates for Authenticated Encryption. Cryptology ePrint report 2014/792 (2014)
5. Alizadeh, J., Aref, M. R., Bagheri, N.: Artemia v1. CAESAR submission (2014)
6. Andreeva, E., Bilgin, B., Bogdanov, A., Luykx, A., Mennink, B., Mouha, N., Yasuda, K.: APE: authenticated permutation-based encryption for lightweight cryptography. In: Cid, C., Rechberger, C. (eds.) FSE 2014. LNCS, vol. 8540, pp. 168–186. Springer, Heidelberg (2015)
7. Andreeva, E., Bogdanov, A., Luykx, A., Mennink, B., Mouha, N., Yasuda, K.: How to securely release unverified plaintext in authenticated encryption. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8873, pp. 105–125. Springer, Heidelberg (2014)
8. Andreeva, E., Bogdanov, A., Luykx, A., Mennink, B., Tischhauser, E., Yasuda, K.: AES-COPA v. 1. CAESAR submission (2014)
9. Andreeva, E., Bogdanov, A., Luykx, A., Mennink, B., Tischhauser, E., Yasuda, K.: Parallelizable and authenticated online ciphers. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part I. LNCS, vol. 8269, pp. 424–443. Springer, Heidelberg (2013)
10. Andreeva, E., Bogdanov, A., Luykx, A., Mennink, B., Tischhauser, E., Yasuda, K.: Parallelizable (Authenticated) Online Ciphers. DIAC presentation (2013)
11. Andreeva, E., Luykx, A., Mennink, B., Yasuda, K.: AES-COBRA v1. CAESAR submission (2014)
12. Andreeva, E., Luykx, A., Mennink, B., Yasuda, K.: COBRA: a parallelizable authenticated online cipher without block cipher inverse. In: Cid, C., Rechberger, C. (eds.) FSE 2014. LNCS, vol. 8540, pp. 187–203. Springer, Heidelberg (2015)
13. Aumasson, J.P., Jovanovic, P., Neves, S.: NORX v1. CAESAR submission (2014)
14. Authenticated Encryption Zoo. <https://aezoo.compute.dtu.dk>
15. Bellare, M., Boldyreva, A., Knudsen, L.R., Namprempre, C.: Online ciphers and the Hash-CBC construction. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 292–309. Springer, Heidelberg (2001)
16. Bellare, M., Rogaway, P.: Encode-then-encipher encryption: how to exploit nonces or redundancy in plaintexts for efficient cryptography. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 317–330. Springer, Heidelberg (2000)
17. Bernstein, D.: Cryptographic competitions: CAESAR. <http://competitions.cr.yp.to>
18. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G., Van Keer, R.: CAESAR submission: Keyak v1. CAESAR submission (2014)
19. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Duplexing the sponge: single-pass authenticated encryption and other applications. In: Miri, A., Vaudenay, S. (eds.) SAC 2011. LNCS, vol. 7118, pp. 320–337. Springer, Heidelberg (2012)

20. Kavun, E.B., Lauridsen, M., Leander, G., Rechberger, C., Schwabe, P., Yalçın, T.: Prøst v1.1. CAESAR submission (2014)
21. Bogdanov, A., Lauridsen, M., Tischhauser, E.: AES-Based Authenticated Encryption Modes in Parallel High-Performance Software. DIAC presentation (2014)
22. Boldyreva, A., Degabriele, J.P., Paterson, K.G., Stam, M.: Security of symmetric encryption in the presence of ciphertext fragmentation. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 682–699. Springer, Heidelberg (2012)
23. Boldyreva, A., Taesombut, N.: Online encryption schemes: new security notions and constructions. In: Okamoto, T. (ed.) CT-RSA 2004. LNCS, vol. 2964, pp. 1–14. Springer, Heidelberg (2004)
24. Chakraborti, A., Nandi, M.: Trivia-ck-v1. CAESAR submission. (2014)
25. Datta, N., Nandi, M.: ELMd v1.0. CAESAR submission. (2014)
26. Datta, N., Nandi, M.: ELMe: a misuse resistant parallel authenticated encryption. In: Susilo, W., Mu, Y. (eds.) ACISP 2014. LNCS, vol. 8544, pp. 306–321. Springer, Heidelberg (2014)
27. Duong, T., Rizzo, J.: Here Come The  $\oplus$  Ninjas. Manuscript (2011)
28. Fleischmann, E., Forler, C., Lucks, S.: McOE: A family of almost foolproof on-line authenticated encryption schemes. In: Canteaut, A. (ed.) FSE 2012. LNCS, vol. 7549, pp. 196–215. Springer, Heidelberg (2012)
29. Fleischmann, E., Forler, C., Lucks, S., Wenzel, J.: McOE: A Foolproof On-line Authenticated Encryption Scheme. Cryptology ePrint report 2011/644 (2013)
30. Fouque, P.-A., Joux, A., Martinet, G., Valette, F.: Authenticated on-line encryption. In: Matsui, M., Zuccherato, R.J. (eds.) SAC 2003. LNCS, vol. 3006. Springer, Heidelberg (2004)
31. Fouque, P.-A., Martinet, G., Poupard, G.: Practical symmetric on-line encryption. In: Johansson, T. (ed.) FSE 2003. LNCS, vol. 2887, pp. 362–375. Springer, Heidelberg (2003)
32. Guo, J.: Marble Specification Version 1.0. CAESAR submission (2014). Also DIAC presentation (2014)
33. Hoang, V.T., Reyhanitabar, R., Rogaway, P., Vizár, D: Online Authenticated-Encryption and its Nonce-Reuse Misuse-Resistance. Cryptology ePrint Archive, Report 2015/189 (2015)
34. Hoang, V.T., Krovetz, T., Rogaway, P.: Robust authenticated-encryption AEZ and the problem that it solves. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 15–44. Springer, Heidelberg (2015)
35. Iwata, I., Minematsu, K., Guo, J., Morioka, S.: CLOC: Compact Low-Overhead CFB. CAESAR submission. (2014)
36. Iwata, T., Minematsu, K., Guo, J., Morioka, S., Kobayashi, E.: SILC: Simple Lightweight CFB. CAESAR submission. (2014)
37. Jean, J., Nikolić, I., Peyrin, T.: Deoxys v1. CAESAR submission. (2014)
38. Jean, J., Nikolić, I., Peyrin, T.: Joltik v1. CAESAR submission. (2014)
39. Jean, J., Nikolić, I., Peyrin, T.: KIASU v1. CAESAR submission. (2014)
40. Joux, A., Martinet, G., Valette, F.: Blockwise-adaptive attackers: revisiting the (In)security of some provably secure encryption modes: CBC, GEM, IACBC. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 17–30. Springer, Heidelberg (2002)
41. Katz, J., Yung, M.: Unforgeable encryption and chosen ciphertext secure modes of operation. In: Schneier, B. (ed.) FSE 2000. LNCS, vol. 1978, pp. 284–299. Springer, Heidelberg (2001)

42. Krovetz, T., Rogaway, P.: The OCB Authenticated-Encryption Algorithm. RFC 7253. Internet Research Task Force (IRTF) and Crypto Forum Research Group (CFRG) (2014)
43. Liskov, M., Rivest, R., Wagner, D.: Tweakable Block Ciphers. *J. Cryptology* **24**(3), 588–614 (2011)
44. Lucks, S.: Personal communication (2014)
45. Abed, F., Fluhrer, S., Forler, C., List, E., Lucks, S., McGrew, D., Wenzel, J.: Pipelineable on-line encryption. In: Cid, C., Rechberger, C. (eds.) FSE 2014. LNCS, vol. 8540, pp. 205–223. Springer, Heidelberg (2015)
46. Miaw, W.: Netflix/mssl (2014). <https://github.com/Netflix/mssl/wiki>
47. Minematsu, K.: AES-OTR v1. CAESAR submission (2014)
48. Minematsu, K.: Parallelizable Rate-1 Authenticated Encryption from Pseudorandom Functions. *Cryptology ePrint Archive*, Report 2013/628 (2013)
49. Möller, B.: Security of CBC Ciphersuites in SSL/TLS: Problems and Countermeasures. <http://web.archive.org/web/20120630143111/http://www.openssl.org/~bodo/tls-cbc.txt>
50. Morawiecki, P., Gaj, K., Homsirikamol, E., Matusiewicz, K., Pieprzyk, J., Rogawski, M., Srebrny, M., Wójcik, M.: ICEPOLE v1. CAESAR submission (2014)
51. Recacha, F.: ++AE v1.0. CAESAR submission (2014)
52. Rogaway, P.: Authenticated-Encryption with Associated-Data. In: ACM CCS 2002, pp. 98–107. ACM Press (2002)
53. Rogaway, P.: Problems with Proposed IP Cryptography. Manuscript (1995)
54. Rogaway, P., Bellare, M., Black, J., Krovetz, T.: OCB: A block-cipher mode of operation for efficient authenticated encryption. In: ACM CCS 2001, pp. 196–205. ACM Press (2001)
55. 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)
56. Rogaway, P., Zhang, H.: Online ciphers from tweakable blockciphers. In: Kiayias, A. (ed.) CT-RSA 2011. LNCS, vol. 6558, pp. 237–249. Springer, Heidelberg (2011)
57. Saarinen, M.-J.O.: The CBEAMr1 Authenticated Encryption Algorithm. CAESAR submission (2014)
58. Saarinen, M.-J.O.: The STRIBOBr 1 Authenticated Encryption Algorithm. CAESAR submission (2014)
59. Sasaki, Y., Todo, Y., Aoki, K., Naito, Y., Sugawara, T., Murakami, Y., Matsui, M., Hirose, S.: Minalpher v1. CAESAR submission (2014)
60. Tousef, S.: Streaming API to Authenticated Encryption. *Cryptography Stack Exchange*, 16 January 2013. <http://crypto.stackexchange.com/questions/6008>
61. Tsang, P., Solomakhin, R., Smith, S.: Authenticated Streamwise On-line Encryption. Dartmouth Computer Science Technical report TR2009-640 (2009)
62. Vaudenay, S.: Security flaws induced by CBC padding - applications to SSL, IPSEC, WTLS. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 534–545. Springer, Heidelberg (2002)
63. Wang, L.: SHELL v1. CAESAR submission (2014)
64. Wu, H., Huang, T.: JAMBU Lightweight Authenticated Encryption Mode and AES-JAMBU (v1). CAESAR submission (2014)
65. Wu, H., Huang, T.: The Authenticated Cipher MORUS (v1). CAESAR submission (2014)
66. Zhang, L, Wu, W., Sui, H., Wang, P.: iFeed[AES] v1. CAESAR submission (2014)

# Relational Hash: Probabilistic Hash for Verifying Relations, Secure Against Forgery and More

Avradip Mandal and Arnab Roy<sup>(✉)</sup>

Fujitsu Laboratories of America, Sunnyvale, CA, USA  
{amandal,aroy}@us.fujitsu.com

**Abstract.** Traditional cryptographic hash functions allow one to easily check whether the original plaintexts are equal or not, given a pair of hash values. Probabilistic hash functions extend this concept where given a probabilistic hash of a value and the value itself, one can efficiently check whether the hash corresponds to the given value. However, given distinct probabilistic hashes of the same value it is not possible to check whether they correspond to the same value. In this work we introduce a new cryptographic primitive called *Relational Hash* using which, given a pair of (relational) hash values, one can determine whether the original plaintexts were related or not. We formalize various natural security notions for the Relational Hash primitive - one-wayness, twin one-wayness, unforgeability and oracle simulatability.

We develop a Relational Hash scheme for discovering linear relations among bit-vectors (elements of  $\mathbb{F}_2^n$ ) and  $\mathbb{F}_p$ -vectors. Using the linear Relational Hash schemes we develop Relational Hashes for detecting proximity in terms of hamming distance. The proximity Relational Hashing schemes can be adapted to a privacy preserving biometric identification scheme, as well as a privacy preserving biometric authentication scheme secure against passive adversaries.

**Keywords:** Probabilistic hash functions · Functional encryption · Biometric authentication

## 1 Introduction

Traditional cryptographic hash functions, like MD-5 and SHA-3, enable checking for equality while hiding the plaintexts. Since these are deterministic functions, this just involves checking if the hashes are identical. The notion of probabilistic hash functions was developed in [Can97, CMR98]. In this setting, the computation of hashes is randomized and thus no two independently generated hashes of the same plaintext look same. However, given the plaintext and a hash, it can be checked efficiently if the hash corresponds to the plaintext. Probabilistic hashes can provably enable strong privacy guarantees in standard model, like oracle simulatability, which deterministic hash functions cannot provide. Oracle simulatability captures the notion that a hash reveals nothing about the value except enabling equality checking. This typically has come at the price

of efficiency. In addition, the property of compression, which is desirable for deterministic hash functions, is no longer at the forefront.

However, probabilistic hashes suffer from the drawback that for verification of equality the plaintext has to be provided in the clear, which deterministic hashes do not require. Probabilistic hashes do not allow checking whether the plaintexts are equal, given two distinct hash values. This drawback can preclude use of probabilistic hashes in certain scenarios where it is desirable to hide the plaintext from the verifier as well. For example, consider a scenario where password equality is to be checked by a server. If the server uses deterministic hashes, then only the hash of the password could be transmitted to the server. However, with probabilistic hashes, the actual password has to be sent to the server for verification<sup>1</sup>. Therefore question arises whether we can build probabilistic hashes which allow verification given two distinct hashes of the plaintexts.

So suppose we had a probabilistic hash function  $ph$  which allows efficient checking of equality of plaintexts  $x_1$  and  $x_2$ , given  $ph(x_1, r_1)$  and  $ph(x_2, r_2)$ , where the  $r_i$ 's are randomnesses used for hashing. Now we run into a different problem. The existence of such a functionality implies that a secrecy property called 2-value perfect one-wayness (2-POW) [CMR98] would no longer hold. This property states that the distribution of two probabilistic hashes of the same value is computationally indistinguishable from the distribution of probabilistic hashes of two independent values. The property trivially breaks down if we have an efficient mechanism for checking if two hashes correspond to the same plaintext. In addition to being a strong security notion, this property also implies oracle simulatability [CMR98]. So now the question is:

How do we develop probabilistic hashes which enable equality checking just given hashes but at the same time preserve 2-value perfect one-wayness?

*Our Contributions.* We propose a cryptographic primitive called *Relational Hash* which attempts to model the question above. One of the key ideas is to have distinct, but related, hashing systems for the individual co-ordinates, i.e., have two probabilistic hash functions  $ph_1$  and  $ph_2$  and enable checking of  $x_1 \stackrel{?}{=} x_2$ , given  $ph_1(x_1, r_1)$  and  $ph_2(x_2, r_2)$ . Having two hashing systems leaves open the possibility that they can individually be 2-POW. Extending equality, we define *Relational Hash* with respect to a relation  $R$ , such that given two hashes  $ph_1(x_1, r_1)$  and  $ph_2(x_2, r_2)$ , we can efficiently determine whether  $R(x_1, x_2)$  holds. It may also be desirable to compute ternary relations  $R'$  on  $x_1, x_2$  and a third plaintext parameter  $z$ , so that given  $ph_1(x_1, r_1), ph_2(x_2, r_2)$  and  $z$ , we can efficiently determine whether  $R'(x_1, x_2, z)$  holds. For any Relational Hash primitive, we formalize a few natural and desirable security properties, namely one-wayness, unforgeability, twin one-wayness and oracle simulatability. The notion of oracle simulatability was introduced in [Can97, CMR98] for the equality relation. Here we extend this concept for arbitrary relations.

<sup>1</sup> We need additional protocol steps to ensure security against replay attacks and so on. However, for now, we focus on the core property of the hashes themselves.

For the equality relation, there is a simple construction which extends Canetti’s scheme in [Can97]. While the [Can97] probabilistic hash on a plaintext  $m$  and randomness  $r$  is  $(\mathbf{g}^r, \mathbf{g}^{rm})$ , one can consider bilinear groups  $\mathbb{G}_1$  and  $\mathbb{G}_2$  with a pairing  $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$  and define  $ph_1(x_1, r_1) := (\mathbf{g}^{r_1}, \mathbf{g}^{r_1 x_1})$  and  $ph_2(x_2, r_2) := (\mathbf{h}^{r_2}, \mathbf{h}^{r_2 x_2})$  with  $\mathbf{g} \in \mathbb{G}_1$  and  $\mathbf{h} \in \mathbb{G}_2$ . Plaintext equality of two hashes  $(c_1, c_2)$  and  $(d_1, d_2)$  of different types can be done as:  $e(c_1, d_2) \stackrel{?}{=} e(c_2, d_1)$ . We do not develop this construction formally in the body of the paper, additionally relegating some proof sketches to the full version [MR14]<sup>2</sup>.

For hamming proximity relations among vectors, especially low characteristic ones, the constructions turn out to be far more sophisticated and form the main thrust of our paper. Towards that end, we first develop a construction for a linear Relational Hash scheme. In our scheme, for any  $x, y, z \in \mathbb{F}_2^n$ , given just the hashes of  $x$  and  $y$  and the plaintext  $z$ , it is possible to verify whether  $x + y \stackrel{?}{=} z$ . A linear Relational Hash scheme is also trivially an equality Relational Hash scheme, by taking  $z$  to be all 0’s. We also extend our construction to verify linear relations over  $\mathbb{F}_p^n$ . We show that our linear Relational Hash constructions satisfy all four security notions: one-wayness, unforgeability, twin one-wayness and oracle simulatability. Next we show that using a linear Relational Hash and error correcting codes it is possible to build Relational Hashing schemes which can verify proximity relations and enjoy one-wayness, unforgeability and a stronger version of twin one-wayness. It remains open to build a proximity Relational Hash scheme which is oracle simulation secure.

*Application.* A motivating application of the proximity relation hash primitive is a privacy preserving biometric identification scheme. Consider a scenario where there is a database of fingerprints of known criminals. The database should not reveal the actual fingerprints, even internally. An investigative officer might want to check, whether a candidate fingerprint digest matches with the database. Using a Relational Hash scheme for proximity relation, one can build a biometric identification scheme which guarantees complete template privacy (to the server, as well as to the investigating officer). While storing the fingerprints in the database, hashes of type 1 are used. On the other hand, the officer gets access to type 2 hash of the fingerprint template. The Relational Hash scheme will guarantee that, with access to a relational secret key the server can only verify whether the original templates are close to each other or not. To construct authentication schemes, rather than identification schemes, additional protocol layers are needed to address replay attacks and so on. Merely providing a type 2 hash of the challenge biometric template does not suffice as that can easily be replayed. We leave open the construction of such protocols building on the Relational Hash primitive. However, we show that for the case of a passive adversary attempting to recover the biometric template, a Relational Hash can be seen as a biometric authentication mechanism (Sect. 4).

<sup>2</sup> We thank Mehdi Tibouchi for observing this example.



*Relation to Fuzzy Extractor/Secure Sketch Based Schemes.* Existing biometric authentication schemes, e.g. fuzzy vault [JS02], fuzzy commitment [JW99] and secure sketch [DRS04, DS05] based schemes guarantee template privacy only during the registration phase. Boyen solved this issue in [Boy04], by constructing a “Zero Storage remote biometric authentication scheme”, which provides complete template privacy. Boyen’s construction only assumes that the biometric template comes from a high entropy distribution. Compared to that, we only achieve a passive adversary secure biometric authentication scheme assuming uniform distribution of biometric templates. On the positive side, our biometric authentication scheme is much simpler, in particular during authentication the client generates the authentication token on its own, without requiring any intervention from the server. Moreover, for our primary application - the non-interactive biometric identification mechanism, the advantage becomes more apparent. It is not readily clear whether one can build such identification mechanism based on fuzzy extractors/secure sketches.

*Relation to Multi-input Functional Encryption (MIFE).* Goldwasser et al. proposed the concept of MIFE in [GGG+14], which is a functional encryption which enables the computation of  $f(x_1, x_2, \dots, x_n)$  given the encryptions of  $x_1, x_2, \dots, x_n$ . The paper [GGG+14] is a merge of two independent and concurrent works [GGJS13, GKL+13]. While a Relational Hash scheme for a relation  $R$  can be considered an MIFE for evaluating the relation  $R$ , there are several important differences between the MIFE work of [GGG+14] and Relational Hash. We only consider the fully public key model where encryption keys for all the co-ordinates are given to the adversary.

We first remark that an indistinguishability based functional encryption security definition (FE-IND) for the equality relation is a rather trivial notion. The FE-IND notion asks the adversary to query two sets of  $n$ -tuples, and the challenger randomly selects which set to encrypt. We observe that even a standard CPA secure public-key encryption scheme satisfies this notion, where the functional key is simply the secret key for decryption. The FE-IND security notion is satisfied for equality because the restriction on the adversary’s queries forces it to choose equal sets of messages to the challenger. So in the end the adversary has information theoretically no clue about which of the messages was chosen for encryption by the challenger. In a Relational Hash scheme, even when given the relational key, the encryption of the plaintexts is required to be at least one-way secure. No such guarantee is provided by the standard CPA scheme, since giving the full decryption key fully exposes the plaintext to the functional key recipient.

Thus we have to resort to the simulation based security notion (FE-SIM) for any meaningful assurance of security. The only possibility result in the fully public key setting is given by [GKL+13], who give a construction of FE-SIM secure encryption scheme for a class of functionalities they call “learnable” functions. They also prove that if an FE-SIM secure scheme exists for a class of functionalities, then this class must be learnable. Briefly, a 2-ary function  $f(., .)$  is learnable if, given a description of  $f$  and oracle access to  $f(x, .)$ , one can output the description of a function that is indistinguishable from  $f_x(., .)$ , which is

the restriction of  $f$  on fixing the first input to  $x$ . This has to hold true with high probability even if the distinguisher is given  $x$ . One can immediately see that equality is not a learnable function. When  $x$  comes from high min-entropy distribution, it is not possible to learn the value of  $x$  efficiently by querying  $f(x, \cdot)$  on various inputs. A distinguisher can immediately thwart any such ‘learnt’ function by simply testing it on  $x$ .

Thus these work(s) effectively show that there is no FE-SIM secure functional encryption scheme for the function testing equality. How does our construction get around this impossibility? The reason is that the security properties that we consider: one-wayness and unforgeability do not imply FE-SIM. The property closest to FE-SIM is oracle simulatability, but it differs from FE-SIM in that the adversary does not choose the messages to be encrypted, rather they are sampled from a distribution and only their encryption is given to the adversary.

*Relation to Property Preserving (Tagged) Encryption (PPE).* PPE [PR12] is a special case of MIFE in the symmetric key setting. PPE offers IND based security guarantees, where attacker queries are constrained such that the preserved property values cannot be trivially used for distinguishing purposes. Moreover, PPE involves a secret key, whereas for Relational Hashes all the keys are public. For our public key case, the trivial construction which makes the functional key the same as the decryption key, is IND secure and does not provide any meaningful security guarantee. On the other hand, for the symmetric key PPE schemes, chosen message security is non-trivial.

*Relation to Perceptual Image Hashing (PIH).* PIH [KVM04] is a related technique which aims to construct hash of images invariant under geometric transformations which preserve perceptual similarity. There are several differences, most importantly: (1) the primary objective of PIH is the detection of similar inputs, however privacy of the inputs may not be preserved, (2) generating hashes requires a secret key, and (3) while for PIH the hashes are required to be equal for similar images, we require that the hashes are randomized and a verification algorithm is given which uses a key to perform the relation check.

*Organization of the Paper.* In Sect. 2, we formally define the notion of Relational Hash and its desired security properties. In Sect. 3, we construct a Relational Hash for linearity over  $\mathbb{F}_2^n$ , with extension to  $\mathbb{F}_p^n$ . In Sect. 4, we show how to construct a proximity (in terms of hamming distance) Relational Hash using a linear Relational Hash and a linear error correcting code. In Sect. 5, we describe relations among notions of security for constructing Relational Hashes for various relations. Standard hardness assumptions are summarized in Appendix A. We defer the proof of unproven theorems in this paper to the full version [MR14].

*Notations.* We denote a sequence  $x_j, \dots, x_k$  as  $\langle x_i \rangle_{i=j}^k$ . We treat  $\mathbb{F}_p^n$  as an  $\mathbb{F}_p$  vector space and write  $x \in \mathbb{F}_p^n$  also as  $\langle x_i \rangle_{i=1}^n$ . Group elements are written in bold font:  $\mathbf{g}$ ,  $\mathbf{f}$ . The security parameter is denoted as  $\lambda$ .

## 2 Relational Hash

The concept of *Relational Hash* is an extension of regular probabilistic hash functions. In this work, we only consider 3-tuple relations. Suppose  $R \subseteq X \times Y \times Z$  be a 3-tuple relation, that we are interested in. We abuse the notation a bit, and often use the equivalent functional notation  $R : X \times Y \times Z \rightarrow \{0, 1\}$ . The Relational Hash for the relation  $R$ , will specify two hash algorithms  $\text{HASH}_1$  and  $\text{HASH}_2$  which will output the hash values  $\text{HASH}_1(x)$  and  $\text{HASH}_2(y)$  for any  $x \in X$  and  $y \in Y$ . Any Relational Hash must also specify a verification algorithm  $\text{VERIFY}$ , which will take  $\text{HASH}_1(x)$ ,  $\text{HASH}_2(y)$  and any  $z \in Z$  as input and output  $R(x, y, z)$ . Formally, we define the notion of Relational Hash as follows.

**Definition 1 (Relational Hash).** *Let  $\{R_\lambda\}_{\lambda \in \mathbb{N}}$  be a relation ensemble defined over set ensembles  $\{X_\lambda\}_{\lambda \in \mathbb{N}}$ ,  $\{Y_\lambda\}_{\lambda \in \mathbb{N}}$  and  $\{Z_\lambda\}_{\lambda \in \mathbb{N}}$  such that  $R_\lambda \subseteq X_\lambda \times Y_\lambda \times Z_\lambda$ . A Relational Hash for  $\{R_\lambda\}_{\lambda \in \mathbb{N}}$  consists of four efficient algorithms:*

- A randomized key generation algorithm:  $\text{KEYGEN}(1^\lambda)$  outputs key  $pk$  from key space  $K_\lambda$ .
- The hash algorithm of first type (possibly randomized):  $\text{HASH}_1 : K_\lambda \times X_\lambda \rightarrow \text{RANGEX}_\lambda$ , here  $\text{RANGEX}_\lambda$  denotes the range of  $\text{HASH}_1$  for security parameter  $\lambda$ .
- The hash algorithm of second type (possibly randomized):  $\text{HASH}_2 : K_\lambda \times Y_\lambda \rightarrow \text{RANGERY}_\lambda$ , here  $\text{RANGERY}_\lambda$  denotes the range of  $\text{HASH}_2$  for security parameter  $\lambda$ .
- The deterministic verification algorithm:  
 $\text{VERIFY} : K_\lambda \times \text{RANGEX}_\lambda \times \text{RANGERY}_\lambda \times Z_\lambda \rightarrow \{0, 1\}$ .

Treating the third parameter  $z$  differently from the first two might strike as odd. Our reason behind the choice of this asymmetric definition is to convey the intention that we are not trying to hide  $z$  and that the verifier or attacker can choose the value of  $z$  to test relations.

In the rest of the paper we will drop the subscript  $\lambda$  for simplicity and it will be implicitly assumed in the algorithm descriptions. Often, we will also denote the 1 output of  $\text{VERIFY}$  as  $\text{ACCEPT}$ , and the 0 output as  $\text{REJECT}$ . The definition of Relational Hashing consists of two requirements: *Correctness* and *Security* (or *Secrecy*).

*Correctness:* Informally speaking, the correctness condition is, if an honest party evaluates  $\text{VERIFY}(\text{HASH}_1(pk, x), \text{HASH}_2(pk, y), z)$  for some key  $pk$  which is the output of  $\text{KEYGEN}$  and any  $(x, y, z) \in X \times Y \times Z$ , the output can differ from  $R(x, y, z)$  only with negligible probability (the probability is calculated over the internal randomness of  $\text{KEYGEN}$ ,  $\text{HASH}_1$  and  $\text{HASH}_2$ ). Formally,

**Definition 2 (Relational Hash - Correctness).** *A Relational Hash scheme  $(\text{KEYGEN}, \text{HASH}_1, \text{HASH}_2, \text{VERIFY})$  for a relation  $R \subseteq X \times Y \times Z$  satisfies correctness if the following holds for all  $(x, y, z) \in X \times Y \times Z$ :*

$$\Pr \left[ \begin{array}{l} pk \leftarrow \text{KEYGEN}(1^\lambda) \\ hx \leftarrow \text{HASH}_1(pk, x) \\ hy \leftarrow \text{HASH}_2(pk, y) \end{array} : \text{VERIFY}(pk, hx, hy, z) \equiv R(x, y, z) \right] \approx 1.$$

*Security:* The notion of security for a Relational Hash will depend on the context where the Relational Hash is going to be used and also on the a priori information available to the adversary. Recall that for a regular hash function one of the weakest form of security is one-wayness. We will consider Probabilistic Polynomial Time (PPT) adversaries for our security definitions.

**Definition 3 (Security of Relational Hash - One-Way).** *Let  $\mathcal{X}$  and  $\mathcal{Y}$  be (independent) probability distributions over  $X$  and  $Y$ . We define a Relational Hash scheme  $(\text{KEYGEN}, \text{HASH}_1, \text{HASH}_2, \text{VERIFY})$  to be one-way secure for the probability distributions  $\mathcal{X}$  and  $\mathcal{Y}$ , if the following hold:*

- $pk \leftarrow \text{KEYGEN}(1^\lambda)$ ,  $x \leftarrow \mathcal{X}$ ,  $y \leftarrow \mathcal{Y}$ ,  $hx \leftarrow \text{HASH}_1(pk, x)$ ,  $hy \leftarrow \text{HASH}_2(pk, y)$
- For any PPT adversary  $A_1$ , there exists a negligible function  $\text{negl}()$ , such that  $\Pr[A_1(pk, hx) = x] < \text{negl}(\lambda)$ .
- For any PPT adversary  $A_2$ , there exists a negligible function  $\text{negl}()$ , such that  $\Pr[A_2(pk, hy) = y] < \text{negl}(\lambda)$ .

Here the probabilities are calculated over the internal randomness of  $\text{KEYGEN}$ ,  $\text{HASH}_1$  and  $\text{HASH}_2$ , internal randomness of the adversarial algorithms  $A_1$  and  $A_2$  as well as the probability distributions  $\mathcal{X}$  and  $\mathcal{Y}$ .

The above definition captures the security notion in case the adversary has access to either type 1 or type 2 hash values. We observe that if the distributions  $\mathcal{X}$  and  $\mathcal{Y}$  remain independent, Relational Hash still remains one-way secure, even if the adversary has access to both type of hash values. However for correlated  $x$  and  $y$ , sampled from a joint probability distribution  $\Psi$  over  $X \times Y$ , the previous security notion does not provide sufficient security guarantee when the attacker has access to both types of hash values. For these kind of distributions we define a stronger security notion called *twin one-wayness* as follows.

**Definition 4 (Security of Relational Hash - Twin One-Way).** *Let  $\Psi$  be a probability distribution over  $X \times Y$ . We define a Relational Hash scheme  $(\text{KEYGEN}, \text{HASH}_1, \text{HASH}_2, \text{VERIFY})$  to be twin one-way secure for the probability distribution  $\Psi$ , if the following hold:*

- $pk \leftarrow \text{KEYGEN}(1^\lambda)$ ,  $(x, y) \leftarrow \Psi$ ,  $hx \leftarrow \text{HASH}_1(pk, x)$ ,  $hy \leftarrow \text{HASH}_2(pk, y)$ .
- For any PPT adversary  $A_1$ , there exists a negligible function  $\text{negl}()$ , such that  $\Pr[A_1(pk, hx, hy) = x] < \text{negl}(\lambda)$ .
- For any PPT adversary  $A_2$ , there exists a negligible function  $\text{negl}()$ , such that  $\Pr[A_2(pk, hx, hy) = y] < \text{negl}(\lambda)$ .

Here the probabilities are calculated over the internal randomness of  $\text{KEYGEN}$ ,  $\text{HASH}_1$  and  $\text{HASH}_2$ , internal randomness of the adversarial algorithms  $A_1$  and  $A_2$  as well as the probability distribution  $\Psi$ .

Note that the twin one-wayness property is actually a stronger version of correlated input security due to Rosen and Segev [RS09]. We require each coordinate to be one-way, whereas correlated input security requires the input involving all coordinates should be one-way.

*Remark 1.* For our application scenarios: biometric identification and authentication, the twin one-wayness property plays a key role. Intuitively, this guarantees that even if the server has access to both type of hashes coming from biometric templates (possibly generated at different times) of the same person, the template still remains one-way to the server<sup>3</sup>.

In this work, we are mostly interested in *sparse* relations (Definition 7). Informally speaking, for a sparse relation  $R \subseteq X \times Y \times Z$  and unknown  $x$  it is hard to output  $y$  and  $z$  such that  $(x, y, z) \in R$ . A Relational Hash scheme is called *unforgeable* if given  $hx = \text{HASH}_1(pk, x)$  and  $pk$  it is hard to output  $hy, z$ , such that  $\text{VERIFY}(pk, hx, hy, z)$  outputs 1. Formally,

**Definition 5 (Security of Relational Hash - Unforgeable).** *Let  $\mathcal{X}$  and  $\mathcal{Y}$  be (independent) probability distributions over  $X$  and  $Y$ . A Relational Hash scheme  $(\text{KEYGEN}, \text{HASH}_1, \text{HASH}_2, \text{VERIFY})$  is unforgeable for the probability distributions  $\mathcal{X}$  and  $\mathcal{Y}$ , if the following holds:*

- $pk \leftarrow \text{KEYGEN}(1^\lambda)$ ,  $x \leftarrow \mathcal{X}$ ,  $y \leftarrow \mathcal{Y}$ ,  $hx \leftarrow \text{HASH}_1(pk, x)$ ,  $hy \leftarrow \text{HASH}_2(pk, y)$ .
- For any PPT adversary  $A_1$ , there exists a negligible function  $\text{negl}()$ , such that:  $\Pr[(hy', z) \leftarrow A_1(pk, hx) \wedge \text{VERIFY}(pk, hx, hy', z) = 1] < \text{negl}(\lambda)$ .
- For any PPT adversary  $A_2$ , there exists a negligible function  $\text{negl}()$ , such that:  $\Pr[(hx', z) \leftarrow A_2(pk, hy) \wedge \text{VERIFY}(pk, hx', hy, z) = 1] < \text{negl}(\lambda)$ .

For Relational Hash functions, the strongest form of security notion is based on oracle simulations. The concept of oracle simulation was introduced in [Can97]. However, over there the author was interested in regular probabilistic hash functions. In case of Relational Hash functions, we want to say that: having  $hx = \text{HASH}_1(pk, x)$  gives no information on  $x$ , besides the ability to evaluate the value of  $R(x, y, z)$  for any  $y, z$  chosen from their respective domains. Similarly,  $hy = \text{HASH}_1(pk, y)$  should not provide any extra information other than the ability to evaluate the value of  $R(x, y, z)$  for any  $x \in X$  and  $z \in Z$ . Also, having access to both  $hx$  and  $hy$ , one should be able to only evaluate  $R(x, y, z)$  for any  $z \in Z$ .

For any relation  $R \subseteq X \times Y \times Z$  and  $x \in X, y \in Y$ , let  $R_x(\cdot, \cdot) : Y \times Z \rightarrow \{0, 1\}$ ,  $R_y(\cdot, \cdot) : X \times Z \rightarrow \{0, 1\}$  and  $R_{x,y}(\cdot) : Z \rightarrow \{0, 1\}$  be the oracles defined as follows:

- For any  $y' \in Y, z' \in Z$ ,  $R_x(y', z') = 1$  if and only if  $(x, y', z') \in R$ .
- For any  $x' \in X, z' \in Z$ ,  $R_y(x', z') = 1$  if and only if  $(x', y, z') \in R$ .
- For any  $z' \in Z$ ,  $R_{x,y}(z') = 1$  if and only if  $(x, y, z') \in R$ .

We note that giving oracle access to  $R_{x,y}$  on top of  $R_x$  and  $R_y$  is not superfluous as both  $x$  and  $y$  are generated and kept unknown from the adversary.

---

<sup>3</sup> Strictly speaking, we need a stronger a security criterion, i.e. not only the server should be able to recover exact  $x$  or  $y$ , it should not be able to recover any nearby  $x'$  from  $x$  or  $y$ . Theorem 4 in Sect. 4, in fact guarantees this stronger security notion.

**Definition 6 (Security of Relational Hash - Oracle Simulation).** Let  $\Psi$  be a probability distribution over  $X \times Y$ . A Relational Hash scheme (KEYGEN, HASH<sub>1</sub>, HASH<sub>2</sub>, VERIFY) is said to be oracle simulation secure with respect to the distribution  $\Psi$  if for any PPT adversary  $C$ , there exists a PPT simulator  $S$  such that for any predicate  $P(\cdot, \cdot, \cdot) : K \times X \times Y \rightarrow \{0, 1\}$  (where  $K$  is the range of KEYGEN), there exists a negligible function  $\text{negl}(\cdot)$ , such that

$$\left| \Pr[C(pk, \text{HASH}_1(pk, x), \text{HASH}_2(pk, y)) = P(pk, x, y)] - \Pr[S^{R_x, R_y, R_{x,y}}(pk) = P(pk, x, y)] \right| < \text{negl}(\lambda),$$

where  $(x, y) \leftarrow \Psi$  and  $pk \leftarrow \text{KEYGEN}(1^\lambda)$ .

### 3 Relational Hash for Linearity in $\mathbb{F}_2^n$

We now construct a Relational Hash scheme for the domains  $X, Y, Z = \mathbb{F}_2^n$  and the relation  $R = \{(x, y, z) \mid x + y = z \wedge x, y, z \in \mathbb{F}_2^n\}$ .

KEYGEN: Given the security parameter, bilinear groups  $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$  are generated of prime order  $q$ , exponential in the security parameter, and with a bilinear pairing operator  $e$ . Now we sample generators  $\mathbf{g}_0 \leftarrow \mathbb{G}_1$  and  $\mathbf{h}_0 \leftarrow \mathbb{G}_2$ . Next we sample  $\langle a_i \rangle_{i=1}^{n+1}$  and  $\langle b_i \rangle_{i=1}^{n+1}$ , all randomly from  $\mathbb{Z}_q^*$ . Define  $\mathbf{g}_i = \mathbf{g}_0^{a_i}$  and  $\mathbf{h}_i = \mathbf{h}_0^{b_i}$ . Now we define the output of KEYGEN as  $pk := (pk_1, pk_2, pk_R)$ , defined as follows:

$$pk_1 := \langle \mathbf{g}_i \rangle_{i=0}^{n+1}, \quad pk_2 := \langle \mathbf{h}_i \rangle_{i=0}^{n+1}, \quad pk_R := \sum_{i=1}^{n+1} a_i b_i.$$

HASH<sub>1</sub>: Given plaintext  $x = \langle x_i \rangle_{i=1}^n \in \mathbb{F}_2^n$  and  $pk_1 = \langle \mathbf{g}_i \rangle_{i=0}^{n+1}$ , the hash is constructed as follows: Sample a random  $r \in \mathbb{Z}_q^*$  and then compute the following:

$$hx := \left( \mathbf{g}_0^r, \left\langle \mathbf{g}_i^{(-1)^{x_i r}} \right\rangle_{i=1}^n, \mathbf{g}_{n+1}^r \right).$$

HASH<sub>2</sub>: Given plaintext  $y = \langle y_i \rangle_{i=1}^n \in \mathbb{F}_2^n$  and  $pk_2 = \langle \mathbf{h}_i \rangle_{i=0}^{n+1}$ , the hash is constructed as follows: Sample a random  $s \in \mathbb{Z}_q^*$  and then compute the following:

$$hy := \left( \mathbf{h}_0^s, \left\langle \mathbf{h}_i^{(-1)^{y_i s}} \right\rangle_{i=1}^n, \mathbf{h}_{n+1}^s \right).$$

VERIFY: Given hashes  $hx = \langle hx_i \rangle_{i=0}^{n+1}$  and  $hy = \langle hy_i \rangle_{i=0}^{n+1}$ , the quantity  $z = \langle z_i \rangle_{i=1}^n \in \mathbb{F}_2^n$  and  $pk_R$ , the algorithm VERIFY checks the following equality:

$$e(hx_0, hy_0)^{pk_R} \stackrel{?}{=} e(hx_{n+1}, hy_{n+1}) \prod_{i=1}^n e(hx_i, hy_i)^{(-1)^{z_i}}.$$

*Correctness.* Correctness of the scheme follows from standard algebraic manipulation of pairing operations. Details are given in [MR14].

*One-Wayness.* This Relational Hash can be shown to be one-way secure based on the SXDH assumption, and a new hardness assumption we call Binary Mix DLP. The assumption says if we choose a random  $x$  from  $\mathbb{F}_2^n$  (for sufficiently large  $n$ ),  $n$  random elements  $\mathbf{g}_1, \dots, \mathbf{g}_n$  from group  $\mathbb{G}$  then given the product  $\prod_{i=1}^n \mathbf{g}_i^{(-1)^{x_i}}$  it is hard to find any candidate  $x$ .

**Assumption 1. (Binary Mix DLP) :** *Assuming a generation algorithm  $\mathcal{G}$  that outputs a tuple  $(n, q, \mathbb{G})$  such that  $\mathbb{G}$  is a group of prime order  $q$ , the Binary Mix DLP assumption asserts that given random elements  $\langle \mathbf{g}_i \rangle_{i=1}^n$  from the group  $\mathbb{G}$  and  $\prod_{i=1}^n \mathbf{g}_i^{(-1)^{x_i}}$ , for a random  $x \leftarrow \mathbb{F}_2^n$ , it is computationally infeasible to output  $y \in \mathbb{F}_2^n$  such that*

$$\prod_{i=1}^n \mathbf{g}_i^{(-1)^{x_i}} = \prod_{i=1}^n \mathbf{g}_i^{(-1)^{y_i}}.$$

There is an interesting parallel between the Binary Mix DLP assumption and the Discrete Log hardness assumption which may appeal to the appreciation of its hardness at an intuitive level. The Discrete Log problem asks to find  $w \in \mathbb{Z}_q^*$  given a random element  $\mathbf{g} \in \mathbb{G}$  and  $\mathbf{g}^w$ . Consider the sequence of elements  $\mathbf{g}_1 = \mathbf{g}, \mathbf{g}_2 = \mathbf{g}^2, \dots, \mathbf{g}_\lambda = \mathbf{g}^{2^\lambda}$ , where  $\lambda = \lg q$ . When we think of the binary expansion of  $w = \overline{w_\lambda \dots w_0}$  and interpret the vector  $W = w_\lambda \dots w_0$  in  $\mathbb{F}_2^{\lambda+1}$ , then equivalently we are asking for computing  $W$ , given the product  $\prod_{i=0}^\lambda \mathbf{g}_i^{w_i}$ .

In the Binary Mix DLP problem, the difference is that the  $\mathbf{g}_i$ 's are independently random and that instead of raising the  $\mathbf{g}_i$ 's to the powers 0 or 1, we raise them to the powers  $\pm 1$ . This is, of course, not a formal proof of its hardness. In [MR14], we show that the Binary Mix DLP assumption can actually be reduced to the more standard Random Modular Subset Sum assumption [Lyu05]. As an added assurance, in [MR14], we show that the Binary Mix DLP assumption is also secure in the Generic Group Model [Sho97].

The Binary Mix DLP assumption is similar to [BGG95], where Bellare et al. define a hash function to be a subset product of publicly given random group elements based on the bits of the plaintext. In our case, we either use a random group element or its inverse depending on the bit. They achieve reduction from DLP to collision resistance. In contrast, this does not work for one-wayness, as for certain admissible values of  $(q, n)$  our function (as also [BGG95]) may turn out to be collision-free.

**Theorem 1.** *The above algorithms (KEYGEN, HASH<sub>1</sub>, HASH<sub>2</sub>, VERIFY) constitute a Relational Hash scheme for the relation  $R = \{(x, y, z) \mid x + y = z \wedge x, y, z \in \mathbb{F}_2^n\}$ . The scheme is one-way secure under the SXDH and Binary Mix DLP assumptions, when  $x$  and  $y$  are sampled uniformly from  $\mathbb{F}_2^n$ .*

*Twin One-Wayness.* Until now, we have shown this Relational Hash is one-way when the adversary has access to only one type of hash values. However, an important scenario to consider is the case when adversary has access to both type of hash values for any  $x$  uniformly drawn from  $\mathbb{F}_2^n$ . The following theorem claims our scheme is indeed twin one-way secure in this case and is proved in [MR14].

**Theorem 2.** *The above algorithms (KEYGEN, HASH<sub>1</sub>, HASH<sub>2</sub>, VERIFY) constitute a Relational Hash scheme for the relation  $R = \{(x, y, z) \mid x + y = z \wedge x, y, z \in \mathbb{F}_2^n\}$ . The scheme is twin one-way secure in the generic group model, when  $x$  is sampled uniformly from  $\mathbb{F}_2^n$  and  $y = x$ .*

*Unforgeability and Oracle Simulation Security.* In Sect. 2, we show this Relational Hash is in fact a 2-value perfectly one-way function, albeit under a stronger hardness assumption. By Theorem 8 from Sect. 5, that will imply this Relational Hash construction is also unforgeable and oracle simulation secure.

*Remark 2.* This linear Relational Hash construction is weakly homomorphic, in the sense that, given

$$\text{HASH}_2(y) = (hy_0, \langle hy_i \rangle_{i=1}^n, hy_{n+1}) = \left( \mathbf{h}_0^s, \left\langle \mathbf{h}_i^{(-1)^{y_i} s} \right\rangle_{i=1}^n, \mathbf{h}_{n+1}^s \right),$$

it is easy to construct

$$\text{HASH}_2(y + t) = \left( hy_0, \left\langle hy_i^{(-1)^{t_i}} \right\rangle_{i=1}^n, hy_{n+1} \right) = \left( \mathbf{h}_0^s, \left\langle \mathbf{h}_i^{(-1)^{y_i+t_i} s} \right\rangle_{i=1}^n, \mathbf{h}_{n+1}^s \right)$$

for any  $t \in \mathbb{F}_2^n$ . HASH<sub>1</sub> is also homomorphic in a similar manner. However, this does not really refute any of our security claims. In fact, in next section we will see this linear homomorphism gives us strong security guarantee for relation hash construction for hamming proximity (Theorem 4).

*Remark 3.* Theorem 2 and Remark 2 imply that given HASH<sub>1</sub>( $x$ ), HASH<sub>2</sub>( $y$ ) and  $x + y$  it is hard to output either of  $x$  or  $y$ , for uniformly sampled  $x$  and  $y$  from  $\mathbb{F}_2^n$ .

**Relational Hash for Linearity in  $\mathbb{F}_p^n$ :** For any prime  $p$ , we can choose the order  $q$  of the bilinear groups to be exponential in the security parameters as well as equal to 1 (mod  $p$ ). This means the group  $\mathbb{Z}_q^*$  has a subgroup  $\mathbb{J}_p$  of prime order  $p$ . Let  $\omega$  be an arbitrary generator of  $\mathbb{J}_p$ . We can publish this arbitrary generator as part of the public key. For HASH<sub>1</sub> evaluation (similarly in HASH<sub>2</sub>), we can simply calculate  $hx_i$  as  $\mathbf{g}_i^{\omega^{x_i} r}$  (instead of  $\mathbf{g}_i^{(-1)^{x_i} r}$ ). Similarly during verification, instead of checking  $e(hx_0, hy_0)^{pk_R} \stackrel{?}{=} e(hx_{n+1}, hy_{n+1}) \prod_{i=1}^n e(hx_i, hy_i)^{(-1)^{z_i}}$ , we can just check  $e(hx_0, hy_0)^{pk_R} \stackrel{?}{=} e(hx_{n+1}, hy_{n+1}) \prod_{i=1}^n e(hx_i, hy_i)^{\omega^{-z_i}}$ . We provide the details in [MR14].

## 4 Relational Hash for Hamming Proximity

In this section we construct a Relational Hash for the domains  $X, Y = \mathbb{F}_2^n$  and the relation<sup>4</sup>  $R_\delta = \{(x, y) \mid \text{dist}(x, y) \leq \delta \wedge x, y \in \mathbb{F}_2^n\}$ , where **dist** is

<sup>4</sup> Note that Relational Hash is defined over 3-tuple relations (Definition 2). However, here proximity encryption is defined over 2-tuple relations. 2-tuple relations can be regarded as special cases of 3-tuple relations, where the third entry does not matter. E.g. the relation  $R'_\delta \subseteq \mathbb{F}_2^n \times \mathbb{F}_2^n \times Z$  (where  $Z$  is any non empty domain) and  $(x, y, z) \in R'_\delta$  if and only if  $(x, y) \in R_\delta$ .



the hamming distance and  $\delta$  is a positive integer less than  $n$ . Specifically, we construct a Relational Hash for proximity from a family of binary  $(n, k, d)$  linear error correcting codes (ECC)  $\mathcal{C}$  and a Relational Hash for linearity in  $\mathbb{F}_2^k$ : (KEYGENLINEAR, HASHLINEAR<sub>1</sub>, HASHLINEAR<sub>2</sub>, VERIFYLINEAR).

For any  $C \in \mathcal{C}$ , ENCODE and DECODE are the encoding and decoding algorithms of the  $(n, k, d)$  error correcting code  $C$ . For any  $x \in \mathbb{F}_2^n$ ,  $\text{weight}(x)$  is the usual *hamming* weight of  $x$ , denoting the number of one's in the binary representation of  $x$ . For any error vector  $e \in \mathbb{F}_2^n$ , with  $\text{weight}(e) \leq d/2$  and  $m \in \mathbb{F}_2^k$  we have,

$$\text{DECODE}(\text{ENCODE}(m) + e) = m.$$

If  $\text{weight}(e) > d/2$ , the decoding algorithm DECODE is allowed to return  $\perp$ .

KEYGEN: Given the security parameter, choose a binary  $(n, k, 2\delta + 1)$  linear error correcting code  $C$ , where  $k$  is of the order of the security parameter. Run KEYGENLINEAR and let  $pk_{lin}$  be its output. Publish,

$$pk := (\text{ENCODE}, \text{DECODE}, pk_{lin}).$$

HASH<sub>1</sub>: Given plaintext  $x \in \mathbb{F}_2^n$  and  $pk = (\text{ENCODE}, \text{DECODE}, pk_{lin})$ , the hash value is constructed as follows: Sample a random  $r \leftarrow \mathbb{F}_2^k$  and then compute the following:

$$\begin{aligned} hx_1 &:= x + \text{ENCODE}(r) \\ hx_2 &:= \text{HASHLINEAR}_1(pk_{lin}, r) \end{aligned}$$

Publish the final hash value  $hx := (hx_1, hx_2)$ .

HASH<sub>2</sub> is defined similarly.

VERIFY: Given the hash values  $hx = (hx_1, hx_2)$ ,  $hy = (hy_1, hy_2)$  and  $pk = (\text{ENCODE}, \text{DECODE}, pk_{lin})$  verification is done as follows.

- Recover  $z$  as  $z := \text{DECODE}(hx_1 + hy_1)$ .
- Output REJECT if DECODE returns  $\perp$  or  $\text{dist}(\text{ENCODE}(z), hx_1 + hy_1) > \delta$
- Output VERIFYLINEAR( $pk_{lin}, hx_2, hy_2, z$ ).

**Theorem 3.** *The above algorithms (KEYGEN, HASH<sub>1</sub>, HASH<sub>2</sub>, VERIFY) constitute a Relational Hash for the relation  $R_\delta = \{(x, y) \mid \text{dist}(x, y) \leq \delta \wedge x, y \in \mathbb{F}_2^n\}$ . The scheme is one-way secure with respect to the uniform distributions on  $\mathbb{F}_2^n$  if the linear Relational Hash is a one-way secure with respect to the uniform distributions on  $\mathbb{F}_2^k$ . The scheme is unforgeable for the uniform distributions on  $\mathbb{F}_2^n$  if the linear Relational Hash is unforgeable with respect to the uniform distributions on  $\mathbb{F}_2^k$ .*

*Twin One-Wayness.* For our target application scenarios (biometric identification/authentication), we need a slightly stronger security property compared to the Twin one-wayness as defined in Definition 4. We only consider a passive adversary looking at the communication transcripts between the entities. Consideration of active adversaries would require an additional challenge-response

mechanism which we do not develop in this paper. In particular, we should show that if an attacker has access to  $\text{HASH}_1(x)$  and a number of samples of  $\text{HASH}_2(y_i)$  (where  $x$  and the  $y_i$ 's are biometric templates generated by same individual), the attacker cannot output any other biometric template  $z$  near to  $x$ . If we assume that every individual's biometric template has full entropy we can model the scenario as follows:

$$x \leftarrow \mathbb{F}_2^n, y_i = x + e_i,$$

where the  $e_i$ 's are sampled from some known noise distribution  $\Xi$ , such that with high probability we have  $\text{weight}(e_i) \leq \delta$ . We now show that, given  $\text{HASH}_1(x)$  and any number of samples<sup>5</sup>  $\text{HASH}_2(y_i)$ , the attacker cannot output  $z$ , such that  $\text{dist}(x, z) \leq \delta$ . The proof, which is a reduction to twin one-wayness of the linear Relational Hash is given in [MR14].

**Theorem 4.** *If the above Relational Hash for  $R_\delta = \{(x, y) \mid \text{dist}(x, y) \leq \delta \wedge x, y \in \mathbb{F}_2^n\}$ , is instantiated by the twin one-way secure linear Relational Hash in Sect. 3, then for a random  $x \leftarrow \mathbb{F}_2^n$  and for any polynomially bounded number of error samples  $e_1, \dots, e_t \leftarrow \Xi$ , given  $(\text{HASH}_1(x), \text{HASH}_2(x + e_1), \dots, \text{HASH}_2(x + e_t))$  it is hard to output  $x' \in \mathbb{F}_2^n$  such that  $\text{dist}(x', x) \leq \delta$ .*

**Privacy Preserving Biometric Authentication Scheme.** Suppose we have a biometric authentication scheme, where during registration phase a particular user generates a biometric template  $x \in \{0, 1\}^n$  and sends it to the server. During authentication phase the user generates a new biometric template  $y \in \{0, 1\}^n$  and sends  $y$  to server. The server authenticates the user if  $\text{dist}(x, y) \leq \delta$ . The drawback of this scheme is the lack of template privacy. However, if we have a Relational Hash ( $\text{KEYGEN}, \text{HASH}_1, \text{HASH}_2, \text{VERIFY}$ ) for the relation  $R_\delta = \{(x, y) \mid \text{dist}(x, y) \leq \delta \wedge x, y \in \mathbb{F}_2^n\}$ , we readily get a privacy preserving biometric authentication scheme as follows: 1. A trusted third party runs  $\text{KEYGEN}$  and publishes  $pk \leftarrow \text{KEYGEN}$ . 2. During Registration, the client generates biometric template  $x \in \{0, 1\}^n$  and sends  $hx = \text{HASH}_1(pk, x)$  to the server. 3. During Authentication, the client generates biometric template  $y \in \{0, 1\}^n$  and sends  $hy = \text{HASH}_2(pk, y)$  to the server. 4. The server authenticates the client iff  $\text{VERIFY}(pk, hx, hy)$  returns ACCEPT.

If we assume that the biometric templates of individuals follow uniform distribution over  $\{0, 1\}^n$ , then Theorem 3 would imply that the server can never recover the original biometric template  $x$ . Moreover, the unforgeability property guarantees that even if the server's database gets leaked to an attacker then also the attacker cannot come up with a forged  $hy'$ , which would authenticate the attacker. Theorem 4 will guarantee that even with access to the registered hash and several authentication transcripts from the same individual, the biometric template will remain private to the server.

In spite of these strong guarantees there is a significant drawback of our privacy preserving authentication scheme. One basic premise of this scheme is

<sup>5</sup> Limited only by the time complexity of the attacker.

that the biometric template  $x$  comes from a uniform distribution over  $\{0, 1\}^n$ . From a practical point of view this is really a strong assumption. One interesting open problem in this direction is whether we can build a privacy preserving biometric authentication scheme when  $x$  comes from a distribution with high min-entropy which is not necessarily uniform.

## 5 Relation Among Notions of Security for Relational Hashes

In Sect. 2 we introduced three natural definitions of security for Relational Hash functions: one-wayness, unforgeability and oracle simulation security. In this section we define the notion of *sparse* and *biased* relations. We show, if a Relational Hash function is unforgeable, that implies the relation must be sparse. Following [CMR98], we extend the notion of *2-value Perfectly One-Way* (2-POW) function. We show if a Relational Hash function is 2-POW, then the relation must be biased. We also show that the 2-POW property is actually a sufficient condition for oracle simulation security, as well as unforgeability (when the relation is sparse). These implications are summarized in Fig. 1.

We begin by asking the question: What kind of relations can support the existence of an unforgeable Relational Hash? It is easy to see that certain relations cannot support unforgeability. Take, for example, the relation  $R(x, y, z)$ , where  $x, y \in \mathbb{F}_2^n$  and  $z \in \mathbb{F}_2$  which returns 1 iff the parity of  $x + y$  is equal to the bit  $z$ . One cannot construct an unforgeable hash for this relation because given the type 1 hash of a random  $x$ , it is easy to construct a type 2 hash of a  $y$  such that the relational verification outputs 1, without knowing  $x$ : We just pick an arbitrary  $y$ , compute a type 2 hash of the arbitrary  $y$  and verify with the relational key with the type 1 hash of  $x$  for both  $z$  values 0 and 1.

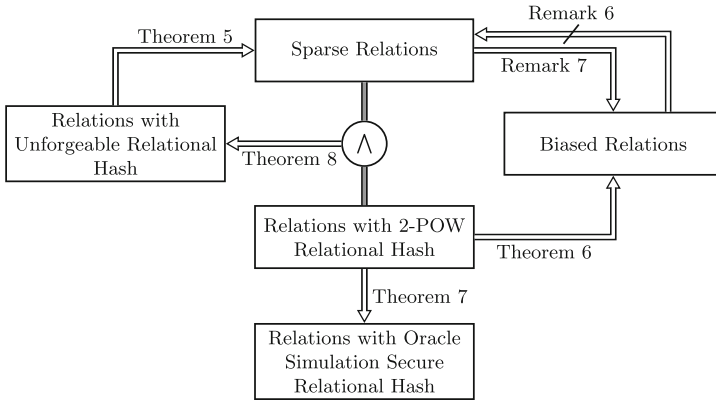
So the intuitive property of relations supporting unforgeability is that without knowing  $x$ , it should be hard to come up with  $(y, z)$ , such that  $R(x, y, z)$  holds. We formalize this intuition below in defining *sparse* relations.

**Definition 7.** A relation  $R \subseteq X \times Y \times Z$  is called a *sparse relation* in the first co-ordinate with respect to a probability distribution  $\mathcal{X}$  over  $X$ , if for all PPTs  $A$ :

$$\Pr[x \leftarrow \mathcal{X}, (y, z) \leftarrow A(\lambda) : (x, y, z) \in R] < \text{negl}(\lambda)$$

Similarly, we can define a *sparse relation* in the second co-ordinate with respect to a probability distribution  $\mathcal{Y}$  over  $Y$ . A relation  $R \subseteq X \times Y \times Z$  is called a *sparse relation* with respect to probability distributions  $\mathcal{X}$  over  $X$  and  $\mathcal{Y}$  over  $Y$ , if it is a sparse relation in first coordinate with respect to  $\mathcal{X}$ , as well as a sparse relation in second coordinate with respect to  $\mathcal{Y}$ .

*Remark 4.* Similar to Sect. 2, the definitions given in this sections are actually defined with respect to ensemble of probability distributions  $\mathcal{X}_\lambda, \mathcal{Y}_\lambda, \mathcal{K}_\lambda$ , ensemble of sets  $X_\lambda, Y_\lambda, Z_\lambda, K_\lambda$  and ensemble of relation  $R_\lambda$ . However, for simplicity we drop the subscript  $\lambda$ .



**Fig. 1.** Relationship among Types of Relations. Arrowhead indicates direction of implication. Strike on an arrow indicates the existence of a counter-example.

Now, we show if a Relational Hash function is unforgeable, that implies the relation must be sparse.

**Theorem 5.** *If a Relational Hash scheme  $(\text{KEYGEN}, \text{HASH}_1, \text{HASH}_2, \text{VERIFY})$  for a relation  $R$  is unforgeable for probability distributions  $\mathcal{X}$  over  $X$  and  $\mathcal{Y}$  over  $Y$ , then the relation  $R$  is sparse with respect to  $\mathcal{X}$  and  $\mathcal{Y}$ .*

*Proof.* Suppose, the relation  $R$  is not sparse over first coordinate, and there exists a PPT attacker  $A$  such that  $\Pr[x \leftarrow \mathcal{X}, (y, z) \leftarrow A(\lambda) : (x, y, z) \in R]$  is non-negligible. Now, given an unforgeability challenge  $(pk, cx)$ , such that  $pk \leftarrow \text{KEYGEN}(1^\lambda)$  and  $cx \leftarrow \text{HASH}_1(pk, x)$  for some  $x \leftarrow \mathcal{X}$ ; we can just get  $(y, z) \leftarrow A(\lambda)$  and output  $(\text{HASH}_2(pk, y), z)$ . From the correctness of the Relational Hash function, it follows that this output is a valid forgery with non-negligible probability.  $\square$

Following [CMR98], we recall the definition of 2-value perfectly one-way (POW) functions. Intuitively, this property states that the distribution of two probabilistic hashes of the same value is computationally indistinguishable from the distribution of probabilistic hashes of two independent values. This is a useful property, because if we can show a Relational Hash function is 2-POW, we show that it would immediately imply the Relational Hash function is oracle simulation secure, as well as unforgeable (if the relation is sparse).

**Definition 8 (2-value Perfectly One-Way function).** *Let  $\mathcal{X}$  be a probability distribution over  $X$ . Let  $H = \{h_k\}_{k \in K}$  be a keyed probabilistic function family with domain  $X$  and randomness space  $U$ , where the key  $k$  gets sampled from a probability distribution  $\mathcal{K}$  over  $K$ .  $H$  is 2-value perfectly one-way (POW) with respect to  $\mathcal{X}$  and  $\mathcal{K}$  if for any PPT distinguisher  $D$ ,*

$$\left| \begin{array}{l} \Pr[D(k, h_k(x, r_1), h_k(x, r_2)) = 1] \\ - \Pr[D(k, h_k(x_1, r_1), h_k(x_2, r_2)) = 1] \end{array} \right| < \text{negl}(\lambda),$$

where  $x, x_1, x_2$  are drawn independently from  $\mathcal{X}$ ,  $k$  is drawn from  $\mathcal{K}$  and  $r_1, r_2$  are generated uniformly at random from the randomness space  $U$ .

*Remark 5.* In [CMR98], the key  $k$  was universally quantified, and the function family  $H$  was called 2-POW if the inequality was true for all  $k \in \mathcal{K}$ . However, for our purpose it is sufficient if we consider random  $k$  coming from the distribution  $\mathcal{K}$  (or KEYGEN).

Now we ask what kind of relations can support the existence of 2-POW Relational Hashes? Intuitively, we require that it should be hard to distinguish two distinct samples  $x$  and  $w$  from the distribution  $\mathcal{X}$  by testing relations with a  $(y, z)$  tuple which is efficiently computable without knowing the samples. That is we should have  $R(x, y, z)$  and  $R(w, y, z)$  come out equal most of the time. This intuition is formalized in the following definition of *biased* relations.

**Definition 9.** A relation  $R \subseteq X \times Y \times Z$  is called a *biased relation* in the first co-ordinate with respect to a probability distribution  $\mathcal{X}$  over  $X$ , if for all PPTs  $A$ :

$$\Pr[x, w \leftarrow \mathcal{X}, (y, z) \leftarrow A(\lambda) : R(x, y, z) \neq R(w, y, z)] < \text{negl}(\lambda)$$

Similarly, we can define a *biased relation* in the second co-ordinate with respect to a probability distribution  $\mathcal{Y}$  over  $Y$ . A relation  $R \subseteq X \times Y \times Z$  is called a *biased relation* with respect to independent probability distributions  $\mathcal{X}$  over  $X$  and  $\mathcal{Y}$  over  $Y$ , if it is a biased relation in first coordinate with respect to  $\mathcal{X}$ , as well as a biased relation in second coordinate with respect to  $\mathcal{Y}$ .

*Remark 6.* We observe that if a relation  $R$  is biased, then its complement  $\bar{R}$  is also biased. Now one might begin to think that maybe for a biased relation  $R$ , either  $R$  or  $\bar{R}$  is sparse. However, the following counterexample shows that this is not the case. Consider the relation  $R(x, y, z)$  which outputs the first bit of  $y$ . This is a biased relation, but neither  $R$ , nor its complement  $\bar{R}$  is sparse.

*Remark 7.* The other direction is actually an implication, that is, if a relation  $R$  is sparse then it is also biased. The proof intuition is as follows: Given an algorithm  $A$  breaking the biased-ness of  $R$ , we construct an algorithm breaking the sparse-ness of  $R$ . Let  $A$  output  $(y, z)$ , such that with probability  $p$  over the choice of  $x \leftarrow \mathcal{X}$ ,  $R(x, y, z) = 1$  and therefore with probability  $1 - p$ ,  $R(x, y, z) = 0$ . The probability of breaking the biased-ness of  $R$  is thus  $2p(1 - p)$  which should be non-negligible. Hence  $p$  should be non-negligible. Now observe that  $p$  is the probability of breaking the sparse-ness of  $R$ .

Now, we show if a Relational Hash is 2-POW, then the relation must be biased.

**Theorem 6.** For a Relational Hash scheme  $(\text{KEYGEN}, \text{HASH}_1, \text{HASH}_2, \text{VERIFY})$  for a relation  $R$ , if  $\text{HASH}_1$  is 2-value Perfectly One-Way with respect to  $\mathcal{X}$  and KEYGEN, then  $R$  is a biased relation in the 1st co-ordinate with respect to  $\mathcal{X}$ .

*Proof.* We are given that,

$$\forall PPT D : \left| \begin{array}{l} \Pr[D(k, \text{HASH}_1(k, x, r_1), \text{HASH}_1(k, x, r_2)) = 1] \\ - \Pr[D(k, \text{HASH}_1(k, x_1, r_1), \text{HASH}_1(k, x_2, r_2)) = 1] \end{array} \right| < \text{negl}(\lambda)$$

Suppose  $R$  is not a biased relation in the 1st co-ordinate. Then, there exists an efficient algorithm  $A$ , which outputs  $(y, z) \in Y \times Z$ , such that  $\Pr[x \leftarrow X, (y, z) \leftarrow A(\lambda) : R(x, y, z) \neq R(w, y, z)]$  is non-negligible in the security parameter. So now given  $(k, \text{HASH}_1(k, x, r_1), \text{HASH}_1(k, w, r_2))$ , we generate  $(y, z) \leftarrow A(\lambda)$ , compute  $\text{HASH}_2(k, y, r')$  and then compute  $\text{VERIFY}(k, \text{HASH}_1(k, x, r_1), \text{HASH}_2(k, y, r'), z)$  and  $\text{VERIFY}(k, \text{HASH}_1(k, w, r_2), \text{HASH}_2(k, y, r'), z)$ . By the correctness of the Relational Hash scheme, these boolean results are  $R(x, y, z)$  and  $R(w, y, z)$  respectively. In the case  $R(x, y, z) = R(w, y, z)$ , the distinguisher  $D$  outputs 1, else 0. By the non-sparseness of  $R$ ,  $D$  will have a non-negligible chance of distinguishing the distributions. Hence we get a contradiction.  $\square$

Theorem 7, stated below, shows that if a Relational Hash is 2-POW, then it is also oracle simulation secure.

**Theorem 7.** *For a Relational Hash scheme  $(\text{KEYGEN}, \text{HASH}_1, \text{HASH}_2, \text{VERIFY})$ , if the algorithms  $\text{HASH}_1$  and  $\text{HASH}_2$  are individually 2-value Perfectly One-Way for distributions  $(\mathcal{X}, \text{KEYGEN})$  and  $(\mathcal{Y}, \text{KEYGEN})$  respectively, then the Relational Hash scheme is Oracle Simulation Secure for the distribution  $\mathcal{X} \times \mathcal{Y}$ . Formally, for all PPT  $C$ , there exists a PPT  $S$ , such that:*

$$\left| \begin{array}{l} \Pr[C(pk, \text{HASH}_1(pk, x), \text{HASH}_2(pk, y)) = P(pk, x, y)] \\ - \Pr[S^{R_x, R_y, R_{x,y}}(pk) = P(pk, x, y)] \end{array} \right| < \text{negl}(\lambda),$$

where  $pk \leftarrow \text{KEYGEN}, x \leftarrow \mathcal{X}, y \leftarrow \mathcal{Y}$ .

Finally, we show that if a Relational Hash is 2-POW as well as sparse, then it must be unforgeable.

**Theorem 8.** *If  $(\text{KEYGEN}, \text{HASH}_1, \text{HASH}_2, \text{VERIFY})$  is a Relational Hash scheme for a sparse relation  $R$  with respect to independent probability distributions  $\mathcal{X}$  and  $\mathcal{Y}$  and  $\text{HASH}_1$  ( $\text{HASH}_2$ ) is 2-value Perfectly One-Way for distribution  $\mathcal{X}$  ( $\mathcal{Y}$ ) and  $\text{KEYGEN}$ , then the Relational Hash scheme is unforgeable for the distribution  $\mathcal{X} \times \mathcal{Y}$ .*

*Proof.* Assume that the scheme is not unforgeable. This means that given  $(pk, \text{HASH}_1(pk, x, r))$  for  $x \leftarrow \mathcal{X}$ , there is an attacker  $A$ , which outputs  $\text{HASH}_2(pk, y, s)$  and  $z$ , such that  $R(x, y, z) = 1$ , with non-negligible probability. Using  $A$ , we now build an attacker  $B$  which distinguishes the distributions  $(pk, \text{HASH}_1(pk, x, r_1), \text{HASH}_1(pk, x, r_2))$  and  $(pk, \text{HASH}_1(pk, x, r_1), \text{HASH}_1(pk, x', r_2))$  with non-negligible probability. Given  $(pk, \text{HASH}_1(pk, x, r_1), \text{HASH}_1(pk, w, r_2))$ ,  $B$  sends  $\text{HASH}_1(pk, x, r_1)$  to  $A$ . With non-negligible probability  $A$  outputs  $\text{HASH}_2(pk, y, s)$  and  $z$ , such that  $R(x, y, z) = 1$ . Now since  $R$  is a sparse relation, if  $w \neq x$ , then with non-negligible probability  $R(w, y, z) = 0$ , whereas if  $w = x$ , then  $R(w, y, z) = 1$ . Now  $R(w, y, z)$  can be efficiently evaluated by computing  $\text{VERIFY}(pk, \text{HASH}_1(pk, w, r_2), \text{HASH}_2(pk, y, s), z)$ . Thus,  $B$  will have a non-negligible probability of breaking the 2-value POW security of  $\text{HASH}_1$ .  $\square$

### Stronger Security Properties for the Relational Hash Constructions.

In Theorem 9, we show that the Relational Hash construction for linearity over  $\mathbb{F}_2^n$  from Sect. 3 is actually a 2-value perfectly one-way function. This property is based on a stronger hardness assumption called Decisional Binary Mix (Assumption 2). In [MR14] we show that this assumption holds in Generic Group Model [Sho97]. One can easily verify that the linearity relation over  $\mathbb{F}_2^n$ ,  $R = \{(r, s, z) \mid r + s = z \wedge r, s, z \in \mathbb{F}_2^n\}$  is actually a sparse relation with respect to uniform distributions over  $\mathbb{F}_2^n$ . Hence, by Theorems 7 and 8 we get that the Relational Hash construction from Sect. 3 is actually oracle simulation secure as well as unforgeable with respect to the independent uniform distributions over  $\mathbb{F}_2^n$ .

**Assumption 2 (Decisional Binary Mix).** *Assuming a generation algorithm  $\mathcal{G}$  that outputs a tuple  $(n, q, \mathbb{G})$  such that  $\mathbb{G}$  is a group of prime order  $q$ , the Decisional Binary Mix assumption asserts that for random  $x, y \leftarrow \mathbb{F}_2^n$ , given random elements  $\langle \mathbf{g}_i \rangle_{i=1}^n, \langle \mathbf{f}_i \rangle_{i=1}^n$  from the group  $\mathbb{G}$  it is hard to distinguish the following distributions:*

$$\left( \prod_{i=1}^n \mathbf{g}_i^{(-1)^{x_i}}, \prod_{i=1}^n \mathbf{f}_i^{(-1)^{x_i}} \right) \text{ and } \left( \prod_{i=1}^n \mathbf{g}_i^{(-1)^{x_i}}, \prod_{i=1}^n \mathbf{f}_i^{(-1)^{y_i}} \right).$$

**Theorem 9.** *The algorithms (KEYGEN, HASH<sub>1</sub>, VERIFY) in Sect. 3 constitute a 2-value Perfectly One Way Function for the uniform distribution on  $\mathbb{F}_2^n$ , under the Decisional Binary Mix and DDH assumptions.*

### On Stronger Security Properties for the Proximity Hash Constructions.

We observe that our proximity hash construction is not 2-POW secure. This is readily seen by considering the first component of the proximity hash, which is  $x + c$ , where  $x$  is the plaintext and  $c$  is a codeword. Two independent hashes of  $x$  will have first components  $x + c$  and  $x + c'$ , and therefore adding them will lead to  $c + c'$ , which is a codeword. However for the hash of an independently randomly generated  $y$ , the first component will be  $y + c''$ . If we add the first components we get  $x + y + c + c''$ , which is unlikely to be a codeword. Therefore there is an efficient distinguisher for the 2-POW distributions. Our construction is also not Oracle Simulation secure, because it reveals the syndrome of the plaintext with respect to the ECC used - this is more information than what the simulation world can provide. We leave it as an open problem to construct 2-POW and Oracle Simulation secure Relational Hashes for proximity.

## A Hardness Assumptions

We summarize the standard hardness assumptions used in this paper.

**Assumption 3 (DDH [DH76]).** *Assuming a generation algorithm  $\mathcal{G}$  that outputs a tuple  $(q, \mathbb{G}, \mathbf{g})$  such that  $\mathbb{G}$  is of prime order  $q$  and has generator  $\mathbf{g}$ , the DDH assumption asserts that it is computationally infeasible to distinguish*

between  $(\mathbf{g}, \mathbf{g}^a, \mathbf{g}^b, \mathbf{g}^c)$  and  $(\mathbf{g}, \mathbf{g}^a, \mathbf{g}^b, \mathbf{g}^{ab})$  for  $a, b, c \leftarrow \mathbb{Z}_q^*$ . More formally, for all PPT adversaries  $A$  there exists a negligible function  $\text{negl}()$  such that

$$\left| \Pr[(q, \mathbb{G}, \mathbf{g}) \leftarrow \mathcal{G}(1^\lambda); a, b, c \leftarrow \mathbb{Z}_q^* : A(\mathbf{g}, \mathbf{g}^a, \mathbf{g}^b, \mathbf{g}^c) = 1] - \Pr[(q, \mathbb{G}, \mathbf{g}) \leftarrow \mathcal{G}(1^\lambda); a, b \leftarrow \mathbb{Z}_q^* : A(\mathbf{g}, \mathbf{g}^a, \mathbf{g}^b, \mathbf{g}^{ab}) = 1] \right| < \text{negl}(\lambda).$$

**Assumption 4 (SXDH [BBS04]).** Consider a generation algorithm  $\mathcal{G}$  taking the security parameter as input, that outputs a tuple  $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \mathbf{g}_1, \mathbf{g}_2)$ , where  $\mathbb{G}_1, \mathbb{G}_2$  and  $\mathbb{G}_T$  are groups of prime order  $q$  with generators  $\mathbf{g}_1, \mathbf{g}_2$  and  $e(\mathbf{g}_1, \mathbf{g}_2)$  respectively and which allow an efficiently computable  $\mathbb{Z}_q^*$ -bilinear pairing map  $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ . The Symmetric eXternal decisional Diffie-Hellman (SXDH) assumption asserts that the Decisional Diffie-Hellman (DDH) problem is hard in both the groups  $\mathbb{G}_1$  and  $\mathbb{G}_2$ .

**Assumption 5 (Random Modular Subset Sum [Lyu05]).** Assuming a generation algorithm  $\mathcal{G}$  that outputs a tuple  $(n, q)$ , where  $q$  is prime, the Random Modular Subset Sum assumption asserts that given random elements  $\langle a_i \rangle_{i=1}^n$  from the group  $\mathbb{Z}_q$  and  $c = \sum_{i=1}^n \epsilon_i a_i$  for a random  $\epsilon \leftarrow \{0, 1\}^n$ , it is hard to output  $\eta \in \{0, 1\}^n$  such that

$$\sum_{i=1}^n \eta_i a_i = c \pmod{q}.$$

More formally, for all PPT  $A$ , there exists a negligible function  $\text{negl}()$  such that

$$\Pr \left[ \begin{array}{l} (n, q) \leftarrow \mathcal{G}(1^\lambda), \langle a_i \rangle_{i=1}^n \leftarrow \mathbb{Z}_q \\ \epsilon \leftarrow \{0, 1\}^n, c = \sum_{i=1}^n \epsilon_i a_i : \sum_{i=1}^n \eta_i a_i = c \pmod{q} \\ \eta \leftarrow A(\langle a_i \rangle_{i=1}^n, c) \end{array} \right] < \text{negl}(\lambda).$$

## References

BBS04. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)

BGG95. Bellare, M., Goldreich, O., Goldwasser, S.: Incremental cryptography and application to virus protection. In: 27th ACM STOC, pp. 45–56. ACM Press, May/June 1995

Boy04. Boyen, X.: Reusable cryptographic fuzzy extractors. In: Atluri, V., Pfitzmann, B., McDaniel, P. (eds.) ACM CCS 2004, pp. 82–91. ACM Press, New York (2004)

Can97. 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)

CMR98. Canetti, R., Micciancio, D., Reingold, O.: Perfectly one-way probabilistic hash functions (preliminary version). In: 30th ACM STOC, pp. 131–140. ACM Press, May 1998

DH76. Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Trans. Inf. Theor. **22**(6), 644–654 (1976)



- DRS04. Dodis, Y., Reyzin, L., Smith, A.: Fuzzy extractors: how to generate strong keys from biometrics and other noisy data. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 523–540. Springer, Heidelberg (2004)
- DS05. Dodis, Y., Smith, A.: Correcting errors without leaking partial information. In: Gabow, N.H., Fagin, R. (eds.) 37th ACM STOC, pp. 654–663. ACM Press, New York (2005)
- GGG+14. Goldwasser, S., Gordon, S.D., Goyal, V., Jain, A., Katz, J., Liu, F.-H., Sahai, A., Shi, E., Zhou, H.-S.: Multi-input functional encryption. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 578–602. Springer, Heidelberg (2014)
- GGJS13. Goldwasser, S., Goyal, V., Jain, A., Sahai, A.: Multi-input functional encryption. Cryptology ePrint Archive, Report 2013/727 (2013). <http://eprint.iacr.org/2013/727>
- GKL+13. Dov Gordon, S., Katz, J., Liu, F.-H., Shi, E., Zhou, H.-S.: Multi-input functional encryption. Cryptology ePrint Archive, Report 2013/774 (2013). <http://eprint.iacr.org/2013/774>
- JS02. Juels, A., Sudan, M.: A fuzzy vault scheme. Cryptology ePrint Archive, Report 2002/093 (2002). <http://eprint.iacr.org/2002/093>
- JW99. Juels, A., Wattenberg, M.: A fuzzy commitment scheme. In: ACM CCS 99, pp. 28–36. ACM Press (1999)
- KVM04. Kozat, S.S., Venkatesan, R., Mihçak, M.K.: Robust perceptual image hashing via matrix invariants. In: 2004 International Conference on Image Processing 2004, ICIP 2004, vol. 5, pp. 3443–3446. IEEE (2004)
- Lyu05. Lyubashevsky, V.: On random high density subset sums. Electronic Colloquium on Computational Complexity (ECCC) vol. 12, no. 7 (2005)
- MR14. Mandal, A., Roy, A.: Relational hash. Cryptology ePrint Archive, Report 2014/394 (2014). <http://eprint.iacr.org/2014/394>
- PR12. Pandey, O., Rouselakis, Y.: Property preserving symmetric encryption. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 375–391. Springer, Heidelberg (2012)
- RS09. 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)
- Sho97. 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)

# Explicit Non-malleable Codes Against Bit-Wise Tampering and Permutations

Shashank Agrawal<sup>1</sup>, Divya Gupta<sup>2</sup>, Hemanta K. Maji<sup>2,4</sup>, Omkant Pandey<sup>3</sup>,  
and Manoj Prabhakaran<sup>1</sup>(✉)

<sup>1</sup> University of Illinois Urbana-Champaign, Champaign, USA  
{sagrawl2,mmp}@illinois.edu

<sup>2</sup> Los Angeles and Center for Encrypted Functionalities,  
University of California, Los Angeles, USA  
{divyag,hmaji}@cs.ucla.edu

<sup>3</sup> University of California, Berkeley, USA  
omkant@gmail.com

<sup>4</sup> Purdue University, West Lafayette, USA

**Abstract.** A non-malleable code protects messages against various classes of tampering. Informally, a code is non-malleable if the message contained in a tampered codeword is either the original message, or a completely unrelated one. Although existence of such codes for various rich classes of tampering functions is known, *explicit* constructions exist only for “compartmentalized” tampering functions: i.e. the codeword is partitioned into *a priori fixed* blocks and each block can *only be tampered independently*. The prominent examples of this model are the family of bit-wise independent tampering functions and the split-state model.

In this paper, for the first time we construct explicit non-malleable codes against a natural class of non-compartmentalized tampering functions. We allow the tampering functions to *permute the bits* of the codeword and (optionally) perturb them by flipping or setting them to 0 or 1. We construct an explicit, efficient non-malleable code for arbitrarily long messages in this model (unconditionally).

We give an application of our construction to non-malleable commitments, as one of the first direct applications of non-malleable codes to computational cryptography. We show that non-malleable *string* commitments can be “entirely based on” non-malleable *bit* commitments.

---

S. Agrawal, D. Gupta, O. Pandey and M. Prabhakaran—Research supported in part by NSF grant 1228856.

D. Gupta, H.K. Maji and O. Pandey—Research supported in part from a DARPA/ONR PROCEED award, NSF Frontier Award 1413955, NSF grants 1228984, 1136174, 1118096. This material is based upon work supported by the Defense Advanced Research Projects Agency through the U.S. Office of Naval Research under Contract N00014-11-1-0389. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense, the National Science Foundation, or the U.S. Government.

## 1 Introduction

Non-malleability is a cryptographic notion [16] which requires that an encoding (encryption, commitment etc.) of a message cannot be used to create a valid encoding of a “related” message by a (computationally) crippled adversary. Non-malleable codes [18] is a special case of this idea: here, the encoding is in the form of a single string (rather than an interactive protocol), but the attacker is heavily crippled in that the tampering function it can apply on a codeword must belong to very simple classes (e.g., bit-wise functions). Non-malleable codes are required to be secure without relying on any computational restriction on the adversary, but instead – as is the case for other information-theoretic cryptographic primitives like secret-sharing and information-theoretically secure multi-party computation – relies on limitations of the adversary’s access<sup>1</sup> to information. The limited access is captured by limiting the class of tampering functions.

Ever since non-malleable codes were explicitly introduced, there has been a tremendous body of work on the topic [1, 2, 9–11, 17, 18]. Even when there are severe restrictions on the tampering class, it has been a challenge to obtain *explicit constructions* of non-malleable codes. All prior explicit constructions of non-malleable codes rely on the “compartmentalized” structure of the tampering function, i.e. the codeword is partitioned into *a priori fixed* blocks and *any tampering function should tamper with each block independently*. The prominent examples of this model are the family of bit-wise independent tampering functions and the split-state model.

In this work, we seek to build explicit non-malleable codes (with efficient encoding and decoding algorithms) for certain non-compartmentalized tampering functions. In particular, we consider bit-permutation attacks composed with arbitrary bit-wise functions. Our result could be seen as a first step towards one of the major problems in non-malleable codes: *to construct explicit codes that are non-malleable against low-complexity function classes like  $NC^0$  and  $AC^0$* . Indeed, the tampering functions we consider are very low-complexity circuits, with only unary gates and no fan-outs.

We point out that existential results and efficient randomized constructions are known for non-malleable codes against a very broad classes of tampering functions. However, given the theoretical importance of non-malleable codes (as evidenced by a deep and rich literature), explicit constructions are of great interest. Further, randomized constructions, even when they are efficient, are not suitable for some cryptographic applications. Indeed, in this work, we present a novel cryptographic application of non-malleable codes to non-malleable string commitments. Among other things, this is an instance of an application where neither party can be trusted to carry out the code construction honestly. We discuss this application further, later in this section.

*Construction Sketch.* Our non-malleable code construction consists of four steps, that are sketched below. We present a more detailed overview and further motivation behind these steps in the full version [3].

<sup>1</sup> Here access refers to both the ability to read and write the information in the system.

- We start with a large-alphabet randomized encoding which has a large enough distance and whose positions are  $t$ -wise independent for a large enough  $t$  (e.g., a “packed secret-sharing scheme” based on the Reed-Solomon code suffices), and make it resistant to permutations by incorporating into each character its position value; i.e., the character at the  $i^{\text{th}}$  position in a codeword  $x_i$  is re-encoded as  $\langle i, x_i \rangle$ , and allowed to occur at any position in the new codeword.
- The above code uses a large alphabet. It is concatenated with a binary inner code that is also resistant to permutations: each character in the outer code’s alphabet is mapped to a positive integer (in a certain range) and is encoded by a block of bits whose weight (number of positions with a 1) equals this integer. Note that a permutation may move bits across the different blocks. To resist such attacks, we keep the bits within each block randomly permuted, and also, ensure that a good fraction of the weights do not correspond to a valid block (achieved, for instance, by requiring that the weight of each block is a multiple of  $3^2$ ), so that blindly mixing together bits from different blocks has *some* probability of creating an invalid block. A careful combinatorial argument can be used to show that, despite dependencies among the blocks caused by a permutation attack, the probability of having all attacked blocks remaining valid decreases multiplicatively with the number of blocks being attacked thus. This, combined with the fact that the outer code has a large distance, ensures that the probability of creating a different valid codeword by this attack is negligible. However, we need to ensure not only that the attack has negligible chance of modifying one codeword into a different valid codeword, but also that the probability of creating an invalid codeword is (almost) independent of the actual message. Roughly, this is based on the large independence of the outer code.
- The resulting code is not necessarily resistant to attacks which can set/reset several bits. Towards achieving resistance to such attacks as well, we consider an intermediate 2-phase attack family: here the adversary can set/reset bits at *random positions*, learn which positions were subjected to this attack, and then specify a permutation attack.<sup>3</sup> To resist such attacks, we encode each bit in the above codeword into a bundle, using an additive secret-sharing. Then, if one or more bits in a bundle are set/reset, all the other bits in the bundle turn uniformly random. Hence, unless the adversary chooses to set/reset a very large number of positions (in which case almost every bundle is touched, and all information about the original message is lost), for every bit which has been set/reset, there will be several that are uniformly random. Now, even though the adversary can apply a permutation to rearrange these random bits (into as few bundles as possible), to ensure that there are only

<sup>2</sup> In our actual analysis, we also allow the attacker to flip any subset of bits. This prevents us from having valid weights to be 0 modulo 2, as flipping an even number of positions preserves this parity.

<sup>3</sup> In the actual analysis, we need to consider a slightly stronger 2-phase attack, in which the adversary can also learn the values of the bits in a small number of positions before specifying a permutation (and flipping a subset of bits).

a few bundles with a random bit, the adversary is forced to set/reset at most a few bundles' worth of bits. We note that our actual analysis follows a somewhat different argument, but fits the above intuition.

- Finally, the above code is modified as follows: a random permutation over the bits of the code is applied to a codeword; the permutation itself is encoded using a code of large distance, and appended to the above (permuted) codeword. Then it can be shown that a full-fledged attack (involving arbitrary set/reset and permutations) on such a codeword translates to a 2-phase attack of the above kind. Note that we do *not* rely on the permutation itself to be encoded in a non-malleable fashion. Indeed, the adversary can be allowed to learn and modify the encoded permutation *after* it has committed to the set/reset part of its attack on the rest of the codeword; in the 2-phase attack, this is modeled by the fact that the adversary can learn which positions in the codeword were set and reset, before deciding on the permutation attack.

As sketched above, the rate of our code is zero, as the codewords are polynomially longer than the message. However, the generic compiler provided in [4] when instantiated with our code yields a rate 1 code (i.e., the codewords are only marginally longer than the messages, with the increase being sub-linear in the length of the message).

*An Application.* One motivation behind the class of attacks considered in this work comes from the following intriguing question:

*Can non-malleable string-commitments be “entirely based” on non-malleable bit-commitments?*

To formalize this problem, we may consider an idealized model of bit commitments using physical tokens: to commit a bit to Bob, Alice can create a small physical token which has the bit “locked” inside (and later, she can send him a “key” to open the token). This completely hides the bit from Bob until Alice reveals it to him; on the other hand, Alice cannot change the bit inside the token once she has sent it to Bob. Further, this is a non-malleable bit commitment scheme, in that if Bob plays a man-in-the-middle adversary, and wants to send a commitment to Carol, he can only send the token from Alice as it is, or create a new token himself, independent of the bit committed to by Alice.

Now, we ask whether, in this model, one can make non-malleable string commitments (relying on no computational assumptions). *This is a question about non-malleable codes in disguise!* Indeed, if we required the commitment protocol to involve just a single round of sending a fixed number of tokens, then a commitment protocol is nothing but a non-malleable encoding of a string into bits, and the class of tampering functions we need to protect against is that of *bit-level permutations* and bit-wise set/reset.<sup>4</sup> Though we presented this

<sup>4</sup> For this application, bit-flipping need not be part of the admissible tampering functions. However, even if we restricted ourselves to this simpler class, our construction does not become significantly simpler. Indeed, handling permutations and set/reset present the biggest technical challenges in our construction. By handling bit-flipping as well, our tampering function family subsumes the bit-wise tampering function family.

string commitment scheme in an idealized setting involving tokens, it can be translated to a *reduction of non-malleable string commitment to CCA-secure bit commitment (as defined in [6])*.

As mentioned above, the non-malleable codes we build can withstand a slightly larger class of tampering attacks, which corresponds to the ability of the adversary to apply any set of functions from  $\{0,1\}$  to  $\{0,1\}$  to the bits stored in the tokens (i.e., set, reset, flip or keep), before applying the permutation attack. As such, in the above application, we do not actually require the bit commitment scheme to be CCA secure. We also present a variant of the above construction to illustrate this, which we base on a specific (non-standard) assumption on a PRG.

This application also illustrates why explicit constructions can be of importance to cryptographic constructions. While there indeed is an efficient randomized construction of non-malleable codes that can resist permutations [20], it will not be suitable in this case, because neither the sender nor the receiver in a commitment scheme can be trusted to pick the code honestly (Bob could play either role), and non-malleable codes are not guaranteed to stay non-malleable if the description of the code itself can be tampered with. While one may address this issue using a more complex protocol which securely samples a non-malleable code using (malleable) string commitments, this undermines the simplicity of our protocol (which involves no interaction beyond carrying out the bit commitments) and introduces more rounds of interaction.

## 1.1 Prior Work

Cramer et al. [14] introduced the notion of arithmetic manipulation detection codes, which is a special case of non-malleable codes; AMD codes with optimal parameters have been recently provided by [15]. Dziembowski et al. motivated and formalized the more general notion of non-malleable codes in [18]. They showed existence of a constant rate non-malleable code against the class of all bit-wise independent tampering functions. Existence of rate 1 non-malleable codes against various classes of tampering functions is known. For example, existence of such codes with rate  $(1 - \alpha)$  was shown against any tampering function family of size  $2^{2^{\alpha n}}$ ; but this scheme has inefficient encoding and decoding [10]. For tampering functions of size  $2^{\text{poly}(n)}$ , rate 1 codes (with efficient encoding and decoding) exist with overwhelming probability [20].

On the other hand, explicit constructions of non-malleable codes have remained elusive, except for some well structured tampering function classes. Recently, an explicit rate 1 code for the class of bit-wise independent tampering function was proposed by [11]. Note that a tampering function in this class tampers each bit independently. For a more general compartmentalized model of tampering, in which the codeword is partitioned into separate blocks and each block can be tampered arbitrarily but independently, an encoding scheme was proposed in [12]. In the most general compartmentalized model of tampering, where there are only two compartments (known as the split-state model), an

explicit encoding scheme for bits was proposed by [17]. Recently, in a breakthrough result, an explicit scheme (of rate 0) was proposed for arbitrary length messages by [2]. Subsequently, a constant rate construction for 10 states was provided in [9] and, building on that, a constant rate construction for the split state was proposed in [1].

*Codes Under Computational Assumptions.* The idea of improving the rate of error-correcting codes by considering computationally limited channels has been explored in a large body of work [8, 26, 27, 30, 32, 34]. In the setting of non-malleable codes as well, constructions based on computational assumptions have been explored, for example [19, 31].

*Non-malleable Commitments.* There is extensive literature on non-malleable commitments starting from the work of Dolev, Dwork and Naor [16] leading to recent constant-round constructions based on one-way functions [24, 25, 29]. Our application of nonmalleable codes to non-malleable commitments is similar in spirit to the work of Meyers and Shelat [33] on the completeness of bit encryption. Concurrently, and independently of our work, Chandran et al. [7] relate non-malleable commitments to a new notion of non-malleable codes, called *blockwise non-malleable codes*.

*Application of Non-malleable Codes to Cryptographic Constructions.* AMD codes have found several applications in information-theoretic cryptography, for secret-sharing, randomness extraction and secure multi-party computation (e.g., [5, 14, 21, 23]). However, the more general notion of non-malleable codes have had few other applications, outside of the direct application to protecting the contents of device memories against tampering attacks. Our application to non-malleable commitment is one of the few instances where non-malleable codes have found an application in a natural cryptographic problem that is not information-theoretic in nature. A similar application appears in the recent independent work of Coretti et al. [13].

## 1.2 Our Contribution

The class of tampering functions which permutes the bits of the codeword is represented by  $\mathcal{S}_N$ . The set of all tampering functions which allow the adversary to tamper a *bit* by passing it through a channel is denoted by  $\mathcal{F}_{\{0,1\}}$ ; this includes forwarding a bit unchanged, toggling it, setting it to 1, or resetting it to 0. The class of tampering functions which allows the adversary to do apply both: i.e., tamper bits followed by permuting them is represented by:  $\mathcal{F}_{\{0,1\}} \circ \mathcal{S}_N$ . Our main result is a non-malleable code against this class of tampering functions.

**Theorem 1 (Non-malleable Code).** *There exists an explicit and efficient non-malleable code for multi-bit messages against the tampering class  $\mathcal{F}_{\{0,1\}} \circ \mathcal{S}_N$ .*

Our main non-malleable encoding which is robust to  $\mathcal{F}_{\{0,1\}} \circ \mathcal{S}_N$  relies on a basic encoding scheme. The basic encoding scheme is robust to a weaker class of

tampering functions, but it provides slightly stronger security guarantees. More specifically, the basic scheme protects only against  $\tilde{\mathcal{F}}_{\{0,1\}} \circ \mathcal{S}_N$  class, where  $\tilde{\mathcal{F}}_{\{0,1\}}$  is the class of functions which either forward a bit unchanged or toggle it but do not set or reset it. The stronger security guarantee given by basic scheme is that it allows the adversary to *adaptively* choose the tampering function  $\tilde{\mathcal{F}}_{\{0,1\}} \circ \mathcal{S}_N$ . The adversary first specifies  $n_0$  and  $n_1$ , i.e. number of indices it wants to reset to 0 and number of indices it wants to set to 1. It is provided a random subset of indices of size  $n_0$  which is all reset to 0; and a (disjoint) random subset of indices of size  $n_1$  which is all set to 1. Given this information, the adversary can adaptively choose the tampering function in  $\tilde{\mathcal{F}}_{\{0,1\}} \circ \mathcal{S}_N$ . Even given this additional power, the adversary cannot tamper the codeword to produce related messages (except with negligible probability).

We present the basic encoding scheme and prove its non-malleability in Sect. 5. Theorem 1 is proved via a reduction to the basic scheme. This proof is provided in the full version [3].

*Non-malleable Commitments.* As noted earlier, we consider the question of constructing simple string non-malleable commitments from bit non-malleable commitments. For example, if we simply encode the given string and commit to each of its bit using the given non-malleable bit-commitment, does it result in a secure non-malleable string commitment schemes? What are the conditions we need on the underlying bit commitment scheme?

For this question, we are interested in a really simple reduction, as opposed to, e.g. “merely” black-box reductions. Indeed, if we ask for a merely black-box construction we can invoke known (but complex) reductions: a bit commitment scheme (even malleable) implies a one-way function, which in turn imply string commitments in a black box way [25]. Such reductions are not, what we call *totally* black-box. For example, if we switch to a model where we are given the bit-commitment scheme as a *functionality* which can be executed only a bounded number of times, such as a one-time program [22] or a hardware token [28], then we do not necessarily have standard one-way functions. Therefore, the reduction should avoid assuming additional complexity assumptions such as OWFs or signatures. In fact, for this reason, the reduction should also not rely on using tags and “tag-based” non-malleability [35]. It should work with standard *non-tag-based* non-malleable bit-commitments.

Our reduction actually satisfies these conditions provided that we start with a (non-tag-based) CCA-secure bit-commitment scheme [6]. We show that (perhaps the simplest construction where) if we just commit to each bit of a random codeword of the given string works! This gives us the following theorem:

**Theorem 2 (CCA Bit-Commitment to Non-malleable String Commitment).** *There exists a simple and efficient black-box compiler which, when provided with:*

- A non-malleable encoding robust to  $\mathcal{F}_{\{0,1\}} \circ \mathcal{S}_N$ , and
- A  $r$ -round (possibly **non-tag-based**) CCA-secure bit-commitment scheme yields a  $r$ -round non-malleable string-commitment scheme.



We note that the theorem statement is **unconditional**: it does not assume any computational assumption beyond the given non-malleable bit-commitment. In particular, the theorem holds even if the bit-commitment is implemented in a model which does not necessarily imply OWFs. Furthermore, in our full version [3], we prove that in fact, the theorem holds even if the bit-commitment is not CCA-secure but only satisfies a much weaker notion which we call *bounded-parallel* security.

Finally, we show the power of our non-malleable codes by demonstrating that even if we start with a seemingly much weaker scheme which allows partial malleability, e.g., it may allow the MIM to toggle the committed bit, our non-malleable codes can “boost” it to full-fledged malleability. See [3] for details.

## 2 Preliminaries

We denote the set  $\{1, \dots, n\}$  by  $[n]$ . If  $a \in [b - \varepsilon, b + \varepsilon]$ , then we represent it as:  $a = b \pm \varepsilon$ .

Probability distributions are represented by bold capital alphabets, for example  $\mathbf{X}$ . The distribution  $\mathbf{U}_S$  represents a uniform distribution over the set  $S$ . Given a distribution  $\mathbf{X}$ ,  $x \sim \mathbf{X}$  represents that  $x$  is sampled according to the distribution  $\mathbf{X}$ . And, for a set  $S$ ,  $x \stackrel{\$}{\leftarrow} S$  is equivalent to  $x \sim \mathbf{U}_S$ . For a joint variable  $\mathbf{X} = (\mathbf{X}_1, \dots, \mathbf{X}_n)$  and  $S = \{i_1, \dots, i_{|S|}\} \subseteq [n]$ , we define the random variable  $\mathbf{X}_S = (\mathbf{X}_{i_1}, \dots, \mathbf{X}_{i_{|S|}})$ . We also define hamming distance between two samples  $u = (u_1, u_2, \dots, u_n)$  and  $v = (v_1, v_2, \dots, v_n)$  drawn from the distribution  $\mathbf{X}$  as the number of indices at which they differ, i.e.,  $\text{HD}(u, v) = |\{i \in [n] \mid u_i \neq v_i\}|$ . For a function  $f(\cdot)$ , the random variable  $\mathbf{Y} = f(\mathbf{X})$  represents the following distribution: Sample  $x \sim \mathbf{X}$ ; and output  $f(x)$ . Further,  $f(x_{[n]})$  represents the vector  $f(x_1) \dots f(x_n)$ .

The statistical distance between two distributions  $\mathbf{S}$  and  $\mathbf{T}$  over a finite sample space  $I$  is defined as:  $\text{SD}(\mathbf{S}, \mathbf{T}) := \frac{1}{2} \sum_{i \in I} |\Pr_{x \sim \mathbf{S}}[x = i] - \Pr_{x \sim \mathbf{T}}[x = i]|$ .

### 2.1 Classes of Tampering Functions

We shall consider the following set of tampering functions.

1. Family of Permutations. Let  $\mathcal{S}_N$  denote the set of all permutations  $\pi : [N] \rightarrow [N]$ . Given an input codeword  $x_{[N]} \in \{0, 1\}^N$ , tampering with function  $\pi \in \mathcal{S}_N$  yields the following codeword:  $x_{\pi^{-1}(1)} \dots x_{\pi^{-1}(N)} =: x_{\pi^{-1}([N])}$ .
2. Family of Fundamental Channels. The set of fundamental channels over  $\{0, 1\}$ , represented as  $\mathcal{F}_{\{0,1\}}$ , contains the following binary channels  $f$ : (a)  $f(x) = x$ , (b)  $f(x) = 1 \oplus x$ , (c)  $f(x) = 0$ , or (d)  $f(x) = 1$ . These channels are, respectively, called *forward*, *toggle*, *reset* and *set* functions.
3. Family of Sensitive Channels. The set of *sensitive* functions  $\tilde{\mathcal{F}}_{\{0,1\}}$  contains only forward and toggle channels. In other words, tampering involves XOR-ing an  $N$ -bit input string with a fixed  $N$ -bit string.

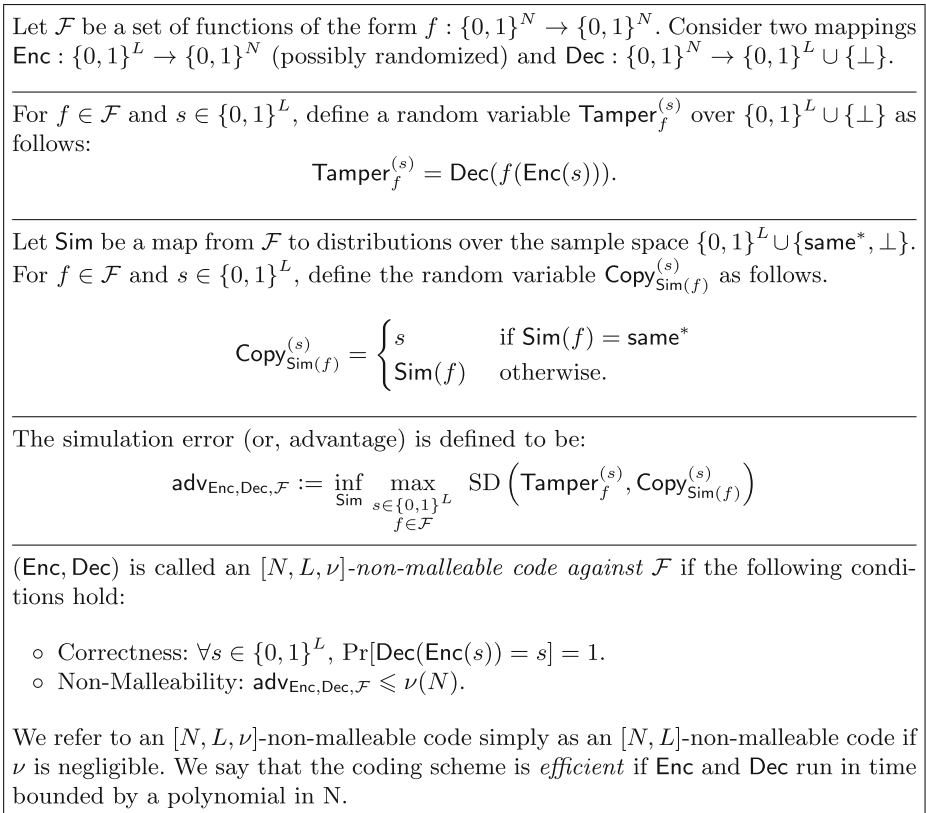
We can define more complex tampering function classes by composition of these function classes. For example, composition of  $\mathcal{S}_N$  with  $\mathcal{F}_{\{0,1\}}$  yields the following class of tampering functions. For any  $\pi \in \mathcal{S}_N$  and  $f_1, \dots, f_N \in \mathcal{F}_{\{0,1\}}$ , it transforms a codeword  $x_{[N]}$  into  $f_1(x_{\pi^{-1}(1)}) \dots f_N(x_{\pi^{-1}(N)}) =: f_{1, \dots, N}(x_{\pi^{-1}([N])})$ . This class is represented by:  $\mathcal{F}_{\{0,1\}} \circ \mathcal{S}_N$ .

### 2.2 Non-malleable Codes

We define non-malleable codes formally in Fig. 1. Our main result provides an efficient non-malleable code against the tampering class  $\mathcal{F}_{\{0,1\}} \circ \mathcal{S}_N$ .

## 3 Building Blocks

In this section, we define various types of secret-sharing schemes relevant to our construction. First we present the basic notion.



**Fig. 1.** Definition of non-malleable codes

**Definition 1 (Secret-Sharing Scheme (SSS)).** Let  $\mathbf{S} = (\mathbf{X}_0, \mathbf{X}_1, \dots, \mathbf{X}_M)$  be a joint distribution over  $\Lambda^L \times \Sigma^M$ , such that the support of  $\mathbf{X}_0$  is all of  $\Lambda^L$ . (The random variable  $\mathbf{X}_0$  represents the secret being shared and  $\mathbf{X}_i$  for  $i \in [m]$  represents the  $i$ -th share.)

We say that  $\mathbf{S}$  is an  $[M, L, T, D]_{\Lambda, \Sigma}$  secret-sharing scheme if the following conditions hold:

1.  $T$ -privacy:  $\forall s, s' \in \Lambda^L, \forall J \subseteq [M]$  such that  $|J| \leq T$ , we have

$$\text{SD}((\mathbf{X}_J | \mathbf{X}_0 = s), (\mathbf{X}_J | \mathbf{X}_0 = s')) = 0.$$

2.  $D$ -distance: For any two distinct  $c, c' \in \text{Supp}(\mathbf{X}_{[M]})$ , the hamming distance between them,  $\text{HD}(c, c')$ , is at least  $D$ , where  $\text{Supp}(\mathbf{X}_{[M]})$  denotes the support of distribution  $\mathbf{X}_{[M]}$ .
3. Reconstruction: For any  $s, s' \in \Lambda^L$  such that  $s \neq s'$ , we have

$$\text{SD}((\mathbf{X}_{[M]} | \mathbf{X}_0 = s), (\mathbf{X}_{[M]} | \mathbf{X}_0 = s')) = 1.$$

In the remainder of the paper, by an SSS scheme, we shall implicitly refer to a family of SSS schemes indexed by  $M$ , i.e.,  $[M, L(M), T(M), D(M)]$ -SSS schemes for each positive integer  $M$ . We define the *rate* of such a scheme to be  $\lim_{M \rightarrow \infty} \frac{L(M)}{M}$ . We will be interested in *efficient* SSS schemes. For this, we define two algorithms for encoding and decoding as follows:

- $\text{Enc}_{\text{SSS}}(s)$ : This is a randomized algorithm that takes  $s \in \Lambda^L$  as input and outputs a sample from the distribution  $(\mathbf{X}_{[M]} | \mathbf{X}_0 = s)$ .
- $\text{Dec}_{\text{SSS}}(\tilde{c})$ : This algorithm takes a  $\tilde{c} \in \Sigma^M$  as input, and outputs a secret  $s \in \Lambda^L$  such that  $\tilde{c} \in \text{Supp}(\mathbf{X}_{[M]} | \mathbf{X}_0 = s)$ . If such a secret does not exist, it outputs  $\perp$ .

Note that the uniqueness of the output of algorithm  $\text{Dec}_{\text{SSS}}$  is guaranteed by the reconstruction property. An SSS scheme is said to be *efficient* if the two algorithms defined above run in time bounded by a polynomial in  $M$ .

We can instantiate a secret-sharing scheme with all the properties described above using Reed-Solomon codes. Let  $n, k$  and  $\ell$  be any three positive integers such that  $n \geq k \geq \ell$ . Let  $\mathbb{F}$  be a finite field of size at least  $n + \ell$ . Let  $\{u_{-\ell}, \dots, u_{-1}, u_1, \dots, u_n\} \subseteq \mathbb{F}$ . The secret-sharing of a message  $(s_1, \dots, s_\ell) \in \mathbb{F}^\ell$  is done by choosing a random polynomial  $p(\cdot)$  of degree  $< k$  conditioned on  $(p(u_{-1}), \dots, p(u_{-\ell})) = (s_1, \dots, s_\ell)$ . The shares  $\{y_1, \dots, y_n\}$  are evaluations of  $p(\cdot)$  at  $\{u_1, \dots, u_n\}$  respectively. It is known that efficient encoding and decoding procedures exist using Lagrange interpolation. Further, this encoding has privacy  $k - \ell$  and distance  $n - k + 1$ .

**SSS with Independence.** A secret-sharing scheme with independence, or  $i$ -SSS in short, is defined in the same way as an SSS, except that instead of privacy, it has a stronger independence property:

- $T$ -independence:  $\forall s \in \Lambda^L, \forall J \subseteq [M]$  such that  $|J| \leq T$ , we have

$$\text{SD}((\mathbf{X}_J | \mathbf{X}_0 = s), \mathbf{U}_{\Sigma^{|J|}}) = 0.$$

In simple words, any subset of at most  $T$  shares are uniformly distributed over the corresponding codeword space (whereas privacy only guarantees that these shares have a distribution independent of the secret being shared). The encoding and decoding algorithms for i-SSS are denoted by  $\text{Enc}_{i\text{-SSS}}$  and  $\text{Dec}_{i\text{-SSS}}$  respectively. Reed-Solomon codes, as defined above, actually give independence and not just privacy.

**Augmented SSS.** A secret-sharing scheme (with or without independence) can be augmented to have each share also specify *which* share it is – the first, the second, etc. More formally, if we have an  $[M, L, T, D]_{A, \Sigma}$ -SSS (or i-SSS) with  $(\text{Enc}_{\text{SSS}}, \text{Dec}_{\text{SSS}})$  algorithms, we define algorithms  $\text{Enc}_{a\text{-SSS}}$  and  $\text{Dec}_{a\text{-SSS}}$  for the augmented secret-sharing scheme over  $\Lambda^L \times ([M] \times \Sigma)^M$  as follows:

- $\text{Enc}_{a\text{-SSS}}(s)$ : Run  $\text{Enc}_{\text{SSS}}(s)$  to obtain  $c_1, \dots, c_M$ . Output  $(1, c_1), \dots, (M, c_M)$ .
- $\text{Dec}_{a\text{-SSS}}(\tilde{c})$ : Let  $\tilde{c} = ((i_1, \tilde{c}_1), \dots, (i_M, \tilde{c}_M))$ . Sort the shares according to the first element in each tuple, check that each index occurs exactly once, and then output  $\text{Dec}_{\text{SSS}}((\tilde{c}_1, \dots, \tilde{c}_M))$ .

It is easy to observe that a-SSS defined in this way has  $T$ -privacy and  $D$ -distance.

**Additive Secret Sharing.** An  $[M, L]_{A, \Sigma}$  additive secret sharing scheme, referred to simply as **add**, is an  $[M, L, T, D]_{A, \Sigma}$ -i-SSS with  $T = M - 1$  and  $D = 1$ . One can instantiate such sharing schemes over any Abelian group  $(G, +)$ . The joint distribution  $(\mathbf{X}_0, \dots, \mathbf{X}_M)$  is defined via the following sampling procedure: pick  $x_1, \dots, x_M \stackrel{\$}{\leftarrow} G$  and set  $x_0 = \sum_{i \in [M]} x_i$ . It is easy to see that there exist efficient encoding and decoding algorithms, which are denoted by  $\text{Enc}_{\text{add}}$  and  $\text{Dec}_{\text{add}}$  respectively.

**Balanced Unary Encoding.** This scheme is parameterized by a message space  $F$  and a positive integer  $p$ . Let  $\pi: F \rightarrow \mathbb{Z}_{|F|}$  be a bijection and  $m = 3p|F| + 1$ . Then, given a message  $s \in F$ , the encoding  $\text{Enc}_{\text{unary}}(s)$  is performed as follows: Sample a random set  $S$  of  $[m]$  of weight  $\lceil m/3 \rceil + p\pi(s)$ . The codeword is defined to be the characteristic vector of set  $S$ . Note that this scheme has efficient encoding and decoding algorithms,  $\text{Enc}_{\text{unary}}$  and  $\text{Dec}_{\text{unary}}$ , respectively. For any  $s \in F$  and any set  $S$  used for encoding  $s$ , the total weight of the final shares lie in  $\lceil m/3, 2m/3 \rceil$ . Hence, the name *balanced unary* secret sharing scheme.

We now define how to combine two or more schemes.

**Definition 2 (Concatenating Sharing Schemes).** Consider two secret sharing schemes, an outer scheme  $\mathbf{S}^{(\text{out})} = (\mathbf{X}_0^{(\text{out})}, \mathbf{X}_1^{(\text{out})}, \dots, \mathbf{X}_n^{(\text{out})})$  over  $\Lambda^L \times \Sigma^N$  and an inner scheme  $\mathbf{S}^{(\text{in})} = (\mathbf{X}_0^{(\text{in})}, \mathbf{X}_1^{(\text{in})}, \dots, \mathbf{X}_m^{(\text{in})})$  over  $\Sigma \times \Gamma^M$ . The concatenation of the outer scheme with the inner scheme is defined as the joint distribution  $\mathbf{S}^{(\text{concat})} = (\mathbf{X}_0^{(\text{concat})}, \mathbf{X}_1^{(\text{concat})}, \dots, \mathbf{X}_{NM}^{(\text{concat})})$  over  $\Lambda^L \times \Gamma^{MN}$ . Given a secret  $s \in \Lambda^L$ , sample  $\mathbf{x}_{[NM]}^{(\text{concat})} \sim (\mathbf{X}_{[NM]}^{(\text{concat})} \mid \mathbf{X}_0^{(\text{concat})} = s)$  as follows: first sample  $\mathbf{x}_{[N]}^{(\text{out})} \sim (\mathbf{X}_{[N]}^{(\text{out})} \mid \mathbf{X}_0^{(\text{out})} = s)$ , and then for each  $i \in [N]$ , sample  $\mathbf{x}_{(i-1)m+[M]}^{(\text{concat})} \sim (\mathbf{X}_{[M]}^{(\text{in})} \mid \mathbf{X}_0^{(\text{in})} = \mathbf{x}_i^{(\text{out})})$ . We use  $\mathbf{S}^{(\text{concat})} = \mathbf{S}^{(\text{out})} \circ \mathbf{S}^{(\text{in})}$  to represent the concatenation of  $\mathbf{S}^{(\text{out})}$  with  $\mathbf{S}^{(\text{in})}$ .

If the encoding and decoding procedures for outer and inner schemes are  $(\text{Enc}^{(\text{out})}, \text{Dec}^{(\text{out})})$  and  $(\text{Enc}^{(\text{in})}, \text{Dec}^{(\text{in})})$  respectively, then the corresponding procedures for the concatenated scheme are denoted by  $(\text{Enc}^{(\text{out})} \circ \text{Enc}^{(\text{in})}, \text{Dec}^{(\text{out})} \circ \text{Dec}^{(\text{in})})$ . Note that the final encoding and decoding procedures are efficient if the corresponding procedures are efficient for inner and outer schemes.

Moreover, we emphasize that we do not focus on error correcting codes. In particular, if any of inner or outer decoding procedures fails, we output  $\perp$  as the decoding of the overall code.

## 4 Our Non-malleable Encoding Scheme

In this section, we describe our non-malleable encoding scheme against the class of tampering functions  $\mathcal{F}_{\{0,1\}} \circ \mathcal{S}_N$ . It proceeds in following two steps.

1. **Basic Encoding Scheme.** Though this scheme will offer non-malleability against a weaker class of tampering functions, it will offer stronger guarantees beyond standard non-malleability. We refer to this as “2-Phase Non-malleability” property. The security proof of our main construction described below reduces to the 2-phase non-malleability of our basic scheme.

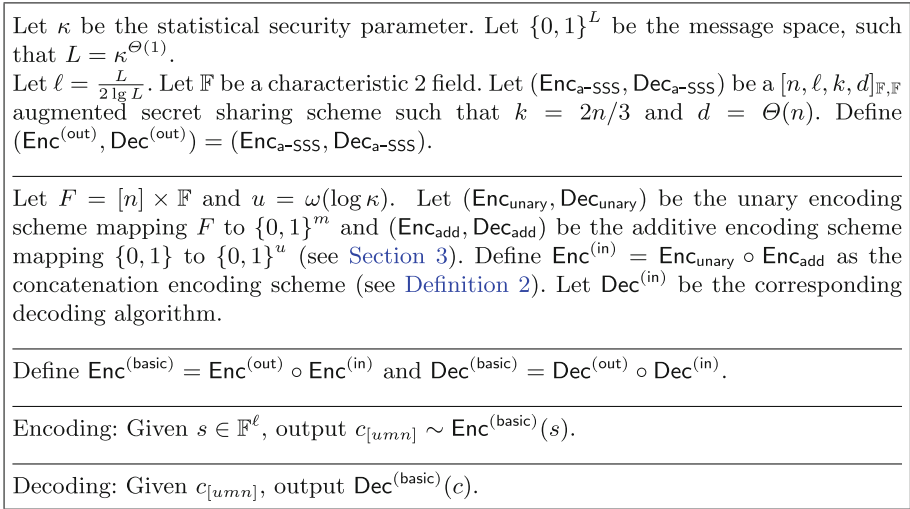
The basic encoding scheme is described formally in Fig. 2. As a high level, our encoding scheme is a concatenation code (see Definition 2) which does the following: Given a message  $s$ , it samples an outer code according to augmented Reed-Solomon code based secret sharing. Then for each outer code element, it samples an inner codeword which itself is a concatenation code using balanced unary secret sharing scheme and additive sharing scheme.

2. **Main Construction.** Our main non-malleable coding scheme resistant against the class of attacks  $\mathcal{F}_{\{0,1\}} \circ \mathcal{S}_N$  is built on top of the basic coding scheme. In order to encode a message  $s$ , we choose a random permutation  $\sigma$ . The codeword consists of two parts: the first part is the basic encoding of  $s$  with  $\sigma$  applied on it, and the second part is a secret sharing of  $\sigma$  with high distance and independence encoding. Intuitively, applying a random permutation ensures that setting/resetting bits in the main codeword results in random positions being modified in the basic codeword, exactly the kind of attack basic code can handle. The scheme is described formally in Fig. 3.

In the following section, we first describe the 2-phase security of basic encoding scheme and then our main construction.

## 5 Basic Encoding Scheme and 2-Phase Non-malleability

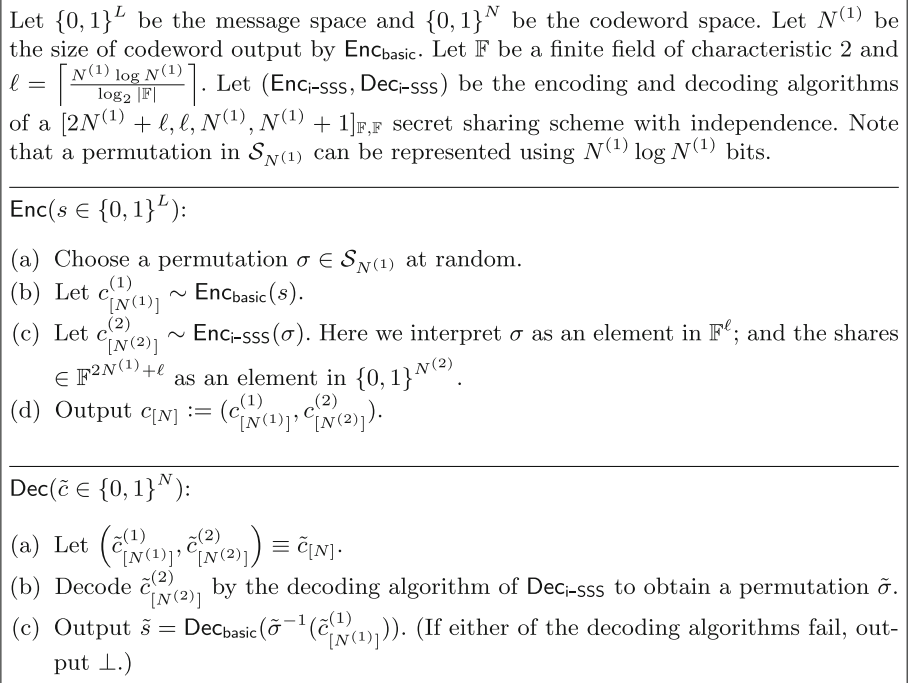
2-Phase Non-Malleability is a two-phase attack experiment where the adversary gets additional information about the codeword in the first phase before it gets to choose the tampering function in second phase.



**Fig. 2.** Basic non-malleable Code achieving 2-phase non-malleability.

1. In the first phase the adversary sends message  $s$  and  $n_0, n_1, n_p \in [N]$  such that  $n_0 + n_1 + n_p \leq N$  and  $n_p \leq \log^2 \kappa$ . Here  $n_0$  and  $n_1$  refer to the number of bits in the tampered codeword that will be set to 0 and 1, respectively.  $n_p$  refers to the number of bits of the original codeword which will be revealed to the adversary before he chooses the final tampering function.
2. The challenger picks an index set  $I \xleftarrow{\$} \binom{[N]}{n_0 + n_1 + n_p}$  and randomly partitions  $I$  into  $I_0, I_1$  and  $I_p$  of size  $n_0, n_1$  and  $n_p$ , respectively. It picks  $c = \text{Enc}(s)$ . Then it sends  $(I_0, I_1, I_p, c_{I_p})$  to the adversary. Here  $I_0$  and  $I_1$  refer to the indices which will be set to 0 and 1, respectively.  $I_p$  refers to the indices of  $c$  which are revealed in  $c_{I_p}$ .
3. In the second phase, the adversary sends a tampering function  $f \in \tilde{\mathcal{F}}_{\{0,1\}} \circ \mathcal{S}_N$ , using the information obtained from first phase.
4. The tampered codeword is created by setting the bits at positions  $I_0$  and  $I_1$  to 0 and 1, respectively, to obtain  $c'$ . Then,  $f$  is applied to  $c'$ .

Observe that in the above experiment the adversary can specify  $n_0, n_1, n_p$  and a function  $\text{map}$  in advance, and then the challenger can carry out the entire experiment on its own. The function  $\text{map}$  takes three disjoint subsets of indices  $I_0, I_1, I_p \subseteq [N]$  of size  $n_0, n_1$ , and  $n_p$  respectively, and a bit string of length  $|I_p| \leq \log^2 \kappa$ , and outputs a function  $f \in \tilde{\mathcal{F}}_{\{0,1\}} \circ \mathcal{S}_N$ . Let  $\mathcal{F}^*$  be a tampering function family where an  $f^* \in \mathcal{F}^*$  is specified by  $n_0, n_1, n_p$  and  $\text{map}$ , and tampers the codeword as the challenger does. Our security requirement can now be simply stated as non-malleability against the tampering class  $\mathcal{F}^*$ . The only issue is that the functions in this family are randomized and we have defined non-malleability w.r.t. deterministic functions. However, we could just define the randomness of



**Fig. 3.** Main non-malleable code

$\text{Tamper}_f^{(s)}$  in Fig. 1 to be over the coin tosses of not only the encoding function but also over the tampering function, and take care of this problem.

**Theorem 3 (2-Phase Non-malleability).** *There exists an explicit and efficient non-malleable code for multi-bit messages against the tampering class  $\mathcal{F}^*$ .*

### 5.1 Proof of Theorem 3

In this section we show that the basic encoding scheme described in Fig. 2 satisfies Theorem 3.

**Useful Terminology.** We will call the field elements of augmented secret sharing scheme  $\text{Enc}^{(\text{out})}$  as “elements”. The encoding of each element via inner encoding  $\text{Enc}^{(\text{in})}$  will be called a “block” or “inner codeword”. We shall visualize our inner code blocks as a two-dimensional objects, where each “column” represents the additive secret shares of a bit in the unary encoding scheme.

Below we crucially rely on the notion of “equivalence of codes” and “dirty inner codewords” or “dirty blocks” as defined below.

**Equivalence of Codewords for our Scheme.** Here we describe equivalence of codes for  $\text{Enc}^{(\text{in})}$  and  $\text{Enc}^{(\text{basic})}$ .

**1. Inner Codes.** Two inner codewords,  $g_{[um]}^{(in)}$  and  $h_{[um]}^{(in)}$  are equivalent codes if they encode the same message according to the inner encoding scheme  $\text{Enc}^{(in)}$ .

**2. Non-Malleable Codes.** Two codewords  $g_{[umn]}^{(basic)}$  and  $h_{[umn]}^{(basic)}$  are equivalent codes if their outer codes under  $\text{Enc}^{(out)}$  are identical<sup>5</sup>. That is, following holds. For all  $i \in [n]$ , define  $g_i^{(in)} = g_{(i-1)um+[um]}^{(basic)}$  and  $h_i^{(in)} = h_{(i-1)um+[um]}^{(basic)}$ . Then, there exists a  $\pi : [n] \rightarrow [n]$  such that for all  $i \in [n]$ ,  $g_i^{(in)} \cong h_{\pi(i)}^{(in)}$ .

**Criteria for Valid Inner Codeword or Block.** For a block to be valid, the corresponding unary code should have parity  $0 \pmod 3$ .

**Classification of Blocks.** We classify the inner codewords or blocks in following three categories.

**Fixed Blocks.** We say that a block is *completely fixed* if all its bits are obtained from bits in  $I_0 \cup I_1$ . That is, the tampering function explicitly writes down the whole block using bits from  $I_0 \cup I_1$ . Note that some of these bits might have been toggled.

**Copied Blocks.** We say that a block is *completely copied* if it is generated by copying one whole block in  $c$  and (possibly) performing column preserving permutations to the block. Also, even number of toggles might have been applied to any of these columns. Note that copied blocks are valid with probability 1.

**Dirty Blocks.** We say that an inner codeword or a block in  $\tilde{c}$  is dirty if it is neither fixed nor copied. In other words, one of the following holds:

1. The block receives its bits partially from one block of  $c$ . To clarify, it can be the case that it receives bits from more than one blocks, or it receives bits from one block but some of its bits are obtained from  $I_0 \cup I_1$ .
2. (The block receives all its bits from one block of  $c$  but) The permutation within the block is not column preserving. That is, there exists a column which receives bits from more than one columns of the same inner codeword.
3. (The block receives all its bits from one block and the permutation is column preserving but) There exists a column which has odd number of toggle gates.

We show that a dirty block fails to be a valid block (according to inner encoding scheme) with at least a constant probability (see [3] for details).

We denote the number of dirty blocks by  $n_{dirty}$ , fixed blocks by  $n_{fixed}$ , and copied blocks by  $n_{copy}$ . Note that  $n = n_{dirty} + n_{fixed} + n_{copy}$ . Finally, we define peeked blocks as follows:

**Peeked Blocks.** We say that a block is a peeked if one of its bits has been copied from  $I_p$ . Let  $n_{peek}$  be the number of such blocks. Note that  $n_{peek} \leq n_p$ .

---

<sup>5</sup> Note that we only insist that  $g_{[umn]}^{(basic)}$  and  $h_{[umn]}^{(basic)}$  not only encode the same message  $s$  but also every outer codeword element is identical. Note that we allow for permutation of outer code elements.



### 5.2 Key Steps of the Proof

In this section, we give a high level proof idea by doing a case analysis on  $n_0 + n_1$  and explaining how **Sim** is determined on each case (depending on **map**). The threshold value  $\log^{10} \kappa$  chosen below for analysis is arbitrary; any suitable  $\text{poly log } \kappa$  will suffice. For a formal proof of non-malleability, please refer to the full version [3].

**Case 1.**  $\log^6 \kappa \leq n_0 + n_1 \leq N - um \log^3 \kappa$ .

In this case, **Sim** outputs  $\perp$  with probability 1. This incurs  $\text{negl}(\kappa)$  simulation error. This is because, as we can show,  $n_{\text{dirty}} \geq \log^3 \kappa$  with probability  $1 - \text{negl}(\kappa)$ . Further, as described below, if  $n_{\text{dirty}}$  is large, then the probability that **Tamper**<sup>(s)</sup> outputs something other than  $\perp$  is negligible.

**Case 2.**  $n_0 + n_1 \leq \log^6 \kappa$ .

Note that **Sim** has  $n_0, n_1, n_p$  and **map**. It begins by sampling  $I_0, I_1, I_p, c_{I_p}$  and  $f \in \mathcal{F}$  according to **map** as in real execution. Based on these we have the following cases on  $n_{\text{dirty}}$ , the number of dirty blocks.

**Case 2.1.**  $n_{\text{dirty}} \geq \log^3 \kappa$ .

In this case, **Sim** outputs  $\perp$  with probability 1. Again, as explained below, this incurs a negligible simulation error.

**Case 2.2.**  $n_{\text{dirty}} < \log^3 \kappa$ .

In this case,  $n_{\text{copy}} = n - n_{\text{dirty}} - n_{\text{fixed}} \geq n - \log^3 \kappa - \log^6 \kappa / um$ . That is, the tampering function copies most of the blocks identically into the tampered codeword. Now, let  $n'_{\text{copy}}$  be the number of copied blocks which do not contain any bit from  $I_p$ . Then  $n'_{\text{copy}} \geq n_{\text{copy}} - n_{\text{peek}} \geq n - 2 \log^6 \kappa$ .

So, the tampered codeword can either be invalid or (if valid) equivalent to the original codeword (because distance of the outer codeword  $\gg \Theta(\log^6 \kappa)$ ). Now the probability that the completely fixed, dirty and peeked blocks are each identical to their counterparts in the input codeword does not depend the message  $s$  because the privacy of the outer codeword is  $\gg n_{\text{dirty}} + n_{\text{fixed}} + n_{\text{peek}}$ . This probability  $\sigma$  can be computed exhaustively by **Sim** and does not depend on the message  $s$ . So, given these  $I_0, I_1, I_p, c_{I_p}, f$ , **Sim** outputs **same**\* with probability  $\sigma$ ; otherwise outputs  $\perp$ . This is clearly identical to the real execution random variable **Tamper**<sup>(s)</sup> on  $I_0, I_1, I_p, c_{I_p}, f$ .

**Case 3.**  $n_0 + n_1 \geq N - um \log^3 \kappa$ .

In this case,  $n_{\text{fixed}} = n - n_{\text{dirty}} - n_{\text{copy}} > n - \log^3 \kappa$ . In this case, the tampered code word is either invalid or (if valid) equivalent to the codeword consistent with the fixed inner codewords (because the distance of the outer encoding scheme is much greater than  $\Theta(\log^3 \kappa)$ ).

Now **Sim** samples  $I_0, I_1, I_p, c_{I_p}$  and  $f \in \mathcal{F}$  according to **map** as in real execution. We say that  $(I_0, I_1)$  is *good* if it contains at least one bit from each column of  $c$ .

Since we have  $n_0 + n_1 > N - um \log^3 \kappa$ , we can show that  $\Pr[(I_0, I_1) \text{ is good}] = 1 - \text{negl}(\kappa)$ . If  $(I_0, I_1)$  is not good, then we define **Sim** to output  $\perp$  with probability

1 on this  $(I_0, I_1)$ . This incurs additional  $\text{negl}(\kappa)$  expected simulation error over the choices of  $(I_0, I_1)$ .

Now we are in the case that  $(I_0, I_1)$  is good. Given this,  $\text{Sim}$  first checks whether the fixed inner blocks can define a valid outer codeword. If not, then  $\text{Sim}$  outputs  $\perp$ . Simulation error in this case is 0.

Finally, we are in the case when  $(I_0, I_1)$  are good and the fixed blocks define a valid outer codeword, say  $g^*$  and a message  $s^*$ . Now  $\text{Sim}$  needs to check that remaining blocks are consistent with  $g^*$  or not. Note that since  $(I_0, I_1)$  is good, the bits restricted to  $[N] \setminus (I_0 \cup I_1 \cup I_p)$  are uniform bits. This is because all proper subsets of any column are uniform random bits. Now the probability of forming a codeword for  $s^*$  can be exhaustively computed starting from uniform random bits for  $[N] \setminus (I_0 \cup I_1 \cup I_p)$  and taking fixed value for bits in  $I_p$  as  $c_{I_p}$ . Let this probability be  $\sigma$ . Note that peeked blocks cannot intersect with set of fixed blocks. Now,  $\text{Sim}$  outputs  $s^*$  with probability  $\sigma$  and  $\perp$  otherwise. The simulation error in this case is again 0.

*Analysis for  $n_{\text{dirty}} \geq \log^3 \kappa$ .* In Case 1 and Case 2.1 above we relied on the claim that if  $n_{\text{dirty}}$  is large, then the probability of the tampered codeword being valid is negligible. To prove this, we need to consider different ways in which a dirty block can be formed. In all cases, it is easy to argue that each dirty block has a constant positive probability of being an invalid block (the unary codeword encoded by the block will not be  $0 \pmod{3}$ ). However, these events need not be independent of each other. The main reason for dependence is that bits from a single block can be moved into two different blocks. Nevertheless, we can show that there is an ordering of the dirty blocks that at least  $n_{\text{dirty}}/2$  dirty blocks are “risky,” i.e., the probability of it being an invalid block is a positive constant, *conditioned on all previous blocks being valid*. Technically, the  $i^{\text{th}}$  block (according to the given ordering) is risky if there is some block  $u$  in the original codeword, such that the  $i^{\text{th}}$  block as well as the  $j^{\text{th}}$  block, for some  $j > i$ , each has at least one bit copied to it from  $u$ ; further a sufficiently large number of bits from  $u$  should be copied to blocks numbered  $i$  or larger. We can consider any arbitrary ordering, and its reverse ordering, and argue that every dirty block is risky in one of the two orderings. This guarantees that in one of the two orderings, there are at least  $n_{\text{dirty}}/2$  risky blocks. The above analysis assumes that the adversary does not see the bits from the original codeword that it copies, which is not true since it is given the bits in  $I_p$ . To account for this we restrict to orderings in which the  $n_{\text{peek}}$  blocks containing the bits from  $I_p$  appear at the beginning, and also define a block to be risky only if it is not one of these blocks. Since  $n_{\text{peek}}$  is much smaller than  $n_{\text{dirty}}$ , we can argue that the number of risky blocks remains large.

## 6 Application to Non-malleable String Commitment

For a bit  $b$  and auxiliary input  $z$ , let  $\text{STR}_b(\langle C, R \rangle, A, n, z)$  denote the output of the following experiment: on input  $(1^n, z)$ ,  $A$  adaptively chooses two *strings*  $(v_0, v_1)$  of length  $n$  (where  $n$  is the security parameter), and receives a commitment to  $v_b$  while simultaneously  $A$  also interacts with a receiver, attempting to

commit to some value. Define  $\tilde{v}$  to be the value contained in the right commitment<sup>6</sup>. If  $A$ 's commitment transcript on right is either not accepting, or identical to the transcript on left, then output a special symbol  $\perp$ ; if  $\tilde{v} \in \{v_0, v_1\}$ , output a special symbol **same\***; otherwise, output  $\tilde{v}$ .<sup>7</sup>

**Definition 3 (Non-malleable String Commitments).** *We say that  $\langle C, R \rangle$  is a non-malleable string commitment scheme (for all strings in  $\{0, 1\}^n$ ) if for every PPT  $A$  and every  $z \in \{0, 1\}^*$  it holds that*

$$\text{STR}_0(\langle C, R \rangle, A, n, z) \approx_c \text{STR}_1(\langle C, R \rangle, A, n, z).$$

We remark that the definition above requires the message space to be large (i.e., at least super-polynomial in  $n$ ). Otherwise, the definition cannot be achieved. This definition, however, is equivalent to a standard simulation-based formulation of non-malleability (see Theorem A.1 in [18]).

Below we give our construction for non-malleable string commitments from non-malleable bit commitments. We prove the security of our scheme in the full version [3].

**Construction.** Given a bit commitment scheme  $\langle C, R \rangle$ , we construct a string commitment scheme  $\langle C', R' \rangle$  for  $\{0, 1\}^n$  as follows. Let **nm-code** be a non-malleable coding scheme for messages of length  $n$  that is robust to  $\mathcal{F} := \mathcal{F}_{0,1} \circ \mathcal{S}_N$ , and let  $t(n)$  denote the length of the codewords for some fixed polynomial  $t$ . Let **Enc** and **Dec** be encoding and decoding algorithms. To commit to a string  $v \in \{0, 1\}^n$ ,  $C'$  generates a random codeword  $w \leftarrow \text{Enc}(v)$ , and commits to each bit of  $w$  independently, and in parallel using the bit-commitment protocol  $\langle C, R \rangle$ . The receiver checks that no two bit-commitment transcripts, out of  $t$  such transcripts, are identical. If the check fails, or if any of the bit-commitment transcripts are not accepting, the receiver rejects; otherwise it accepts the commitment. To decommit to  $v$ , the receiver sends  $v$  along with decommitment information for each bit of  $w$  denoted by  $(w_i, d_i)$  for every  $i \in [t]$ ; the receiver accepts  $v$  if and only if all recommitments verify and the resulting codeword decodes to  $v$ .

## References

1. Aggarwal, D., Dodis, Y., Kazana, T., Obremski, M.: Non-malleable reductions and applications. STOC (2015, to appear)
2. Aggarwal, D., Dodis, Y., Lovett, S.: Non-malleable codes from additive combinatorics. In: STOC, pp. 774–783 (2014)

<sup>6</sup> Note that  $\tilde{v}$  is unique w.h.p. and there exists  $\tilde{d}$  s.t.  $\text{open}(\tilde{c}, \tilde{v}, \tilde{d}) = 1$  where  $\tilde{c}$  is the right commitment.

<sup>7</sup> Following [16], this definition allows MIM to commit to the same value. It is easy to prevent MIM from committing the same value generically in case of string commitments: convert the scheme to tag based by appending the tag with  $v$ , and then sign the whole transcript using the tag.

3. Agrawal, S., Gupta, D., Maji, H.K., Pandey, O., Prabhakaran, M.: Explicit non-malleable codes resistant to permutations and perturbations. *Cryptology ePrint Archive*, Report 2014/841 (2014). <http://eprint.iacr.org/>
4. Agrawal, S., Gupta, D., Maji, H.K., Pandey, O., Prabhakaran, M.: A rate-optimizing compiler for non-malleable codes against bit-wise tampering and permutations. In: Dodis, Y., Nielsen, J.B. (eds.) *TCC 2015, Part I*. LNCS, vol. 9014, pp. 375–397. Springer, Heidelberg (2015)
5. Broadbent, A., Tapp, A.: Information-theoretic security without an honest majority. In: Kurosawa, K. (ed.) *ASIACRYPT 2007*. LNCS, vol. 4833, pp. 410–426. Springer, Heidelberg (2007)
6. Canetti, R., Lin, H., Pass, R.: Adaptive hardness and composable security in the plain model from standard assumptions. In: *FOCS*, pp. 541–550 (2010)
7. Chandran, N., Goyal, V., Mukherjee, P., Pandey, O., Upadhyay, J.: Block-wise non-malleable codes. *Cryptology ePrint Archive*, Report 2015/129 (2015). <http://eprint.iacr.org/>
8. Chandran, N., Kanukurthi, B., Ostrovsky, R.: Locally updatable and locally decodable codes. In: Lindell, Y. (ed.) *TCC 2014*. LNCS, vol. 8349, pp. 489–514. Springer, Heidelberg (2014)
9. Chattopadhyay, E., Zuckerman, D.: Non-malleable codes against constant split-state tampering. In: *STOC*, pp. 306–315 (2014)
10. Cheraghchi, M., Guruswami, V.: Capacity of non-malleable codes. In: *ITCS*, pp. 155–168 (2014)
11. Cheraghchi, M., Guruswami, V.: Non-malleable coding against bit-wise and split-state tampering. In: Lindell, Y. (ed.) *TCC 2014*. LNCS, vol. 8349, pp. 440–464. Springer, Heidelberg (2014)
12. Choi, S.G., Kiayias, A., Malkin, T.: BiTR: built-in tamper resilience. In: Lee, D.H., Wang, X. (eds.) *ASIACRYPT 2011*. LNCS, vol. 7073, pp. 740–758. Springer, Heidelberg (2011)
13. Coretti, S., Maurer, U., Tackmann, B., Venturi, D.: From single-bit to multi-bit public-key encryption via non-malleable codes. *Cryptology ePrint Archive*, Report 2014/324 (2014). <http://eprint.iacr.org/>
14. Cramer, R., Dodis, Y., Fehr, S., Padró, C., Wichs, D.: Detection of algebraic manipulation with applications to robust secret sharing and fuzzy extractors. In: Smart, N.P. (ed.) *EUROCRYPT 2008*. LNCS, vol. 4965, pp. 471–488. Springer, Heidelberg (2008)
15. Cramer, R., Padró, C., Xing, C.: Optimal algebraic manipulation detection codes in the constant-error model. *Cryptology ePrint Archive*, Report 2014/116 (2014). <http://eprint.iacr.org/>
16. Dolev, D., Dwork, C., Naor, M.: Non-malleable cryptography (extended abstract). In: *STOC*, pp. 542–552 (1991)
17. Dziembowski, S., Kazana, T., Obremski, M.: Non-malleable codes from two-source extractors. In: Canetti, R., Garay, J.A. (eds.) *CRYPTO 2013, Part II*. LNCS, vol. 8043, pp. 239–257. Springer, Heidelberg (2013)
18. Dziembowski, S., Pietrzak, K., Wichs, D.: Non-malleable codes. In: *ICS*, pp. 434–452 (2010)
19. Faust, S., Mukherjee, P., Nielsen, J.B., Venturi, D.: Continuous non-malleable codes. In: Lindell, Y. (ed.) *TCC 2014*. LNCS, vol. 8349, pp. 465–488. Springer, Heidelberg (2014)
20. Faust, S., Mukherjee, P., Venturi, D., Wichs, D.: Efficient non-malleable codes and key-derivation for poly-size tampering circuits. In: Nguyen, P.Q., Oswald, E. (eds.) *EUROCRYPT 2014*. LNCS, vol. 8441, pp. 111–128. Springer, Heidelberg (2014)

21. Genkin, D., Ishai, Y., Prabhakaran, M., Sahai, A., Tromer, E.: Circuits resilient to additive attacks with applications to secure computation. In: STOC, pp. 495–504 (2014)
22. 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)
23. Gordon, D., Ishai, Y., Moran, T., Ostrovsky, R., Sahai, A.: On complete primitives for fairness. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 91–108. Springer, Heidelberg (2010)
24. Goyal, V.: Constant round non-malleable protocols using one way functions. In: STOC, pp. 695–704 (2011)
25. Goyal, V., Lee, C., Ostrovsky, R., Visconti, I.: Constructing non-malleable commitments: a black-box approach. In: FOCS, pp. 51–60 (2012)
26. Guruswami, V., Smith, A.: Codes for computationally simple channels: explicit constructions with optimal rate. In: FOCS, pp. 723–732 (2010)
27. Hemenway, B., Ostrovsky, R.: Public-key locally-decodable codes. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 126–143. Springer, Heidelberg (2008)
28. 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)
29. Lin, H., Pass, R.: Constant-round non-malleable commitments from any one-way function. In: STOC, pp. 705–714 (2011)
30. Lipton, R.J.: A new approach to information theory. In: STACS, pp. 699–708 (1994)
31. Liu, F.-H., Lysyanskaya, A.: Tamper and leakage resilience in the split-state model. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 517–532. Springer, Heidelberg (2012)
32. Micali, S., Peikert, C., Sudan, M., Wilson, D.A.: Optimal error correction against computationally bounded noise. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 1–16. Springer, Heidelberg (2005)
33. Myers, S., Shelat, A.: Bit encryption is complete. In: FOCS, pp. 607–616 (2009)
34. Ostrovsky, R., Pandey, O., Sahai, A.: Private locally decodable codes. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 387–398. Springer, Heidelberg (2007)
35. Pass, R., Rosen, A.: New and improved constructions of non-malleable cryptographic protocols. In: STOC, pp. 533–542 (2005)

# **Assumptions**

# Cryptanalysis of the Co-ACD Assumption

Pierre-Alain Fouque<sup>1</sup>, Moon Sung Lee<sup>2</sup>, Tancrede Lepoint<sup>3</sup>,  
and Mehdi Tibouchi<sup>4</sup>✉

<sup>1</sup> Université de Rennes 1 and Institut Universitaire de France, Rennes, France  
fouque@irisa.fr

<sup>2</sup> Seoul National University (SNU), Seoul, South Korea  
moolee@snu.ac.kr

<sup>3</sup> CryptoExperts, Paris, France  
tancrede.lepoint@cryptoexperts.com

<sup>4</sup> NTT Secure Platform Laboratories, Tokyo, Japan  
tibouchi.mehdi@lab.ntt.co.jp

**Abstract.** At ACM-CCS 2014, Cheon, Lee and Seo introduced a new number-theoretic assumption, the Co-Approximate Common Divisor (Co-ACD) assumption, based on which they constructed several cryptographic primitives, including a particularly fast additively homomorphic encryption scheme. For their proposed parameters, they found that their scheme was the “most efficient of those that support an additive homomorphic property”. Unfortunately, it turns out that those parameters, originally aiming at 128-bit security, can be broken in a matter of seconds.

Indeed, this paper presents several lattice-based attacks against the Cheon–Lee–Seo (CLS) homomorphic encryption scheme and of the underlying Co-ACD assumption that are effectively devastating for the proposed constructions. A few known plaintexts are sufficient to decrypt any ciphertext in the symmetric-key CLS scheme, and small messages can even be decrypted without any known plaintext at all. This breaks the security of both the symmetric-key and the public-key variants of CLS encryption as well as the underlying decisional Co-ACD assumption. Moreover, Coppersmith techniques can be used to solve the search variant of the Co-ACD problem and mount a full key recovery on the CLS scheme.

**Keywords:** Cryptanalysis · Lattice reduction · Coppersmith theorem · Homomorphic encryption · Co-ACD problem

## 1 Introduction

At ACM-CCS 2014, Cheon, Lee and Seo [CLS14] introduced a new hardness assumption called the co-approximate common divisor assumption (Co-ACD). Informally, the decisional Co-ACD assumption states that it is hard to distinguish (without knowing the primes  $p_i$ 's) between the uniform distribution over  $\mathbb{Z}_{p_1} \times \cdots \times \mathbb{Z}_{p_n}$  and the distribution which outputs  $(e \cdot Q \bmod p_i)_{i=1}^n$ , where  $Q$  is a *known* value and  $e$  is some uniformly distributed noise in  $(-2^\rho, 2^\rho)$ . It is

assumed that  $\max\{p_1, \dots, p_k\} < 2^\rho \cdot Q < \prod_i p_i$  to make the problem non trivial. The search Co-ACD assumption states that, given arbitrarily many samples from the distribution above, it is hard to recover the  $p_i$ 's themselves. To validate the plausible hardnesses of these assumptions, the authors provided a crypt-analytic survey [CLS14, Sect. 4] based on known and new dedicated attacks: the algebraic approach due to Chen–Nguyen [CN12, CNT12], orthogonal lattices [NS98b, NS99] and Coppersmith's theorem [Cop97, How01, CH12].

Based on the hardness of the decisional problem, the authors then proposed a very efficient additive homomorphic encryption scheme which outperformed competitors such as [Pai99, NLV11, JL13] by several orders of magnitude. In this scheme, a message  $m \in \mathbb{Z}_Q$  is encrypted as  $(c_1, c_2) = (m + e \cdot Q \bmod p_1, m + e \cdot Q \bmod p_2)$  for large enough primes  $p_1, p_2$  which form the secret key. The hope is that  $eQ > p_1, p_2$  will hide the message  $m$  for each component  $c_1$  and  $c_2$  (which is indeed the case if the decisional Co-ACD assumption holds), while still allowing decryption using the Chinese Remainder Theorem for users who know the secret key, since  $eQ < p_1 p_2$ . This is a symmetric-key scheme, but can be converted to public-key using a transformation similar to the one from [DGHV10].

As the name suggests, the Co-ACD assumption has some similarity with the (extended) approximate common divisor (ACD) assumption, which has been used to construct various primitives including fully homomorphic encryption [DGHV10, CCK+13, CLT14]. In the ACD problem, the goal is to recover  $p$  given samples of the form  $x = pq + r$  where  $q$  and  $r$  are uniformly distributed in  $[0, 2^\gamma/p)$  and  $(-2^\rho, 2^\rho)$ , respectively. This problem has been introduced by Howgrave-Graham in [How01] and lattice reduction algorithms have been used to solve this problem using Coppersmith's theorem [Cop97, CH12], as well as other algebraic techniques [CN12, CNT12]. In view of that similarity, the parameter choice in [CLS14] seems rather bold: for example, the authors claim 128 bits of security with ciphertexts as small as 3000 bits, whereas even if we restrict the recent ACD-based fully homomorphic encryption scheme [CLT14] to homomorphic additions, ciphertext size for that scheme is still at least cubic in the security parameter, so ciphertexts have to be millions of bits long for 128 bits of security.

**Our Contributions.** In this paper, we present three new attacks, targeting the Cheon–Lee–Seo (CLS) encryption scheme (in both its symmetric-key and public-key incarnations) as well as the decisional and search variants of the Co-ACD assumption. Our new attacks severely reduce the security of all proposed constructions from [CLS14].

Our first attack, described in Sect. 3, is a known-plaintext attack against the symmetric-key CLS scheme. We establish that a few known plaintext-ciphertext pairs are sufficient to decrypt any ciphertext. This attack breaks the one-wayness of the symmetric-key CLS scheme, and it is straightforward to use it to break the one-wayness of the public-key CLS scheme and the decisional Co-ACD assumption as well. The algorithm runs in polynomial time on a wide range of parameters, and while it is possible to select parameters outside of that range, they need to be huge to achieve security, resulting in a scheme of little practical use.



Our second attack, discussed in Sect. 4, is a ciphertext-only attack on the symmetric-key CLS scheme. It allows an attacker to decrypt, without any known plaintext, a set of ciphertexts corresponding to small messages. Combined with the first attack, this makes it possible to decrypt arbitrary ciphertexts given only a few ciphertexts corresponding to small messages. This stronger attack uses the more advanced “doubly orthogonal lattice” technique of Nguyen–Stern, which makes it heuristic, but we find that it is very effective as well in practice.

Finally, our third attack, discussed in Sect. 5, solves the search variant of the Co-ACD problem in a wide range of parameters, and can in particular be used to factor the modulus of the public-key CLS scheme, revealing the entire private key. This attack combines a pure lattice step together with a generalization of Coppersmith’s theorem due to Alexander May.

We present each of these attacks both in the case when  $n = 2$  (i.e. the modulus  $N$  is a product of two primes), which is the one considered by Cheon et al. to construct their encryption schemes, and in the case of larger  $n$ , for which the Co-ACD assumptions are still defined, and the encryption schemes admit natural generalizations. We also provide extensive experiments that show that our attacks completely break the parameters proposed in [CLS14] in a very concrete way.

**Related Work.** Our attacks use orthogonal lattice techniques (as discussed e.g. in [NT12]), originally introduced by Nguyen and Stern in several attacks [NS97, NS98b, NS98a, NS99] against the Qu–Vanstone knapsack-based scheme, the Itoh–Okamoto–Mambo cryptosystem, Béguin–Quisquater server-aided RSA protocol, and the hidden subset sum problem. Similar techniques were also used in other cryptanalytic works, but work only with one modulus  $p_1$  (either known but hard to factor [DGHV10, CNT10], or unknown [LT15]). In the original paper [CLS14], orthogonal lattice attacks were already considered to set the parameter  $\rho = (n - 1)\eta + 2\lambda$  for  $\lambda$  bits of security; we obtain better attacks, however, by considering different lattices as well as extended attack techniques.

**Notation.** For any integer  $n$ , we denote by  $[n]$  the set  $\{1, \dots, n\}$  and by  $\mathbb{Z}_n$  the ring of integers modulo  $n$ . We use  $a \stackrel{\$}{\leftarrow} A$  to denote the operation of uniformly sampling an element  $a$  from a finite set  $A$ . If  $\chi$  is a distribution, the notation  $a \leftarrow \chi$  refers to sampling  $a$  according to the distribution  $\chi$ . We let  $\lambda$  be the security parameter. We use bold letters for vectors and the inner product of two vectors  $\mathbf{u}, \mathbf{v}$  is denoted by  $\langle \mathbf{u}, \mathbf{v} \rangle$ .

## 2 Preliminaries

In this section, we recall the additive homomorphic schemes (symmetric and public-key) proposed by Cheon–Lee–Seo (CLS) at ACM-CCS 2014 [CLS14], and its underlying security assumption called the co-approximate common divisor assumption (Co-ACD). We also give some background on lattices, orthogonal lattices and Coppersmith’s algorithm to find small roots of modular polynomial equations.

**Table 1.** Parameters for the CLS scheme for  $\lambda = 128$  bits of security

Parameters	$Q$	$\eta$	$\rho$	$\tau$	$\nu$
Set-I	$2^{256}$	1536	1792	3328	142
Set-II	$2^{256}$	2194	2450	4645	142
Set-III	$2^{256}$	2706	2962	5669	142

### 2.1 CLS Somewhat Additively Homomorphic Encryption Schemes

**Secret-Key Scheme.** Given the security parameter  $\lambda$ , we use the following parameters:  $\eta$  the bit-length of the secret key elements  $p_i$ 's,  $Q$  the size of the message space  $\mathbb{Z}_Q$ ,  $\rho$  the bit-length of the error. The CLS scheme then consists of the following algorithms.

CLS.KeyGen( $1^\lambda$ ): Generate two random prime integers  $p_1, p_2$  of  $\eta$  bits, and output  $\text{sk} = \{p_1, p_2\}$ .

CLS.Encrypt( $\text{sk}, m \in \mathbb{Z}_Q$ ): Generate a random noise  $e \xleftarrow{\$} (-2^\rho, 2^\rho)$ , and output  $\mathbf{c} = (c_1, c_2) = (m + e \cdot Q \bmod p_1, m + e \cdot Q \bmod p_2)$ .

CLS.Decrypt( $\text{sk}, \mathbf{c}$ ): Parse  $\mathbf{c} = (c_1, c_2)$ . Compute  $e' = c_1 + p_1 \cdot (p_1^{-1} \bmod p_2) \cdot (c_2 - c_1) \bmod (p_1 p_2)$  and output  $e' \bmod Q$ .

This completes the description of the scheme using Garner's formula to improve the decryption. When  $2^\rho \cdot Q \leq 2^{2\eta-2} < p_1 p_2$ , the previous scheme is obviously correct. As shown in [CLS14], this scheme is also somewhat additively homomorphic when adding the ciphertexts componentwise over  $\mathbb{Z}$ , i.e. a limited number of homomorphic additions (at least  $2^{\eta-3-\rho-\lceil \log_2 Q \rceil}$ ) can be performed on fresh ciphertexts while preserving correctness.

**Public-Key Variant.** A public key variant of the latter scheme was also proposed in [CLS14]. The public key  $\text{pk}$  then consists of the public modulus  $N = p_1 p_2$ , and  $\mathbf{x}_1, \dots, \mathbf{x}_\tau, \mathbf{b}_1, \mathbf{b}_2 \leftarrow \text{CLS.Encrypt}(\text{sk}, 0)$ .

To encrypt a message  $m \in \mathbb{Z}_Q$ , one samples  $(s_i)_{i=1}^\tau \xleftarrow{\$} \{0, 1\}^\tau$  and  $t_1, t_2 \leftarrow [0, 2^\nu)$  and outputs  $\mathbf{c} = (m, m) + \sum_{i=1}^\tau s_i \cdot \mathbf{x}_i + t_1 \cdot \mathbf{b}_1 + t_2 \cdot \mathbf{b}_2$ . The parameters are chosen so that  $(\tau + 2^{\nu+1}) \cdot 2^\rho \cdot Q < 2^{\eta-2}$  to ensure that  $\mathbf{c}$  decrypts correctly with CLS.Decrypt. The  $\mathbf{x}_i$ 's,  $\mathbf{b}_1$ , and  $\mathbf{b}_2$  are specially crafted (using rejection sampling) in order to apply the leftover hash lemma over lattices of Coron, Lepoint and Tibouchi [CLT13, Sect. 4]; this step gives conditions on  $\nu$  and  $\tau$  but is irrelevant for our purposes—we refer to [CLS14] for a rigorous description.

**Practical Parameters.** Some specific parameters (and implementation results) are proposed by Cheon et al. The parameters are chosen from their cryptanalysis survey [CLS14, Sect. 4] and aim at a security level of 128 bits. We recall these parameters in Table 1.

### 2.2 The Co-ACD Assumptions

**Definition 1 (Co-ACD).** Let  $n, Q, \eta, \rho \geq 1$  and denote  $\pi$  the uniform distribution over the  $\eta$ -bit prime integers. The Co-ACD distribution for a given  $\mathbf{p} = (p_1, \dots, p_n) \in \mathbb{Z}^n$  is the set of tuples  $(e \cdot Q \bmod p_1, \dots, e \cdot Q \bmod p_n)$  where  $e \xleftarrow{\$} (-2^\rho, 2^\rho)$ .

- The search-Co-ACD problem is: For a vector  $\mathbf{p} \leftarrow \pi^n$  and given arbitrarily many samples from the Co-ACD distribution for  $\mathbf{p}$ , to compute  $\mathbf{p}$ .
- The decisional-Co-ACD problem is: For some fixed vector  $\mathbf{p} \leftarrow \pi^n$  and given arbitrarily many samples from  $\mathbb{Z}_{p_1} \times \dots \times \mathbb{Z}_{p_n}$ , to distinguish whether the samples are distributed uniformly or whether they are distributed as the Co-ACD distribution for  $\mathbf{p}$ .

We sometimes use notation like  $(\rho, \eta, n; Q)$ -Co-dACD to mean the assumption that no polynomial-time adversary solve the decisional Co-ACD problem with these parameters. In [CLS14, Theorem 1 and 3], the authors prove that the (somewhat) additive homomorphic encryption schemes of Sect. 2.1 are semantically secure under the  $(\rho, \eta, 2; Q)$ -Co-dACD assumption when  $Q < 2^{\eta-3-2\lambda}$ .

### 2.3 Background on Lattices

In this section, we recall some useful facts about lattices, orthogonal lattices and Coppersmith’s technique. We refer to [MR09, NS01, Ngu10] for more details.

**Lattices.** A  $d$ -dimensional Euclidean lattice  $L \subset \mathbb{Z}^t$  is the set of all integer linear combinations of some linearly independent vectors  $\mathbf{b}_1, \dots, \mathbf{b}_d \in \mathbb{Z}^t$ , and we write

$$L = \left\{ \sum_{i \leq d} x_i \mathbf{b}_i : (x_i)_{i \leq d} \in \mathbb{Z}^d \right\} = \mathbb{Z} \mathbf{b}_1 \oplus \dots \oplus \mathbb{Z} \mathbf{b}_d.$$

We denote  $\text{vol}(L)$  the volume of the lattice, defined as  $\text{vol}(L) = |\det(BB^t)^{1/2}|$  for any basis matrix  $B = (\mathbf{b}_1, \dots, \mathbf{b}_d)$ . (The determinant of a lattice is well-defined since it is independent of the choice of the basis). A theorem due to Minkowski bounds the length of the shortest vector in  $L$  in terms of  $\text{vol}(L)$ .

**Theorem 1 ([Ngu10], Cor. 3).** Any  $d$ -dimensional lattice  $L$  in  $\mathbb{Z}^t$  contains a nonzero vector  $\mathbf{x}$  such that  $\|\mathbf{x}\| \leq \sqrt{d} \cdot \text{vol}(L)^{1/d}$ .

And in fact, in most lattices, that is close to the right order of magnitude not just for the length  $\lambda_1(L)$  of the shortest vector, but for all successive minima  $\lambda_i(L)_{1 \leq i \leq d}$ : according to the Gaussian heuristic, one expects  $\lambda_i(L) \approx \sqrt{\frac{d}{2\pi e}} \text{vol}(L)^{1/d}$  for all  $i$ .

Among all the bases of a lattice  $L$ , some are “better” than others, in the sense that they consist of shorter and “more orthogonal” vectors. Shortening the vectors of a lattice basis and making them more orthogonal is the purpose of lattice reduction algorithm. The LLL algorithm of Lenstra, Lenstra and Lovász [LLL82]

is the best known such algorithm; it runs in polynomial time and returns a basis  $(\mathbf{b}_1, \dots, \mathbf{b}_d)$  such that  $\|\mathbf{b}_i\| \leq 2^{\chi \cdot d} \cdot \lambda_i(L)$  for some absolute constant  $\chi$ . More generally, with other reduction algorithms such as BKZ, one can achieve better approximation factors.

**Orthogonal Lattices.** For any vectors  $\mathbf{u}, \mathbf{v} \in \mathbb{Z}^t$ , we say that  $\mathbf{u}$  and  $\mathbf{v}$  are orthogonal if  $\langle \mathbf{u}, \mathbf{v} \rangle = 0$ , and we denote it  $\mathbf{u} \perp \mathbf{v}$ . For any lattice  $L \subset \mathbb{Z}^t$ , we denote by  $L^\perp$  its orthogonal lattice, i.e. the set of vectors in  $\mathbb{Z}^t$  orthogonal to the points in  $L$ :  $L^\perp = \{\mathbf{v} \in \mathbb{Z}^t \mid \forall \mathbf{u} \in L, \langle \mathbf{u}, \mathbf{v} \rangle = 0\}$ . Note that it is shown that  $\dim(L) + \dim(L^\perp) = t$  and  $\text{vol}(L^\perp) \leq \text{vol}(L)$  in [NS97]. We have the following theorem [NS97]:

**Theorem 2.** *There exists an algorithm which, given any basis  $\{\mathbf{b}_1, \dots, \mathbf{b}_d\}$  of a lattice  $L$  in  $\mathbb{Z}^t$  of dimension  $d$ , outputs an LLL-reduced basis of the orthogonal lattice  $L^\perp$ , and whose running time is polynomial with respect to  $t, d$  and any upper bound on the bit-length of the  $\|\mathbf{b}_j\|$ 's.*

*Remark 1.* A simple algorithm for Theorem 2 consists in a single call to LLL: to compute an LLL-reduced basis  $\{\mathbf{u}_1, \dots, \mathbf{u}_{t-d}\}$  of the orthogonal lattice  $L^\perp \subset \mathbb{Z}^t$  to  $L = \mathbb{Z}\mathbf{b}_1 \oplus \dots \oplus \mathbb{Z}\mathbf{b}_d \subset \mathbb{Z}^t$ , one applies LLL to the lattice in  $\mathbb{Z}^{d+t}$  generated by the rows of the following matrix:

$$\begin{pmatrix} \gamma \cdot b_{1,1} & \dots & \gamma \cdot b_{d,1} & 1 & & 0 \\ \vdots & & \vdots & & \ddots & \\ \gamma \cdot b_{1,t} & \dots & \gamma \cdot b_{d,t} & 0 & & 1 \end{pmatrix},$$

where  $\mathbf{b}_i = (b_{i,j})_{j=1}^t$  and  $\gamma$  is a suitably large constant, and keeps only the  $t$  last coefficients of each resulting vector.

**Coppersmith's Technique.** In [Cop97], Coppersmith presents a method based on lattice reduction to find small roots of univariate modular polynomials. In this paper, we use the more general formulation of his theorem due to May [May03, Theorem 7].

**Theorem 3 (Coppersmith).** *Let  $N$  be an integer of unknown factorization, which has a divisor  $b \geq N^\beta$ . Let  $f(x)$  be a univariate polynomial of degree  $\delta$ . Then, we can find all solutions  $x_0$  to the equation  $f(x) \equiv 0 \pmod b$  which satisfy  $|x_0| \leq N^{\beta^2/\delta}$  in time polynomial in  $(\log N, \delta)$ .*

### 3 Known Plaintext Attack Against the CLS Scheme

Let  $m_1, \dots, m_t$  be  $t$  messages with their respective ciphertexts  $\mathbf{c}_1, \dots, \mathbf{c}_t$ , where  $\mathbf{c}_i \leftarrow \text{Encrypt}(\text{sk}, m_i)$ . Throughout this section, we assume that the first message  $m_1$  is unknown while the other  $(t - 1)$  messages  $m_2, \dots, m_t$  are known, and we show that if  $t$  is large enough,  $m_1$  can be recovered efficiently. This means that the symmetric-key CLS scheme is not one-way against known-message attacks (not OW-KPA-secure), and as a direct corollary, it follows that the public-key CLS scheme is not one-way either.

Moreover, this also implies that we can solve the decisional Co-ACD problem efficiently for suitable parameters. We can either see this from the original security reduction for the symmetric-key [CLS14, Theorem 1], or much more directly: if we have samples from a distribution which is either the Co-ACD distribution or uniformly random, we can use these samples to “encrypt” randomly chosen messages  $m_1, \dots, m_t$ , and then apply our attack. It will recover the correct value of  $m_1$  with significant probability if the distribution was Co-ACD, but returns a random element of  $\mathbb{Z}_Q$  for the uniform distribution, so that we solve the decisional Co-ACD problem with significant advantage.

We present our attack in the sections below: we first present the attack in Sect. 3.1 in the case when  $n = 2$ , i.e.  $N = p_1 p_2$ , as in the original CLS encryption schemes. Then, we show in Sect. 3.2 how it generalizes naturally to higher values of  $n$ , thus breaking the decisional Co-ACD assumption for a wide range of parameters.

### 3.1 Message Recovery Using Known Plaintexts for $N = p_1 p_2$

Denote  $\mathbf{c} = (c_1, \dots, c_t)$ . We can write for each  $i \in [t]$ :

$$\begin{cases} c_{i,1} = m_i + e_i \cdot Q + k_{i,1} \cdot p_1 \\ c_{i,2} = m_i + e_i \cdot Q + k_{i,2} \cdot p_2, \end{cases}$$

where  $|e_i| < 2^\rho$  and  $k_{i,1}$  (resp.  $k_{i,2}$ ) is the quotient in the Euclidean division of  $m_i + e_i \cdot Q$  by  $p_1$  (resp.  $p_2$ ). If we write  $\mathbf{e} = (e_i)_{i \in [t]}$ ,  $\mathbf{k}_j = (k_{i,j})_{i \in [t]}$  and  $\mathbf{C}_j = (c_{i,j})_{i \in [t]}$  for  $j = 1, 2$ , we have:

$$\begin{cases} \mathbf{C}_1 = \mathbf{m} + \mathbf{e} \cdot Q + p_1 \cdot \mathbf{k}_1 \\ \mathbf{C}_2 = \mathbf{m} + \mathbf{e} \cdot Q + p_2 \cdot \mathbf{k}_2. \end{cases} \tag{1}$$

In particular, this yields the following equation:

$$\mathbf{C}_1 - \mathbf{C}_2 = p_1 \cdot \mathbf{k}_1 - p_2 \cdot \mathbf{k}_2. \tag{2}$$

Since only  $\mathbf{C}_1 - \mathbf{C}_2$  is known, Eq. (2) can be seen as a variant of the hidden subset sum problem as considered by Nguyen and Stern [NS99]. However, while in the hidden subset sum setting the hidden vectors are random independent binary vectors, in our case the unknown vectors  $\mathbf{k}_1$  and  $\mathbf{k}_2$  are nearly parallel and have entries of roughly  $(\rho + \log Q - \eta)$  bits. As a result, it turns out that  $\mathbf{k}_1$  and  $\mathbf{k}_2$  cannot be obtained directly from the much shorter reduced basis of the lattice they generate, and therefore we do not know how to recover the secret primes  $p_1, p_2$  from the ciphertext difference  $\mathbf{C}_1 - \mathbf{C}_2$  alone. Nevertheless, we can still obtain the unknown message  $m_1$  in two steps:

- (1) Find a short vector  $\mathbf{u}$  in the orthogonal lattice  $L^\perp$  to the lattice  $L = \mathbb{Z}(\mathbf{C}_1 - \mathbf{C}_2)$  generated by  $\mathbf{C}_1 - \mathbf{C}_2$ . If  $\mathbf{u}$  is short enough, we get  $\langle \mathbf{u}, \mathbf{k}_1 \rangle = \langle \mathbf{u}, \mathbf{k}_2 \rangle = 0$ .

- (2) Reducing the linear equation  $\langle \mathbf{u}, \mathbf{m} + \mathbf{e} \cdot Q - \mathbf{C}_1 \rangle = \langle \mathbf{u}, -p_1 \cdot \mathbf{k}_1 \rangle = -p_1 \langle \mathbf{u}, \mathbf{k}_1 \rangle = 0$  modulo  $Q$ , we eliminate  $\mathbf{e}$ , and recover the message from  $\langle \mathbf{u}, \mathbf{m} - \mathbf{C}_1 \rangle \equiv 0 \pmod{Q}$ .

For the first step, we use the algorithm of Nguyen and Stern to obtain a basis of the orthogonal lattice  $L^\perp \subset \mathbb{Z}^t$  of rank  $t - 1$ , where  $L = \mathbb{Z}(\mathbf{C}_1 - \mathbf{C}_2)$  (see Theorem 2). Let  $\mathbf{u}$  be a vector in  $L^\perp$ . Then, the following holds:

$$\begin{aligned} \langle \mathbf{u}, \mathbf{m} + \mathbf{e} \cdot Q - \mathbf{C}_1 \rangle &\equiv \langle \mathbf{u}, \mathbf{C}_1 - \mathbf{C}_1 \rangle = 0 \pmod{p_1}, \\ \langle \mathbf{u}, \mathbf{m} + \mathbf{e} \cdot Q - \mathbf{C}_1 \rangle &\equiv \langle \mathbf{u}, \mathbf{C}_2 - \mathbf{C}_1 \rangle = 0 \pmod{p_2}. \end{aligned}$$

Thus, we get the following equation:

$$\langle \mathbf{u}, \mathbf{m} + \mathbf{e} \cdot Q - \mathbf{C}_1 \rangle \equiv 0 \pmod{N}.$$

Now if  $\|\mathbf{u}\|$  is less than  $N/\|\mathbf{m} + \mathbf{e} \cdot Q - \mathbf{C}_1\| \approx 2^{2\eta - \rho - \log Q}$ , then

$$\|\langle \mathbf{u}, \mathbf{m} + \mathbf{e} \cdot Q - \mathbf{C}_1 \rangle\| \leq \|\mathbf{u}\| \cdot \|\mathbf{m} + \mathbf{e} \cdot Q - \mathbf{C}_1\| < N,$$

which implies that the inner product  $\langle \mathbf{u}, \mathbf{m} + \mathbf{e} \cdot Q - \mathbf{C}_1 \rangle$  is actually zero over the integers. Finally,  $\mathbf{u}$  satisfies  $\langle \mathbf{u}, \mathbf{k}_1 \rangle = 0$  from Eq. (1), and similarly we obtain that  $\langle \mathbf{u}, \mathbf{k}_2 \rangle = 0$ .

In the second step, we actually recover the message  $m_1$ . Using the vector  $\mathbf{u} = (u_1, \dots, u_t) \in L^\perp$  obtained in the first step, we have  $\langle \mathbf{u}, \mathbf{m} + \mathbf{e} \cdot Q - \mathbf{C}_1 \rangle = 0$ . Viewing this equation modulo  $Q$ , we obtain

$$\langle \mathbf{u}, \mathbf{m} - \mathbf{C}_1 \rangle = 0 \pmod{Q}. \tag{3}$$

Solving the Eq. (3) modulo  $Q$  reveals  $m_1$  completely as soon as  $\gcd(u_1, Q) = 1$ , which happens with significant probability  $\phi(Q)/Q = \Omega(1/\log \log Q)$  if we assume that  $u_1$  is randomly distributed modulo  $Q$ . More generally, we always obtain  $m_1$  modulo  $(Q/\gcd(u_1, Q))$ , which gives  $\gcd(u_1, Q)$  candidates for  $m_1$  (this is usually small, and polynomially bounded on average for a random  $Q$  by [Bro01, Theorem 4.3]), and we can of course obtain more information on  $m_1$  with a different, independent short vector  $\mathbf{u}$  or with more known plaintexts, making recovery very fast in practice.

We now discuss the value of  $t$  needed to find a short enough vector  $\mathbf{u}$ . Since the volume of  $L^\perp$  is  $\text{vol}(L^\perp) \leq \text{vol}(L) = \|\mathbf{C}_1 - \mathbf{C}_2\| \approx 2^\eta$  and it has rank  $t - 1$ , Minkowski's theorem guarantees that it contains a vector  $\mathbf{u}$  of length at most  $\sqrt{t-1} \cdot \text{vol}(L^\perp)^{1/(t-1)} \approx 2^{\eta/(t-1)}$ . Such a vector is short enough to carry out our attack provided that:

$$\eta/(t-1) < 2\eta - \rho - \log Q \iff t > 1 + \frac{\eta}{2\eta - \rho - \log Q}.$$

Setting  $t = 4$  is enough for all proposed parameters in [CLS14]. For such a small lattice dimension, it is straightforward to find the actual shortest vector of the lattice, and we can easily recover  $m_1$  in practice in a fraction of a second, even accounting for occasional repetitions when more than one candidate is found.

We can also analyze the attack asymptotically as follows. For large lattice dimensions, a lattice reduction algorithm may not find the shortest vector of  $L^\perp$ , but only an approximation within a factor  $2^{\chi \cdot (t-1)}$ , where the value  $\chi$  depends on the algorithm; we can achieve a given value of  $\chi$  in time  $2^{\Theta(1/\chi)}$  using BKZ-reduction with block size  $\Theta(1/\chi)$ . With such an algorithm, a short enough vector  $\mathbf{u}$  will be found provided that:

$$\chi \cdot (t - 1) + \frac{\eta}{t - 1} < 2\eta - \rho - \log Q.$$

The left-hand side is minimal for  $t - 1 = \sqrt{\eta/\chi}$ , and is then equal to  $2\sqrt{\chi\eta}$ . Moreover, the right-hand side is a lower bound on the additive homomorphicity of the encryption scheme (denoted by  $\log_2 A$  in [CLS14]), and should thus be at least as large as the security parameter  $\lambda$  for the scheme to be of interest. The condition to find  $\mathbf{u}$  then becomes  $\chi < \lambda^2/4\eta$ . Thus, we obtain an attack with complexity  $2^{\Omega(\eta/\lambda^2)}$ , which means that our algorithm runs in polynomial time for parameters such that  $\eta = \tilde{O}(\lambda^2)$ , and that we should have at least  $\eta = \Omega(\lambda^3)$  to achieve  $\lambda$  bits of security, making the scheme quite inefficient.

More concretely, 128-bit security roughly corresponds to  $2^\chi \approx 1.007$  [CN11, vdPS13]. Hence, a conservative choice of  $\eta$  for 128 bits of security should satisfy:

$$\eta \gtrsim \frac{128^2}{4 \cdot \log_2(1.007)} > 400,000,$$

making the scheme quite impractical!

### 3.2 Generalization to $n \geq 2$

In this section, we extend the attack to  $n \geq 2$ . Consider a modulus  $N = \prod_{j=1}^n p_j$  (which may be kept secret). Using the same notation as before, we have:

$$\mathbf{C}_j = \mathbf{m} + \mathbf{e} \cdot Q + p_j \cdot \mathbf{k}_j, \text{ for all } j \in [n].$$

Recall that we know the plaintexts  $m_2, \dots, m_t$  where  $\mathbf{m} = (m_1, m_2, \dots, m_t)$ . Our goal is to find  $m_1 \in \mathbb{Z}_Q$ .

We first prove that a vector orthogonal to  $\mathbf{C}_j - \mathbf{C}_1$  for all  $j \in [n]$  is either large, or orthogonal to  $\mathbf{k}_j$  for all  $j \in [n]$ .

**Lemma 1.** *Let  $\mathbf{u} \in \mathbb{Z}^t$ . If  $\mathbf{u} \perp (\mathbf{C}_j - \mathbf{C}_1)$  for all  $j \in [n]$ , then it verifies one of the following condition:*

- (1)  $\mathbf{u} \perp \mathbf{k}_j$  for all  $j \in [n]$ ;
- (2)  $\|\mathbf{u}\| \geq 2^{n(\eta-1)} / (Q \cdot 2^{\rho+1} \cdot t^{1/2})$ .

*Proof.* Let  $\mathbf{u} \in \mathbb{Z}^t$  be such that  $\mathbf{u} \perp (\mathbf{C}_j - \mathbf{C}_1)$  for all  $j \in [n]$ , and not satisfying condition (2). Now for all  $j \in [n]$ ,

$$0 = \langle \mathbf{u}, \mathbf{C}_j - \mathbf{C}_1 \rangle = \langle \mathbf{u}, \mathbf{m} + \mathbf{e} \cdot Q + p_j \cdot \mathbf{k}_j - \mathbf{C}_1 \rangle = \langle \mathbf{u}, \mathbf{m} + \mathbf{e} \cdot Q - \mathbf{C}_1 \rangle + p_j \cdot \langle \mathbf{u}, \mathbf{k}_j \rangle.$$

In particular,  $N = \prod_{j=1}^n p_j > 2^{n(\eta-1)}$  divides  $\langle \mathbf{u}, \mathbf{m} + \mathbf{e} \cdot Q - \mathbf{C}_1 \rangle$ . Now the Cauchy-Schwarz inequality yields

$$|\langle \mathbf{u}, \mathbf{m} + \mathbf{e} \cdot Q - \mathbf{C}_1 \rangle| \leq \|\mathbf{u}\| \cdot (\|\mathbf{m}\| + Q \cdot \|\mathbf{e}\| + \|\mathbf{C}_1\|) < \|\mathbf{u}\| \cdot Q \cdot 2^{\rho+1} \cdot t^{1/2} < 2^{n(\eta-1)},$$

which implies  $\langle \mathbf{u}, \mathbf{m} + \mathbf{e} \cdot Q - \mathbf{C}_1 \rangle = 0$ , and thus  $\langle \mathbf{u}, \mathbf{k}_j \rangle = 0$  for all  $j \in [n]$ .  $\square$

Let  $L^\perp$  be the orthogonal lattice to the lattice  $L = \mathbb{Z}(\mathbf{C}_2 - \mathbf{C}_1) \oplus \dots \oplus \mathbb{Z}(\mathbf{C}_n - \mathbf{C}_1)$  generated by the vectors  $\mathbf{C}_j - \mathbf{C}_1$ . As before, we can use lattice reduction to find a short vector  $\mathbf{u}$  in  $L^\perp$ , and by Lemma 1, if  $\mathbf{u}$  is sufficiently short it must satisfy  $\langle \mathbf{u}, \mathbf{m} + \mathbf{e} \cdot Q - \mathbf{C}_1 \rangle = 0$ . Reducing modulo  $Q$ , we obtain  $\langle \mathbf{u}, \mathbf{m} - \mathbf{C}_1 \rangle = 0 \pmod Q$ . And  $m_1$  can be recovered provided that  $\gcd(u_1, Q) = 1$  (which again happens with significant probability).

As before, let us estimate the condition on  $t$  for such a short vector  $\mathbf{u}$  to exist. Since  $L^\perp$  is of rank  $m = t - n + 1$  and volume  $\text{vol}(L^\perp) \leq \text{vol}(L) \leq \prod_{i=2}^n \|\mathbf{C}_i - \mathbf{C}_1\| \approx 2^{(n-1)\eta}$ , Minkowski's theorem ensures that it contains a vector of length at most  $\sqrt{m} \cdot \text{vol}(L^\perp)^{1/m} \approx 2^{(n-1)\eta/m}$ . Taking logarithms and ignoring logarithmic factors, the condition can be written as:

$$\frac{(n-1)\eta}{m} < n(\eta-1) - \log Q - \rho \iff t > \frac{(n-1)\eta}{n(\eta-1) - \log Q - \rho} + n - 1. \quad (4)$$

Again, if we can find the shortest vector in a  $t$ -dimensional lattice for some  $t$  satisfying (4), we can break the CLS scheme (and the decisional Co-ACD assumption) for the corresponding choice of parameters. For the parameters suggested in [CLS14], where the authors take  $\rho = (n-1)\eta + 2\lambda$ , the required  $t$  is quite small: it suffices to choose  $t \approx 3n$  if  $2\lambda + \log Q < \eta/2$ . Therefore, it is easy to break such parameters for small values of  $n$ .

More generally, we can mimic the asymptotic analysis of the previous section to take larger parameters into account. We show in the full version of this paper [FLLT14] that, in this case, our attack on the scheme (and the corresponding Co-ACD assumption) runs in time  $2^{\Omega(n\eta/\lambda^2)}$ . Therefore,  $n\eta$  (the size in bits of the modulus  $N$ ) should be chosen as  $\Omega(\lambda^3)$  for  $\lambda$  bits of security, and no smaller than 400,000 bits at the 128-bit security level.

### 3.3 Experimental Results

We implemented our attack on the parameters proposed by [CLS14] (see Table 1), and on other sets of parameters for  $n \geq 2$ . The reduced basis for  $L^\perp$  is computed using the Nguyen–Stern algorithm (cf. Remark 1), and we choose  $\mathbf{u}$  among short enough vectors in the reduced basis such that  $\gcd(u_1, Q)$  is minimal. As reported in Table 2, the attack takes *much less than a second* for  $n = 2$ , and under 40s even for  $n = 5$  and a much larger  $\rho$ . On average, the number of candidates for  $m_1$  is always less than 2.

## 4 Ciphertext-Only Attack Against the CLS Scheme

We now present a somewhat stronger attack against the symmetric-key CLS encryption scheme, which works without any known plaintext. We assume that



**Table 2.** Known Plaintext Attack on the CLS scheme with message space  $\mathbb{Z}_{2^{256}}$  using  $(t-1)$  plaintext-ciphertext pairs (average value over 100 experiments using Sage [S+14] on a single 2.8 Ghz Intel CPU).

(a) Attack against the proposed parameters claiming 128 bits of security

Parameters	$t$	Time in seconds	Success rate	Average # of candidates
<b>Set-I</b>	4	0.005s	100%	1.21
<b>Set-II</b>	4	0.006s	100%	1.52
<b>Set-III</b>	4	0.007s	100%	1.33

(b) Various parameters for  $n \geq 2$  with  $\eta = 1536$

$n$	2		3		4		5	
$\rho$	1792	2688	3328	4224	4864	5760	6400	7296
$t$	4	14	6	28	8	42	11	58
Time in seconds	0.005s	0.122s	0.027s	1.95s	0.081s	10.8s	0.22s	39.1s
Success rate	100%	100%	100%	95%	100%	95%	100%	92%
Average # of candidates	1.21	1	1.08	1	1.07	1	1.03	1

we obtain the ciphertexts  $\mathbf{c}_i \leftarrow \text{Encrypt}(\text{sk}, m_i)$  corresponding to  $t$  messages  $m_1, \dots, m_t$  that are unknown but *small*, and we show that all the  $m_i$ 's can be recovered efficiently.

Combining this attack with the one from the previous section, this means that we can break the one-wayness of the symmetric-key CLS scheme without any known plaintexts, as long as we get a few ciphertexts associated with small messages (a very common situation in a homomorphic setting!).

From a technical standpoint, this stronger attack is still based on Nguyen–Stern orthogonal lattices, but uses the “doubly orthogonal” technique introduced in [NS97].

We present our attack in the sections below: we first present the attack in Sect. 4.1 in the case when  $n = 2$ , i.e.  $N = p_1 p_2$ . Then, we explain in Sect. 4.2 that it generalizes naturally to higher values of  $n$ .

#### 4.1 (Small) Message Recovery Using Known Ciphertexts for $N = p_1 p_2$

We use the same notation as in Sect. 3.1. Our attack proceeds in two steps:

- (1) Find  $t - 3$  short vectors  $\mathbf{u}_1, \dots, \mathbf{u}_{t-3}$  in the orthogonal lattice  $L^\perp$  to the lattice  $L = \mathbb{Z}\mathbf{C}_1 \oplus \mathbb{Z}\mathbf{C}_2$ . If the  $\mathbf{u}_i$  are short enough, we will get that  $\langle \mathbf{u}_i, \mathbf{m} + \mathbf{e} \cdot Q \rangle = 0$ .
- (2) Rewriting  $\langle \mathbf{u}_i, \mathbf{m} + \mathbf{e} \cdot Q \rangle = \langle \mathbf{u}_i, \mathbf{m} \rangle + Q \cdot \langle \mathbf{u}_i, \mathbf{e} \rangle = 0$  and reducing modulo  $Q$ , we get that  $\langle \mathbf{u}_i, \mathbf{m} \rangle = 0 \pmod{Q}$ . If  $\mathbf{u}_1, \dots, \mathbf{u}_{t-4}$  are short enough, the previous equation holds over  $\mathbb{Z}$  and  $\mathbf{m} \in (L')^\perp$  where  $L' = \mathbb{Z}\mathbf{u}_1 \oplus \dots \oplus \mathbb{Z}\mathbf{u}_{t-4}$ . One should recover the small vector  $\mathbf{m}$  as the shorter vector of  $(L')^\perp$ .

For the first step, we once again use the algorithm of Nguyen and Stern to obtain a basis  $\mathbf{u}_1, \dots, \mathbf{u}_{t-2}$  of  $L^\perp \subset \mathbb{Z}^t$  of rank  $t - 2$ . Similarly to Lemma 1, we have that a vector  $\mathbf{u}_i$  orthogonal to both  $\mathbf{C}_1$  and  $\mathbf{C}_2$  is either large, or orthogonal to  $\mathbf{k}_1, \mathbf{k}_2$  and  $\mathbf{m} + \mathbf{e} \cdot Q$ .

**Lemma 2.** *Let  $\mathbf{u} \in \mathbb{Z}^t$ . If  $\mathbf{u} \perp \mathbf{C}_1$  and  $\mathbf{u} \perp \mathbf{C}_2$ , then it verifies one of the following condition:*

- (1)  $\mathbf{u} \perp (\mathbf{m} + \mathbf{e} \cdot Q)$ ,  $\mathbf{u} \perp \mathbf{k}_1$  and  $\mathbf{u} \perp \mathbf{k}_2$ ;
- (2)  $\|\mathbf{u}\| \geq 2^{2(\eta-1)} / (Q \cdot 2^{\rho+1} \cdot t^{1/2})$ .

*Proof.* Let  $\mathbf{u} \in \mathbb{Z}^t$  be such that  $\mathbf{u} \perp \mathbf{C}_1$  and  $\mathbf{u} \perp \mathbf{C}_2$ , and not satisfying condition (2). Now

$$\begin{cases} 0 = \langle \mathbf{u}, \mathbf{C}_1 \rangle = \langle \mathbf{u}, \mathbf{m} + \mathbf{e} \cdot Q \rangle + p_1 \cdot \langle \mathbf{u}, \mathbf{k}_1 \rangle \\ 0 = \langle \mathbf{u}, \mathbf{C}_2 \rangle = \langle \mathbf{u}, \mathbf{m} + \mathbf{e} \cdot Q \rangle + p_2 \cdot \langle \mathbf{u}, \mathbf{k}_2 \rangle. \end{cases}$$

In particular,  $p = p_1 \cdot p_2 > 2^{2(\eta-1)}$  divides  $\langle \mathbf{u}, \mathbf{m} + \mathbf{e} \cdot Q \rangle$ . Now the Cauchy–Schwarz inequality yields

$$|\langle \mathbf{u}, \mathbf{m} + \mathbf{e} \cdot Q \rangle| \leq \|\mathbf{u}\| \cdot (\|\mathbf{m}\| + Q \cdot \|\mathbf{e}\|) < \|\mathbf{u}\| \cdot Q \cdot 2^{\rho+1} \cdot t^{1/2} < 2^{2(\eta-1)},$$

which implies  $\langle \mathbf{u}, \mathbf{m} + \mathbf{e} \cdot Q \rangle = 0$ , and thus  $\langle \mathbf{u}, \mathbf{k}_1 \rangle = \langle \mathbf{u}, \mathbf{k}_2 \rangle = 0$ . □

In particular, if  $\mathbf{u}_1, \dots, \mathbf{u}_{t-3}$  do not verify condition (2) of Lemma 2, they are such that  $\langle \mathbf{u}_i, \mathbf{m} + \mathbf{e} \cdot Q \rangle = 0$ .

For the second step, we similarly prove that if a vector  $\mathbf{u}$  is orthogonal to  $\mathbf{m} + \mathbf{e} \cdot Q$ , then it is either large or orthogonal to both  $\mathbf{m}$  and  $\mathbf{e}$ .

**Lemma 3.** *Let  $\mathbf{u} \in \mathbb{Z}^t$ . If  $\mathbf{u} \perp (\mathbf{m} + \mathbf{e} \cdot Q)$ , then it verifies one of the following condition:*

- (1)  $\mathbf{u} \perp \mathbf{m}$  and  $\mathbf{u} \perp \mathbf{e} \cdot Q$ ;
- (2)  $\|\mathbf{u}\| \geq Q / (2^\mu \cdot t^{1/2})$ .

In particular, if  $\mathbf{u}_1, \dots, \mathbf{u}_{t-4}$  do not verify condition (2) of Lemma 3, then  $\mathbf{m}, \mathbf{e}, \mathbf{k}_1$  and  $\mathbf{k}_2$  are in  $(L')^\perp$  where  $L' = \mathbb{Z}\mathbf{u}_1 \oplus \dots \oplus \mathbb{Z}\mathbf{u}_{t-4}$ . By applying Nguyen and Stern technique, one can hope to recover  $\mathbf{m}$  as the shortest vector of  $(L')^\perp$ .

We now discuss the conditions so that

- (a)  $\mathbf{u}_1, \dots, \mathbf{u}_{t-3}$  do not verify condition (2) of Lemma 2,
- (b)  $\mathbf{u}_1, \dots, \mathbf{u}_{t-4}$  do not verify condition (2) of Lemma 3.

Let us start with (a). For linearly independent  $\mathbf{m} + \mathbf{e} \cdot Q, \mathbf{k}_1$  and  $\mathbf{k}_2$ , the first condition of Lemma 2 cannot hold for all  $\mathbf{u}_k$  with  $k \in [t - 2]$  (for reasons of dimensions). In particular, the largest  $\mathbf{u}_k$ , say  $\mathbf{u}_{t-2}$ , must satisfy  $\|\mathbf{u}_{t-2}\| \geq 2^{2(\eta-1)} / (Q \cdot 2^{\rho+1} \cdot t^{1/2})$ . Now the other vectors form a lattice  $\Lambda = \mathbb{Z}\mathbf{u}_1 \oplus \dots \oplus \mathbb{Z}\mathbf{u}_{t-3}$  of rank  $t - 3$  and of volume

$$V = \text{vol}(\Lambda) \approx \frac{\text{vol}(L^\perp)}{\|\mathbf{u}_{t-2}\|} \leq \frac{\|\mathbf{C}_1\| \cdot \|\mathbf{C}_2\|}{2^{2(\eta-1)} / (Q \cdot 2^{\rho+1} \cdot t^{1/2})} \leq Q \cdot 2^{\rho+3} \cdot t^{3/2}.$$

Heuristically, we can expect  $\Lambda$  to behave as a random lattice; assuming the Gaussian heuristic, we should have  $\|\mathbf{u}_k\| \approx \sqrt{t-3} \cdot V^{1/(t-3)}$ . Thus, the condition for all the  $\mathbf{u}_k$ 's to be orthogonal to  $\mathbf{k}_1, \mathbf{k}_2$  and  $\mathbf{m} + \mathbf{e} \cdot Q$  becomes

$$t^{-1/2} \cdot 2^{-2} \cdot \left(Q \cdot 2^{\rho+3} \cdot t^{3/2}\right)^{1+1/(t-3)} \ll 2^{2 \cdot (\eta-1)} \leq 2^{2 \cdot \eta}.$$

Taking logarithms and ignoring logarithmic factors, this means:

$$t \gtrsim 3 + \frac{\rho + 3 + \log Q}{2\eta - \log Q - \rho - 3} = 3 + \frac{\alpha}{1 - \alpha} \quad \text{where} \quad \alpha = \frac{\rho + 3 + \log Q}{2\eta}. \quad (5)$$

In the following, we assume that condition (5) is satisfied; therefore the vectors  $\mathbf{m} + \mathbf{e} \cdot Q, \mathbf{k}_1$  and  $\mathbf{k}_2$  belong to  $\Lambda^\perp$ .

Next, let us focus on (b). Similarly, for linearly independent  $\mathbf{m}, \mathbf{e}, \mathbf{k}_1$  and  $\mathbf{k}_2$ , condition (2) of Lemma 3 cannot hold for all  $k \in [t-3]$ , and therefore  $\|\mathbf{u}_{t-3}\| \geq Q/(2^\mu \cdot t^{1/2})$ . Now we want to select  $t$  large enough so that all the  $\|\mathbf{u}_k\|$  for  $k \leq t-4$  verifies condition (1) of Lemma 3. We have that

$$\|\mathbf{u}_k\| = \mathcal{O}(t^2 \cdot Q^{1/(t-3)} \cdot 2^{(\rho+3)/(t-3)}),$$

so the  $\mathbf{u}_k$ 's do not verify condition (2) of Lemma 3 (and therefore verify condition (1)) when

$$t^{5/2} \cdot Q^{1/(t-3)} \cdot 2^{(\rho+3)/(t-3)} \cdot 2^\mu \ll Q.$$

Taking logarithms and ignoring logarithmic factors, this means:

$$t \gtrsim 3 + \frac{\log Q + \rho + 3}{\log Q - \mu}. \quad (6)$$

Finally, assuming condition (6) is satisfied,  $\mathbf{m}$  is a really short vector (of norm  $\approx 2^\mu \cdot t^{1/2}$ ) orthogonal to  $\mathbf{u}_k$  for all  $k \in [t-4]$ . It follows that one should recover  $\mathbf{m}$  as the first vector of the reduced basis of  $(L')^\perp$ , at least in the case of small lattice dimensions. Our experiments, presented in Sect. 4.3, show that this condition is well verified in practice.

Moreover, one can carry out an asymptotic analysis as in Sect. 3 to take larger lattice dimensions into account. The computations are very similar, but due to the heuristic nature of the present attack, they are less meaningful.

### 4.2 Generalization to $n \geq 2$

Once again, our technique generalizes directly to  $n \geq 2$ . The steps of the generalized attack are similar:

- (1) Find  $t - n - 1$  short vectors  $\mathbf{u}_1, \dots, \mathbf{u}_{t-n-1}$  in the orthogonal lattice  $L^\perp$  to the lattice  $L = \mathbb{Z}\mathbf{C}_1 \oplus \dots \oplus \mathbb{Z}\mathbf{C}_n$ . If the  $\mathbf{u}_i$  are short enough, we will get that  $\langle \mathbf{u}_i, \mathbf{m} + \mathbf{e} \cdot Q \rangle = 0$  (and  $\langle \mathbf{u}_i, \mathbf{k}_j \rangle = 0$  for all  $j \in [n]$ ).

**Table 3.** Attack of Sect. 4.1 on the CLS scheme with message space  $\mathbb{Z}_{2^{256}}$  (average value over 500 experiments using Sage [S+14] on a single 3.4 Ghz Intel Core i7 CPU).

(a) **Parameter Set-I**

$\mu$	0	16	32	64	128	192	224
Minimal $t$ from Eq. (6)	12	12	13	14	20	36	68
Minimal $t$ in practice	12	12	13	14	20	39	80
Running time (in seconds)	0.16s	0.16s	0.21s	0.28s	1.10s	13.9s	169s
Success rate	100%						

(b) **Parameter Set-II**

$\mu$	0	16	32	64	128	192	224
Minimal $t$ from Eq. (6)	14	15	16	18	25	46	88
Minimal $t$ in practice	14	15	16	18	25	47	98
Running time (in seconds)	0.38s	0.49s	0.62s	0.99s	3.65s	37.1s	521s
Success rate	100%						72.8%

(2) Rewriting  $\langle \mathbf{u}_i, \mathbf{m} + \mathbf{e} \cdot Q \rangle = \langle \mathbf{u}_i, \mathbf{m} \rangle + Q \cdot \langle \mathbf{u}, \mathbf{e} \rangle = 0$  and reducing modulo  $Q$ , we get that  $\langle \mathbf{u}_i, \mathbf{m} \rangle = 0 \pmod Q$ . If  $\mathbf{u}_1, \dots, \mathbf{u}_{t-n-2}$  are short enough, the previous equation holds over  $\mathbb{Z}$  and  $\mathbf{m} \in (L')^\perp$  where  $L' = \mathbb{Z}\mathbf{u}_1 \oplus \dots \oplus \mathbb{Z}\mathbf{u}_{t-n-2}$ . One should recover the small vector  $\mathbf{m}$  as the shorter vector of  $(L')^\perp$ .

Condition (6) becomes:

$$t \gtrsim n + 1 + \frac{\log Q + \rho + n + 1}{\log Q - \mu}.$$

### 4.3 Experimental Results

We ran our attacks against the parameters of Table 1. Once again, our attack is really efficient; it amounts to applying LLL twice (cf. Remark 1) and runs in a matter of seconds. Results are collected in Table 3.

## 5 Breaking the Search Co-ACD Assumption

In this section, we break the search Co-ACD assumption when  $N = \prod_i p_i$  and  $Q$  are known (as in the public-key CLS scheme): given a few samples  $\{(e_i \cdot Q \pmod{p_1}, \dots, e_i \cdot Q \pmod{p_n})\}_i$  from the Co-ACD distribution, we show that one can recover the  $p_i$ 's in heuristic polynomial time, at least for certain ranges of parameters. In particular, in the public-key CLS encryption scheme, the private key can be recovered from the public key alone!

### 5.1 Description of the Attack

For simplicity, we first consider the case  $n = 2$  (as in the CLS scheme). We use the same notation as in Sect. 4 with  $\mathbf{m} = \mathbf{0}$ , and assume that  $N = p_1 p_2$  is known. Hence, we have that

$$(\mathbf{C}_1 - \mathbf{e} \cdot Q) \cdot (\mathbf{C}_2 - \mathbf{e} \cdot Q) = \mathbf{0} \pmod N, \tag{7}$$

where the multiplication  $\cdot$  is done componentwise. We start from the following equation:

$$\mathbf{e} \cdot Q = (\mathbf{C}_1 - \mathbf{C}_2) \cdot \bar{p}_1 + \mathbf{C}_2 \pmod N, \tag{8}$$

where  $\bar{p}_1 = p_2 \cdot (p_2^{-1} \pmod{p_1}) \pmod N$  is the first CRT coefficient for  $(p_1, p_2)$ . Multiplying  $Q^{-1} \pmod N$ , we obtain

$$\mathbf{e} = (\mathbf{C}_1 - \mathbf{C}_2) \cdot \bar{p}_1 Q^{-1} + \mathbf{C}_2 \cdot Q^{-1} \pmod N.$$

Similar to the lattice used against the ACD problem in [DGHV10], considering the above equation, we construct a lattice  $L$  generated by the rows of the following  $(t + 2) \times (t + 1)$  matrix:

$$\begin{pmatrix} \mathbf{C}_1 - \mathbf{C}_2 & \mathbf{0} \\ \mathbf{C}_2 \cdot Q^{-1} \pmod N & 2^\rho \\ N \cdot I_{t \times t} & \mathbf{0} \end{pmatrix}$$

Since  $\mathbb{Z}^{t+1}/L$  is the quotient of  $(\mathbb{Z}/N\mathbb{Z})^t \times (\mathbb{Z}/2^\rho\mathbb{Z})$  by  $(\mathbf{C}_1 - \mathbf{C}_2 \pmod N, 0)$ , we have  $\text{vol}(L) = 2^\rho \cdot N^{t-1}$  with overwhelming probability. Moreover, the lattice  $L$  contains the following short distinguished vectors:

$$\mathbf{v}_1 = (\mathbf{C}_1 - \mathbf{C}_2, 0) \text{ and } \mathbf{v}_2 = (\mathbf{e}, 2^\rho), \tag{9}$$

of respective norms  $\|\mathbf{v}_1\| \approx 2^\eta$  and  $\|\mathbf{v}_2\| \approx 2^\rho$ , and when  $t$  is large enough, we expect those vectors to be much shorter than other independent vectors in  $L$  (see the discussion below). As a result, if  $\mathbf{x}_1$  and  $\mathbf{x}_2$  are the first two vectors of a reduced basis of the lattice  $L$ , we expect to have, up to some explicit sign change,  $\mathbf{v}_1 = \mathbf{x}_1$  and:

$$(\mathbf{e}, 2^\rho) = \mathbf{v}_2 = \mathbf{x}_2 + \alpha \mathbf{x}_1 \tag{10}$$

for some unknown integer coefficient  $\alpha \in \mathbb{Z}$ .

Now, plugging the previous equality into Eq. (7) and considering the first components of the corresponding vectors, we obtain:

$$(c_{1,1} - Q(x_{2,1} + \alpha \cdot x_{1,1})) \cdot (c_{2,1} - Q(x_{2,1} + \alpha \cdot x_{1,1})) = 0 \pmod N.$$

This yields a univariate quadratic equation modulo  $N$  which admits  $\alpha$  as a solution. Moreover, that solution  $\alpha$  is short, in the sense that

$$|\alpha| = \frac{\|\mathbf{v}_2 - \mathbf{x}_2\|}{\|\mathbf{x}_1\|} \leq \frac{\|\mathbf{v}_2\| + \|\mathbf{x}_2\|}{\|\mathbf{x}_1\|} \lesssim 2^{1+\rho-\eta} < \sqrt{N}.$$

As a result, we can use (the original, univariate version of) Coppersmith’s theorem [Cop97] to solve this equation in polynomial time, obtain  $\alpha$ , and recover  $\mathbf{e}$  from (10). It is then straightforward to factor  $N$  by computing  $\gcd(\mathbf{C}_1 - \mathbf{e} \cdot Q, N)$  componentwise.

Finally, we analyze how  $t$  should be chosen. Since our target vector  $\mathbf{v}_2$  is much longer than the shortest vector  $\mathbf{v}_1$ , the best we can hope is that the second shortest vector in  $L$  is  $\mathbf{v}_2$  (modulo  $\mathbf{v}_1$ ). Using  $\lambda_1(L) \approx 2^\eta$ ,  $\text{vol}(L) = 2^\rho \cdot N^{t-1} \approx 2^{\rho+2\eta(t-1)}$ , and  $(\prod_{i=1}^{t+1} \lambda_i(L))^{1/(t+1)} \leq \sqrt{t+1} \text{vol}(L)^{1/(t+1)}$ , we know that  $\lambda_2(L) \lesssim (\text{vol}(L)/\lambda_1(L))^{1/t} \approx (2^{\rho+2\eta(t-1)-\eta})^{1/t} = 2^{2\eta + \frac{\rho-3\eta}{t}}$ . If this quantity is much larger than  $\|\mathbf{v}_2\| \approx 2^\rho$ , we can expect to find  $\mathbf{v}_2$  in  $L$  (modulo  $\mathbf{v}_1$ ). This yields the following condition on  $t$  when our attack works:

$$t > \frac{3\eta - \rho}{2\eta - \rho}. \tag{11}$$

### 5.2 Extension to $n \geq 3$

In this section, we extend the attack against search Co-ACD assumption to the case  $n \geq 3$ . Unlike the case  $n = 2$ , we will see that this extended attack is only applicable in a certain range for  $\rho$ , but it always breaks non trivial instances of the search Co-ACD problem.

Similar to the case  $n = 2$ , we start from the following equation:

$$\mathbf{e} \cdot Q = (\mathbf{C}_1 - \mathbf{C}_n) \cdot \bar{p}_1 + \dots + (\mathbf{C}_{n-1} - \mathbf{C}_n) \cdot \bar{p}_{n-1} + \mathbf{C}_n \pmod{N}, \tag{12}$$

where the  $\bar{p}_i$ ’s are the CRT coefficients  $\bar{p}_i = \frac{N}{p_i} \cdot (\frac{p_i}{N} \text{ mod } p_i)$ . Multiplying  $Q^{-1} \text{ mod } N$ , we again get

$$\mathbf{e} = (\mathbf{C}_1 - \mathbf{C}_n) \cdot \bar{p}_1 Q^{-1} + \dots + (\mathbf{C}_{n-1} - \mathbf{C}_n) \cdot \bar{p}_{n-1} Q^{-1} + \mathbf{C}_n \cdot Q^{-1} \pmod{N}.$$

Therefore, if we consider the lattice  $L$  generated by the rows of the following matrix:

$$\begin{pmatrix} \mathbf{C}_1 - \mathbf{C}_n & 0 \\ \vdots & \vdots \\ \mathbf{C}_{n-1} - \mathbf{C}_n & 0 \\ \mathbf{C}_n \cdot Q^{-1} \text{ mod } N & 2^\rho \\ N \cdot I_{t \times t} & \mathbf{0} \end{pmatrix}$$

it is full rank and contains the following short distinguished vectors:  $\mathbf{v}_i = (\mathbf{C}_i - \mathbf{C}_n, 0)$  for  $i = 1, \dots, n-1$ , which are all of norm  $\approx 2^\eta$ , and  $\mathbf{v}_n = (\mathbf{e}, 2^\rho)$  of norm  $\approx 2^\rho$ . With high probability, these vectors are linearly independent, and when  $t$  is large enough, we expect them to be much shorter than other independent vectors in the lattice (see the discussion below).

As a result, and since  $\mathbf{v}_n$  is much longer than the  $\mathbf{v}_i$ ’s for  $i < n$ , applying lattice reduction to  $L$  should yield a reduced basis  $(\mathbf{x}_1, \dots, \mathbf{x}_{t+1})$  such

that  $\bigoplus_{i=1}^r \mathbb{Z}\mathbf{x}_i = \bigoplus_{i=1}^r \mathbb{Z}\mathbf{v}_i$  for  $r = n - 1$  and  $r = n$ . In particular,  $(\mathbf{v}_1, \dots, \mathbf{v}_{n-1}, \mathbf{x}_n, \dots, \mathbf{x}_{t+1})$  is a basis of  $L$ , and writing  $\mathbf{v}_n$  over that basis yields:

$$(\mathbf{e}, 2^\rho) = \mathbf{v}_n = \alpha \mathbf{v}_1 + \mathbf{y}$$

for some  $\alpha \in \mathbb{Z}$  and  $\mathbf{y} \in \mathbb{Z}\mathbf{v}_2 \oplus \dots \oplus \mathbb{Z}\mathbf{v}_{n-1} \oplus \mathbb{Z}\mathbf{x}_n$ . Plugging that relation into Eq. (12) gives:

$$(\alpha \mathbf{v}_1 + \mathbf{y}) \cdot Q \equiv \bar{p}_1 \mathbf{v}_1 + \dots + \bar{p}_{n-1} \mathbf{v}_{n-1} + \mathbf{w} \pmod{N}$$

where  $\mathbf{w} = (\mathbf{C}_n, 2^\rho Q)$ . Now choose a vector  $\mathbf{u} \in \mathbb{Z}^{t+1}$  orthogonal to  $\mathbf{v}_2, \dots, \mathbf{v}_{n-1}, \mathbf{x}_n$  but not to  $\mathbf{v}_1$  modulo  $N$  (such a vector exists with overwhelming probability, and when it does, it can be found in deterministic polynomial time using the Nguyen–Stern algorithm [NS97]). Taking the inner product with  $\mathbf{u}$  yields:

$$Q\alpha \cdot \langle \mathbf{v}_1, \mathbf{u} \rangle + 0 \equiv \bar{p}_1 \langle \mathbf{v}_1, \mathbf{u} \rangle + 0 + \dots + 0 + \langle \mathbf{w}, \mathbf{u} \rangle \pmod{N},$$

or equivalently:

$$\bar{p}_1 \equiv Q\alpha + \omega \pmod{N} \quad \text{where} \quad \omega = -\frac{\langle \mathbf{w}, \mathbf{u} \rangle}{\langle \mathbf{v}_1, \mathbf{u} \rangle} \pmod{N}. \tag{13}$$

Moreover,  $\alpha$  is still small compared to  $N$ , of size about  $\rho - \eta$  bits. Therefore, we can proceed as before and deduce a polynomial relation from (13) so as to apply Coppersmith’s theorem to recover  $\alpha$ . We propose two ways of doing so. Note that once  $\alpha$  is found, we obtain a non trivial factor of  $N$  straight away by computing  $\gcd(Q\alpha + \omega, N) = N/p_1$ .

One first approach to computing  $\alpha$  is to observe that  $\bar{p}_1$  is an idempotent element of  $\mathbb{Z}_N$ : it satisfies  $\bar{p}_1^2 \equiv \bar{p}_1 \pmod{N}$ . It follows that  $\alpha$  is a root of the quadratic polynomial  $F_2(X) = (Q \cdot X + \omega)^2 - (Q \cdot X + \omega)$  modulo  $N$ . It is thus possible to compute  $\alpha$  in polynomial time using Coppersmith’s theorem when  $2^{\rho-\eta} < \sqrt{N} \approx 2^{n\eta/2}$ , i.e.  $\rho < \frac{n+2}{2} \cdot \eta$ . Since we already know that  $\rho > (n - 1)\eta$  for security, that condition is only non trivial for  $n = 2$  (providing a slightly different formulation of the attack from the previous section) and  $n = 3$  (in which case we can break parameters  $\rho < 5\eta/2$ ).

A second approach is to see that Eq. (13) implies:

$$Q\alpha + \omega \equiv 0 \pmod{N/p_1}.$$

Therefore,  $\alpha$  is a small root of the linear polynomial  $F_1(X) = Q \cdot X + \omega$  modulo some large unknown factor of  $N$  of size  $\approx N^{1-1/n}$ . Alexander May’s extension of Coppersmith’s theorem guarantees that we can then recover  $\alpha$  in deterministic polynomial time provided that  $2^{\rho-\eta} < N^{(1-1/n)^2} \approx 2^{(n-2+1/n)\eta}$ , i.e.  $\rho < (n - 1 + 1/n)\eta$ . That condition is always non trivial, and thus we obtain an attack for all values of  $n$ . For  $n = 3$ , however, the previous approach should be preferred as it gives a better bound for  $\rho$  ( $5\eta/2$  instead of  $7\eta/3$ ).

Finally, let us evaluate the condition on  $t$  for the attack to succeed. As before, the condition says that the  $n$ -th minimum of the lattice  $L$  should

**Table 4.** Attack of Sect. 5.1 on the search Co-ACD assumption with  $Q = 2^{256}$  (average value over 100 experiments using Sage on a single 2.8 Ghz Intel CPU).

$\rho$	1792	2192	2592	2792	2892	2992
Minimal $t$ from Eq. (11)	3	3	5	7	10	21
Minimal $t$ in practice	3	3	5	7	10	22
Running time of the attack (in seconds)	0.31s	0.26s	1.07s	1.07s	17.3s	1886s
Success rate	100%			99%		86%

(a)  $\eta = 1536$  ( $\rho = 1792$  for 128 bits of security)

$\rho$	2450	2950	3450	3700	3950	4200
Minimal $t$ from Eq. (11)	3	3	4	5	7	13
Minimal $t$ in practice	3	3	4	5	7	14
Running time of the attack (in seconds)	0.57s	0.55s	0.41s	2.0s	2.1s	203s
Success rate	100%					

(b)  $\eta = 2194$  ( $\rho = 2450$  for 128 bits of security)

be at least  $2^\rho$ , while the first  $n - 1$  minima are at most  $2^n$ . The volume of  $L$  is  $\text{vol}(L) = 2^\rho \cdot N^{t-n+1}$ , and the expected  $n$ -th minimum is roughly  $\ell = (\text{vol}(L)/2^{(n-1)\eta})^{1/(t+1-(n-1))}$ . Thus, the condition can be written as:  $(t - n + 2) \cdot \rho < \rho + (t - n + 1) \cdot n\eta - (n - 1)\eta$ , or equivalently:

$$t > \frac{(n + 1)\eta - \rho}{n\eta - \rho} \cdot (n - 1),$$

which is a direct generalization of Condition (11). For  $n \geq 4$ , since our best attack only works for  $\rho < (n - 1 + 1/n)\eta$ , this condition simplifies to  $t > \frac{(n+1)-(n-1+1/n)}{n-(n-1+1/n)}(n - 1) = 2n - 1$ , i.e.  $t \geq 2n$ .

### 5.3 Experimental Results

We have implemented the attack of Sect. 5.1 in Sage. Timings are reported in Table 4. The initial lattice reduction step is very fast, and the Coppersmith computation, where most of the CPU time is spent, also takes on the order of seconds at most for practically all parameters we tested (despite the fact that Sage’s `small_roots` command is relatively poorly optimized compared to more recent implementation efforts such as [BCF+14]).

We also implemented the attack for larger  $n$ , and found for example that  $N$  can be factored in a few seconds with only 5 samples for  $(n, \eta, \rho) = (3, 1000, 2300)$ .

**Acknowledgments.** The authors thank Jung Hee Cheon, Paul Kirchner, Changmin Lee, Guénaél Renault, Jae Hong Seo, and Yong Soo Song for helpful discussions. The second author was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (No. 2012R1A1A2039129).



## References

- [BCF+14] Bi, J., Coron, J.-S., Faugère, J.-C., Nguyen, P.Q., Renault, G., Zeitoun, R.: Rounding and chaining LLL: finding faster small roots of univariate polynomial congruences. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 185–202. Springer, Heidelberg (2014)
- [Bro01] Broughan, K.A.: The gcd-sum function. *J. Integer Sequences* **4**, 01.2.2, 19 (2001)
- [CCK+13] Cheon, J.H., Coron, J.-S., Kim, J., Lee, M.S., Lepoint, T., Tibouchi, M., Yun, A.: Batch fully homomorphic encryption over the integers. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 315–335. Springer, Heidelberg (2013)
- [CH12] Cohn, H., Heninger, N.: Approximate common divisors via lattices. In: ANTS X (2012)
- [CLS14] Cheon, J.H., Lee, H.T., Seo, J.H.: A new additive homomorphic encryption based on the Co-ACD problem. In: Ahn, G.-J., Yung, M., Li, N. (eds.) ACM CCS, pp. 287–298. ACM, New York (2014)
- [CLT13] Coron, J.-S., Lepoint, T., Tibouchi, M.: Practical multilinear maps over the integers. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 476–493. Springer, Heidelberg (2013)
- [CLT14] Coron, J.-S., Lepoint, T., Tibouchi, M.: Scale-invariant fully homomorphic encryption over the integers. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 311–328. Springer, Heidelberg (2014)
- [CN11] Chen, Y., Nguyen, P.Q.: BKZ 2.0: Better lattice security estimates. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 1–20. Springer, Heidelberg (2011)
- [CN12] Chen, Y., Nguyen, P.Q.: Faster algorithms for approximate common divisors: breaking fully-homomorphic-encryption challenges over the integers. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 502–519. Springer, Heidelberg (2012)
- [CNT10] Coron, J.-S., Naccache, D., Tibouchi, M.: Fault attacks against EMV signatures. In: Pieprzyk, J. (ed.) CT-RSA 2010. LNCS, vol. 5985, pp. 208–220. Springer, Heidelberg (2010)
- [CNT12] Coron, J.-S., Naccache, D., Tibouchi, M.: Public key compression and modulus switching for fully homomorphic encryption over the integers. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 446–464. Springer, Heidelberg (2012)
- [Cop97] Coppersmith, D.: Small solutions to polynomial equations, and low exponent RSA vulnerabilities. *J. Cryptology* **10**(4), 233–260 (1997)
- [DGHV10] van Dijk, M., Gentry, C., Halevi, S., Vaikuntanathan, V.: Fully homomorphic encryption over the integers. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 24–43. Springer, Heidelberg (2010)
- [FLLT14] Fouque, P.-A., Lee, M.S., Lepoint, T., Tibouchi, M.: Cryptanalysis of the Co-ACD assumption. *Cryptology ePrint Archive*. Full version of this paper, Report 2014/1024 (2014). <http://eprint.iacr.org/>
- [How01] Howgrave-Graham, N.: Approximate integer common divisors. In: Silverman, J.H. (ed.) CaLC 2001. LNCS, vol. 2146, p. 51. Springer, Heidelberg (2001)
- [JL13] Joye, M., Libert, B.: Efficient cryptosystems from  $2^k$ -th power residue symbols. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 76–92. Springer, Heidelberg (2013)

- [LLL82] Lenstra, A.K., Lenstra Jr., H.W., Lovász, L.: Factoring polynomials with rational coefficients. *Math. Ann.* **261**(4), 515–534 (1982)
- [LT15] Lepoint, T., Tibouchi, M.: Cryptanalysis of a (somewhat) additively homomorphic encryption scheme used in PIR. In: WAHC (2015)
- [May03] May, A.: New RSA Vulnerabilities Using Lattice Reduction Methods. Ph.D. thesis, University of Paderborn (2003)
- [MR09] Micciancio, D., Regev, O.: Lattice-based cryptography. In: Bernstein, D.J., Buchmann, J., Dahmen, E. (eds.) *Post-Quantum Cryptography*, pp. 147–191. Springer, Berlin (2009)
- [Ngu10] Nguyen, P.Q.: Hermite’s constant and lattice algorithms. In: Nguyen, P.Q., Vallée, B. (eds.) *The LLL Algorithm, Information Security and Cryptography*, pp. 19–69. Springer, Berlin (2010)
- [NLV11] Naehrig, M., Lauter, K.E., Vaikuntanathan, V.: Can homomorphic encryption be practical? In: Cachin, C., Ristenpart, T. (eds), *ACM CCSW*, pp. 113–124, ACM (2011)
- [NS97] Nguyen, P.Q., Stern, J.: Merkle-hellman revisited: a cryptanalysis of the quanstone cryptosystem based on group factorizations. In: Kaliski Jr., B.S. (ed.) *CRYPTO 1997*. LNCS, vol. 1294, pp. 198–212. Springer, Heidelberg (1997)
- [NS98a] Nguyen, P.Q., Stern, J.: The béguin-quisquater server-aided RSA protocol from crypto 1995 is not secure. In: Ohta, K., Pei, D. (eds.) *ASIACRYPT 1998*. LNCS, vol. 1514, pp. 372–379. Springer, Heidelberg (1998)
- [NS98b] Nguyen, P.Q., Stern, J.: Cryptanalysis of a fast public key cryptosystem presented at SAC 1997. In: Tavares, S., Meijer, H. (eds.) *SAC 1998*. LNCS, vol. 1556, p. 213. Springer, Heidelberg (1999)
- [NS99] Nguyen, P.Q., Stern, J.: The hardness of the hidden subset sum problem and its cryptographic implications. In: Wiener, M. (ed.) *CRYPTO 1999*. LNCS, vol. 1666, p. 31. Springer, Heidelberg (1999)
- [NS01] Nguyen, P.Q., Stern, J.: The two faces of lattices in cryptology. In: Silverman, J.H. (ed.) *CaLC 2001*. LNCS, vol. 2146, p. 146. Springer, Heidelberg (2001)
- [NT12] Nguyen, P.Q., Tibouchi, M.: Lattice-based fault attacks on signatures. In: Joye, M., Tunstall, M. (eds.) *Fault Analysis in Cryptography, Information Security and Cryptography*, pp. 201–220. Springer, Berlin (2012)
- [Pai99] Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) *EUROCRYPT 1999*. LNCS, vol. 1592, p. 223. Springer, Heidelberg (1999)
- [S+14] Stein, W. et al.: Sage Mathematics Software (Version 6.4). The Sage Development Team (2014). <http://www.sagemath.org>
- [vdPS13] van de Pol, J., Smart, N.P.: Estimating key sizes for high dimensional lattice-based systems. In: Stam, M. (ed.) *IMACC 2013*. LNCS, vol. 8308, pp. 290–303. Springer, Heidelberg (2013)

# Last Fall Degree, HFE, and Weil Descent Attacks on ECDLP

Ming-Deh A. Huang<sup>1</sup>, Michiel Kisters<sup>2</sup>(✉), and Sze Ling Yeo<sup>3</sup>

<sup>1</sup> USC, Los Angeles, California  
mdhuang@usc.edu

<sup>2</sup> TL@NTU, Singapore, Singapore  
kisters@gmail.com

<sup>3</sup> I2R, Singapore, Singapore  
slyeo@i2r.a-star.edu.sg

**Abstract.** Weil descent methods have recently been applied to attack the Hidden Field Equation (HFE) public key systems and solve the elliptic curve discrete logarithm problem (ECDLP) in small characteristic. However the claims of quasi-polynomial time attacks on the HFE systems and the subexponential time algorithm for the ECDLP depend on various heuristic assumptions.

In this paper we introduce the notion of the last fall degree of a polynomial system, which is independent of choice of a monomial order. We then develop complexity bounds on solving polynomial systems based on this last fall degree.

We prove that HFE systems have a small last fall degree, by showing that one can do division with remainder after Weil descent. This allows us to solve HFE systems unconditionally in polynomial time if the degree of the defining polynomial and the cardinality of the base field are fixed. For the ECDLP over a finite field of characteristic 2, we provide computational evidence that raises doubt on the validity of the first fall degree assumption, which was widely adopted in earlier works and which promises sub-exponential algorithms for ECDLP. In addition, we construct a Weil descent system from a set of summation polynomials in which the first fall degree assumption is unlikely to hold. These examples suggest that greater care needs to be exercised when applying this heuristic assumption to arrive at complexity estimates.

These results taken together underscore the importance of rigorously bounding last fall degrees of Weil descent systems, which remains an interesting but challenging open problem.

**Keywords:** HFE · ECDLP · Weil descent · Solving equations · First fall degree · Last fall degree

# 1 Introduction

## 1.1 Zero-Dimensional Polynomial Systems and Weil Descent Attacks

Zero-dimensional multivariate polynomial systems over finite fields arise in many practical areas of interest including in cryptography and coding theory. As such, solving these systems has both practical and theoretical interest. For instance, a major application is in the design of multivariate cryptosystems. One of the earliest proposals for the multivariate cryptosystem was the HFE public-key cryptosystem [21]. In recent years, polynomial solving also arises in elliptic curve cryptography, specifically in the index calculus approach to solve the elliptic curve discrete logarithm problem (ECDLP).

Many different approaches had been proposed to solve multivariate polynomial equations over finite fields. The most common approach for generic polynomial systems is via Gröbner basis algorithms [1, 9, 10]. Typically, a Gröbner basis with respect to the degree reverse lexicographical ordering is first computed via algorithms  $F_4$  or  $F_5$  [9, 10]. It is then converted to a Gröbner basis with respect to the lexicographical ordering by algorithms such as the FGLM algorithm [11] which contains equations where variables are eliminated. This enables the variables to be solved one at a time. In general, it is very difficult to determine the complexity of the Gröbner basis algorithm. Various authors have used the term “the degree of regularity” to describe properties of a system that can be used to obtain complexity results. However not all definitions of this term are equivalent.

Another approach to solve multivariate polynomial systems is the XL algorithm and its variants [2–5, 17]. This class of algorithms performs well when the system under consideration is overdetermined, that is, the number of equations far exceeds the number of variables.

In this present paper, we first introduce the notion of the last fall degree of a polynomial system over a finite field. Our definition is intrinsic to the polynomial system itself, independent of the choice of a monomial order. With this notion at our disposal, we present an explicit algorithm to find all the roots of a zero-dimensional multivariate polynomial system, bounding the complexity by the last fall degree.

When the polynomial systems are over a field of cardinality  $q^n$ , where  $q$  is a prime power and  $n$  a positive integer, one can convert this system via Weil descent to a system over its subfield with  $q$  elements (see Sect. 3 for more details). This results in a polynomial system over a smaller field, but at the expense of more variables. For example, Weil descent has been adopted to solve the HFE system as well as the index calculus method for ECDLP. In this paper, we will describe Weil descent systems arising from a polynomial in one variable and study the relations among various polynomial systems. Analogous definitions hold for a multivariate polynomial system.

## 1.2 The HFE Cryptosystem

Let  $k$  be a finite field of cardinality  $q^n$ , with subfield  $k'$  of cardinality  $q$ . Let  $f \in k[X]$  be a polynomial over  $k$  with a relatively small degree. Using factorization

algorithms, one can easily factorize this polynomial to find its roots in  $k$ . One can transform this system using Weil descent and two transformations into a system in  $n$  variables over  $k'$ . At first glance, this system seems to be hard to solve and this is the basis of the Hidden Field Equations (HFE) cryptosystem (see [21] and Subsect. 4.1). Computational and heuristic evidence show that such a system is not secure [7, 8, 16]: the degree of regularity of a Weil descent system is small and does not depend on  $n$  and hence the system can be solved efficiently using Gröbner basis algorithms. In particular in [16], the authors claimed that the HFE system can be solved efficiently under a heuristic assumption on the complexity of the Gröbner basis computations to solve the system. More recently, Christophe Petit, in a preprint [22], gives a proof of this observation by doing manipulations using his successive resultant algorithm on the descent side. In this article, we prove that HFE systems have a small last fall degree, by showing that one can do division with remainder after Weil descent. This allows us to solve the HFE systems unconditionally in polynomial time if the degree of the defining polynomial and the cardinality of the base field are fixed.

We have a natural right action of  $\text{Aff}_n(k') = k'^n \times \text{GL}_n(k')$  on the ring  $R' = k'[Y_0, \dots, Y_{n-1}]$  by acting as affine change of coordinates. If  $M \in \text{Aff}_n(k')$  and  $g \in R'$  we write  $gM$  for this action. The main theorem is the following, which allows one to solve HFE systems efficiently. We stress that our results hold for a larger class of polynomial systems as we do not require the resulting Weil descent system to be quadratic.

For  $r \in \mathbb{Z}_{\geq 0}$  and  $c \in \mathbb{Z}_{\geq 1}$ , we set

$$\psi(r, c) = \max(\lfloor 2(c - 1)(\log_c(r) + 1) \rfloor, 0).$$

**Main Theorem 1.** *Let  $q$  be a prime power and let  $k$  be a finite field of cardinality  $q^n$  with subfield  $k'$  of cardinality  $q$ . Let  $f \in k[X]$  nonzero which has at most  $e$  different roots over  $k$  and let  $\mathcal{F} = \{f\}$ . Let  $\mathcal{F}'_f \subset k'[Y_0, \dots, Y_{n-1}]$  be a Weil descent system of  $\mathcal{F}$  (Subsect. 3.1). Let  $M \in \text{Aff}_n(k')$ ,  $N \in \text{GL}_n(k')$ . Define  $g_i$ ,  $i = 0, \dots, n - 1$ , by*

$$\begin{pmatrix} g_0 \\ \vdots \\ g_{n-1} \end{pmatrix} = N \begin{pmatrix} [f]_0 M \\ \vdots \\ [f]_{n-1} M \end{pmatrix}.$$

*Set  $d = \max(\psi(\deg(f), q), q, e)$ . Then given  $\mathcal{G} = \{g_0, \dots, g_{n-1}, Y_0^q - Y_0, \dots, Y_{n-1}^q - Y_{n-1}\} \subset k'[Y_0, \dots, Y_{n-1}]$ , one can deterministically find all solutions to  $\mathcal{G}$  in time polynomial in  $(n + d)^d$ .*

If one fixes  $q$  and  $\deg(f)$ , then the complexity to solve systems in Main Theorem 1 is polynomial in  $n$ . Note that  $e \leq \deg(f)$ , but usually it is much smaller. Furthermore, in practical applications, one wants  $e$  to be small, say bounded by a constant: in this case one can solve the above system in quasi-polynomial time if  $q$  is fixed and  $\deg(f)$  grows like  $n^\alpha$ .

It is an open question whether variants of HFE, such as HFEv-, can be attacked by our approach.

### 1.3 Polynomial Systems from ECDLP

A major application of solving multivariate polynomial equations over a finite field  $k$  of cardinality  $q^n$  is in the relation search step of the index calculus algorithm for elliptic curves over the field [6, 14, 23]. Indeed, let  $E : y^2 + a_1xy + a_3 = x^3 + a_2x^2 + a_4x + a_6$ , where  $a_1, a_2, a_3, a_4, a_6 \in k$ , be an elliptic curve defined over  $k$ . Let  $P$  be a point on  $E$  and let  $Q$  be a point in the cyclic group generated by  $P$ . The elliptic curve discrete logarithm problem seeks for an integer  $a$  such that  $Q = aP$ .

The most important step in the index calculus approach is to generate sufficiently many relations among suitable points on the elliptic curve  $E$ . To this end, summation polynomials provide a way to achieve this (see Subsect. 5.1). In particular, this transforms the problem of finding relations among points to solving a system of polynomial equations over  $k$  via the summation polynomials.

Cryptographic applications of Weil descent were first suggested by Frey [12], and Weil descent attacks were initially applied to elliptic curves of composite degrees over  $\mathbb{F}_2$  [12, 15]. In [6, 14], Weil descent was exploited to solve the ECDLP by applying the Weil descent to the summation polynomials over  $k$ . In [6], for instance, sub-exponential time estimates via this Weil descent approach were obtained for certain classes of  $q$  and  $n$ . Here, the author relied on a geometric approach by Rojas to solve a Weil descent system.

In [23], Petit et al. studied Weil descent systems arising from polynomial systems over fields of characteristic 2. Their results are based on a certain heuristic assumption, called the first fall degree assumption, which asserts that the first fall degree of a polynomial system is close to the degree of regularity. More specifically, they obtained a sub-exponential time complexity of  $2^{O(n^{2/3} \log n)}$  on the basis of this assumption.

In this article, we provide computational evidence that raises doubt on the validity of the first fall degree assumption when applied to elliptic curves over fields of characteristic 2. In addition, we construct a Weil descent system from a set of summation polynomials in which the first fall degree assumption is unlikely to hold. These examples suggest that greater care needs to be exercised when applying this heuristic assumption to arrive at complexity estimates.

### 1.4 Our Contributions

The contributions of this paper are three-fold.

- First, we introduce the notion of the last fall degree for a finite set of polynomials. Intuitively, this last fall degree determines the minimum degree at which operations on the generating polynomials need to be performed for all other polynomials to be generated. Our definition is intrinsic to the generating system and is independent of any monomial order. This allows us to provide an explicit and generic algorithm to find all the zeroes of a zero-dimensional set of polynomials whose time complexity depends on this last fall degree. While our approach may be similar to existing Gröbner basis algorithms, we stress

that we have developed a generic framework that applies to any multivariate zero-dimensional polynomial system with a time complexity dependent on a well-defined parameter.

- Second, we prove that the polynomials from the HFE system can be solved in polynomial time if the degree of the defining polynomial and the cardinality of the base field are fixed. Our proof is elementary and complete, without relying on any unproved assumptions or results. We do this by bounding the last fall degree of the zero-dimensional system and then exploit the aforementioned algorithm to solve the system. Besides, our proof works for any univariate polynomial  $f(X)$  bounded by some degree in contrast to the original HFE system which restricts the monomials in  $f(X)$  to be of a certain form. More importantly, our approach can be applied to analyze zero-dimensional polynomial systems of other types (see [20]).
- Finally, we consider an important application of solving a zero-dimensional multivariate polynomial system, namely in finding relations for index calculus algorithms to solve the elliptic curve discrete logarithm problem. Here, we revisit the first fall degree assumption adopted in [23] to derive a sub-exponential time estimate to solve the ECDLP. We illustrate two examples which raise some doubts on the correctness of this assumption on Weil descent systems arising from summation polynomials. From such examples, we believe that more evidence has to be presented before applying the first fall degree assumption to make complexity claims on the ECDLP.

## 1.5 Organization of the Paper

The rest of this article is organized as follows. We begin in Sect. 2 by defining a vector space of polynomials obtained with operations within a certain degree starting from a set of polynomials. We then use this set to define the notion of the last fall degree of a polynomial system. With these notions, we present an algorithm to find all the zeros of a zero-dimensional multivariate polynomial system over a finite field. Next in Sect. 3, we define the notion of a Weil descent system and of a fake Weil descent system arising from a system of univariate polynomials over a field of cardinality  $q^n$  and we discuss the relations between both systems. This is followed by our attack on the HFE system in Sect. 4. The main result in this section is Main Theorem 1. In the final section, we provide a brief discussion and some comments on Weil descent attacks on ECDLP.

## 2 Constructible Polynomials

Let  $k$  be a field and let  $R = k[X_0, \dots, X_{n-1}]$  be a polynomial ring. Let  $\mathcal{F}$  be a finite subset of  $R$  and let  $I \subseteq R$  be the ideal generated by  $\mathcal{F}$ . We set  $\deg(\mathcal{F}) = \max(\deg(f) : f \in \mathcal{F})$ .

**Definition 1.** For  $i \in \mathbb{Z}_{\geq 0}$ , we let  $V_{\mathcal{F}, i}$  be the smallest  $k$ -vector space of  $R$  such that

1.  $\{f \in \mathcal{F} : \deg(f) \leq i\} \subseteq V_{\mathcal{F},i};$
2. if  $g \in V_{\mathcal{F},i}$  and if  $h \in R$  with  $\deg(hg) \leq i$ , then  $hg \in V_{\mathcal{F},i}$ .

We set  $V_{\mathcal{F},\infty} = I$ . For convenience, we set  $V_{\mathcal{F},-1} = \emptyset$ . If  $\mathcal{F}$  is fixed, we often write  $V_i$  instead of  $V_{\mathcal{F},i}$ .

Intuitively,  $V_i$  is the largest subset of  $I$  which can be constructed from  $\mathcal{F}$  by doing ideal operations without exceeding degree  $i$ .

Note that  $V_i$  is a finite-dimensional  $k$ -vector space of dimension  $\dim_k(V_i) \leq \binom{n+i}{i} \leq (n+i)^i$ .

If  $\mathcal{F}$  is fixed and  $g_1, g_2 \in R$ , then we write  $g_1 \equiv_i g_2$  whenever  $g_1 - g_2 \in V_i$ . Note that for  $h_1, h_2, h_3 \in R$  with  $h_1 \equiv_r h_2$ , one has

$$h_1 h_3 \equiv_{\max(r, \deg(h_1 h_3), \deg(h_2 h_3))} h_2 h_3.$$

We write  $g_1 \equiv g_2$  if  $g_1 - g_2 \in I$ .

**Definition 2.** Let  $\mathcal{F}$  be a finite subset of  $R$  and let  $I$  be the ideal generated by  $\mathcal{F}$ . The minimal  $c \in \mathbb{Z}_{\geq 0} \cup \{\infty\}$  such that for all  $f \in I$  one has  $f \in V_{\max(c, \deg(f))}$ , is called the last fall degree of  $\mathcal{F}$ , and is denoted by  $d_{\mathcal{F}}$ .

A monomial order  $\leq$  on  $R$  is called degree refining if for monomials  $M, N$  with  $\deg(M) < \deg(N)$ , one has  $M < N$ .

**Lemma 1.** The following hold:

1. One has  $d_{\mathcal{F}} \in \mathbb{Z}_{\geq 0}$ .
2. Let  $\mathcal{B}$  be a Gröbner basis of  $I$  with respect to some degree refining monomial order on  $R$ . Then there is an integer  $c \in \mathbb{Z}_{\geq 0}$  such that  $\mathcal{B} \subseteq V_{\mathcal{F},c}$  and one has  $d_{\mathcal{F}} \leq c$ .

*Proof.* Since (1) follows from (2), we will prove (2). Let  $\{g_1, \dots, g_s\}$  be a Gröbner basis of  $I$  with respect to some monomial order which refines the degree. Set  $c$  to be the minimal  $i$  such that  $g_j \in V_i$  for all  $j$ . Let  $f \in I$ . Since  $\mathcal{B}$  is a Gröbner basis of  $I$  with respect to a degree refining order, we can write  $f = \sum_{i=1}^s a_i g_i$  with  $\deg(a_i g_i) \leq \deg(f)$  for  $i = 1, \dots, s$ . Then one easily finds  $f \in V_{\max(\deg(f), c)}$ .

Note that the bound  $c$  on  $d_{\mathcal{F}}$  given in Lemma 1 is constructed with respect to a fixed monomial order. However, the last fall degree  $d_{\mathcal{F}}$  is intrinsic to the set  $\mathcal{F}$  and independent of the choice of a monomial order.

Let  $\mathcal{G}$  be obtained from  $\mathcal{F}$  through an invertible linear transformation of equations. Then one has  $\max(d_{\mathcal{F}}, \deg(\mathcal{F})) = \max(d_{\mathcal{G}}, \deg(\mathcal{F}))$ . Note that  $\deg(\mathcal{F}) = \deg(\mathcal{G})$ . Further, since any affine transformation of the variables  $X_0, \dots, X_{m-1}$  is degree-preserving, the last fall degree is invariant under such transformations. Finally, enlarging the field  $k$  does not change the last fall degree.

*Remark 1.* The name last fall degree has been chosen because there is a similar concept called the first fall degree, which is used to heuristically bound the complexity of Gröbner basis algorithms, see [23]. The *first fall degree* of a system



$\mathcal{F}$  is the smallest  $d \geq \deg(\mathcal{F})$  such that there exists  $g_f \in R$  for  $f \in \mathcal{F}$  such that  $d = \max_{f \in \mathcal{F}}(\deg(g_f f))$  and  $\deg(\sum_{f \in \mathcal{F}} g_f f) < d$  and  $\sum_{f \in \mathcal{F}} g_f f \neq 0$ .

An equivalent definition of the last fall degree is the following:  $d_{\mathcal{F}}$  is the largest  $c \in \mathbb{Z}_{\geq 0}$  such that  $V_c \cap R_{\leq c-1} \neq V_{c-1}$ , where  $R_{\leq c-1}$  denotes the set of all polynomials in  $R$  with degree less than or equal to  $c-1$ . This definition has the same flavour as the definition of the first fall degree. This equivalent definition of the last fall degree allows one to compute the last fall degree if an upper bound, for example from Lemma 1, is known. It would be of great interest to find a direct method for computing the last fall degree.

### 2.1 An Explicit Construction of $V_i$

Next, we describe an algorithm to construct  $V_i$  explicitly.

Let  $V$  be a finite-dimensional  $k$ -vector subspace of  $k[X_0, \dots, X_{n-1}]$ . We say that  $B$  is a reduced basis of  $V$  if  $B$  is a basis of  $V$  and for all  $h = \sum_{g \in B} a_g g, a_g \in k$ , we have  $\deg(h) = \max_{g \in B} \deg(a_g g)$ . For instance, a reduced basis can be constructed if we order the monomials with respect to their degrees and apply linear algebra operations to obtain a basis with different leading monomials.

Fix an integer  $i \geq 0$  and let  $\mathcal{F} = \{f_1, \dots, f_r\} \subset k[X_0, \dots, X_{n-1}]$ . We construct  $V_i$  inductively as follows.

Let  $W_0$  be the  $k$ -linear span of  $\{f_j : j = 1, \dots, r, \deg(f_j) \leq i\}$ . By linear algebra operations, construct a reduced basis  $B_0$  of  $W_0$ . For  $j = 1, 2, \dots$ , define  $W_j = \text{Span}_k\{tg : g \in B_{j-1}, t \text{ is a monomial and } \deg(tg) \leq i\}$ . Construct a reduced basis  $B_j$  of  $W_j$  from using linear algebra. Note that  $W_j$  contains  $W_{j-1}$ . Since  $W_0 \subseteq W_1 \subseteq \dots \subseteq V_i$ , this process must terminate and we conclude that there exists some  $l$  such that  $W_l = W_{l+1}$ .

We claim that  $W_l = V_i$ . Suppose not. Then there must exist some  $g \in W_l$  and  $h \in k[X_0, \dots, X_{n-1}]$  such that  $gh \notin W_l$  and  $\deg(gh) \leq i$ . Let  $B_l = \{g_1, \dots, g_s\}$  be a reduced basis of  $W_l$ . Then one has  $g = a_1 g_1 + a_2 g_2 + \dots + a_s g_s$  with  $a_1, a_2, \dots, a_s \in k$ . Since  $B_l$  is a reduced basis of  $W_l$ ,  $\max_j(\deg(a_j g_j)) = \deg(g)$ . Hence,  $gh = \sum_j g a_j g_j$  and  $\max_j(\deg(g a_j g_j)) \leq i$  so that  $W_{l+1} \neq W_l$ .

Assume  $k$  is a finite field of cardinality  $q$ . Since  $l$  is bounded by  $(n+i)^i$ , it follows from the above arguments that one can compute  $V_i$  in time polynomial in  $r, \log(q)$  and  $(n+i)^i$ . Furthermore, using linear algebra, one can determine if a polynomial  $f$  with  $\deg(f) \leq i$  lies in  $V_i$  with the same time bound.

### 2.2 Solving a Zero-Dimensional Polynomial System

Consider a system of  $r$  multivariate polynomial equations over a field  $k$  of cardinality  $q$ , namely,  $f_1 = f_2 = \dots = f_r = 0$  in  $n$  variables  $X_0, X_1, \dots, X_{n-1}$ . Suppose that the algebraic set defined by this system is zero-dimensional, that is, there are finitely many solutions over an algebraic closure  $\bar{k}$  of  $k$ . The next proposition gives a generic approach to solve the system via the above construction of  $V_i$ .

**Proposition 1.** *Let  $k$  be a finite field of cardinality  $q$ . Let  $\mathcal{F} \subset R$  be a finite subset and let  $I$  be the ideal generated by  $\mathcal{F}$ . Assume that  $I$  is radical and that the system has at most  $e$  solutions over  $\bar{k}$ . Set  $d = \max(d_{\mathcal{F}}, e)$ . Then one can find all solutions of  $I$  in  $k$*

- probabilistically in time polynomial in the input size of  $\mathcal{F}$ ,  $\log(q)$  and  $(n + d)^d$ ;
- deterministically in time polynomial in the input size of  $\mathcal{F}$ ,  $q$  and  $(n + d)^d$ .

*Proof.* First, note that one can factor a polynomial of degree  $s$  over  $k$  deterministically in time polynomial in  $q$  and  $s$ , and probabilistically in time polynomial in  $\log(q)$  and  $s$  [13].

Compute  $V_d$  in time polynomial in the input size of  $\mathcal{F}$ ,  $\log(q)$  and  $(n + d)^d$  (Subsect. 2.1).

Assume that all solutions over  $\bar{k}$  of the system are

$$(a_{0,0}, \dots, a_{0,n-1}), \dots, (a_{t,0}, \dots, a_{t,n-1}) \in \bar{k}^n$$

with  $t < e$ . Since  $I$  is a radical ideal, by the Nullstellensatz and Galois theory, one has

$$h_0 = \prod_{a \in \{a_{i,0} : i=0, \dots, t\}} (X_0 - a) \in I.$$

Using linear algebra, one can find  $h_0$  as the nonzero polynomial of minimal degree  $d_0$  in  $V_d \cap \text{Span}_k\{1, X_0, \dots, X_0^e\}$ . Factor  $h_0$ . Assume that  $a_0$  is a root of  $h_0$  in  $k$ . We will find all solutions over  $k$  with  $X_0 = a_0$ . Set  $h'_0 = h_0 / (X_0 - a_0)$  of degree  $d_0 - 1$ . By the Nullstellensatz and Galois theory, one has

$$h_1 = h'_0 \prod_{a \in \{a_{i,1} : i=0, \dots, t, a_{i,0}=a_0\}} (X_1 - a) \in I.$$

Using linear algebra, one finds  $h_1$  as the polynomial of minimal degree  $d_1$  in  $V_d \cap \text{Span}_k\{h'_0, X_1 h'_0, \dots, X_1^{e-d_0+1} h'_0\}$ . Factor  $h_1/h'_0$  over  $k$ . Pick a solution  $a_1$  over  $k$  and find all solutions with  $X_0 = a_0, X_1 = a_1$  using the similar recursive procedure. Hence one can find all solutions over  $k$  as required.

*Remark 2.* See [20] for a comparison between our approach for solving systems, MutantXL and Gröbner basis algorithms.

### 3 Weil Descent

Let  $q$  be a prime power. Let  $n \in \mathbb{Z}_{\geq 1}$  and let  $k$  be a finite field of cardinality  $q^n$  with subfield  $k'$  of cardinality  $q$ . Let  $\mathcal{F}$  be a finite subset of  $k[X]$ . In this section, we introduce a Weil descent system of  $\mathcal{F}$ , which is a system in  $k'[Y_0, \dots, Y_{n-1}]$ . Furthermore, we introduce the fake Weil descent system of  $\mathcal{F}$ , which is a system in  $k[X_0, \dots, X_{n-1}]$ . The analysis in this section can be easily extended to  $m$  variables for any positive integer  $m$  (see Remark 3).

Let  $\mathcal{F} \subset k[X]$  be a finite set of polynomials. Suppose we want to find the common zeros of these polynomials in  $k$ . Let  $I$  be the ideal generated by

$$\mathcal{F}_f = \mathcal{F} \cup \{X^{q^n} - X\}.$$

### 3.1 Weil Descent

Let  $\alpha_0, \dots, \alpha_{n-1}$  be a basis of  $k/k'$ . Write  $X = \sum_{i=0}^{n-1} \alpha_i Y_i$  and for  $f \in k[X]$ , define  $[f]_j \in k'[Y_0, \dots, Y_{n-1}]$  by

$$f\left(\sum_{j=0}^{n-1} \alpha_j Y_j\right) \equiv \sum_{j=0}^n [f]_j \alpha_j \pmod{Y_0^q - Y_0, \dots, Y_{n-1}^q - Y_{n-1}}$$

where  $[f]_j \in k'[Y_0, \dots, Y_{n-1}]$  is chosen of minimal degree, that is,  $\deg_{Y_i}([f]_j) < q$ . Consider the systems

$$\mathcal{F}' = \{[f]_j : f \in \mathcal{F}, j = 0, \dots, n-1\}$$

and

$$\mathcal{F}'_f = \{[f]_j : f \in \mathcal{F}, j = 0, \dots, n-1\} \cup \{Y_i^q - Y_i : i = 0, \dots, n-1\}.$$

The latter is called a Weil descent system of  $\mathcal{F}$ . Notice that the ideal generated by  $\mathcal{F}'_f$  is always a radical ideal, as  $k'[Y_0, \dots, Y_{n-1}]/(Y_i^q - Y_i : i = 0, \dots, n-1)$  is isomorphic to a product of fields (Chinese remainder theorem). One easily sees that solutions of  $\mathcal{F}$  or  $\mathcal{F}_f$  in  $k$  correspond to solutions of  $\mathcal{F}'$  or  $\mathcal{F}'_f$  over  $k'$ .

A different choice of  $\alpha_i$  merely results in a linear change of the variables  $Y_i$  and the polynomials  $[f]_i$ . An interesting choice for the  $\alpha_i$  is a normal basis, that is, a basis with  $\alpha_i = \theta^{q^i}$  for some  $\theta \in k$ .

### 3.2 Fake Weil Descent

To study the complexity of solving a Weil descent system, we relate a Weil descent system to another system in  $k[X_0, \dots, X_{n-1}]$ , which we refer to as the fake Weil descent system.

Let  $R = k[X_0, \dots, X_{n-1}]$ . Let  $e \in \mathbb{Z}_{\geq 0}$ . Let  $X^{e'}$  be the remainder of division of  $X^e$  by  $X^{q^n} - X$  in  $k[X]$ . Write  $e' = \sum_{j=0}^{n-1} e'_j q^j$  in base  $q$  with  $e'_j \in \{0, 1, \dots, q-1\}$ . We set

$$\overline{X^e} = X_0^{e'_0} \dots X_{n-1}^{e'_{n-1}} \in R.$$

We extend this definition  $k$ -linearly for all polynomials in  $R$ . This gives a map  $\bar{\cdot} : k[X] \rightarrow R$ . We set, where by convention  $X_n = X_0$ ,

$$\overline{\mathcal{F}} = \{\overline{f} : f \in \mathcal{F}\}$$

and

$$\overline{\mathcal{F}}_f = \{\overline{f} : f \in \mathcal{F}\} \cup \{X_0^q - X_1, \dots, X_{n-1}^q - X_n\}.$$

We let  $\overline{I}$  be the ideal generated by  $\overline{\mathcal{F}}_f$ . We call  $\overline{\mathcal{F}}_f$  the fake Weil descent system of  $\mathcal{F}$ . Note that  $\overline{I}$  is a radical ideal. Indeed the  $k$ -algebra morphism

$R/(X_0^q - X_1, \dots, X_{n-1}^q - X_n) \rightarrow k[X_0]/(X_0^{q^n} - X_0)$  which sends  $X_i$  to  $X_0^{q^i}$  is an isomorphism, because it is a surjective morphism on  $k$ -vector spaces of the same dimension. The latter ring is isomorphic to  $k^k$  by the Chinese remainder theorem. In the ring  $k^k$  all ideals are radical.

There is a bijection between the set of solutions of  $I$  and those of  $\bar{I}$  over  $k$  (or  $\bar{k}$ ). If  $X = a \in \bar{k}$  is a zero of  $I$ , then  $(X_0, \dots, X_{n-1}) = (a, a^q, \dots, a^{q^{n-1}})$  is a zero of  $\bar{I}$ . Conversely, if  $(X_0, \dots, X_{n-1}) = (a_0, \dots, a_{n-1})$  is a solution of  $\bar{I}$ , then  $X = a_0$  is a solution of  $I$ .

We will now prove a couple of lemmas which will be useful later.

**Lemma 2.** *Let  $h_1, h_2 \in R, g \in k[X]$ . One has, where  $\equiv_i$  is defined with respect to  $\bar{\mathcal{F}}_f$ :*

1.  $\overline{h_1 + h_2} \equiv_{\max(\deg(\bar{h}_1), \deg(\bar{h}_2))} \bar{h}_1 + \bar{h}_2$ ;
2.  $\overline{h_1 \cdot h_2} \equiv_{\deg(\bar{h}_1) + \deg(\bar{h}_2)} \bar{h}_1 \bar{h}_2$ ;
3. *There is  $h_3 \in k[X]$  with  $\deg(h_3) < q^n$  such that  $g \equiv_{\deg(g)} \bar{h}_3$ .*

*Proof.* One reduces to the case of monomials and the result then follows easily.

We have a morphism of  $k$ -algebras  $\varphi : R \rightarrow k[X]$  which maps  $X_i$  to  $X^{q^i}$ . This map has the following properties.

**Lemma 3.** *Let  $h \in k[X]$ . The following statements hold:*

1.  $\varphi(\bar{h}) \equiv h \pmod{X^{q^n} - X}$ ;
2.  $h \in I$  if and only if  $\bar{h} \in \bar{I}$ .

*Proof.* 1: Follows directly.

2: Let  $h \in I$ . We will show  $\bar{h} \in \bar{I}$ . One can write  $h = b(X^{q^n} - X) + \sum_{f \in \mathcal{F}} a_f f$ .

Modulo  $\bar{I}$  we find with Lemma 2:

$$\bar{h} = \overline{b(X^{q^n} - X) + \sum_{f \in \mathcal{F}} a_f f} \equiv \bar{b}(X_0 - X_0) + \sum_{f \in \mathcal{F}} \bar{a}_f \bar{f} \equiv 0.$$

Conversely, let  $h \in k[X]$  and assume  $\bar{h} \in \bar{I}$ . Write  $\bar{h} = \sum_{j=0}^{n-1} c_j (X_j^q - X_{j+1}) + \sum_{f \in \mathcal{F}} b_f \bar{f}$ . One finds, using 1,

$$\begin{aligned} \varphi(\bar{h}) &= \sum_{j=0}^{n-1} \varphi(c_j) \varphi(X_j^q - X_{j+1}) + \sum_{f \in \mathcal{F}} \varphi(b_f) \varphi(\bar{f}) \\ &\equiv \varphi(c_{n-1})(X^{q^n} - X) + \sum_{f \in \mathcal{F}} \varphi(b_f) f \pmod{X^{q^n} - X}. \end{aligned}$$

We conclude  $\varphi(\bar{h}) \in I$ .

### 3.3 Summary of Notation

Let us recall some notation we have introduced thus far. Let  $\mathcal{F} \subset k[X]$  be a finite subset, where  $k$  is a finite field of cardinality  $q^n$  and let  $k'$  be its subfield of cardinality  $q$  with an implicit choice of basis of  $k$  over  $k'$ . We let  $I$  be the ideal generated by

$$\mathcal{F}_f = \mathcal{F} \cup \{X^{q^n} - X\}.$$

We have systems in  $k'[Y_0, \dots, Y_{n-1}]$  defined by

$$\mathcal{F}' = \{[f]_j : f \in \mathcal{F}, j = 0, \dots, n - 1\}$$

and a Weil descent system

$$\mathcal{F}'_f = \{[f]_j : f \in \mathcal{F}, j = 0, \dots, n - 1\} \cup \{Y_i^q - Y_i : i = 0, \dots, n - 1\}.$$

Finally, we have systems in  $k[X_0, \dots, X_{n-1}]$  defined by

$$\overline{\mathcal{F}} = \{\overline{f} : f \in \mathcal{F}\}$$

and the fake Weil descent system

$$\overline{\mathcal{F}}_f = \{\overline{f} : f \in \mathcal{F}\} \cup \{X_0^q - X_1, \dots, X_{n-1}^q - X_n\}.$$

We let  $\overline{I}$  be the ideal generated by  $\overline{\mathcal{F}}_f$ .

### 3.4 Relating Both Types of Descent

This subsection seeks to connect the last fall degrees of a Weil descent system and the fake Weil descent system presented in Subsects. 3.1 and 3.2. We follow the formulation in [16] which essentially shows that the two systems are linked by suitable transformations. We have the following result.

**Proposition 2.** *One has*

$$\max(d_{\mathcal{F}'_f}, q, \deg(\mathcal{F}')) \leq \max(d_{\overline{\mathcal{F}}_f}, q, \deg(\mathcal{F}')).$$

*Proof (Sketch).* We follow [16]. The details can be found in [20]. One has  $\deg(\mathcal{F}') = \deg(\overline{\mathcal{F}})$ . After a linear change, we may assume that a Weil descent in  $\mathcal{F}'$  is with respect to a normal basis  $\{\theta^{q^i} : i = 0, \dots, n - 1\}$ . Consider the system  $\mathcal{F}' \subseteq k[Y_0, \dots, Y_{n-1}]$ , which has the same last fall degree as considered over  $k'$ . Using some linear changes of the polynomials and linear changes of variables as in Sect. 4 of [16], we obtain the system  $\mathcal{F}'' = \{\overline{f}, \overline{f^q}, \dots, \overline{f^{q^{n-1}}} : f \in \mathcal{F}\} \cup \{Y_0^q - Y_1, \dots, Y_{n-1}^q - Y_n\}$ . One has

$$\max(d_{\mathcal{F}'_f}, q, \deg(\mathcal{F}')) = \max(d_{\mathcal{F}''}, q, \deg(\mathcal{F}')).$$

Note that  $\overline{\mathcal{F}} \subseteq \mathcal{F}''$  and that both sets generate the same ideal (Lemma 2(2)). Hence the result follows.

*Remark 3.* In this section, we have presented a Weil descent system and its related fake Weil descent system corresponding to a polynomial system in one variable  $X$  over  $k$ . This definition can be easily extended to a system of  $r$  polynomials in  $m$  variables over  $k$  such that each variable corresponds to  $n$  descent variables. This gives rise to  $rn$  polynomials in  $mn$  variables and all the results follow accordingly.

## 4 Solving the HFE System

In this section, our primary goal is to prove that the HFE system and its variants can be solved efficiently by employing the tools we have developed so far. Although such results were shown previously (see for example [16]), their proofs were based on some heuristics. Our proof, on the other hand, is rigorous and free from any unproven conjecture or heuristics. Another claim for a proof can be found in [22].

We begin by reviewing the general description of the HFE system. Throughout this section,  $k$  will denote a field of cardinality  $q^n$ , while  $k'$  will denote its subfield of cardinality  $q$ .

### 4.1 Description of the HFE Encryption

The HFE public key cryptosystem was first introduced by Patarin [21]. Briefly, let  $f(X)$  be a univariate polynomial in  $k[X]$  with degree bounded by  $q^t$ . In practice, the nonconstant monomials in  $f$  are chosen to be either of the form  $X^{q^i+q^j}$  or  $X^{q^i}$  for integers  $i, j \geq 0$ . However, we will remove this restriction in this paper and allow  $f$  to be an arbitrary polynomial with degree bounded by  $q^t$ .

Let  $\mathcal{F} = \{f\} \subset k[X]$  and consider the Weil descent system

$$\mathcal{F}'_f = \{[f]_0, \dots, [f]_{n-1}\} \cup \{Y_i^q - Y_i : i = 0, \dots, n-1\} \subseteq k'[Y_0, \dots, Y_{n-1}]$$

as in Subsect. 3.1 with respect to some basis of  $k/k'$ . We have a natural right action of  $\text{Aff}_n(k')$  on  $R' = k'[Y_0, \dots, Y_{n-1}]$  by an affine transformation of variables. Let  $M \in \text{Aff}_n(k')$ . For  $g \in k'[Y_0, \dots, Y_{n-1}]$ , we write  $gM$  for this action. Let  $N \in \text{GL}_n(k')$ . Define

$$\begin{pmatrix} g_0 \\ \vdots \\ g_{n-1} \end{pmatrix} = N \begin{pmatrix} [f]_0 M \\ \vdots \\ [f]_{n-1} M \end{pmatrix}.$$

The public key of the system is the set of equations  $\{g_0, g_1, \dots, g_{n-1}\} \subset k'[Y_0, \dots, Y_{n-1}]$  while the private key comprises  $f$ , the basis choice  $k/k'$  and the transformations  $M$  and  $N$ . To encrypt a message,  $(m_0, m_1, \dots, m_{n-1}) \in k'^n$ , one computes

$$(c_0, \dots, c_{n-1}) = (g_0(m_0, \dots, m_{n-1}), \dots, g_{n-1}(m_0, \dots, m_{n-1})).$$

Using the private key and a factorization algorithm, one can find the message efficiently.

Let  $\mathcal{G} = \{g_0 - c_0, \dots, g_{n-1} - c_{n-1}\} \cup \{Y_i^q - Y_i : i = 0, 1, \dots, n-1\}$ . Observe that the message  $(m_0, \dots, m_{n-1})$  can be recovered if we can solve  $\mathcal{G}$ . This can be achieved deterministically via Proposition 1 in time polynomial in  $q$  and  $(n+d)^d$ , where  $d$  is bounded by the maximum of the last fall degree of  $\mathcal{G}$  and the number of solutions  $e$  of  $\mathcal{G}$ . Notice that we are now in the situation of the main theorem (Main Theorem 1) which we now proceed to prove.

### 4.2 An Upper Bound on the Last Fall Degree

Let  $q$  be a prime power and let  $k$  be a finite field of cardinality  $q^n$ . Let  $\mathcal{F} \subset k[X]$  be a finite set. Consider a fake Weil descent system  $\mathcal{F}_f$  to the subfield of cardinality  $q$ . Define  $\equiv_j$  with respect to  $\mathcal{F}_f$ . For  $e \in \mathbb{Z}_{\geq 0}$  with  $e = \sum_i a_i q^i$  in base  $q$ , we set  $w(e) = \sum_i a_i$ . For  $f = \sum_i b_i X^i$ , we set  $w(f) = \max(w(i) : b_i \neq 0)$ . Note that  $w(f) \leq \deg(f)$ , with equality if  $\deg(f) < q^n$ .

We start with a technical lemma. Recall the following. For  $r \in \mathbb{Z}_{\geq 0}$  and  $c \in \mathbb{Z}_{\geq 1}$ , we set

$$\psi(r, c) = \max(\lfloor 2(c-1)(\log_c(r) + 1) \rfloor, 0).$$

Let  $g \in k[X] \setminus k$ . Then one has

$$\deg(\bar{g}) \leq (q-1)(\log_q(\deg(g)) + 1).$$

It follows that  $\deg(\bar{g}) \leq \psi(\deg(g), q)/2$ .

**Lemma 4.** *Let  $h_2 \in k[X]$  nonzero of degree  $d$ . Set  $u = \psi(d, q)$ . Assume  $\overline{h_2} \equiv_u 0$ . Let  $h_1 \in k[X]$ . Let  $h_3$  be the remainder of division of  $h_1$  by  $h_2$ . Then one has  $\overline{h_1} \equiv_{\max(u, w(h_1))} \overline{h_3}$ .*

*Proof.* If  $d = 0$ , the result follows easily. Assume  $d > 0$ .

Fix  $h_2$  and write  $h_2 = \sum_{i=0}^d b_i X^i$  where  $b_d \neq 0$ . Since taking remainders is additive, it suffices to prove the result for  $h_1 = X^e$ . Let  $r_e$  be the remainder of division of  $X^e$  by  $h_2$ . For  $g \in k[X]$  with  $\deg(g) \leq d$ , one has  $\deg(\bar{g}) \leq u/2$ . In particular, we have  $\deg(\overline{r_e}) \leq u/2$ .

We will prove the following statements successively:

1. for  $e \in \{0, 1, \dots, qd-1\}$ , we have  $\overline{X^e} \equiv_u \overline{r_e}$ ;
2. if  $e, e' \in \mathbb{Z}_{\geq 0}$  satisfy  $w(e) + w(e') \leq u$ ,  $\overline{X^e} \equiv_u \overline{r_e}$  and  $\overline{X^{e'}} \equiv_u \overline{r_{e'}}$ , then  $\overline{X^{e+e'}} \equiv_u \overline{r_{e+e'}}$ ;
3. for  $e \in \mathbb{Z}_{\geq 0}$  with  $w(e) \leq u$ , we have  $\overline{X^e} \equiv_u \overline{r_e}$ ;
4. for all  $e \in \mathbb{Z}_{\geq 0}$  one has  $\overline{X^e} \equiv_{\max(u, w(e))} \overline{r_e}$ .

1: For  $e = 0, \dots, d-1$ , the remainder is  $X^e$  itself and the result follows. One has  $r_d = \frac{-1}{b_d} \sum_{i=0}^{d-1} b_i X^i$  and this gives  $\overline{X^d} \equiv_u \overline{r_d}$ . We continue by induction. Assume the statement holds for cases smaller than  $e$  and that  $e \leq qd-1$ . We will prove the statement for  $e$ . Write  $r_{e-1} = \sum_{j=0}^{d-1} c_j X^j$ . Note that  $r_e$  is the

remainder of division of  $Xr_{e-1}$  by  $h_2$ , which gives  $r_e = \sum_{j=0}^{d-1} c_j r_{j+1}$ . Note that  $e - 1 \leq qd - 2 = q^{\log_q(d)+1} - 2$ . Hence we have (as  $d > 0$ ):

$$\begin{aligned} \deg(\overline{X}) + \deg(\overline{X^{e-1}}) &\leq 1 + \lfloor (q - 1) (\log_q(d) + 2) - 1 \rfloor \\ &= \lfloor (q - 1) (\log_q(d) + 2) \rfloor \leq u. \end{aligned}$$

Using Lemma 2 and the induction hypothesis, we find

$$\overline{X^e} \equiv_u \overline{X} \cdot \overline{X^{e-1}} \equiv_u \overline{X} \cdot \overline{r_{e-1}} \equiv_u \overline{\sum_{j=0}^{d-1} c_j X^{j+1}} \equiv_u \overline{\sum_{j=0}^{d-1} c_j r_{j+1}},$$

and this gives the required remainder.

2: Assume without loss of generality that  $w(e') \leq u/2$ . Then one has  $u \geq \max(w(e) + w(e'), \deg(\overline{r_e}) + w(e'), \deg(\overline{r_e}) + \deg(\overline{r_{e'}}))$  and one has  $\deg(r_e r_{e'}) \leq 2d - 2 \leq qd - 1$ . Lemma 1 and 2 give

$$\overline{X^{e+e'}} \equiv_u \overline{X^e} \cdot \overline{X^{e'}} \equiv_u \overline{r_e} \cdot \overline{X^{e'}} \equiv_u \overline{r_e} \cdot \overline{r_{e'}} \equiv_u \overline{r_e r_{e'}} \equiv_u \overline{r_{e+e'}}.$$

3: Using 2 and induction, we easily reduce to the case where  $e = q^i, i \geq 0$ . Note that for  $i \geq 1, q^i = q \cdot q^{i-1}$  and that  $u \geq q$ . We can then apply 2 and the proof follows by induction.

4: We prove this statement by induction on  $w(e) > u$ . Write  $e = e_1 + e_2$  with  $u \leq w(e_1) < w(e)$ , and  $w(e_1) + w(e_2) = w(e)$ . One has (Lemma 2 and part 3)

$$\begin{aligned} \overline{X^e} &\equiv_{\max(u, w(e))} \overline{X^{e_1}} \cdot \overline{X^{e_2}} \equiv_{\max(u, w(e))} \overline{r_{e_1}} \cdot \overline{X^{e_2}} \\ &\equiv_{\max(u, w(e))} \overline{r_{e_1}} \cdot \overline{r_{e_2}} \equiv_{\max(u, w(e))} \overline{r_e}. \end{aligned}$$

**Proposition 3.** *Assume  $\mathcal{F} = \{f\}$  with  $f \in k[X]$  nonzero. Set  $u = \psi(\deg(f), q)$  and set  $g = \gcd(f, X^{q^n} - X)$ . Then we have  $\overline{g} \in V_u$ .*

*Proof.* Let  $f_1$  be the remainder of division of  $X^{q^n} - X$  by  $f$ . By Lemma 4, we have  $\overline{f_1} \equiv_u 0$ . Let  $f_2$  be the remainder of division of  $f$  by  $f_1$ . Similarly, we find  $\overline{f_2} \equiv_u 0$ . Hence we can follow the Euclidean algorithm and we obtain  $\overline{g} \in V_u$ .

### 4.3 Proof of the Main Theorem

We can finally prove Main Theorem 1.

*Proof.* [of Main Theorem 1] We first study the last fall degree of  $\mathcal{G}$ . One has (Proposition 2)

$$d_{\mathcal{G}} \leq \max(d_{\mathcal{G}}, q, \deg(\mathcal{F}')) = \max(d_{\mathcal{F}'}, q, \deg(\mathcal{F}')) \leq \max(d_{\overline{\mathcal{F}'}, q, \deg(\mathcal{F}'))).$$

Hence we study the last fall degree of  $\overline{\mathcal{F}'}$ . Set  $g = \gcd(f, X^{q^n} - X)$ . From Proposition 3, we have  $\overline{g} \in V_u$  with  $u = \psi(\deg(f), q)$ .

Let  $h \in \overline{I}$ , the ideal generated by  $\overline{\mathcal{F}'}$ . Define the relations  $\equiv_i$  with respect to  $\overline{\mathcal{F}'}$ . By Lemma 2(3), one has  $h \equiv_{\deg(h)} \overline{h_2}$  for some  $h_2 \in k[X]$  with  $\deg(h_2) < q^n$ .



Since  $\overline{h_2} \in \overline{I}$ , it follows from Lemma 3(2) that  $h_2 \in I$ . Hence  $h_2$  has remainder 0 when divided by  $g$ . From Lemma 4, we conclude (as  $\deg(\overline{h_2}) \leq \deg(h)$ ),

$$h \equiv_{\max(\deg(h), u)} \overline{h_2} \equiv_{\max(\deg(h), u)} 0.$$

This shows  $d_{\mathcal{F}} \leq u$ . Hence one finds, as  $\deg(\mathcal{F}') \leq u$ ,

$$d_{\mathcal{G}} \leq \max(d_{\mathcal{F}_f}, q, \deg(\mathcal{F}')) \leq \max(u, q, \deg(\mathcal{F}')) = \max(u, q).$$

Notice that  $u \geq q$ . Apply Proposition 1 to solve the system  $\mathcal{G}$  in the required time.

## 5 Weil Descent Attacks on ECDLP

### 5.1 ECDLP and Summation Polynomials

Let  $E$  be an elliptic curve over a field  $k$  of cardinality  $q^n$  and let  $k'$  be its subfield of cardinality  $q$ . One possible approach to solve the elliptic curve discrete logarithm problem (ECDLP) is via the index calculus method. Essentially, sufficiently many relations between  $k$ -points on the curve  $E$  need to be generated and the time to construct such relations has a direct impact on the complexity of the entire index calculus approach.

In [6, 24], summation polynomials were used to find relations between points on the curve. Here, we recall the definition of a summation polynomial.

Let  $F$  be a field. Let  $A = (a_1, a_2, a_3, a_4, a_6) \in F^5$ . Set

$$\begin{aligned} b_2 &= a_1^2 + 4a_2, \\ b_4 &= a_1a_3 + 2a_4, \\ b_6 &= a_3^2 + 4a_6, \\ b_8 &= a_1^2a_6 - a_1a_3a_4 + a_2a_3^2 + 4a_2a_6 - a_4^2. \end{aligned}$$

We define

$$S_{A,2} = X_0 - X_1 \in F[X_0, X_1].$$

We define the third summation polynomial  $S_{A,3} \in F[X_0, X_1, X_2]$  of degree 4 by:

$$\begin{aligned} S_{A,3} &= (X_0^2X_1^2 + X_0^2X_2^2 + X_1^2X_2^2) - 2 \cdot (X_0^2X_1X_2 + X_0X_1^2X_2 + X_0X_1X_2^2) \\ &\quad - b_2 \cdot (X_0X_1X_2) - b_4 \cdot (X_0X_1 + X_0X_2 + X_1X_2) - b_6 \cdot (X_0 + X_1 + X_2) - b_8. \end{aligned}$$

We will quite often write  $S_A$  instead of  $S_{A,3}$ . For  $r \in \mathbb{Z}_{>3}$ , we recursively define the  $r$ th summation polynomial by

$$S_{A,r} = \text{Res}_X (S_{A,r-1}(X_0, \dots, X_{r-3}, X), S_{A,3}(X_{r-2}, X_{r-1}, X)) \in F[X_0, \dots, X_{r-1}],$$

where  $\text{Res}_X$  denotes the resultant with respect to  $X$ .

We have the following proposition.

**Proposition 4.** *Let  $F$  be a field and let  $E/F$  be an elliptic curve given by  $Y^2 + a_1XY + a_3Y = X^3 + a_2X^2 + a_4X + a_6$ . Let  $r \in \mathbb{Z}_{\geq 2}$  and let  $x_0, \dots, x_{r-1} \in \overline{F}$ . Then there are  $P_0, \dots, P_{r-1} \in E(\overline{F}) \setminus \{0\}$  with  $x(P_i) = x_i$  ( $i = 0, \dots, r-1$ ) such that  $P_0 + \dots + P_{r-1} = 0$  if and only if  $S_{(a_1, a_2, a_3, a_4, a_6), r}(x_0, \dots, x_{r-1}) = 0$ .*

It follows that given a point  $Q \neq 0$  and a positive integer  $m$ , we can represent a point  $Q$  as a sum of  $m$  points by solving  $S_{m+1}(x(Q), X_0, \dots, X_{m-1}) = 0$ .

Assume that  $F = k$ . Further linear constraints were introduced so as to restrict the  $X_i$ 's to a subspace  $V$  of  $k$  of dimension  $n'$  over  $k'$ . Let  $L(X) \in k[X^q] \subset k[X]$  be the additive polynomial whose roots are precisely the elements of the subspace  $V$ . We obtain a system  $\mathcal{F}$  of equations in  $k[X_0, \dots, X_{m-1}]$ , namely,

$$\mathcal{F} = \{S_{m+1}(x(Q), X_0, \dots, X_{m-1}), L(X_0), L(X_1), \dots, L(X_{m-1})\}.$$

Using this set-up and Weil descent (Remark 3), Diem showed that there exist sub-exponential time index calculus algorithms for ECDLP for some families of  $q$  and  $n$ .

The authors of [23] adopted a similar approach and considered ECDLP for  $q = 2^n$ . To solve the system  $\mathcal{F}$ , they considered a Weil descent system  $\mathcal{F}'$  over  $\mathbb{F}_2$  (notation as in Subsect. 3.1). With  $m = O(n^{1/3})$ , the authors claimed that this system can be solved via Gröbner basis algorithms in sub-exponential time of  $2^{O(n^{2/3} \log n)}$ . Essentially, their claim was based on the so-called “first fall degree assumption” which asserts that the first fall degree (see Remark 1) of a Weil descent polynomial system is close to the degree of regularity, the largest degree reached during Gröbner basis computations. More precisely, as the first fall degree of this system is  $O(m^2)$ , they conjectured that the degree of regularity is  $O(m^2)$  as well, thereby giving their heuristic result. According to the authors, they justified this heuristic assumption based on the following:

- The assumption of a constant gap between the first fall degree and the degree of regularity is widely believed to hold for Weil descent systems arising from HFE systems;
- The assumption is verified with experimental data for some multivariate polynomial systems for small parameters of  $n$  and  $m$ .

## 5.2 Discussion on the First Fall Degree Assumption

Here, we wish to highlight some examples where the first fall degree assumption is unlikely to hold.

**One Bivariate Summation Polynomial.** Let  $k$  be a finite field of cardinality  $2^n$ . Let  $E/k$  be a random elliptic curve in Weierstrass form with a random nonzero point  $Q \in E(k)$ . The following table records the degree of regularity for a Weil descent system comprising the bivariate polynomial  $S_3(X_0, X_1, x(Q))$ . Following the formulation in [23], we include linear constraints on  $X_0$  and  $X_1$  to

restrict their values to be in a random  $\mathbb{F}_2$ -subspace of  $k$  with dimension  $\lceil n/2 \rceil$ . Note that in this case, a Weil descent system  $\mathcal{F}'$ , after eliminating variables using the linear constraints, is a system in about  $n$  variables and has about  $n$  quadratic equations together with field equations of the form  $Y_i^2 + Y_i$ . We performed our computations using the “GroebnerBasis()” function in the Magma computer Algebra System and the degree of regularity is read off as the largest step degree where new polynomials are generated while the first fall degree is the smallest step degree at which a new lower-degree polynomial is generated. Here, the last column in the table records the degree of regularity of a system of  $n$  random quadratic equations in  $n$  variables over  $\mathbb{F}_2$  together with the  $n$  field equations. By a quadratic equation over  $\mathbb{F}_2$ , we mean an equation whose terms are a product of at most 2 variables.

$n$	First fall degree	Degree of regularity	Random
12	2	3	4
16	2	3	5
18	2	4	5
20	2	4	5
24	2	4	6
30	2	4	–
40	2	$\geq 5$	–

As the computations require more than 38 GB for  $n = 40$ , we are not able to carry out more experiments for larger values of  $n$ . However, the behaviour of the step degrees, another observable parameter from the Gröbner basis computations, suggests that the degree of regularity follows an increasing pattern as  $n$  increases. The above table raises doubt to the evidence of Assumption 2 from the article [23]: the gap between the degree of regularity and the first fall degree might be dependent on  $n$ .

*Remark 4.* Notice that our  $n = 40$  computation did not terminate. After the submission of this paper, with the help of the Caramel team from Nancy (France), we managed to terminate similar computations: the degree of regularity does seem to increase. See [19] for the details. This paper also contains a proof that the first fall degree in general is 2.

Note that in all our computations, the first fall degree is 2. One can prove that this is almost always the case when  $E$  is ordinary ( $a_1 \neq 0$ ) and  $Q$  is not the point of order 2. After some mathematics, the result follows from the following proposition where a complete proof can be found in [18, Chapter 7, Proposition 5.4]. The result is partially found in [25] as well.

**Proposition 5.** *Let  $E/k$  be an elliptic curve given by  $Y^2 + a_1XY + a_3Y = X^3 + a_2X^2 + a_4X + a_6$ . Assume that  $E$  is ordinary ( $a_1 \neq 0$ ). Then we have a*

surjective group morphism

$$\begin{aligned}
 E(k) &\rightarrow \mathbb{F}_2 \\
 0 &\mapsto 0 \\
 P &\mapsto \text{Tr}_{k/\mathbb{F}_2} \left( \frac{x(P) + a_2}{a_1^2} \right)
 \end{aligned}$$

with kernel  $2E(k)$ .

Note that knowledge of this map can speed up the summation polynomial approach for solving ECDLP, but probably only by a constant.

**Multiple Summation Polynomials.** Let  $k$  be a finite field of cardinality  $2^n$  and let  $E/k$  be an elliptic curve. Let  $Q \in E(k)$  be a nonzero point. Let  $m \in \mathbb{Z}_{\geq 3}$ . Instead of working with the  $(m + 1)$ st summation polynomial, we consider the following sequence of  $m$  sums:

$$\begin{aligned}
 Q &= P_1 + Q_1, \\
 Q_1 &= P_2 + Q_2, \\
 &\dots \quad \dots \\
 Q_{m-2} &= P_{m-1} + P_m.
 \end{aligned}$$

Observe that when this system is satisfied, we have  $Q = P_1 + \dots + P_m$ .

Once again, we let the  $x$ -coordinates of  $P_i$  be restricted in some subspace of dimension  $O(n/m)$ . Consider the set

$$\mathcal{F} = \{S_3(x(Q), X_1, Y_1), \dots, S_3(Y_{m-2}, X_{m-1}, X_m)\},$$

where the  $X_i$ 's are restricted to the subspace and the  $Y_i$  are unrestricted. We perform Weil descent to obtain a system  $\mathcal{F}'$  of equations in  $\mathbb{F}_2$ , where each equation has degree at most 3. According to [23], the first fall degree of this system is no greater than 5 (in fact, it is usually 2). Under the first fall degree assumption, this system will have a constant degree of regularity. In particular, it can be solved in time polynomial in  $m$ . Now, take  $m = O(n)$ . Letting the  $P_i$ 's take some specific points, say  $P_i = 2^i P, i = 1, \dots, m$ , this system will allow us to solve the ECDLP for a large proportion of points  $Q$  and thus, for all points  $Q$ . Consequently, we have a polynomial-time algorithm to solve ECDLP, which is highly improbable. We conclude that the first fall degree assumption is unlikely to hold for this system as well. In a similar way, using the first fall degree assumption, one can prove P=NP [19].

### 5.3 Open Problem on the Last Fall Degree

From the discussion in the preceding subsection, we believe that greater justification needs to be provided before one applies the first fall degree assumption to

a Weil descent system arising from a multivariate polynomial system. Nonetheless, as the above table demonstrates, the degree of regularity of a Weil descent system tends to grow more slowly than a random system with the same number of equations and variables. The big question is, how slowly does it grow. The slower it grows, the better algorithms there will be for ECDLP using Gröbner basis algorithms. As such, we believe that it remains worthwhile to analyze such systems in greater detail in order to get a more rigorous estimate to solve the ECDLP.

In this article, we defined the notion of a last fall degree of a multivariate polynomial system and describe an explicit algorithm to solve a zero-dimensional polynomial system whose time complexity depends on this last fall degree. As the last fall degree is independent of monomial orders, it enables us to give a rigorous bound on the time to solve a Weil descent system coming from univariate polynomials. We believe that this framework will be useful to help us investigate Weil descent systems from multivariate polynomials as well and will hopefully allow us to rigorously bound last fall degrees.

*Remark 5.* After submitting this paper, the authors continued their work in [20], and showed that the last fall degree of a Weil descent system arising from a zero-dimensional system also does not depend on the Weil descent parameter  $n$ . Unfortunately, the results of [20] do not apply to summation polynomials, because such systems are not zero-dimensional without adding field equations.


**Acknowledgements.** The authors would like to thank Bagus Santoso, Chaoping Xing and Yun Yang for their help and support in preparing this manuscript. We are grateful to Steven Galbraith and the anonymous reviewers for their valuable comments. Finally, we would like to thank the Caramel team from Nancy (France) for allowing us to use their computers to do experiments.

## References

1. Buchberger, B.: Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal. Ph.D. thesis, University of Innsbruck (1965)
2. Buchmann, J.A., Ding, J., Mohamed, M.S.E., Mohamed, W.S.A.E.: Mutantxl: solving multivariate polynomial equations for cryptanalysis. In: Handschuh, H., Lucks, S., Preneel, B., Rogaway, P. (eds.) *Symmetric Cryptography (Dagstuhl, Germany, 2009)*. Dagstuhl Seminar Proceedings, vol. 09031. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany (2009)
3. Courtois, N.T., Klimov, A.B., Patarin, J., Shamir, A.: Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In: Preneel, B. (ed.) *EUROCRYPT 2000*. LNCS, vol. 1807, pp. 392–407. Springer, Heidelberg (2000)
4. Courtois, N.T., Pieprzyk, J.: Cryptanalysis of block ciphers with overdefined systems of equations. In: Zheng, Y. (ed.) *ASIACRYPT 2002*. LNCS, vol. 2501, pp. 267–287. Springer, Heidelberg (2002)

5. Courtois, N.T., Patarin, J.: About the XL algorithm over  $GF(2)$ . In: Joye, M. (ed.) CT-RSA 2003. LNCS, vol. 2612, pp. 141–157. Springer, Heidelberg (2003)
6. Diem, C.: On the discrete logarithm problem in elliptic curves. *Compositio Math.* **147**, 75–104 (2011)
7. Ding, J., Hodges, T.J.: Inverting HFE systems is quasi-polynomial for all fields. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 724–742. Springer, Heidelberg (2011)
8. 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)
9. Faugère, J.C.: A new efficient algorithm for computing Gröbner bases ( $F_4$ ). *J. Pure Appl. Algebra* **139**, 61–88 (1999)
10. Faugère, J.C.: A new efficient algorithm for computing Gröbner bases without reduction to zero  $F_5$ . In: Proceedings of ISSAC, pp. 75–83. ACM Press (2002)
11. Faugère, J.C., Gianni, P.M., Lazard, D., Mora, T.: Efficient computation of zero-dimensional Gröbner bases by change of ordering. *J. Symb. Comput.* **16**(4), 329–344 (1993)
12. Galbraith, S.D., Smart, N.P.: A cryptographic application of Weil descent. In: Walker, M. (ed.) Cryptography and Coding 1999. LNCS, vol. 1746, pp. 191–200. Springer, Heidelberg (1999)
13. von zur Gathen, J., Panario, D.: Factoring polynomials over finite fields: a survey. *J. Symbolic Comput.* **31**(1–2), 3–17 (2001). Computational algebra and number theory, (1996)
14. Gaudry, P.: Index calculus for abelian varieties of small dimension and the elliptic curve discrete logarithm problem. *J. Symb. Comput.* **44**(12), 1690–1702 (2009)
15. Gaudry, P., Hess, F., Smart, N.P.: Constructive and destructive facets of Weil descent on elliptic curves. *J. Cryptology* **15**(1), 19–46 (2002)
16. 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)
17. Kipnis, A., Shamir, A.: Cryptanalysis of the HFE public key cryptosystem by relinearization. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 19–30. Springer, Heidelberg (1999)
18. Kusters, M.: Groups and fields in arithmetic. Ph.D. thesis, Universiteit Leiden (2014)
19. Kusters, M., Yeo, S.L.: Notes on summation polynomials. Preprint (2015). <http://arxiv.org/abs/1503.08001>
20. Huang, M.-D.A., Kusters, M., Yang, Y., Yeo, S.L.: On the last fall degree of zero-dimensional Weil descent systems. Preprint (2015). <http://arxiv.org/abs/1505.02532>
21. 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)
22. Petit, C.: Bounding HFE with SRA. Preprint (2013). [http://www0.cs.ucl.ac.uk/staff/c.petit/files/SRA\\_GB.pdf](http://www0.cs.ucl.ac.uk/staff/c.petit/files/SRA_GB.pdf)
23. Petit, C., Quisquater, J.-J.: On polynomial systems arising from a Weil descent. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 451–466. Springer, Heidelberg (2012)
24. Semaev, I.: Summation polynomials and the discrete logarithm problem on elliptic curves. Preprint (2004). <https://eprint.iacr.org/2004/031.pdf>
25. Seroussi, G.: Compact representation of elliptic curve points over  $\mathbb{F}_2^n$  research contribution to IEEE P1363 (1998)

# A Quasipolynomial Reduction for Generalized Selective Decryption on Trees

Georg Fuchsbauer<sup>1</sup>, Zahra Jafargholi<sup>2</sup>, and Krzysztof Pietrzak<sup>1</sup>

<sup>1</sup> Institute of Science and Technology Austria, Vienna, Austria  
`{gfuchsbauer,pietrzak}@ist.ac.at`

<sup>2</sup> Northeastern University, Boston, USA  
`z.jafargholi@gmail.com`

**Abstract.** Generalized Selective Decryption (GSD), introduced by Panjwani [TCC’07], is a game for a symmetric encryption scheme  $\text{Enc}$  that captures the difficulty of proving adaptive security of certain protocols, most notably the Logical Key Hierarchy (LKH) multicast encryption protocol. In the GSD game there are  $n$  keys  $k_1, \dots, k_n$ , which the adversary may adaptively corrupt (learn); moreover, it can ask for encryptions  $\text{Enc}_{k_i}(k_j)$  of keys under other keys. The adversary’s task is to distinguish keys (which it cannot trivially compute) from random. Proving the hardness of GSD assuming only IND-CPA security of  $\text{Enc}$  is surprisingly hard. Using “complexity leveraging” loses a factor exponential in  $n$ , which makes the proof practically meaningless.

We can think of the GSD game as building a graph on  $n$  vertices, where we add an edge  $i \rightarrow j$  when the adversary asks for an encryption of  $k_j$  under  $k_i$ . If restricted to graphs of depth  $\ell$ , Panjwani gave a reduction that loses only a factor exponential in  $\ell$  (not  $n$ ). To date, this is the only non-trivial result known for GSD.

In this paper we give almost-polynomial reductions for large classes of graphs. Most importantly, we prove the security of the GSD game restricted to trees losing only a quasi-polynomial factor  $n^{3 \log n + 5}$ . Trees are an important special case capturing real-world protocols like the LKH protocol. Our new bound improves upon Panjwani’s on some LKH variants proposed in the literature where the underlying tree is not balanced. Our proof builds on ideas from the “nested hybrids” technique recently introduced by Fuchsbauer et al. [Asiacrypt’14] for proving the adaptive security of constrained PRFs.

## 1 Introduction

Proving security of protocols where an adversary can make queries and/or corrupt players *adaptively* is a notoriously hard problem. Selective security, where the adversary must commit to its queries before the protocol starts, often allows for an easy proof, but in general does not imply (the practically relevant) adaptive security notion [CFGN96].

---

G. Fuchsbauer and K. Pietrzak—Supported by the European Research Council, ERC Starting Grant (259668-PSPC).

Panjwani [Pan07] argues that the two common approaches to achieving adaptive security, namely requiring that all parties erase past data [BH93], or using *non-committing* encryption [CFGN96] are not satisfactory. He introduces the *generalized selective decryption* (GSD) problem and uses it as an abstraction of security requirements of multicast encryption protocols [WGL00, MP06]. GSD is defined by a very simple game that captures the difficulty of proving adaptive security of some interesting protocols.

**The Generalized Selective Decryption (GSD) Game.** In the GSD game we consider a symmetric encryption scheme  $\text{Enc}$  and a parameter  $n \in \mathbb{N}$ . Initially, we sample  $n$  random keys  $k_1, \dots, k_n$  and a bit  $b \in \{0, 1\}$ . During the game the adversary  $A$  can make two types of queries. Encryption query: on input  $(i, j)$  she receives  $c = \text{Enc}_{k_i}(k_j)$ ; corruption query: on input  $i$ , she receives  $k_i$ . At some point,  $A$  chooses some  $i$  to be challenged on. If  $b = 0$ , she gets the key  $k_i$ ; if  $b = 1$ , she gets a uniformly random  $r_i$ .<sup>1</sup> Finally,  $A$  outputs a guess bit  $b'$ . The goal is prove that for any efficient  $A$ ,  $|\Pr[b = b'] - 1/2|$  is negligible (or, equivalently,  $k_i$  is pseudorandom) assuming only that  $\text{Enc}$  is a secure encryption scheme. We only allow one challenge query, but this notion is equivalent to allowing any number of challenge queries by a standard hybrid argument (losing a factor that is only the number of challenge queries).

It is convenient to think of the GSD game as dynamically building a graph, which we call key graph. We start with a graph with  $n$  vertices labeled  $1, \dots, n$ , where we associate vertex  $i$  with key  $k_i$ . On an encryption query  $\text{Enc}_{k_i}(k_j)$  we add a directed edge  $i \rightarrow j$ . On a corruption query  $i$  we label the vertex  $i$  as corrupted. Note that if  $i$  is corrupted then  $A$  also learns all keys  $k_j$  for which there is a path from  $i$  to  $j$  in the key graph by simply decrypting the keys along that path. To make the game non-trivial, challenge queries are thus only allowed for keys that are not reachable from any corrupted key. Another restriction we must make is to disallow encryption cycles, i.e., loops in the graph. Otherwise we cannot hope to prove security assuming only standard security (in our case IND-CPA) of the underlying encryption scheme, as this would require circular (or key-dependent-message) security [BRS03], which is stronger than IND-CPA [ABBC10]. Finally, we require that the challenge query is a leaf in the graph; this restriction too is necessary unless we make additional assumptions on the underlying encryption scheme (cf. Footnote 9).

**SELECTIVE SECURITY OF GSD.** In order to prove security of the GSD game, one must turn an adversary  $A$  that breaks the GSD game with some advantage  $\epsilon = |\Pr[b = b'] - 1/2|$  into an adversary  $B$  that breaks the security of  $\text{Enc}$  with some advantage  $\epsilon' = \epsilon'(\epsilon)$ . The security notion we consider is the standard

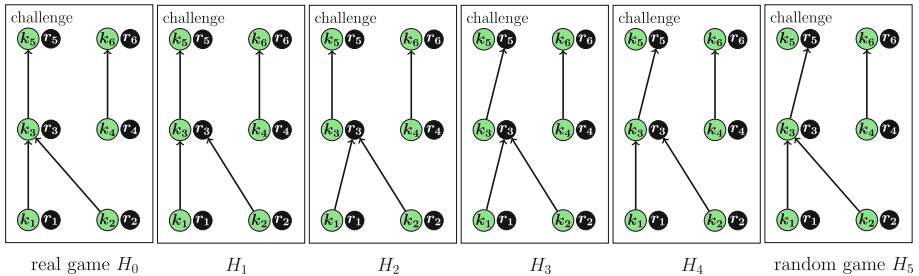
<sup>1</sup> Below, we will consider a (seemingly) different experiment and output  $k_i$  in both cases ( $b = 0$  and  $b = 1$ ), but if  $b = 1$ , then on any query  $(j, i)$ , we will encrypt  $\text{Enc}_{k_j}(r_i)$  and not  $\text{Enc}_{k_j}(k_i)$ . This is just a semantic change assuming the following: during the experiment we always answer encryption queries of the form  $(a, b)$  with  $\text{Enc}_{k_a}(k_b)$  (note that we don't know if we're encrypting the challenge at this point), and once the adversary chooses a challenge  $i$ , if  $b = 1$ , we simply switch the values of  $r_i$  and  $k_i$  (this trick is already used in [Pan07]).



notion of indistinguishability under chosen plaintext attacks (IND-CPA). Recall that in the IND-CPA game an adversary  $B$  is given access to an encryption oracle  $\text{Enc}_k(\cdot)$ . At some point  $B$  chooses a pair of messages  $(m_0, m_1)$ , then gets a challenge ciphertext  $c = \text{Enc}_k(m_b)$  for a random bit  $b$ , and must output a guess  $b'$ . The advantage of  $B$  is  $|\Pr[b = b'] - 1/2|$ .

It is not at all clear how to construct an adversary  $B$  that breaks IND-CPA from an  $A$  that breaks GSD. This problem becomes much easier if we assume that  $A$  breaks the *selective* security of GSD, where  $A$  must choose all its encryption, corruption and challenge queries before the experiment starts.

In fact, it is sufficient to know the topology of the connected component in the key graph that contains the challenge node. Let  $\alpha$  denote the number of edges in this component. One can now define a sequence of  $2\alpha$  hybrid games  $H_0, \dots, H_{2\alpha-1}$ , where the first game is the real game (i.e., the GSD game with  $b = 0$  where the adversary gets the key), the last hybrid is the random game ( $b = 1$ ), and moreover, from any adversary that distinguishes  $H_i$  from  $H_{i+1}$  with some advantage  $\epsilon'$ , we get an adversary against the IND-CPA security of  $\text{Enc}$  with the same advantage. Thus, given an  $A$  breaking GSD with advantage  $\epsilon$ , we can break the IND-CPA security with advantage  $\epsilon' \geq \epsilon/(2\alpha - 1) \geq \epsilon/n^2$  (as an  $n$  vertex graph has  $\leq n^2$  edges). We illustrate this reduction in Fig. 1.



**Fig. 1.** Hybrids for the selective security proof. Green nodes correspond to keys, dark nodes are random values. The adversary  $A$  commits to encryption queries  $(1, 3)$ ,  $(2, 3)$ ,  $(3, 5)$  and challenge query  $(4, 6)$  is outside the connected component containing the challenge and thus not relevant for the hybrids.  $A$  could also corrupt keys 4 and 6, which are also outside.) Hybrid  $H_0$  is the real game, hybrid  $H_5$  is the random game, where instead of an encryption of the challenge key  $\text{Enc}_{k_3}(k_5)$ , the adversary gets an encryption of the random value  $\text{Enc}_{k_3}(r_5)$ . If an adversary  $A$  can distinguish any two consecutive hybrids  $H_i$  and  $H_{i+1}$  with some advantage  $\delta$ , we can use  $A$  to construct  $B$  which breaks the IND-CPA security of  $\text{Enc}$  with the same advantage  $\delta$ : E.g., assume  $B$  is given an IND-CPA challenge  $C = \text{Enc}_k(z)$  where  $z$  is one of two messages (which we call  $k_5$  and  $r_5$ ). Now  $B$  can simulate game  $H_2$  for  $A$ , but when  $A$  makes the encryption query  $(3, 5)$ ,  $B$  answers with  $C$ . If  $z = k_5$  then  $B$  simulates game  $H_2$ ; but if  $z = r_5$ , it simulates game  $H_3$ . Note that  $B$  can simulate the games because  $k_3$ , which in the simulation is  $B$ 's challenger's key, is not used anywhere else. Thus,  $B$  has the same advantage in the IND-CPA game as  $A$  has in distinguishing  $H_3$  from  $H_4$  (Color figure online).

ADAPTIVE SECURITY OF GSD. In the selective security proof for GSD we crucially relied on the fact that we knew the topology of the underlying key graph. Proving adaptive security, where the adversary decides what queries to ask adaptively during the experiment, is much more difficult. A generic trick to prove adaptive security is “complexity leveraging”, where one simply turns an adaptive adversary into a selective one by initially guessing the adaptive adversary’s choices and committing to those (as required by the selective security game). If during the security game the adaptive choices by the adversary disagree with the guessed ones, we simply abort. The problem with this approach is that assuming the adaptive adversary has advantage  $\epsilon$ , the constructed selective adversary only has advantage  $\epsilon/P$  where  $1/P$  is the probability of that our guess is correct, which is typically exponentially small. Concretely, in the GSD game we need to guess the nodes in the connected component containing the challenge, and as the number of such choices is exponential in the number of keys  $n$ , this probability is  $2^{-\Theta(n)}$ .

No proofs for the adaptive security of GSD with a subexponential (in  $n$ ) security loss are known in general. But remember that the GSD problem abstracts problems we encounter in proving adaptive security of many real-world applications where the underlying key graph is typically not completely arbitrary, but often has some special structure. Motivated by this, Panjwani [Pan07] investigated better reductions assuming some special structure of the key graph. He gives a proof where the security degradation is only exponential in the *depth* of the key graph, as opposed to its size. Concretely, he proves that if the encryption scheme is  $\epsilon$ -IND-CPA secure then the adaptive GSD game with  $n$  keys where the adversary is restricted to key graphs of depth  $\ell$  is  $\epsilon'$ -secure where

$$\epsilon' = \epsilon \cdot O(n \cdot (2n)^\ell).$$

Until today, Panjwani’s bound is the only non-trivial improvement over the  $2^{\Theta(n)}$  loss for GSD.

**Our Result.** The main result of this paper is Theorem 2, which states that GSD restricted to trees can be proven secure with only a quasi-polynomial loss

$$\epsilon' = \epsilon \cdot n^{3 \log(n)+5}.$$

Our bound is actually even stronger as the entire key graph need not be a tree; it is sufficient that the subgraph containing only the nodes from which the challenge node can be reached is a tree (when ignoring edge directions).

The bound above is derived from a more fine-grained bound: assuming that the longest path in the key graph is of length  $\ell$ , the in-degree of every node is at most  $d$  and the challenge node can be reached from at most  $s$  sources (i.e., nodes with in-degree 0) we get

$$\epsilon' = \epsilon \cdot dn((2d+1)n)^{\lceil \log s \rceil} (3n)^{\lceil \log \ell \rceil}.$$

Note that  $\ell, d$  and  $s$  are at most  $n$  and the previous bound was derived from this by setting  $\ell = d = s = n$ . Panjwani [Pan07] uses his bound to give a quasi-polynomial reduction of the Logical Key Hierarchy (LKH) protocol [WGL00].

Panjwani first fixes a flaw in LKH, and calls the new protocol rLKH with “r” for repaired. rLKH is basically the GSD game restricted to a binary tree.<sup>2</sup>

The users correspond to the leaves of this tree, and their keys consists of all the nodes from the root to their leaf. Thus, if the tree is almost full and balanced, then it has only depth  $\ell \approx \log n$  and Panjwani’s bound loses only a quasi-polynomial factor  $n^{\log(n)+2}$  (if  $\ell = \log n$ ). As here  $d = 2, \ell = \log n, s = n$ , our bound gives a slightly worse bound  $n^{\log(n)+\log \log(n)+4}$  for this particular problem, but this is only the case if a large fraction of the keys are actually used, and the adversary gets to see almost all of them. If  $\ell$  is significantly larger than  $\log n$  (e.g., because only few of the keys are active, or the tree is constructed in an unbalanced way like e.g. proposed in [SS00]), our bounds decrease only marginally, as opposed to exponentially fast in  $\ell$  in [Pan07].

**Graphs with Small Cut-Width.** The reason our result is restricted to trees is that in the process of generating the hybrids, we have to guess nodes such that removing this node splits the tree in a “nice” way (this has to be done  $\log n$  times, losing a factor  $n$  in the distinguishing advantage every time).

One can generalize this technique (but we do not work out the details in this paper) to graphs with small “cut-width”, where we say that a graph has cut-width  $w$  if for any two vertices  $u, v$  that are not connected by an edge, there exists a set of at most  $w$  vertices such that removing those disconnects  $u$  from  $v$  (a tree has cut-width  $w = 1$ ). For graphs with cut-width  $w$  we get

$$\epsilon' = \epsilon \cdot n^{(2w+1)\log(n)+4},$$

which is subexponential in  $n$ , and thus beats the existing exponential bound whenever  $w = o(n/\log^2(n))$ . Whether there exists a subexponential reduction which works for any graph is an intriguing open problem.

**Shorter Keys from Better Reduction.** An exponential security loss (as via complexity leveraging) means that, even when assuming exponential hardness of Enc (which is a typical assumption for symmetric encryption schemes like AES), one needs to use keys for Enc whose length is at least linear in  $n$  to get any security guarantee for the hardness of GSD at all. Whereas our bound for trees means that a key of length  $\text{polylog}(n)$  is sufficient to get asymptotically overwhelming security (again assuming Enc is exponentially hard).

**Nested Hybrids.** In a classical paper [GGM86] Goldreich, Goldwasser and Micali constructed a pseudorandom function (PRF) from a pseudorandom generator (PRG). More recently, three papers independently [BW13, KPTZ13, BGI14] observed that this construction is also a so-called *constrained* PRF, where for every string  $x$  one can compute a constrained key  $k_x$  that allows evaluation of the PRF on all inputs with prefix  $x$ . Informally, the security requirement is that

<sup>2</sup> Let us stress that the graph obtained when just adding an edge for every encryption query in rLKH is not a tree after a rekeying operation. But for every node  $v$ , the subgraph we get when only keeping the nodes from which  $v$  can be reached is a tree, and as explained above, this is sufficient.

an adversary that can ask for constrained keys cannot distinguish the output of the PRF on some challenge input from random.

All three papers [BW13, KPTZ13, BGI14] only prove *selective* security of this constrained PRF, where before any queries the adversary must commit to the input on which it wants to be challenged. This proof is a hybrid argument losing a factor  $2m$  in the distinguishing advantage, where  $m$  is the PRF input length. One can then get adaptive security losing a huge exponential factor  $2^m$  via complexity leveraging. Subsequently, Fuchsbauer et al. [FKPR14] gave a reduction that only loses a quasi-polynomial factor  $(3q)^{\log m}$ , where  $q$  denotes the number of queries made by the adversary. Our proofs borrows ideas from their work.

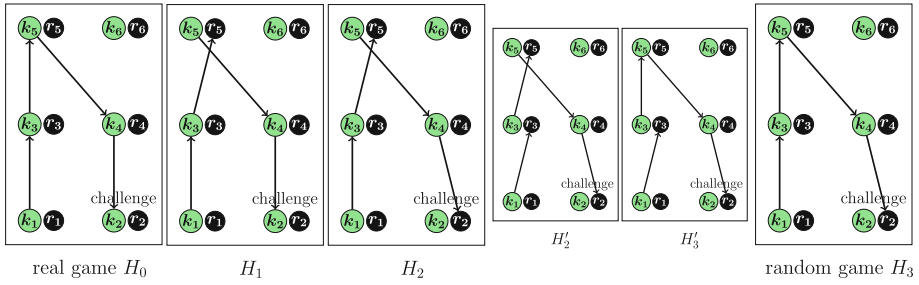
Very informally, the idea behind their proof is the following. In the standard proof for adaptive security using leveraging one first guesses the challenge query (losing a huge factor  $2^m$ ), which basically turns the adaptive attacker into a selective one, followed by a simple hybrid argument (losing a small factor  $2m$ ) to prove selective security. The proof from [FKPR14] also first makes a guessing step, but a much simpler one, namely which of the  $q$  queries made by the adversary is the first to coincide with the challenge query on the first  $m/2$  bits. This is followed by a hybrid argument losing a factor 3, so both steps together lose a factor  $3q$ . At this point the reduction is not finished yet, but intuitively the problem was reduced to itself but on inputs of only half the size  $m/2$ . These two steps can be iterated  $\log m$  times (losing a total factor of  $(3q)^{\log m}$ ) to get a reduction to the security of the underlying PRG.

**Proof Outline for Paths.** Our proof for GSD uses an approach similar to the one just explained, iterating fairly simple guessing steps with hybrid arguments, but the analogy ends here, as the actual steps are very different.

We first outline the proof for the adaptive security of the GSD game for a special case where the adversary is restricted in the sense that the connected component in the key graph containing the challenge must be a path. Even for this very special case, currently the best reduction [Pan07] loses an *exponential* factor  $2^{\Theta(n)}$ . We will now outline a reduction losing only a *quasi-polynomial*  $n^{\log n}$  factor.<sup>3</sup> Recall that the standard way to prove adaptive security is to first guess the entire connected component containing the challenge, and then prove selective security as illustrated in Fig. 1.

Our approach is not to guess the entire path, but in a first step only the node in the middle of the path (as we make a uniform guess, it will be correct with probability  $1/n$ ). This reduces the adaptive security game to a “slightly

<sup>3</sup> Let us mention that it is trivial to prove security of GSD restricted to paths if we additionally assume that for random keys  $k, k'$  the ciphertext  $\text{Enc}_k(k')$  is uniform given  $k'$  (this is e.g. the case for one-time pad encryption  $\text{Enc}_k(k') = k \oplus k'$ ): then the real and random challenge have the same distribution (they're uniform) and thus even a computationally unbounded adversary has zero advantage. (This is because in the path case, every key is used only once to encrypt.) The proof we outline here does not require this special property of  $\text{Enc}$ , and this will be crucial to later generalize it to more interesting graphs.



**Fig. 2.** Illustration of our adaptive security proof for paths.

selective” game where the adversary must commit initially to this middle node, at the price of losing a factor  $n$  in the distinguishing advantage.<sup>4</sup>

Let  $H_0$  and  $H_3$  denote these “slightly selective” real and random GSD games (we also assume that the adversary initially commits to the challenge query, which costs another factor of  $n$ ). We illustrate this with a small example featuring a path of length 4 in Fig. 2. The correct guess for the middle node for the particular run of the experiment illustrated in the figure is  $i = 5$ . As now we know the middle vertex is  $i = 5$ , we can define new games  $H_1$  and  $H_2$  which are derived from  $H_0$  and  $H_3$ , respectively, by replacing the ciphertext  $\text{Enc}_{k_j}(k_i)$  with an encryption  $\text{Enc}_{k_j}(r_i)$  of a random value (in the figure this is illustrated by replacing the edge  $k_j \rightarrow k_i$  with  $k_j \rightarrow r_i$ ).

So, what have we gained? If our adaptive adversary has advantage  $\epsilon$  in distinguishing the real and random games then she has advantage at least  $\epsilon/n$  to distinguish the “slightly selective” real and random games  $H_0$  and  $H_3$ , and thus for some  $i \in \{0, 1, 2\}$  she can distinguish the games  $H_i$  and  $H_{i+1}$  with advantage  $\epsilon/3n$ . Looking at two consecutive games  $H_i$  and  $H_{i+1}$ , we see that they only differ in one edge (e.g., in  $H_2$  we answer the query  $(3, 5)$  with  $\text{Enc}_{k_3}(r_5)$ , in  $H_3$  with  $\text{Enc}_{k_3}(k_5)$ ), and moreover this edge will be at the end of a path that now has only length 2, that is, half the length of the path in our original real and random games.

We can now continue this process, constructing new games where the path length is halved, paying a factor  $3n$  in distinguishing advantage. For example, as illustrated in Fig. 2, we can guess the node that halves the path leading to

<sup>4</sup> We never actually construct this “slightly selective” adversary, but (as in complexity leveraging) we simply commit to a random guess, then run the adaptive adversary, and if its queries are not consistent with our guess, we abort outputting a random value. (We could also output a constant value; the point is that the advantage of the adversary, conditioned on our guess being wrong, is zero; whereas, conditioned on the guess being correct, it is the same as the advantage of the adaptive adversary). However, instead of this experiment it is easier to follow our proof outline by thinking of the adversary actually committing to its choices initially, but the reduction paying a factor (in the distinguishing advantage of the adversary that is allowed to make this choice adaptively) that corresponds to the size of the sample space of this guess.

the differing query in games  $H_2$  and  $H_3$  (for the illustrated path this would be  $i = 3$ ), then define new games where we assume the adversary commits to this node (paying a factor  $n$ ), and then define two new games  $H'_2$  and  $H'_3$ , which are derived from games  $H_2$  and  $H_3$  (which now are augmented by our new guess), respectively, by answering the query  $(j, i)$  that asks for an encryption of this node (in the figure  $(j, i) = (1, 3)$ ) with an encryption  $\text{Enc}_{k_1}(r_3)$  instead of  $\text{Enc}_{k_1}(k_3)$ .

If we start with a path of length  $\ell \leq n$  then after  $\log \ell \leq \log n$  iterations of this process we proved the existence of two consecutive games (call them  $G_0$  and  $G_1$ ) that differ only in a single edge  $j \rightarrow i$  and the vertex  $j$  has in-degree 0. That is, both games are identical, except that in one game the encryption query  $(j, i)$  is answered with  $\text{Enc}_{k_j}(k_i)$  and in the other with  $\text{Enc}_{k_j}(r_i)$ . Moreover, the key  $k_j$  is not used anywhere else in the experiment and we know exactly when this query is made during the experiment (as the adversary committed to  $i$ ).

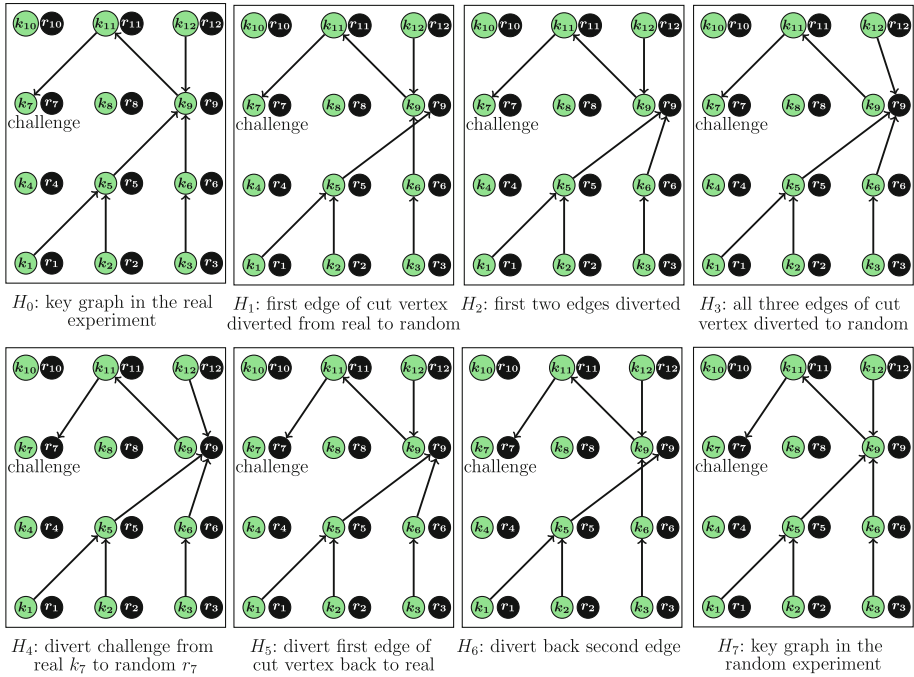
Given a distinguisher  $A$  for  $G_0$  and  $G_1$ , we can now construct an attacker  $B$  that breaks the IND-CPA security of the underlying encryption scheme with the same advantage: in the IND-CPA game  $B$  chooses two random messages  $m_0, m_1$  and asks to be challenged on them.<sup>5</sup> The game samples a random bit  $b$  and returns the challenge  $C = \text{Enc}_k(m_b)$  to  $B$ , which must then output a guess  $b'$  for  $b$ . At this point,  $B$  invokes  $A$  and simulates the game  $G_0$  for it, choosing all keys at random, except that it uses  $C$  to answer the encryption query  $(j, i)$ .<sup>6</sup> Finally,  $B$  forwards  $A$ 's guess  $b'$ . Identifying  $(k, m_0, m_1)$  with  $(k_j, k_i, r_i)$ , we see that depending on whether  $b = 0$  or  $b = 1$ ,  $B$  simulates either  $G_0$  or  $G_1$ . Thus, whatever advantage  $A$  has in distinguishing  $G_0$  from  $G_1$ ,  $B$  will break the IND-CPA security of  $\text{Enc}$  with the same advantage.

**Proof Outline for Trees.** We will now outline our reduction of the adaptive security of GSD to the IND-CPA security of  $\text{Enc}$  for a more general case. Namely, the adversary is only restricted in that the key graph resulting from its queries is such that the connected component containing the challenge is a tree. (Recall that we already disallowed cycles in the key graph as this would require circular security. Being a tree means that we also have no cycles in the key graph when ignoring edge directions). Note that paths as discussed in the previous section are very special trees. The GSD problem on trees is particularly interesting, as it captures some multicast encryption protocols like the Logical Key Hierarchy (LKH) protocol [WGL00]. We refer the reader to [Pan07] for details.

TREES WITH IN-DEGREES  $\leq 1$ . Let us first consider the case where the connected component containing the challenge is a tree, and moreover all its vertices have in-degree 0 or 1. It turns out that the proof outlined for paths goes through with only minor changes for such trees. Note that such a tree has exactly one vertex with in-degree 0, which we call the *root*, and there is a unique path from the root to the challenge node. We can basically ignore all the edges not on this

<sup>5</sup> Note that  $B$  makes no encryption queries at all (which are allowed by the IND-CPA experiment).

<sup>6</sup> Note that since node  $j$  has in-degree 0, we can identify  $k_j$  with the key  $k$  used by the IND-CPA experiment, as we never have to encrypt  $k_j$ .



**Fig. 3.** Illustration of our adaptive security proof for general trees.

path and do a reduction as the one outlined above. The only difference is that now, when simulating the game  $G_b$  (where  $b$  is 0 or 1 depending on the whether the challenge  $C$  with which we answer the encryption query  $(j, i)$  is  $\text{Enc}_{k_j}(k_i)$  or  $\text{Enc}_{k_j}(r_i)$ ), the adversary can also ask for encryption queries  $(j, x)$  for any  $x$ . This might seem like a problem as we do not know  $k_j$  (we identified  $k_j$  with the key used by the IND-CPA challenger). But recall that in the IND-CPA game there is an encryption oracle  $\text{Enc}_{k_j}(\cdot)$ , which we can query for the answer  $\text{Enc}_{k_j}(k_x)$  to such encryption queries.

**GENERAL TREES.** For general trees, where nodes can have in-degree greater than 1, we need to work more. The proof for paths does not directly generalize, as now nodes (in particular, the challenge) can be reached from more than one node with in-degree 0. We call these the *sources* of this node; for example in the tree  $H_0$  in Fig. 3, the (challenge) node  $k_7$  has 4 sources  $k_1, k_2, k_3$  and  $k_{12}$ .

On a high level, our proof strategy will be to start with a tree where the challenge node  $c$  has  $s$  sources (more precisely, we have two games that differ in one edge that points to  $k_i$  in one game, and to  $r_i$  in the other, like games  $H_0$  and  $H_7$  in Fig. 3). We then guess a node  $v$  that “splits” the tree in a nice way, by which we mean the following: Assume  $v$  has in-degree  $d$  and we divert every edge going into  $v$  to a freshly generated node; let’s call them  $v_1, \dots, v_d$ . Then this splits the tree into a forest consisting of  $d + 1$  trees (the component containing

the challenge and one component for every  $v_i$ ). The node  $v$  “well-divides” the tree if after the split the node  $c$  and all of  $v_1, \dots, v_d$  have at most  $\lceil s/2 \rceil$  sources.

As an example, consider again the tree  $H_0$  in Fig. 3, where the challenge node  $k_7$  has 4 sources. The node  $k_9$  would be a good guess, as it well-divides the tree: consider the forest after splitting at this node as described above (creating new nodes  $v_1, v_2, v_3$  and diverting the edges going into  $k_9$  to them, i.e., replacing  $k_5 \rightarrow k_9$  by  $k_5 \rightarrow v_1$ ,  $k_6 \rightarrow k_9$  by  $k_6 \rightarrow v_2$ , and  $k_{12} \rightarrow k_9$  by  $k_{12} \rightarrow v_3$ ). Then we obtain 4 trees, where now  $c = k_7$  has only one source ( $k_9$ ) and the new nodes  $v_1, v_2, v_3$  have 2, 1 and 1 sources, respectively.

Once we have guessed a well-dividing node  $v$  (or equivalently, the adversary has committed to such a node), we define  $2d$  hybrid games (where  $d$  is the degree of the well-dividing node) between the two initial games, which we call  $H_0$  and  $H_{2d+1}$ , as follows.  $H_1$  is derived from  $H_0$  by diverting the first encryption query that asks for an encryption of  $v$  (i.e., that is of the form  $(j, v)$  for some  $j$ ) from real to random; that is, we answer with  $\text{Enc}_{k_j}(r_v)$  instead of  $\text{Enc}_{k_j}(k_v)$ . For  $i \leq d$ ,  $H_i$  is derived from  $H_0$  by diverting the first  $i$  encryption queries.  $H_{d+1}$  is derived from  $H_d$  by diverting the encryption query that asks for an encryption of the challenge  $c$  from real to random. The final  $d-1$  hybrids games are used to switch the encryption of  $v$  back from random to real, one edge at a time. This process is illustrated in the games  $H_0$  to  $H_7$  in Fig. 3.

Because  $v$  was well-dividing (and we show in the full version that such a node always exists), we can prove the following property for any two consecutive games  $H_i$  and  $H_{i+1}$ : they differ in exactly one edge, which for some  $j, v$  in one game is  $k_j \rightarrow k_v$  and  $k_j \rightarrow r_v$  in the other, and moreover,  $k_j$  has at most  $\lceil s/2 \rceil$  sources.

If an adversary can distinguish  $H_0$  and  $H_{2d+1}$  with advantage  $\epsilon$  then it must distinguish two hybrids  $H_i$  and  $H_{i+1}$  with advantage  $\epsilon / ((2d + 1)n)$  (where  $n$  accounts for guessing the well-dividing node). But any such two hybrids now only have at most  $\lceil s/2 \rceil$  sources. If we repeat this guessing/hybrid steps  $\log s$  times, we end up with two games  $G_0$  and  $G_1$  which differ in one edge that has only one source. At this point we can then use our reduction for trees with only one source outlined above.

**ANALYZING THE SECURITY LOSS.** To halve the number of sources, we guess a well-dividing vertex (which costs a factor  $n$  in the reduction), and then must add up to  $2d$  intermediate hybrids (where  $d$  is the maximum in-degree of any node), costing another factor  $2d + 1$ . Assuming that the number of sources is bounded by  $s$ , we have to iterate the process at most  $\log s$  times. Finally, we lose another factor  $d$  (but only once) because our final node can have more than one ingoing edge. Overall, assuming the adversary breaks the GSD game with advantage  $\epsilon$  on trees with at most  $s$  sources and in-degree at most  $d$ , our reduction yields an attacker against the IND-CPA security of  $\text{Enc}$  with advantage

$$\epsilon / dn((2d + 1)n)^{\lceil \log s \rceil} (3n)^{\lceil \log \ell \rceil} .$$

For general trees, since  $s, d \leq n$ , we have  $\epsilon / n^{3 \log n + 5}$ .



## 2 Preliminaries

For  $a \in \mathbb{N}$ , we let  $[a] = \{1, 2, \dots, a\}$  and  $[a]_0 = [a] \cup \{0\}$ . We say adversary (or distinguisher)  $D$  is  $t$ -bounded if  $D$  runs in time  $t$ .

**Definition 1 (Indistinguishability).** *Two distributions  $X$  and  $Y$  are  $(\epsilon, t)$ -indistinguishable, denoted  $Y \sim_{(\epsilon, t)} X$  or  $\Delta_t(Y, X) \leq \epsilon$ , if no  $t$ -bounded distinguisher  $D$  can distinguish them with advantage greater than  $\epsilon$ , i.e.,*

$$\Delta_t(Y, X) \leq \epsilon \iff \forall D_t : |Pr[D_t(X) = 1] - Pr[D_t(Y) = 1]| \leq \epsilon.$$

**Symmetric Encryption.** A pair of algorithms  $(\text{Enc}, \text{Dec})$  with input  $k \in \{0, 1\}^\lambda$ , where  $\lambda$  is the security parameter, and a message  $m$  (or a ciphertext) from  $\{0, 1\}^*$  is a symmetric-key encryption scheme if for all  $k, m$  we have  $\text{Dec}_k(\text{Enc}_k(m)) = m$ . Consider the game  $\text{Exp}_{\text{Enc}, D}^{\text{IND-CPA}-b}$  between a challenger  $C$  and a distinguisher  $D$ :  $C$  chooses a uniformly random key  $k \in \{0, 1\}^\lambda$  and a bit  $b \in \{0, 1\}$ ;  $D$  can make encryption queries for messages  $m$  and receives  $\text{Enc}_k(m)$ ; finally,  $D$  outputs a pair  $(m_0, m_1)$ , is given  $\text{Enc}_k(m_b)$  and outputs a bit  $b' \in \{0, 1\}$ , which is also the output of  $\text{Exp}_{\text{Enc}, D}^{\text{IND-CPA}-b}$ <sup>7</sup>.

**Definition 2.** *Let  $t \in \mathbb{N}^+$  and  $0 < \epsilon < 1$ . An encryption scheme  $(\text{Enc}, \text{Dec})$  is  $(t, \epsilon)$ -IND-CPA secure if for any  $t$ -bounded distinguisher  $D$ , we have*

$$|Pr[\text{Exp}_{\text{Enc}, D}^{\text{IND-CPA}-1} = 1] - Pr[\text{Exp}_{\text{Enc}, D}^{\text{IND-CPA}-0} = 1]| \leq \epsilon.$$

## 3 The GSD Game

In this section we describe the generalized selective decryption game as defined in [Pan07] and give our main theorem. Consider the following game,  $\text{Exp}_{\text{Enc}, A}^{\text{GSD}-(n, b)}$  called the generalized selective decryption (GSD) game, parameterized by an encryption scheme  $\text{Enc}$ ,<sup>8</sup> an integer  $n$  and a bit  $b$ . It is played by the adversary  $A$  and the challenger  $B$ . First  $B$  samples  $n$  keys  $k_1, k_2, \dots, k_n$  uniformly at random from  $\{0, 1\}^\lambda$ .  $A$  can make three types of queries during the game:

- **encrypt:** A query of the form  $\text{encrypt}(i, j)$  is answered with  $c \leftarrow \text{Enc}_{k_i}(k_j)$ .
- **corrupt:** A query of the form  $\text{corrupt}(i)$  is answered with  $k_i$ .
- **challenge:** The response to  $\text{challenge}(i)$  depends on the bit  $b$ : if  $b = 0$ , the answer is  $k_i$ ; if  $b = 1$ , the answer is a random value  $r_i \in \{0, 1\}^\lambda$ .

<sup>7</sup> For this notion to be satisfied,  $\text{Enc}$  must be probabilistic. In this paper one may also consider deterministic encryption, in which case the security definition must explicitly require that the challenge messages are fresh in the sense that  $D$  has not asked for encryptions of them already.

<sup>8</sup> We will never actually use the decryption algorithm  $\text{Dec}$  in the game, and thus will not mention it explicitly.

A can make multiple queries of each type, adaptively and in any order. It can also make several challenge queries at any point in the in the game. Allowing multiple challenge queries models the fact that the respective keys are jointly pseudorandom (as opposed to individual keys being pseudorandom by themselves). Allowing to interleave challenges with other queries models that they remain pseudorandom even after corrupting more keys or seeing further ciphertexts.

We can think of the  $n$  keys that B creates as  $n$  vertices, labeled  $1, 2, \dots, n$ , in a graph. In the beginning of the game there are no edges, but every time A queries  $\text{encrypt}(i, j)$ , we add the edge  $i \rightarrow j$  to the graph. When A queries  $\text{corrupt}(i)$  for some  $i \in [n]$ , we mark  $i$  as a corrupt vertex; when A queries  $\text{challenge}(i)$ , we mark it as a challenge vertex. For an adversary A we call this graph the *key graph*, denoted  $G(A)$  and we write  $V^{\text{corr}}(A)$  and  $V^{\text{chal}}(A)$  for the sets of corrupt and challenge nodes, respectively. (Note that  $G(A)$  is a random variable depending on the randomness used by A and its challenger.)

**Legitimate Adversaries.** Consider an adversary that corrupts a node  $i$  in  $G(A)$  and queries  $\text{challenge}(j)$  for some  $j$  which is reachable from  $i$ . Then A can successively decrypt the keys on the path from  $i$  to  $j$ , in particular  $k_j$ , and thus deduce the bit  $b$ . We only consider non-trivial breaks and require that no challenge node is reachable from a corrupt node in  $G(A)$ .

Two more restrictions must be imposed on  $G(A)$  if we only want to assume that Enc satisfies IND-CPA. First, we do not allow key cycles, that is, queries yielding

$$\text{Enc}_{k_{i_1}}(k_{i_2}), \text{Enc}_{k_{i_2}}(k_{i_3}), \dots, \text{Enc}_{k_{i_{s-1}}}(k_{i_s}), \text{Enc}_{k_s}(k_{i_1}),$$

as this would require the scheme to satisfy key-dependent-message (a.k.a. circular) security [BRS03, CL01].

Second, IND-CPA security does not imply that keys under which one has seen encryptions of random messages remain pseudorandom.<sup>9</sup> Pseudorandomness of keys (assuming only IND-CPA security of the underlying scheme) can thus only hold if their corresponding node does not have any outgoing edges. We thus require that all challenge nodes in the key graph are sinks (i.e., their out-degree is 0). The requirements (as formalized also in [Pan07]) are summarized in the following.

**Definition 3.** *An adversary A is legitimate if in any execution of A in the GSD game the values of  $G(A)$ ,  $V^{\text{corr}}(A)$  and  $V^{\text{chal}}(A)$  are such that:*

- For all  $i \in V^{\text{corr}}(A)$  and  $j \in V^{\text{chal}}(A)$ :  $j$  is unreachable from  $i$  in  $G(A)$ .
- $G(A)$  is a directed acyclic graph (DAG) and every node in  $V^{\text{chal}}(A)$  is a sink.

<sup>9</sup> Consider any IND-CPA-secure scheme (Enc, Dec) and define a new scheme as follows: keys are doubled in length and encryption under  $k = k_1 || k_2$  is defined as  $\text{Enc}_k(m) = \text{Enc}_{k_1}(m) || k_2$ . This scheme is still IND-CPA, but given a ciphertext  $C = \text{Enc}_k(m)$  one can easily distinguish  $k$  from a random value even if  $m$  is random and unknown.

Let  $n \in \mathbb{N}^+$  and  $\mathcal{G}$  be a class of DAGs with  $n$  vertices. We say that a legitimate adversary  $A$  is a  $\mathcal{G}$ -adversary if in any execution the key graph belongs to  $\mathcal{G}$ , i.e.,  $G(A) \in \mathcal{G}$ .

**Definition 4.** Let  $t \in \mathbb{N}^+$ ,  $0 < \epsilon < 1$ . An encryption scheme  $\text{Enc}$  is called  $(n, t, \epsilon, \mathcal{G})$ -GSD secure if for every  $\mathcal{G}$ -adversary  $A$  running in time  $t$ , we have

$$\left| Pr[\mathbf{Exp}_{\text{Enc}, A}^{\text{GSD}-(n,1)} = 1] - Pr[\mathbf{Exp}_{\text{Enc}, A}^{\text{GSD}-(n,0)} = 1] \right| \leq \epsilon.$$

**Assuming One Challenge Query is Enough.** Although the definition of GSD allows the adversary to make any number of corruption queries, Panjwani [Pan07] observes that by a standard hybrid argument one can turn any adversary with advantage  $\epsilon$  (which makes at most  $q \leq n$  challenge queries) into an adversary that makes only one challenge query, but still has advantage at least  $\epsilon/q$ . From now on we therefore only consider adversaries that make exactly one challenge query (keeping in mind that we have to pay an extra factor  $n$  in the final distinguishing advantage for statements about general adversaries).

## 4 Single Source

In this section we will analyze the GSD game for key graphs in which the challenge node is only reachable from one source node. That is, for some  $q \leq n$  there is a path  $p_1 \rightarrow p_2 \rightarrow \dots \rightarrow p_q$  where  $p_1$  has in-degree 0, all nodes  $p_i$ ,  $2 \leq i \leq q$  have in-degree 1 (but arbitrary out-degree) and the (single) challenge query is  $\text{challenge}(p_q)$  (recall that the challenge has out-degree 0). Let  $\mathcal{G}_1$  be the set of all such graphs, and  $\mathcal{G}_1^\ell \subseteq \mathcal{G}_1$  be the subset where this path has length at most  $\ell$ .

**Theorem 1 (GSD on Trees with One Path to Challenge).** Let  $t \in \mathbb{N}$ ,  $0 < \epsilon < 1$  and  $\mathcal{G}_1$  be the class of key graphs just defined. If an encryption scheme is  $(t, \epsilon)$ -IND-CPA secure then it is also  $(n, t', \epsilon', \mathcal{G}_1)$ -GSD secure for

$$\epsilon' = \epsilon \cdot n (3n)^{\lceil \log n \rceil} \quad \text{and} \quad t' = t - Q_{\text{Adv}} T_{\text{Enc}} - \tilde{O}(Q_{\text{Adv}}),$$

where  $T_{\text{Enc}}$  denotes the time required to encrypt a key, and  $Q_{\text{Adv}}$  denotes an upper bound on the number of queries made by the adversary.<sup>10</sup> More generally, if we replace  $\mathcal{G}_1$  with  $\mathcal{G}_1^\ell$ , we get

$$\epsilon' = \epsilon \cdot n (3n)^{\lceil \log \ell \rceil} \quad \text{and} \quad t' = t - Q_{\text{Adv}} T_{\text{Enc}} - \tilde{O}(Q_{\text{Adv}}).$$

**GSD on Single-Source Graphs.** For  $b \in \{0, 1\}$ , we consider the GSD game  $\mathbf{Exp}_{\text{Enc}}^{\text{GSD}-(n,b)}$  on  $\mathcal{G}_1$  between  $B$  and an adversary  $A$ . Challenger  $B$  first samples

<sup>10</sup> If  $\text{Enc}$  is deterministic then w.l.o.g. we can assume  $Q_{\text{Adv}} \leq n^2$  as there are at most  $n(n-1)/2$  possible encryption queries (plus  $\leq n$  corruption and challenge queries). If  $\text{Enc}$  is probabilistic then  $A$  is allowed any number of encryption queries.

$n$  random keys  $k_1, k_2, \dots, k_n$  and we assume that already at this point  $B$  samples fake keys  $r_1, \dots, r_n$ . On all  $\text{encrypt}(i, j)$  queries  $B$  returns real responses  $\text{Enc}_{k_i}(k_j)$ . If  $b = 0$ , the response to  $\text{challenge}(z)$  is  $k_z$ ; if  $b = 1$ , the response is  $r_z$ .

We require that the key graph is in  $\mathcal{G}_1$ , that is the connected component of the key graph which contains the challenge  $z$  has a path  $p_1 \rightarrow p_2 \rightarrow \dots \rightarrow p_q = z$  with  $p_1$  having in-degree 0, all other  $p_i$  having in-degree 1 and  $p_q = z$  having out-degree 0 (this means  $A$  made queries  $\text{encrypt}(p_{i-1}, p_i)$ , but no queries  $\text{encrypt}(x, p_i)$  for  $x \neq p_{i-1}$ ).

Eventually,  $A$  outputs a bit  $b' \in \{0, 1\}$ , which is also the output of the game. If the encryption scheme  $\text{Enc}$  is not  $(t', \epsilon', \mathcal{G}_1)$ -GSD secure then there exists a  $\mathcal{G}_1$ -adversary  $A$  running in time  $t'$  such that

$$\left| \Pr[\text{Exp}_{\text{Enc}, A}^{\text{GSD}-(n,0)} = 1] - \Pr[\text{Exp}_{\text{Enc}, A}^{\text{GSD}-(n,1)} = 1] \right| > \epsilon'. \tag{1}$$

**Our Goal.** Suppose we knew that our GSD adversary  $A$  wants to be challenged on a fixed node  $z^*$  and that it will make a query  $\text{encrypt}(y, z^*)$  for some  $y$  which it will not use in any other query. Then we could use  $A$  directly to construct a distinguisher  $D$  as in Definition 2:  $D$  sets up all keys  $k_x, x \in [n]$ , samples a value  $r_{z^*}$  and runs  $A$ , answering  $A$ 's queries using its keys; except when  $\text{encrypt}(y, z^*)$  is queried for any  $y \in [q]$ ,  $D$  queries its own challenger on  $(k_{z^*}, r_{z^*})$  and forwards the answer to  $A$ . Moreover,  $\text{challenge}(z^*)$  is answered with  $k_{z^*}$ . If  $D$ 's challenger  $C$  chose  $b = 0$ , this perfectly simulates the real game for  $A$ . If  $b = 1$  then  $A$  gets an encryption of  $r_{z^*}$  and the challenge query is answered with  $k_{z^*}$ , although in the random GSD game  $A$  expects an encryption of  $k_{z^*}$  and  $\text{challenge}(z^*)$  to be answered with  $r_{z^*}$ . However, these two games are distributed identically, since both  $k_{z^*}$  and  $r_{z^*}$  are uniformly random values that do not occur anywhere else in the game. Thus  $D$  simulates the real game when  $b = 0$  and the random game when  $b = 1$ . Note that  $D$  implicitly set  $k_y$  to the key that  $C$  chose, but that's fine, since we assumed that  $k_y$  is not used anywhere else in the game and thus not needed by  $D$  for the simulation.

Finally, suppose that, in addition to the challenge  $z^*$ , we knew  $y^*$  for which  $A$  will query  $\text{encrypt}(y^*, z^*)$ . Then we could also allow  $A$  to issue queries of the form  $\text{encrypt}(y^*, x)$ , for  $x$  other than  $z^*$ .  $D$  could easily simulate any such query by querying  $k_x$  to its encryption oracle.

Unfortunately, general GSD adversaries can decide adaptively on which node they want to be challenged, and worse, they can make queries  $\text{encrypt}(x, y)$ , where  $y$  is a key that encrypts the challenge.

We will construct a series of hybrids where any two consecutive games **Game** and **Game'** are such that from a distinguisher  $A$  for them, we can construct an adversary  $D$  against the encryption scheme with the same advantage. For this, the two games should only differ in the response of one encryption query on the path to the challenge, say  $\text{encrypt}(y, z)$ , which is responded to with a real ciphertext  $\text{Enc}_{k_y}(k_z)$  in **Game** and with a fake ciphertext  $\text{Enc}_{k_y}(r_z)$  in **Game'**.

Moreover, the key  $k_y$  must not be encrypted anywhere else in the game, as our distinguisher  $D$  will implicitly set  $k_y$  to be the key of its IND-CPA challenger  $C$ .

Thus, in **Game** and **Game'** all queries  $\text{encrypt}(x, y)$ , for any  $x$ , are responded to with a fake ciphertext  $\text{Enc}_{k_x}(r_y)$ . Summing up, we need the two games to have the following properties for some  $y$ :

- Property 1. **Game** and **Game'** are identical except for the response to one query  $\text{encrypt}(y, z)$ , which is replied to with a real ciphertext in **Game** and a fake one in **Game'**.
- Property 2. Queries  $\text{encrypt}(x, y)$  are replied to with a fake response in both games.

If we knew the entire key graph  $G(A)$  before answering  $A$ 's queries then we could define a series of  $2q - 1$  games as in Fig. 1 where we consecutively replace edges from the source to the challenge by fake nodes and then go back replacing fake edges with real ones starting with  $p_{q-2} \rightarrow p_{q-1}$ . Any two consecutive games in such a sequence would satisfy the two properties, so we could use them to break IND-CPA.

The problem is that in general the probability of guessing the connected component containing the challenge is exponentially small in  $n$  and consequently from a GSD adversary's advantage  $\epsilon'$  we will obtain a distinguisher  $D$  with advantage  $\epsilon = \epsilon'/O(n!)$ . To avoid an exponential loss, we thus must avoid guessing the entire component at once.

**The First Step.** Our first step is to define two new games  $\text{Game}_0^{\{q\}}$  and  $\text{Game}_{\{q\}}$ , which are modifications of  $\text{Exp}^{\text{GSD}-0}$  and  $\text{Exp}^{\text{GSD}-1}$ , respectively. Both new games have an extra step at the beginning of the game:  $B$  guesses which key is going to be the challenge key and at the end of the game only if its guess was correct, the output of the game is  $A$ 's output and otherwise it is 0. Clearly  $B$ 's guess is correct with probability  $1/n$ . Aside from this guessing step,  $\text{Game}_0^{\{q\}}$  is identical to  $\text{Exp}^{\text{GSD}-0}$ ; all responses are real. We therefore have  $\Pr[\text{Game}_0^{\{q\}} = 1] = 1/n \cdot \Pr[\text{Exp}^{\text{GSD}-0} = 1]$ .

Analogously, we define an auxiliary game,  $\text{Game}_1^{\{q\}}$ , which is identical to  $\text{Exp}^{\text{GSD}-1}$ , except for the guessing step. Again we have  $\Pr[\text{Game}_1^{\{q\}} = 1] = 1/n \cdot \Pr[\text{Exp}^{\text{GSD}-1} = 1]$ . We then define  $\text{Game}_{\{q\}}$  exactly as  $\text{Game}_1^{\{q\}}$ , except for a syntactical change: Let  $z$  be the guessed value for the challenge node. Then any query  $\text{encrypt}(x, z)$  is replied to with  $\text{Enc}_{k_x}(r_z)$ , that is, an encryption of the *fake* key  $r_z$ . (Note that this game can be simulated, since we “know”  $z$  when guessing correctly.) On the other hand, the query  $\text{challenge}(z)$  is answered with  $k_z$  (rather than  $r_z$  in  $\text{Exp}^{\text{GSD}-1}$ ). Since the difference between  $\text{Game}_1^{\{q\}}$  and  $\text{Game}_{\{q\}}$  is that we have replaced all occurrences of  $k_z$  by  $r_z$  and all occurrences of  $r_z$  by  $k_z$ , which are distributed identically (thus we've merely swapped the names of  $k_z$  and  $r_z$ ), we have  $\Pr[\text{Game}_{\{q\}} = 1] = \Pr[\text{Game}_1^{\{q\}} = 1] = 1/n \cdot \Pr[\text{Exp}^{\text{GSD}-1} = 1]$ .

Together with Eq. (1), we have thus

$$\begin{aligned} & |\Pr[\text{Game}_0^{\{q\}} = 1] - \Pr[\text{Game}_{\{q\}} = 1]| \\ &= 1/n \cdot |\Pr[\text{Exp}^{\text{GSD}-0} = 1] - \Pr[\text{Exp}^{\text{GSD}-1} = 1]| > 1/n \cdot \epsilon'. \end{aligned}$$

We continue to use the notational convention that for sets  $I \subseteq P \subseteq [n]$ , the game  $\text{Game}_I^P$  is derived from the real game by additionally guessing the nodes corresponding to  $P$  and answering encryptions of the nodes in  $I$  with fake keys. This is made formal in Fig. 4 below.

**The Second Step.** Assume  $q$  is a power of 2 and consider  $\text{Game}_0^{\{q/2, q\}}$ , which is identical to  $\text{Game}_0^{\{q\}}$ , except that in addition to the challenge node,  $B$  also guesses which node  $x \in [n]$  is going to be the node in the middle of the path to the challenge, i.e.  $p_{q/2} = x$ . The output of  $\text{Game}_0^{\{q/2, q\}}$  is  $A$ 's output if the guess was correct and 0 otherwise. Since  $B$  guesses correctly with probability  $1/n$ , we have

$$\Pr[\text{Game}_0^{\{q/2, q\}} = 1] = 1/n \cdot \Pr[\text{Game}_0^{\{q/2\}} = 1].$$

By guessing the middle node, we can assume the middle node is *known* and this will enable us to define a hybrid game,  $\text{Game}_{\{q/2\}}^{\{q/2, q\}}$ , in which the query for the encryption of  $k_{p_{q/2}}$  is responded to with a fake answer. In addition, we consider games  $\text{Game}_{\{q\}}^{\{q/2, q\}}$  and  $\text{Game}_{\{q/2, q\}}^{\{q/2, q\}}$  which are similarly defined by making the same changes to game  $\text{Game}_{\{q\}}^{\{q\}}$ , i.e. guessing the middle node and replying to the encryption query of the guessed key with a fake and a real ciphertext respectively. Again, we have  $\Pr[\text{Game}_{\{q\}}^{\{q/2, q\}} = 1] = 1/n \cdot \Pr[\text{Game}_{\{q\}}^{\{q\}} = 1]$ . Therefore  $(t', \epsilon'/n)$ -distinguishability of  $\text{Game}_0^{\{q\}}$  and  $\text{Game}_{\{q\}}^{\{q\}}$  implies that  $\text{Game}_0^{\{q/2, q\}}$  and  $\text{Game}_{\{q\}}^{\{q/2, q\}}$  are  $(t', \epsilon'/n^2)$ -distinguishable, i.e.  $\Delta_t(\text{Game}_0^{\{q/2, q\}}, \text{Game}_{\{q\}}^{\{q/2, q\}}) > \epsilon'/n^2$ , and therefore by the triangle inequality

$$\begin{aligned} \Delta_t\left(\text{Game}_0^{\{q/2, q\}}, \text{Game}_{\{q/2\}}^{\{q/2, q\}}\right) + \Delta_t\left(\text{Game}_{\{q/2\}}^{\{q/2, q\}}, \text{Game}_{\{q/2, q\}}^{\{q/2, q\}}\right) \\ + \Delta_t\left(\text{Game}_{\{q/2, q\}}^{\{q/2, q\}}, \text{Game}_{\{q\}}^{\{q/2, q\}}\right) \geq \Delta_t\left(\text{Game}_0^{\{q/2, q\}}, \text{Game}_{\{q\}}^{\{q/2, q\}}\right) \\ > 1/n^2 \cdot \epsilon'. \end{aligned} \tag{2}$$

By Eq. (2), at least one of the pairs of games on the left-hand side must be  $(t', \epsilon'/3n^2)$ -distinguishable. The two games of every pair differ in exactly one point, as determined by the subscript of each game. For instance, the difference between the last pair  $\text{Game}_{\{q/2, q\}}^{\{q/2, q\}}$  and  $\text{Game}_{\{q\}}^{\{q/2, q\}}$  is the encryption of node  $q/2$ .

Recall that our goal is to construct a pair of hybrids where the differing query  $\text{encrypt}(y, z)$  is such that all queries  $\text{encrypt}(x, y)$  are replied to with  $\text{Enc}_{k_x}(r_y)$ , as formalized as Property 2. Games  $\text{Game}_0^{\{q\}}$  and  $\text{Game}_{\{q\}}^{\{q\}}$  differed in the last query on the path and the only key above it that is not encrypted anywhere is the start of the path. What we have achieved with our games above is to *halve* that distance: the first pair,  $(\text{Game}_0^{\{q/2, q\}}, \text{Game}_{\{q/2, n\}}^{\{q/2, n\}})$ , and the last pair,  $(\text{Game}_{\{q/2, q\}}^{\{q/2, q\}}, \text{Game}_{\{q\}}^{\{q/2, q\}})$ , differ in a node that is only half way down the path; and the middle pair,  $(\text{Game}_{\{q/2\}}^{\{q/2, q\}}, \text{Game}_{\{q/2, q\}}^{\{q/2, q\}})$ , differ in the last node, but half way up the path there is a key, namely  $k_{q/2}$ , which is not encrypted anywhere, as all queries  $\text{encrypt}(x, q/2)$  are answered with  $\text{Enc}_{k_x}(r_{q/2})$ .

**The Remaining Steps.** For any of the three pairs that is  $(t', \epsilon'/3n^2)$ -distinguishable (and by Eq. (2) there must exist one), we can repeat the same process

$\text{Game}_I^P$ , with  $I \subseteq P \subseteq [n]$  is defined as follows:

- For every  $i \in P$ ,  $B$  chooses  $v_i \leftarrow [n]$ , which is  $B$ 's guess for the node at position  $i$  in the final path.
- $B$  chooses  $2n$  keys  $k_1, r_1, k_2, r_2, \dots, k_n, r_n \leftarrow \{0, 1\}^\lambda$  and runs  $A$ .
- Whenever  $A$  makes a query  $\text{encrypt}(x, y)$ ,  $B$  does the following: If  $y = v_i$  for some  $i \in I$  then reply with  $\text{Enc}_{k_x}(r_{v_i})$ ; otherwise reply with  $\text{Enc}_{k_x}(k_y)$ .
- When  $A$  makes the query  $\text{challenge}(z)$ , return  $k_z$ .
- Let  $b' \in \{0, 1\}$  be  $A$ 's output. At the end of the game, consider the longest path  $p_0 \rightarrow p_1 \rightarrow \dots \rightarrow p_q$  in  $G(A)$ , with  $p_q$  being the argument of  $A$ 's challenge query. If for all  $i \in P$ :  $v_i = p_i$  then  $B$  returns  $b'$ ; otherwise,  $B$  returns 0.

**Fig. 4.** Definition of  $\text{Game}_I^P$  for the single-source case.

on the half of the path which ends with the query that is different in the two games. For example, assume this holds for the last pair, that is

$$\Delta_t \left( \text{Game}_{\{q/2, q\}}^{\{q/2, q\}}, \text{Game}_{\{q\}}^{\{q/2, q\}} \right) > \frac{\epsilon'}{3n^2}. \tag{3}$$

We repeat the process of guessing the middle node between the differing node and the random node above (in this case the root of the path), which is thus node  $q/4$ , and obtain a new pair which satisfies

$$\Delta_t \left( \text{Game}_{\{q/2, q\}}^{\{q/4, q/2, q\}}, \text{Game}_{\{q\}}^{\{q/4, q/2, q\}} \right) > \frac{\epsilon'}{3n^3}, \tag{4}$$

by Eq. (3) and the fact that the guess is correctly with probability  $1/n$ . We can now define two intermediate games

$$\text{Game}_{\{q/4, q/2, q\}}^{\{q/4, q/2, q\}} \quad \text{and} \quad \text{Game}_{\{q/4, q\}}^{\{q/4, q/2, q\}} \tag{5}$$

where we replaced the encryption of  $k_{p_{q/4}}$  by one of  $r_{p_{q/4}}$ . As in Eq. (2), we can again define a sequence of games by putting the games in Eq. (5) between the ones in Eq. (4) and argue that by Eq. (4), two consecutive hybrids must be  $(t', \epsilon'/(3^2 n^3))$ -distinguishable. What we have gained is that any pair in this sequence differs by exactly one edge and the closest fake answer above is only a fourth of the path length away.

Repeating these two steps a maximum number of  $\lceil \log q \rceil$  times, we arrive at two consecutive games, where the distance from the differing node to the closest “fake” node above is 1. We have thus found two games that satisfy Properties 1 and 2, meaning we can use a distinguisher  $A$  to construct an adversary  $D$  against the encryption scheme.

Since a path has at most  $n$  nodes, after at most  $\log n$  steps we end up with two games that are  $(t', \epsilon'/n(3n)^{\lceil \log n \rceil})$ -distinguishable and which can be used to break the encryption scheme. If the adversary is restricted to paths of length  $\ell$  (i.e., graphs in  $\mathcal{G}_1^\ell$ ), this improves to  $(t', \epsilon'/n(3n)^{\lceil \log \ell \rceil})$ .

**Proof of Theorem 1.** We formalize our method to give a proof of the theorem. In Fig. 4 we describe game  $\text{Game}_I^P$ , which is defined by the nodes on the path that are guessed (represented by the set  $P$ ) and the nodes where an encryption of a key is replaced with an encryption of a value  $r$  (represented by  $I \subseteq P$ ).

**Lemma 1.** *Let  $I \subseteq P \subseteq [n]$  and  $z \in P \setminus I$ . Also let  $y$  be the largest number in  $I$  such that  $y < z$ , and  $y = 0$  if  $z$  is smaller than all elements in  $I$ . If  $\text{Game}_I^P$  and  $\text{Game}_{I \cup \{z\}}^P$  are  $(t, \epsilon)$ -distinguishable then the following holds.*

- If  $z = y + 1$  then  $\text{Enc}$  is not  $(t + Q_{\text{Adv}}T_{\text{Enc}} + \tilde{O}(Q_{\text{Adv}}), \epsilon)$ -IND-CPA-secure.
- If  $z > y + 1$ , define  $z' = y + \lfloor (z - y)/2 \rfloor$ ,  $P' = P \cup \{z'\}$  and

$$I_1 = I, \quad I_2 = I \cup \{z'\}, \quad I_3 = I \cup \{z', z\}, \quad I_4 = I \cup \{z\}.$$

Then for some  $i \in \{1, 2, 3\}$ , games  $\text{Game}_{I_i}^{P'}$  and  $\text{Game}_{I_{i+1}}^{P'}$  are  $(t, \epsilon/3n)$ -distinguishable.

The proof of this lemma can be found in the full version. Applying Lemma 1 repeatedly  $\lceil \log n \rceil$  times (or  $\lceil \log \ell \rceil$  if we know an upper bound on the path length  $\ell$ ), we obtain the proof of Theorem 1.

## 5 General Trees

For a node  $v$  in a directed graph  $G$  let  $T_v$  denote the subgraph of  $G$  we get when only keeping the edges on paths that lead to  $v$ . In this section we prove bounds for GSD if the underlying key graph is a tree. Concretely, let  $\mathcal{G}_\tau$  be the class of key graphs that contain one designated “challenge node”  $z$  and where the graph  $T_z$  is a tree (when ignoring edge directions).

To give more fine-grained bounds we define a subset  $\mathcal{G}_\tau^{s,d,\ell} \subseteq \mathcal{G}_\tau$  as follows. For  $G \in \mathcal{G}_\tau$ , let  $z$  be the challenge node and  $T_z$  as above. Then  $G \in \mathcal{G}_\tau^{s,d,\ell}$  if the challenge node has at most  $s$  sources (i.e., there are at most  $s$  nodes  $u$  of in-degree 0 s.t. there is a directed path from  $u$  to  $z$ ), every node in  $T_z$  has in-degree at most  $d$  and the longest path in  $T_z$  has length at most  $\ell$ . Note that as  $d < n, s < n$  and  $\ell \leq n$  any  $G \in \mathcal{G}_\tau$  with  $n$  nodes is trivially in  $\mathcal{G}_\tau^{n-1,n-1,n}$ .

**Theorem 2 (Security of GSD on Trees).** *Let  $n, t \in \mathbb{N}$ ,  $0 < \epsilon < 1$  and  $\mathcal{G}_\tau$  be the class of key graphs just defined. If an encryption scheme is  $(t, \epsilon)$ -IND-CPA secure then it is also  $(n, t', \epsilon', \mathcal{G}_\tau)$ -GSD secure for*

$$\epsilon' = \epsilon \cdot n^2(6n^3)^{\lceil \log n \rceil} \leq \epsilon \cdot n^{3\lceil \log n \rceil + 5} \quad \text{and} \quad t' = t - Q_{\text{Adv}}T_{\text{Enc}} - \tilde{O}(Q_{\text{Adv}})$$

(with  $Q_{\text{Adv}}, T_{\text{Enc}}$  as in Theorem 1). If we replace  $\mathcal{G}_\tau$  with  $\mathcal{G}_\tau^{s,d,\ell}$  then

$$\epsilon' = \epsilon \cdot dn((2d + 1)n)^{\lceil \log s \rceil} (3n)^{\lceil \log \ell \rceil} \quad \text{and} \quad t' = t - Q_{\text{Adv}}T_{\text{Enc}} - \tilde{O}(Q_{\text{Adv}}).$$

For space reasons, the proof of this theorem is moved to the full version.



## 6 Conclusions and Open Problems

We showed a quasipolynomial reduction of the GSD game on trees to the security of the underlying symmetric encryption scheme. As already discussed in the introduction, it is an interesting open problem to extend our reduction to general (directed, acyclic) graphs or to understand why this is not possible. This is the second result using the “nested hybrids” technique (after its introduction in [FKPR14] to prove the security of constrained PRFs), and given that it found applications for two seemingly unrelated problems, we believe that there will be further applications in the future.

One candidate is the problem of proving security under selective opening attacks [DNRS99, FHKW10, BHY09], where one wants to prove security when correlated messages are encrypted under different keys. Here, the adversary may adaptively chose to corrupt some keys after seeing all ciphertexts, and one requires that the messages in the unopened ciphertexts are indistinguishable from random messages (sampled so they are consistent with the already opened ciphertexts). This problem is notoriously hard, and no reduction avoiding complexity leveraging to IND-CPA security of the underlying scheme is known.

**Acknowledgements.** We would like to thank the anonymous reviewers for their valuable comments.

## References

- [ABBC10] Acar, T., Belenkiy, M., Bellare, M., Cash, D.: Cryptographic agility and its relation to circular encryption. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 403–422. Springer, Heidelberg (2010)
- [BGI14] Boyle, E., Goldwasser, S., Ivan, I.: Functional signatures and pseudorandom functions. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 501–519. Springer, Heidelberg (2014)
- [BH93] 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)
- [BHY09] Bellare, M., Hofheinz, D., Yilek, S.: Possibility and impossibility results for encryption and commitment secure under selective opening. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 1–35. Springer, Heidelberg (2009)
- [BRS03] Black, J., Rogaway, P., Shrimpton, T.: Encryption-scheme security in the presence of key-dependent messages. In: Nyberg, K., Heys, H.M. (eds.) SAC 2002. LNCS, vol. 2595. Springer, Heidelberg (2003)
- [BW13] Boneh, D., Waters, B.: Constrained pseudorandom functions and their applications. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part II. LNCS, vol. 8270, pp. 280–300. Springer, Heidelberg (2013)
- [CFGN96] Canetti, R., Feige, U., Goldreich, O., Naor, M.: Adaptively secure multi-party computation. In: 28th ACM STOC, pp. 639–648. ACM Press, May 1996

- [CL01] Camenisch, J.L., Lysyanskaya, A.: An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, p. 93. Springer, Heidelberg (2001)
- [DNRS99] Dwork, C., Naor, M., Reingold, O., Stockmeyer, L.J.: Magic functions. In: 40th FOCS, pp. 523–534. IEEE Computer Society Press, October 1999
- [FHKW10] Fehr, S., Hofheinz, D., Kiltz, E., Wee, H.: Encryption schemes secure against chosen-ciphertext selective opening attacks. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 381–402. Springer, Heidelberg (2010)
- [FKPR14] Fuchsbauer, G., Konstantinov, M., Pietrzak, K., Rao, V.: Adaptive security of constrained PRFs. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014, Part II. LNCS, vol. 8874, pp. 82–101. Springer, Heidelberg (2014)
- [GGM86] Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. *J. ACM* **33**(4), 792–807 (1986)
- [KPTZ13] Kiayia, A., Papadopoulos, S., Triandopoulos, N., Zacharias, T.: Delegatable pseudorandom functions and applications. In: Sadeghi, A.-R., Gligor, V.D., Yung, M. (eds.) ACM CCS 13, pp. 669–684. ACM Press, New York (2013)
- [MP06] Micciancio, D., Panjwani, S.: Corrupting one vs. Corrupting many: the case of broadcast and multicast encryption. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4052, pp. 70–82. Springer, Heidelberg (2006)
- [Pan07] Panjwani, S.: Tackling adaptive corruptions in multicast encryption protocols. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 21–40. Springer, Heidelberg (2007)
- [SS00] Selçuk, A.A., Sidhu, D.P.: Probabilistic methods in multicast key management. In: Information Security, Third International Workshop, ISW 2000, 20–21 December 2000, Wollongong, NSW, Australia, pp. 179–193. Proceedings (2000)
- [WGL00] Wong, C.K., Gouda, M., Lam, S.S.: Secure group communications using key graphs. *IEEE/ACM Trans. Netw.* **8**(1), 16–30 (2000)

# **Hash Functions and Stream Cipher Cryptanalysis**

# Practical Free-Start Collision Attacks on 76-step SHA-1

Pierre Karpman<sup>1,2</sup>, Thomas Peyrin<sup>2</sup>, and Marc Stevens<sup>3</sup>✉

<sup>1</sup> Inria, Villeurbanne, France  
pierre.karpman@inria.fr

<sup>2</sup> Nanyang Technological University, Singapore, Singapore  
thomas.peyrin@ntu.edu.sg

<sup>3</sup> Centrum Wiskunde and Informatica, Amsterdam, The Netherlands  
marc.stevens@cwi.nl

**Abstract.** In this paper we analyze the security of the compression function of **SHA-1** against collision attacks, or equivalently free-start collisions on the hash function. While a lot of work has been dedicated to the analysis of **SHA-1** in the past decade, this is the first time that free-start collisions have been considered for this function. We exploit the additional freedom provided by this model by using a new start-from-the-middle approach in combination with improvements on the cryptanalysis tools that have been developed for **SHA-1** in the recent years. This results in particular in better differential paths than the ones used for hash function collisions so far. Overall, our attack requires about  $2^{50}$  evaluations of the compression function in order to compute a one-block free-start collision for a 76-step reduced version, which is so far the highest number of steps reached for a collision on the **SHA-1** compression function. We have developed an efficient GPU framework for the highly branching code typical of a cryptanalytic collision attack and used it in an optimized implementation of our attack on recent GTX 970 GPUs. We report that a single cheap US\$ 350 GTX 970 is sufficient to find the collision in less than 5 days. This showcases how recent mainstream GPUs seem to be a good platform for expensive and even highly-branching cryptanalysis computations. Finally, our work should be taken as a reminder that cryptanalysis on **SHA-1** continues to improve. This is yet another proof that the industry should quickly move away from using this function.

**Keywords:** **SHA-1** · Hash function · Cryptanalysis · Free-start collision · GPU implementation

## 1 Introduction

Cryptographic hash functions are essential components in countless security systems for very diverse applications. Informally, a hash function  $H$  is a function

---

P. Karpman—Partially supported by the Direction Générale de l’Armement and by the Singapore National Research Foundation Fellowship 2012 (NRF-NRFF2012-06).

T. Peyrin—Supported by the Singapore National Research Foundation Fellowship 2012 (NRF-NRFF2012-06).

that takes an arbitrarily long message  $M$  as input and outputs a fixed-length hash value of size  $n$  bits. One of the main security requirements for a cryptographic hash function is to be collision resistant: it should be hard for an adversary to find two distinct messages  $M, \hat{M}$  leading to the same hash value  $H(M) = H(\hat{M})$  in less than  $2^{\frac{n}{2}}$  calls to  $H$ . Most standardized hash functions are based on the Merkle-Damgård paradigm [6, 27] which iterates a compression function  $h$  that updates a fixed-size internal state (also called chaining value) with fixed-size message blocks. This construction allows a simple and very useful security reduction: if the compression function is collision-resistant, then so is the corresponding hash function. Since the compression function has two inputs, an attacker may use this extra freedom to mount attacks that are not possible on the complete hash function; on the other hand, one loses the ability to chain message blocks. We can then distinguish between two classical attack models: a *free-start* collision is a pair of different message and chaining value  $(c, m), (\hat{c}, \hat{m})$  leading to a collision after applying  $h$ :  $h(c, m) = h(\hat{c}, \hat{m})$ . A *semi-free-start* collision works similarly, with the additional restriction that the chaining values  $c$  and  $\hat{c}$  must be equal. It is important to note that the Merkle-Damgård security reduction assumes that any type of collision for the compression function should be intractable for an attacker, including free-start collisions.

The most famous and probably most used hash function as of today is SHA-1 [29]. This function belongs to the MD-SHA family, that originated with MD4 [34]. Soon after its publication, MD4 was believed to be insufficiently secure [9] and a practical collision was later found [11]. Its improved version, MD5 [35], was widely deployed in countless applications, even though collision attacks on the compression function were quickly identified [10]. The function was completely broken and collisions were found in the groundbreaking work from Wang *et al.* [43]. A more powerful type of collision attack called *chosen-prefix collision attack* against MD5 was later introduced by Stevens *et al.* [40]. Irrefutable proof that hash function collisions indeed form a realistic and significant threat to Internet security was then provided by Stevens *et al.* [41] with their construction of a Rogue Certification Authority that in principle completely undermined HTTPS security. This illustrated further that the industry should move away from weak cryptographic hash functions and should not wait until cryptanalytic advances actually prove to be a direct threat to security. Note that one can use *counter-cryptanalysis* [38] to protect against such digital signature forgeries during the strongly advised migration away from MD5 and SHA-1.

Before the impressive attacks on MD5, the NIST had standardized the hash function SHA-0 [28], designed by the NSA and very similar to MD5. This function was quickly very slightly modified and became SHA-1 [29], with no justification provided. A plausible explanation came from the pioneering work of Chabaud and Joux [4] who found a theoretical collision attack that applies to SHA-0 but not to SHA-1. Many improvements of this attack were subsequently proposed [1] and an explicit collision for SHA-0 was eventually computed [2]. However, even though SHA-0 was practically broken, SHA-1 remained free of attacks until the work of Wang *et al.* [42] in 2005, who gave the very first theoretical collision

attack on SHA-1 with an expected cost equivalent to  $2^{69}$  calls to the compression function. This attack has later been improved several times, the most recent improvement being due to Stevens [39], who gave an attack with estimated cost  $2^{61}$ ; yet no explicit collision has been computed so far. With the attacks on the full SHA-1 remaining impractical, the community focused on computing collisions for reduced versions: 64 steps [8] (with a cost of  $2^{35}$  SHA-1 calls), 70 steps [7] (cost  $2^{44}$  SHA-1), 73 steps [13] (cost  $2^{50.7}$  SHA-1) and the latest advances reached 75 steps [14] (cost  $2^{57.7}$  SHA-1) using extensive GPU computation power. As of today, one is advised to use *e.g.* SHA-2 [30] or the hash functions of the future SHA-3 standard [31] when secure hashing is needed.

In general, two main points are crucial when dealing with a collision search for a member of the MD-SHA family of hash functions (and more generally for almost every hash function): the quality of the differential paths used in the attack and the amount and utilization of the remaining freedom degrees. Regarding SHA-0 or SHA-1, the differential paths were originally built by linearizing the step function and by inserting small perturbations and corresponding corrections to avoid the propagation of any difference. These so-called local collisions [4] fit nicely with the linear message expansion of SHA-0 and SHA-1 and made it easy to generate differential paths and evaluate their quality. However, these linear paths have limitations since not so many different paths can be used as they have to fulfill some constraints (for example no difference may be introduced in the input or the output chaining value). In order to relax some of these constraints, Biham *et al.* [2] proposed to use several successive SHA-1 compression function calls to eventually reach a collision. Then, Wang *et al.* [42] completely removed these constraints by using only two blocks and by allowing some part of the differential paths to behave non-linearly (*i.e.* not according to a linear behavior of the SHA-1 step function). Since the non-linear parts have a much lower differential probability than the linear parts, to minimize the impact on the final complexity they may only be used where freedom degrees are available, that is during the first steps of the compression function. Finding these non-linear parts can in itself be quite hard, and it is remarkable that the first ones were found by hand. Thankfully, to ease the work of the cryptanalysts, generating such non-linear parts can now be done automatically, for instance using the guess-and-determine approach of De Cannière and Rechberger [8], or the meet-in-the-middle approach of Stevens *et al.* [15, 37]. In addition, joint local collision analysis [39] for the linear part made heuristic analyzes unnecessary and allows to generate optimal differential paths.

Once a differential path has been chosen, the remaining crucial part is the use of the available freedom degrees when searching for the collision. Several techniques have been introduced to do so. First, Chabaud and Joux [4] noticed that in general the 15 first steps of the differential path can be satisfied for free since the attacker can fix the first 16 message words independently, and thus fulfill these steps one by one. Then, Biham and Chen [1] introduced the notion of neutral bits, that allows the attacker to save conditions for a few additional steps. The technique is simple: when a candidate following the differential path until

step  $x > 15$  is found, one can amortize the cost for finding this valid candidate by generating many more almost for free. Neutral bits are small modifications in the message that are very likely not to invalidate conditions already fulfilled in the  $x$  first steps. In opposition to neutral bits, the aim of message modifications [42] is not to multiply valid candidates but to correct the wrong ones: the idea is to make a very specific modification in a message word, so that a condition not verified at a later step eventually becomes valid with very good probability, but without interfering with previously satisfied conditions. Finally, one can cite the tunnel technique from Klíma [20] and the auxiliary paths (or boomerangs) from Joux and Peyrin [18], that basically consist in pre-set, but more powerful neutral bits. Which technique to use, and where and how to use it are complex questions for the attacker and the solution usually greatly depends on the specific case that is being analyzed.

**Our Contributions.** In this paper, we study the free-start collision security of SHA-1. We explain why a start-from-the-middle approach can improve the current best collision attacks on the SHA-1 compression function regarding two keys points: the quality of the differential paths generated, but also the amount of freedom degrees and the various ways to use them. Furthermore, we present improvements to derive differential paths optimized for collision attacks using an extension of joint local-collision analysis. All these improvements allow us to derive a one-block free-start collision attack on 76-step SHA-1 for a rather small complexity equivalent to about  $2^{50}$  calls to the primitive. We have fully implemented the attack and give an example of a collision in the full version of the paper [19].

We also describe a GPU framework for a very efficient GPU implementation of our attack. The computation complexity is quite small as a single cheap US\$ 350 GTX 970 can find a collision in less than 5 days, and our cheap US\$ 3000 server with four GTX 970 GPUs can find one in slightly more than one day on average. In comparison, the 75-step collision from [14] was computed on the most powerful supercomputer in Russia at the time, taking 1.5 month on 455 GPUs on average. This demonstrates how recent mainstream GPUs can easily be used to perform big cryptanalysis computations, even for the highly branching code used in cryptanalytic collision attacks. Notably, our approach leads to a very efficient implementation where a single GTX 970 is equivalent to about 140 recent high-clocked Haswell cores, whereas the previous work of Grechnikov and Adinets estimates an Nvidia Fermi GPU to be worth 39 CPU cores [14].

Moreover, we emphasize that we have found the collision that has reached the highest number of SHA-1 compression function steps as of today. Finally, this work serves as a reminder that cryptanalysis on SHA-1 continues to improve and that industry should now quickly move away from using this primitive.

**Outline.** In Sect. 2 we first describe the SHA-1 hash function. In Sect. 3 we describe the start-from-the-middle approach and how it can provide an improvement when looking for (semi)-free-start collisions on a hash function. We then

study the case of 76-step reduced SHA-1 with high-level explanations of the application of the start-from-the-middle approach, the differential paths, and the GPU implementation of the attack in Sect. 4. The reader interested by more low-level details can then refer to Sect. 5. Finally, we summarize our results in Sect. 6.

## 2 The SHA-1 Hash Function

We give here a short description of the SHA-1 hash function and refer to [29] for a more exhaustive treatment. SHA-1 is a 160-bit hash function belonging to the MD-SHA family. Like many hash functions, SHA-1 uses the Merkle-Damgård paradigm [6, 27]: after a padding process, the message is divided into  $k$  blocks of 512 bits each. At every iteration of the compression function  $h$ , a 160-bit chaining value  $cv_i$  is updated using one message block  $m_{i+1}$ , i.e.  $cv_{i+1} = h(cv_i, m_{i+1})$ . The initial value  $IV = cv_0$  is a predefined constant and  $cv_k$  is the output of the hash function.

As for most members of the MD-SHA family, the compression function  $h$  uses a block cipher  $Enc$  in a Davies-Meyer construction:  $cv_{i+1} = Enc(m_{i+1}, cv_i) + cv_i$ , where  $Enc(x, y)$  denotes the encryption of plaintext  $y$  with key  $x$ . The block cipher itself is an 80-step (4 rounds of 20 steps each) generalized Feistel network which internal state is composed of five branches (or internal registers)  $(A_i, B_i, C_i, D_i, E_i)$  of 32-bit each. At each step, a 32-bit extended message word  $W_i$  is used to update the five internal registers:

$$\begin{cases} A_{i+1} = (A_i \lll 5) + f_i(B_i, C_i, D_i) + E_i + K_i + W_i, \\ B_{i+1} = A_i, \\ C_{i+1} = B_i \ggg 2, \\ D_{i+1} = C_i, \\ E_{i+1} = D_i. \end{cases}$$

where  $K_i$  are predetermined constants and  $f_i$  are Boolean functions defined in Table 1. Note that all updated registers but  $A_{i+1}$  are just rotated copies of another register, so one can only consider the register  $A$  at each iteration. Thus, we can simplify the step function as:

$$A_{i+1} = (A_i \lll 5) + f_i(A_{i-1}, A_{i-2} \ggg 2, A_{i-3} \ggg 2) + (A_{i-4} \ggg 2) + K_i + W_i.$$

Finally, the extended message words  $W_i$  are computed from the 512-bit message block, which is split into 16 32-bit words  $M_0, \dots, M_{15}$ . These 16 words are then expanded linearly into the 80 32-bit words  $W_i$  as follows:

$$W_i = \begin{cases} M_i, & \text{for } 0 \leq i \leq 15 \\ (W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16}) \lll 1, & \text{for } 16 \leq i \leq 79 \end{cases}$$

For the sake of completeness, since our attacks compute the internal cipher  $Enc$  both in the forward (encryption) and backward (decryption) directions, we also give below the description of the inverse of the state update function:

$$A_i = (A_{i+5} - W_{i+4} - K_{i+4} - f_{i+4}(A_{i+3}, A_{i+2} \ggg 2, A_{i+1} \ggg 2) - (A_{i+4} \lll 5)) \lll 2$$

and of the message expansion:  $W_i = (W_{i+16} \ggg 1) \oplus W_{i+13} \oplus W_{i+8} \oplus W_{i+2}$ .



**Table 1.** Boolean functions and constants of SHA-1

Round	Step $i$	$f_i(B, C, D)$	$K_i$
1	$0 \leq i < 20$	$f_{IF} = (B \wedge C) \oplus (\bar{B} \wedge D)$	0x5a827999
2	$20 \leq i < 40$	$f_{XOR} = B \oplus C \oplus D$	0x6ed6eba1
3	$40 \leq i < 60$	$f_{MAJ} = (B \wedge C) \oplus (B \wedge D) \oplus (C \wedge D)$	0x8fabcbdc
4	$60 \leq i < 80$	$f_{XOR} = B \oplus C \oplus D$	0xca62c1d6

### 3 A Start-from-the-middle Approach

The first example of an attack starting from the middle of a hash function is due to Dobbertin [11], who used it to compute a collision on MD4. Start-from-the-middle methods are also an efficient approach to obtain distinguishers on hash functions, such as Saarinen’s slide distinguisher on the SHA-1 compression function [36]. Rebound attacks [25] and their improvements [12, 16, 21, 24] can be considered as a start-from-the-middle strategy tailored to the specific case of AES-like primitives. Start-from-the-middle has also been used to improve the analysis of functions based on parallel branches, such as RIPEMD-128 [22]. All these attacks leverage the fact that in some specific scenarios, starting from the middle may lead to a better use of the freedom degrees available.

In general, a start-from-the-middle approach for collision search leads to a free-start or semi-free-start attack. Indeed, since one might not use the freedom degrees in the first steps of the compression function anymore, it is harder to ensure that the initial state (*i.e.* the chaining value) computed from the middle is equal to the specific initial value of the function’s specifications. However, if the starting point is not located too far from the beginning, one may still be able to do so; this is for example the case of the recent attack on Grøstl [26]. In this work, we focus on the search of free-start collisions and consider that the IV can be chosen by the attacker. For SHA-1, this adds 160 bits of freedom that can be set in order to fulfill conditions to follow a differential path.

Furthermore, in the case of SHA-1, one can hope to exploit a start-from-the-middle approach even more to improve previous works in two different ways. Firstly, the set of possible differential paths to consider increases. Secondly, the freedom degrees now operate in two directions: forward and backward.

**More Choice for the Differential Paths.** The linear differential paths (generated from the mask of introduced perturbations, also called *disturbance vector* or DV) used in all existing collision attacks on SHA-1 can be concisely described in two families [23]. Both types have the feature that the complexity of following the path is unevenly distributed along the 80 steps of SHA-1 (this remains true for attacks on reduced versions as well). Furthermore, because a typical attack replaces a portion of the linear differential path by a non-linear part, this latter one defines a series of steps where the complexity of the linear path is basically irrelevant. In the case of a start-from-the-middle attack, one can choose where

to locate this non-linear part, and thereby gains more flexibility in the choice of the linear path to use.

**Two-Way Use of the Freedom Degrees.** In a start-from-the-middle setting, one freely chooses an initial state in the middle of the function instead of necessarily starting from the IV in the beginning. Therefore, the differential path conditions may be related both to *forward* and *backward* computations from the middle state, and the same goes for the freedom available in the first 16 words of the message. Because one now has more possibilities to exploit them, one can hope to make a better use of these freedom degrees. For example, we can imagine two-way neutral bits, that is applying neutral bits in both forward and backward directions. This would potentially allow the attacker to obtain a few free steps not only in the forward direction as in previous works, but also in the backward direction. Of course, one must be careful about the non-independence between the forward and backward computations. Obviously, the same reasoning can apply to other freedom degrees utilization techniques such as message modifications, tunnels or boomerangs.

In the next two sections, we detail how we applied this approach to the search of free-start collisions for SHA-1.

## 4 A High-Level View of the SHA-1 Free-Start Collision Attack

### 4.1 Start-from-the-middle

There is one main choice that must be made when using a start-from-the-middle approach for an attack on SHA-1, that is which consecutive 16 steps are used to apply advanced message modification techniques or neutral bits; in our terminology the offset corresponding to this choice is called the *main block offset*. Any simple change in those 16 steps propagates to all other steps, in particular differences propagate backwards from these 16 steps down to step 0 and thereby affect the input chaining values. Note that for regular attacks on SHA-1, the main block offset must be 0 to ensure that the chaining value is never altered.

For our neutral bits, we found that using a main block offset of 6 was optimal. Therefore neutral bits are applied on the 16 message words  $W_{6..21}$  and a neutral bit in  $W_i$  affects steps  $5, 4, \dots, 0$  backwards and steps  $i, i + 1, \dots$  forwards.

Before we can apply the neutral bits, we first need to compute a partial solution over 16 consecutive steps that can be extended using the neutral bits, which we call *base solution* in our terminology. This base solution is also computed with an offset but it is only one, meaning that it consists of state words  $A_{-3}, \dots, A_{17}$ . We can find such a solution using simple message modification, over the message words  $W_1$  to  $W_{16}$ , in particular we choose an initial solution for  $A_8, \dots, A_{12}$  which we first extend backwards using words  $11, \dots, 1$  and then forwards using words  $12, \dots, 16$ .

**Using Neutral Bits to Improve the Probabilistic Phase.** The offset of the base solution being 1, the state may not follow the differential path anymore starting from  $A_{18}$ , and the attacker needs to test many different base solutions to go as far as  $A_{76}$  and get a complete collision. At first sight, it may therefore seem that we gained only little from this particular use of a start-in-the-middle approach. However, by using a main block offset of 6, there remains freedom that can still be exploited in the message words up to  $W_{21}$ . Although their value cannot be changed entirely (as they were fully determined when computing the base solution), we can still use these words to implement neutral bits.

In our attack, we use 51 neutral bits spread on words  $W_{14}$  to  $W_{21}$ , which are neutral with respect to the computation of state words up to  $A_{18\dots26}$  with good probability. This means that up to the computation of  $A_{26}$ , one can take advantage of solutions up to  $A_{18\dots25}$  (that had to be found probabilistically) to find more solutions up to the same step with only a negligible cost. This considerably reduces the complexity of the attack, and we experimentally observed that about 90 % of the computations were done past  $A_{24}$ , which can thus be considered to be where the actual probabilistic phase starts. This is a very noticeable improvement from the original  $A_{17}$  of the base solution. We give a complete list of these neutral bits in the full version of the paper [19].

There is however one caveat when using neutral bits in such a start-in-the-middle fashion, as one will pay an additional cost in complexity if they interact badly with the base solution when they go through the backward message expansion. In our attack, we chose neutral bits that do so with only a small probability, which can even be lowered to a negligible quantity when running the attack by filtering the base solutions from which to start.

In Fig. 1, we summarize our application of start-from-the-middle to SHA-1 with a graphical representation of the attack layout.

## 4.2 Differential Path Construction Improvements

The full differential path for SHA-1 collision attacks are made of two parts. The most important part is the linear part built from a combination of local collisions as described by the disturbance vector, which almost covers the last 3 rounds of SHA-1 and directly contributes a major factor to the overall complexity. The remaining part, the so-called non-linear-part covering mostly round 1, is constructed to link up prescribed IV differences and the linear part into a full differential path.

Current methods to construct the non-linear part are a guess-and-determine approach due to De Cannière *et al.* [8] and a meet-in-the-middle approach due to Stevens *et al.* [37]. For the latter an implementation has been made public at Project HashClash [15] that we used for this work. For the linear part, the state-of-the-art is Joint Local-Collision Analysis (JLCA) [39] which analyzes the entire set of differential paths over the last 3 rounds conforming to the disturbance vector and which exploits redundancy to make it practical. Using JLCA one can extract a minimal set of conditions (consisting of starting differences (say for

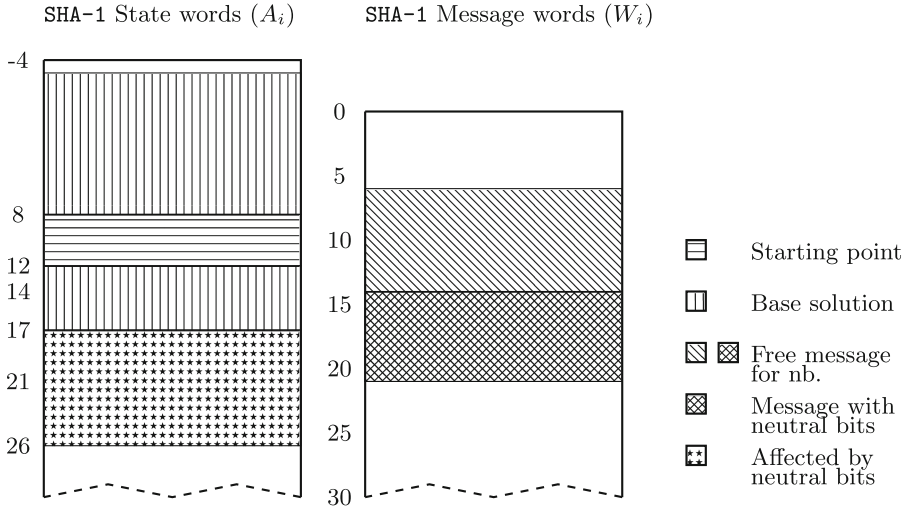


Fig. 1. Illustration of the use of start-from-the-middle for SHA-1.

step 20), message bit-relations and ending differences) that leads to the highest probability. Being of the highest probability implies that the factor contribution of the linear part to the overall complexity is minimal, while a minimal set of conditions maximizes the amount of freedom that can be exploited to speed up the attack.

For our attacks we extended (our own implementation of) JLCA to cover *all* steps and to produce the entire set of sufficient state conditions and message bit-relations as used by collision attacks. In particular, we improved JLCA in the following ways:

1. **Include the Non-linear Part.** Originally JLCA considers the entire set of differential paths that conform to the disturbance vector only over the linear part. This is done by considering sets  $Q_i$  of allowed state differences for each  $A_i$  given the disturbance vector (including carries), see [39]. We extended this by defining sets  $Q_i$  for the non-linear part as the state difference given by a previously constructed differential path of the non-linear part. Here one actually has a few options: only consider exact state difference of the non-linear path or also consider changes in carries and/or signs, as well as include state differences conforming to the disturbance vector. We found that allowing changes in carries and/or signs for the state differences given by the non-linear path made JLCA impractical, yet including state differences conforming to the disturbance vector was practical and had a positive effect on the overall probability of the full differential path.
2. **Do Not Consider Auxiliary Carries Not Used in the Attack.** Originally JLCA considers local collisions with carries as this improves the overall probability, the probability of variants of paths adding up. However, collision attacks employ sufficient conditions for, say, the first 25 steps, where such

auxiliary carries are not used. For these steps JLCA would thus optimize for the wrong model with auxiliary carries. We propose to improve this by not adding up the probability of paths over the first 25 steps, but only to take the maximum probability. We propose to do this by redefining the cumulative probabilities  $p(\mathcal{P},w)$  from [39, Sect. 4.6] to:

$$p(\mathcal{P},w) = \max_{\widehat{\mathcal{P}}_{[0,25]} \in \mathcal{D}_{[0,25]}} \sum_{\substack{\mathcal{P}' \in \mathcal{D}_{[0,t_e]} \\ \mathcal{P}'|_{[0,25]} = \widehat{\mathcal{P}}_{[0,25]} \\ \mathcal{P} = \text{Reduce}(\mathcal{P}'), w = w(\mathcal{P}')}} \Pr[\mathcal{P}' - \mathcal{P}].$$

In our JLCA implementation this can be simply implemented by replacing the addition of two probabilities by taking their maximum conditional on the current SHA-1 step.

3. **Determine Sufficient Conditions.** Originally JLCA only outputted starting differences, ending differences, message bit-relations and the optimal success probability. We propose to extend JLCA to reconstruct the set of differential paths over steps, say,  $[0, 25]$ , and to determine minimal sets of sufficient conditions and message bit-relations. This can be made possible by keeping the intermediate sets of reduced differential paths  $\mathcal{R}_{[t_b, t_e]}$  which were constructed backwards starting at a zero-difference intermediate state of SHA-1. Then one can iteratively construct sets  $\mathcal{O}_{[0, i]}$  of optimal differential paths over steps  $0, \dots, i - 1$ , *i.e.*, differential paths compatible with some combination of the optimal starting differences, ending differences and message bit-relations such that the optimal success probability can be achieved. One starts with the set  $\mathcal{O}_{[0, 0]}$  determined by the optimal starting differences. Given  $\mathcal{O}_{[0, i]}$  one can compute  $\mathcal{O}_{[0, i+1]}$  by considering all possible extensions of every differential path in  $\mathcal{O}_{[0, i]}$  with step  $i$  (under the predefined constraints, *i.e.*  $\Delta Q_j \in \mathcal{Q}_j$ ,  $\delta W_i \in \mathcal{W} - i$ , see [39]). From all those paths, one only stores in  $\mathcal{O}_{[0, i+1]}$  those that can be complemented by a reduced differential path over steps  $i+1, \dots, t_e$  from  $\mathcal{R}_{[i+1, t_e]}$  such that the optimal success probability is achieved over steps  $0, \dots, t_e$ .

Now given, say,  $\mathcal{O}_{[0, 26]}$ , we can select any path and determine its conditions necessary and sufficient for steps  $0, \dots, 25$  and the optimal set of message bit-relations that goes with it. Although we use only one path, having the entire set  $\mathcal{O}_{[0, 26]}$  opens even more avenues for future work. For instance, one might consider an entire subclass of  $2^k$  differential paths from  $\mathcal{O}_{[0, 26]}$  that can be described by state conditions linear in message bits and a set of (linear) message bit-relations. This would provide  $k$  bits more in degrees of freedom that can be exploited by speed up techniques.

In short, we propose several extensions to JLCA that allows us to determine sufficient state conditions and message bit-relations optimized for collision attacks, *i.e.* minimal set of conditions attaining the highest success probability paths (where auxiliary carries are only allowed after a certain step).

### 4.3 Implementation of the Attack on GPUs

We now present a high-level view of the implementation of our attack, focusing on the features that make it efficient on GPUs. Their architecture being noticeably different from the one of CPUs, we first recall a few important points that will help understanding the design decisions<sup>1</sup>.

**Number of Cores and Scheduling.** A modern GPU can feature more than a thousand of small cores, that are packed together in a small number of larger “multiprocessor” execution units. Taking the example of the Nvidia GTX 970 for concreteness, there are 13 multiprocessors of 128 cores each, making 1664 cores in total [33]. The fastest instructions (such as for instance 32-bit bitwise logical operations or modular addition) have a throughput of 1 per core, which means that in ideal conditions 1664 instructions may be simultaneously processed by such a GPU in one clock cycle [32].

Yet, so many instructions cannot be emitted independently, or to put it in another way, one cannot run an independent thread of computation for every core. In fact, threads are grouped together by 32 forming a *warp*, and only warps may be scheduled independently. Threads within a warp may have a diverging control flow, for instance by taking a different path upon encountering a conditional statement, but their execution in this case is serialized. At an even higher level, warps executing the same code can be grouped together as *blocks*.

Furthermore, on each multiprocessor one can run up to 2048 threads simultaneously, which are dynamically scheduled every cycle onto the 128 cores at a warp granularity. Thus while a warp is waiting for the results of a computation or for a (high latency) memory operation to return, another warp can be scheduled. Although having more threads does not increase the computational power of the multiprocessor, such overbooking of cores can be used to hide latencies and thus increase efficiency of a GPU program.

In short, to achieve an optimal performance, one must bundle computations by groups of 32 threads executing the same instructions most of the time and diverging as little as possible and use as many threads as possible.

**Memory Architecture and Thread Synchronization.** In the same way as they feature many execution units, GPUs also provide memory of a generous size, which must however be shared among the threads. The amount of memory available to a single thread is therefore much less than what is typically available on a CPU (it of course highly depends on the number of running threads, but can be lower than 1 MB). This, together with the facts that threads of a same warp do not actually execute independently of each other and that threads of a same block run the same code makes it enticing to organize the memory structure of a program at the block level. Fortunately, this is made rather easy by the fact

---

<sup>1</sup> We specifically discuss these points for Nvidia GPUs of the most recent *Maxwell* generation such as the GTX970 being used in our attacks.

that many efficient synchronization functions are available for the threads, both at the warp and at the block level.

**Balancing the Work Between the GPU and the CPU.** The implementation of our attack can be broadly decomposed in two phases. The first step consists in computing a certain number of base solutions as in Sect. 4.1 and in storing them on disk. Because the total number of base solutions necessary to find a collision in our attack is rather small (about  $2^{25}$ ) and because they can be computed quickly, this can be done efficiently in an offline fashion using CPUs.

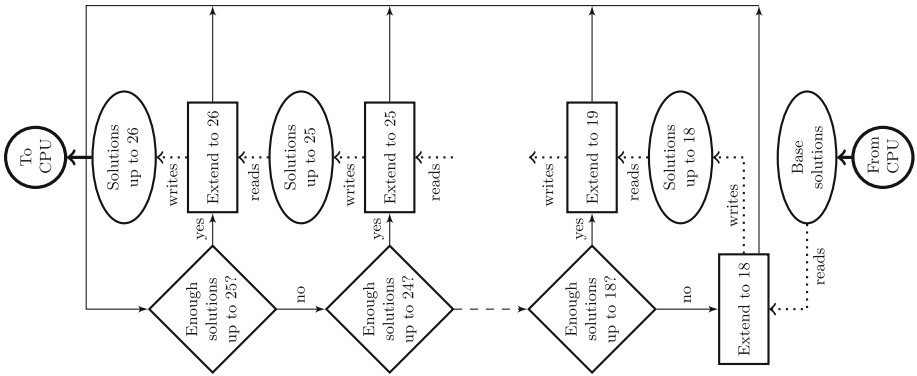
The second phase then consists in trying to extend probabilistically the base solutions (and their variants through the use of neutral bits) to find a collision. This is an intensely parallel task that is well suited to run on GPUs. However, as it was emphasized above, GPUs are most efficient when there is a high coherency between the execution of many threads. For that reason, we must avoid having idle threads that are waiting because their candidate solutions failed to follow the differential paths, while others keep on verifying a more successful one. Our approach to this is to fragment the verification into many small pieces (or *snippets*) that are chosen in a way which ensures that coherency is maintained for every thread of a warp when executing a single snippet, except in only a few small points. This is achieved through a series of intermediary buffers that store inputs and outputs for the snippets; a warp then only executes a given snippet if enough inputs are available for every of its threads. One should note that there is no need to entirely decompose the second step of the attack into snippets, and that a final part can again be run in a more serial fashion. Indeed, if inputs to such a part are scarce, there is no real advantage in verifying them in a highly parallel way.

The sort of decomposition used for the GPU phase of our attack as described above is in no way constrained by the specifics of our attack. In fact, it is quite general, and we believe that it can be successfully applied to many an implementation of cryptographic attacks. We conclude this section by giving more details of the application of this approach to the case of SHA-1.

**Choice of the Snippets.** As it was mentioned in Sect. 4.1, our attack uses neutral bits acting on the state words of step 18 to 26. The choice we made for the decomposition into snippets reflects this use of neutral bits: we use intermediary buffers to store partial solutions up to step 17, 18, etc. Then for each step the corresponding snippet consists in loading one partial solution per thread of a warp and applying every combination of neutral bits for this step. Each combination is tried by every thread at the same time on its own partial solution, thereby maintaining coherency. Then, each thread writes every resulting partial solution extended by one step to the output buffer of the snippet (which is the input buffer of the next snippet) at the condition that it is indeed valid, this being the only part of the code where threads may briefly diverge. For the later steps when no neutral bits can be used anymore, the snippets regroup the computation of several steps together, and eventually the verification that partial solutions

up to step 56 make valid collisions is done on a CPU. This is partly because the amount of available memory makes it hard to use step-by-step snippets until the end, but also because such partial solutions are only produced very slowly (a single GTX 970 produces solutions up to step 56 at a speed of about 0.017 solution per second, that is about 1 per minute).

**Complete Process of the Attack.** When running the attack, every warp tries to work with partial solutions that are up to the latest step possible. If no work is available there, it tries to work with partial solutions up to the second-latest step, etc. Eventually warps resort to using base solutions in the worst case that no work is available anywhere else. As was already said in Sect. 4.1, we experimentally observed that most of the work is done on partial solutions that are at least up to step 24, and work on solutions up to lower steps (and in particular base solutions) is thus done only intermittently. We conclude this description by giving a simplified flow chart (made slightly incorrect for the sake of clarity) of the GPU part of the SHA-1 attack in Fig. 2.



**Fig. 2.** Simplified flow chart for the GPU part of the attack. The start of this infinite loop is in the top left corner. Rectangles “□” represent snippets, ellipses “○” represent shared buffers, plain lines “—” represent control flow, and dotted lines “...” represent data flow.

## 5 Details of the Attack and Its Implementation

### 5.1 The Case of SHA-1

For our 76-step free-start collision attack, we selected disturbance vector II(55,0) (following Manuel’s classification [23]), and this for two reasons. Firstly, JLCA showed it to be one of the best for 76-steps. Secondly, the required IV difference is very sparse and localized on the two lowest bit positions, thus having low



potential for interference of the neutral bits with state conditions on the first few steps.

As explained in Sect. 4.2, we have extended JLCA to determine optimal sets of state conditions and message bit-relations given a non-linear path. For our attack we tried both non-linear differential path construction methods, *i.e.* the guess-and-determine method using our own implementation, and the meet-in-the-middle method using the public HashClash implementation [15]. We have found that the meet-in-the-middle approach generally resulted in fewer conditions and that furthermore we could better position the conditions. Our initial non-linear path was thus generated using the meet-in-the-middle approach, although when considering the full differential path one can encounter contradictions in the message bit-relations and/or an unsolvable highest density part of the differential path. These are expected situations which are easily solvable by considering variations of the non-linear part, which we did using the guess-and-determine approach.

The sufficient conditions over steps 0–35 and the message bit-relations can be found in the full version [19].

## 5.2 GPU Implementation

We complete the description of our approach towards GPU programming from Sect. 4.3 with a few lower-level details about our implementation on GTX 970.

**Block Layout.** A GTX 970 features 13 multiprocessors of 128 cores. Each multiprocessor can host a maximum of 2048 threads regrouped in at least 2 and at most 32 blocks [32]. If every multiprocessor of the GPU hosts 2048 threads, we say that we have reached *full occupancy*. While a multiprocessor can only physically run one thread per core (*i.e.* 128) at a given time, a higher number of resident threads is beneficial to hide computation and memory latencies. These can have a significant impact on the performance as a single waiting thread causes its entire warp of 32 to wait with him; it is thus important in this case for the multiprocessor to be able to schedule another warp in the meantime.

Achieving full occupancy is not however an absolute objective as it may or may not result in optimal performance depending on the resources needed by every thread. Important factors in that respect are the average amount of memory and the number of registers needed by a single thread, both being resources shared among the threads. In our implementation, the threads need to run rather heavy functions and full occupancy is typically not desirable. One reason why it is so is that we need to allocate 64 registers per thread in order to prevent register spilling in some of the most expensive functions; a multiprocessor having “only”  $2^{16}$  registers, this limits the number of threads to 1024. As a result, we use a layout of 26 blocks of 512 threads each, every multiprocessor being then able to host 2 such blocks.

**Buffer Framework.** As it was already said in Sect. 4.3, we use a number of shared buffers in our implementation in order to maximize coherency among threads of a single warp. With the exception of the buffers holding the base solutions and the collision candidates, there is one instance of every buffer per block. This allows to use block-wise instead of global synchronization mechanisms when updating the buffers' content, thence reducing the overhead inherent to the use of such shared data structures.

All of the buffers are cyclic and hold  $2^{20}$  elements of a few different type and size (with the exception of the ones holding solutions after step  $A_{36}$  which are of size only  $2^{10}$ , given their limited number). The different types of buffers are the following:

- The base solution buffer contains the value of 6 words of the solution's state  $A_{12}$  to  $A_{17}$ , and the 16 message words  $W_6$  to  $W_{21}$ , making 22 words in total. Although only 5 state words are necessary to fully determine the base solution, the value of  $A_{12}$  is additionally needed for the computation of some of the neutral bits.
- An *extended base solution* buffer is used after the step  $A_{21}$ ; it holds the value of state words  $A_{17}$  to  $A_{21}$ , message words  $W_{14}$  to  $W_{18}$  and  $W_{20}$ , and the index of the base solution that it extends, using 11 words in total.
- For all the remaining steps with neutral bits, a compact representation is used that only refers to the (extended) base solution from which it is derived and the value of its active neutral bits; all of this can be stored in only two words.
- A candidate solution buffer of 5 state words and 16 message words is used for partial solutions up to step  $A_{36}$  and step  $A_{56}$ .

The decomposition into base and extended base solutions was carefully chosen from the position of the neutral bits. From their description [19], one can see that neutral bits on the message words up to  $W_{18}$  are only used up to step  $A_{21}$ ; similarly, neutral bits on the words  $W_{19}$  to  $W_{21}$  are only used after step  $A_{21}$ . It is then only natural to define extended base solutions as up to  $A_{21}$ . Of course one could have dispensed with such a decomposition altogether, but this would mean that extending a base solution to the later steps (say  $A_{24}$ ) would systematically need to start recomputing many of the earlier steps from  $A_{17}$  before being able to do any useful work and this would be an unnecessary burden on these critical steps. We describe our packing of the neutral bits and of the index to the (extended) base solution in the full version [19]. As a side-note, let us also mention that the use of  $A_{36}$  and  $A_{56}$  as boundaries for the candidate solutions simply comes from the fact that each is the last of a series of 5 state words with no differences.

On the pure implementation side, we also carefully took into account the presence of a limited amount of very fast multiprocessor-specific shared memory. While the 96 KB available per multiprocessor is hardly enough to store the whole buffers themselves, we take advantage of it by dissociating the storage of the buffers and of the meta-data used for their control logic, the latter being held in shared memory. This improves the overall latency of buffer manipulations,

especially in case of heavy contention between different warps. This local shared memory is also very useful to buffer the writes to the buffers themselves. Indeed, only a fraction (often as low as  $\frac{1}{8}$ ) of the threads of a warp have a valid solution to write after having tested a single candidate, and the more unsuccessful threads need to wait while the former write their solution to global memory. It is therefore beneficial to first write the solutions to a small local warp-specific buffer and to flush it to the main block-wise buffer as soon as it holds 32 solutions or more, thence significantly reducing the number of accesses to the slower global memory.

**GPU Tuning.** After our initial implementation, we did some fine tuning of the GPU BIOS settings in order to try having an optimal performance. One first objective was to ensure that the GPU fans work at 100% during the attack, as this was strangely not the case initially, and was obviously not ideal for cooling. We also experimented with various temperature limits (that define when the GPU will start to throttle) and both over-clocking and under-volting. Taken together, these variations can have a significant impact on the overall performance of the program, as can be seen with our 76-step attack below.

## 6 Results and Perspectives

In this last section, we give the statistics for the performance of our implementation of the 76-step attack and estimate the cost of a collision on the full compression function of **SHA-1** using similar methods.

### 6.1 The 76-Step Collisions

The first collision was found when running the attack on a single GPU. Based on the production rate of partial solutions up to step 56, the estimated time to find a collision was slightly less than 5 days, at 4.94 days. This rate was also observed in practice, although we also witnessed significant outliers; as a matter of fact, the first collision was found in less than two days.

We subsequently ran the attack for a longer time on a server with four GPUs, and found 17 additional collisions. By improving the implementation and the GPU settings, we managed to significantly decrease the average time needed to find a collision. For the best configuration we found, the best-performing GPU computed collisions at an expected rate of 1 every 4.16 days, with an average of 4.42 for the 4 GPUs (producing solutions up to step 56 at a rate of 0.0171 per second). The whole server could then be expected to produce one collision every 1.1 day. Our GPU implementation of **SHA-1** can compute about  $2^{31.8}$  **SHA-1** compression functions per second. This means that on the best-performing GPU our attack has a complexity equivalent to  $2^{50.25}$  calls to the compression function. If one takes the average over the 4 GPUs, this increases slightly to  $2^{50.34}$ .

We also implemented our attack to run on a standard CPU, which provides an interesting comparison of the relative performance of the attack versus the speed of raw **SHA-1** computations. On an Haswell Core-i5 running at 3.2 GHz,

the OpenSSL implementation of SHA-1 can compute  $2^{23.47}$  compression functions per second, while our attack program generates solutions up to step 56 at a rate of 0.000124 per second. The total complexity of the attack thus requires about 606.12 core-days and has a complexity of  $2^{49.1}$  compression function calls. This means that a single GTX 970 is worth 322 such CPU cores when computing the SHA-1 compression function, and 138 cores when running our attack program (this increases to 146 for our best-performing GPU). While this drop in relative efficiency was to be expected, it is somehow surprisingly small given the complexity of our implementation and *e.g.* the intensive use of large shared data structures. Our careful implementation thus gives a much better value for the GPUs when compared to previous attempts at running cryptographic attacks on such a platform; in their attack, Grechnikov and Adinets estimated a GPU to be worth 39 CPU cores [14].

### 6.2 Collisions on the Full Compression Function

We are currently working to apply our methods to a free-start collision attack for the full SHA-1. Precisely estimating the cost of such an attack is always difficult before it is actually implemented as several factors may influence the complexity; none the least is the number and the quality of the neutral bits (or of accelerating techniques in general), which is typically hard to determine without a full implementation. We can however provide rather reliable estimates for different disturbance vectors by comparing the cost of the linear parts, as well as the number of conditions over the non-linear parts, and by making an educated guess of where should be the last step with a significant number of neutral bits. This guess is in particular made easier by the fact that we can compare a candidate disturbance vector to the one used for the 76-step attack, for which we have very precise results. As a consequence, we get the estimates in Table 2 for the complexity of an attack starting at  $A_{25}$  for two disturbance vectors. These figures need to be modulated by the fact that different DVs may yield neutral bits of different quality. Both II(55,0) and I(51,0) result in IVs with two differences, though the ones of II(55,0) may be at better positions. As a consequence, one may need to include step  $A_{24}$  and its one condition in the critical computations for I(51,0), thus doubling the complexity. Things are

**Table 2.** Complexity comparison and estimates for an 80-step attack. #C denotes the number of conditions for a given step and Gd is short for GPU-day (the cost as a number of compression function computation is also given as an alternative measure). The use of † denotes an estimated cost.

DV	Steps	Prob. ( $A_{25}$ )	Cost ( $A_{25}$ )	#C ( $A_{24}$ )	#C ( $A_{23}$ )
II(55,0)	76	$2^{-52.59}$	4.4 Gd ( $2^{50.3}$ )	1	3
I(51,0)	80	$2^{-62.27}$	3609† Gd ( $2^{60}$ †)	1	2
II(51,0)	80	$2^{-57.46}$	129† Gd ( $2^{55.2}$ †)	3	3

even worse for  $\text{II}(51,0)$  which yields an IV with five differences. Consequently, one would expect neutral bits to be markedly less efficient, and should probably add the cost of both  $A_{24}$  and  $A_{23}$ , resulting in a 6-bit increase of complexity. Thus, based on these two DVs, we can expect to find a free-start collision for 80 steps for an approximate cost of 7218 GPU-days based on  $\text{I}(51,0)$ , and 8234 GPU-days using  $\text{II}(51,0)$ . With a cluster of 64 GPUs, this represents 4 months of computation or thereabouts. While this gives us a reasonable upper-bound, it is still rather high and hence not entirely satisfactory. We plan to significantly improve the complexity of such an attack by:

1. investigating better disturbance vectors such as  $\text{II}(56,0)$ ,  $\text{II}(58,0)$  or  $\text{II}(59,0)$ ; unfortunately computing their exact probability with JLCA is much harder than for *e.g.*  $\text{II}(51,0)$ ;
2. using better accelerating techniques than the rather simple neutral bits used so far for the 76-step attack.

Both options should result in quite better attacks than the estimates from above. This is a promising and exciting future work, and we hope to achieve significant results in the near future.


## References

1. Biham, E., Chen, R.: Near-collisions of SHA-0. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 290–305. Springer, Heidelberg (2004)
2. Biham, E., Chen, R., Joux, A., Carribault, P., Lemuet, C., Jalby, W.: Collisions of SHA-0 and reduced SHA-1. In: Cramer [5], pp. 36–57
3. Brassard, G. (ed.): CRYPTO 1989. LNCS, vol. 435. Springer, Heidelberg (1990)
4. Chabaud, F., Joux, A.: Differential collisions in SHA-0. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 56–71. Springer, Heidelberg (1998)
5. Cramer, R. (ed.): EUROCRYPT 2005. LNCS, vol. 3494. Springer, Heidelberg (2005)
6. Damgård, I.: A design principle for hash functions. In: Brassard [3], pp. 416–427
7. De Cannière, C., Mendel, F., Rechberger, C.: Collisions for 70-step SHA-1: on the full cost of collision search. In: Adams, C., Miri, A., Wiener, M. (eds.) SAC 2007. LNCS, vol. 4876, pp. 56–73. Springer, Heidelberg (2007)
8. 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)
9. den Boer, B., Bosselaers, A.: An attack on the last two rounds of MD4. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 194–203. Springer, Heidelberg (1992)
10. den Boer, B., Bosselaers, A.: Collisions for the compression function of MD<sub>5</sub>. In: Helleseht, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 293–304. Springer, Heidelberg (1994)
11. Dobbertin, H.: Cryptanalysis of MD4. In: Gollmann, D. (ed.) FSE 1996. LNCS, vol. 1039, pp. 53–69. Springer, Heidelberg (1996)
12. Gilbert, H., Peyrin, T.: Super-sbox cryptanalysis: improved attacks for AES-like permutations. In: Hong, S., Iwata, T. (eds.) FSE 2010. LNCS, vol. 6147, pp. 365–383. Springer, Heidelberg (2010). <http://dx.doi.org/10.1007/978-3-642-13858-4>

13. Grechnikov, E.A.: Collisions for 72-step and 73-step SHA-1: improvements in the method of characteristics. IACR Cryptology ePrint Archive 2010, 413 (2010)
14. Grechnikov, E.A., Adinets, A.V.: Collision for 75-step SHA-1: intensive parallelization with GPU. IACR Cryptology ePrint Archive 2011, 641 (2011)
15. Hashclash project webpage. <https://marc-stevens.nl/p/hashclash/>
16. Jean, J., Naya-Plasencia, M., Peyrin, T.: Improved rebound attack on the finalist Grøstl. In: Canteaut, A. (ed.) FES 2012. LNCS, vol. 7549, pp. 110–126. Springer, Heidelberg (2012)
17. Johansson, T., Nguyen, P.Q. (eds.): EUROCRYPT 2013. LNCS, vol. 7881. Springer, Heidelberg (2013). <http://dx.doi.org/10.1007/978-3-642-38348-9>
18. 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)
19. Karpman, P., Peyrin, T., Stevens, M.: Practical free-start collision attacks on 76-step SHA-1. IACR Cryptology ePrint Archive 2015, 530 (2015)
20. Klíma, V.: Tunnels in hash functions: MD5 collisions within a minute. IACR Cryptology ePrint Archive 2006, 105 (2006)
21. Lamberger, M., Mendel, F., Rechberger, C., Rijmen, V., Schläffer, M.: Rebound distinguishers: results on the full whirlpool compression function. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 126–143. Springer, Heidelberg (2009). <http://dx.doi.org/10.1007/978-3-642-10366-7>
22. Landelle, F., Peyrin, T.: Cryptanalysis of full RIPEMD-128. In: Johansson and Nguyen [17], pp. 228–244. <http://dx.doi.org/10.1007/978-3-642-38348-9>
23. Manuel, S.: Classification and generation of disturbance vectors for collision attacks against SHA-1. Des. Codes Crypt. **59**(1–3), 247–263 (2011)
24. 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). <http://dx.doi.org/10.1007/978-3-642-05445-7>
25. 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). <http://dx.doi.org/10.1007/978-3-642-03317-9>
26. Mendel, F., Rijmen, V., Schläffer, M.: Collision attack on 5 rounds of Grøstl. In: Cid, C., Rechberger, C. (eds.) FSE 2014. LNCS, vol. 8540, pp. 509–521. Springer, Heidelberg (2015). <http://dx.doi.org/10.1007/978-3-662-46706-0>
27. Merkle, R.C.: One way hash functions and DES. In: Brassard [3], pp. 428–446
28. National Institute of Standards and Technology: FIPS 180: Secure Hash Standard, May 1993
29. National Institute of Standards and Technology: FIPS 180–1: Secure Hash Standard, April 1995
30. National Institute of Standards and Technology: FIPS 180–2: Secure Hash Standard, August 2002
31. National Institute of Standards and Technology: Draft FIPS 202: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions, May 2014
32. Nvidia Corporation: Cuda C Programming Guide. <https://docs.nvidia.com/cuda/cuda-c-programming-guide>
33. Nvidia Corporation: Nvidia Geforce GTX 970 Specifications. <http://www.geforce.com/hardware/desktop-gpus/geforce-gtx-970/specifications>

34. 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)
35. Rivest, R.L.: RFC 1321: The MD5 Message-Digest Algorithm, April 1992
36. Saarinen, M.-J.O.: Cryptanalysis of block ciphers based on SHA-1 and MD5. In: Johansson, T. (ed.) FSE 2003. LNCS, vol. 2887, pp. 36–44. Springer, Heidelberg (2003)
37. Stevens, M.: Attacks on Hash Functions and Applications. Ph.D. thesis, Leiden University, June 2012
38. Stevens, M.: Counter-cryptanalysis. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 129–146. Springer, Heidelberg (2013). <http://dx.doi.org/10.1007/978-3-642-40041-4>
39. Stevens, M.: New collision attacks on SHA-1 based on optimal joint local-collision analysis. In: Johansson and Nguyen [17], pp. 245–261. <http://dx.doi.org/10.1007/978-3-642-38348-9>
40. 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). [http://dx.doi.org/10.1007/978-3-540-72540-4\\_1](http://dx.doi.org/10.1007/978-3-540-72540-4_1)
41. 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). <http://dx.doi.org/10.1007/978-3-642-03356-8>
42. 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)
43. Wang, X., Yu, H.: How to break MD5 and other hash functions. In: Cramer [5], pp. 19–35

# Fast Correlation Attacks over Extension Fields, Large-Unit Linear Approximation and Cryptanalysis of SNOW 2.0

Bin Zhang<sup>1,2</sup>, Chao Xu<sup>1</sup>, and Willi Meier<sup>3</sup>

<sup>1</sup> TCA Laboratory, SKLCS, Institute of Software, Chinese Academy of Sciences,  
Beijing, China

<sup>2</sup> State Key Laboratory of Cryptology, P.O.Box 5159, Beijing 100878, China

<sup>3</sup> FHNW, Windisch, Switzerland

`willi.meier@fhnw.ch`

**Abstract.** Several improvements of fast correlation attacks have been proposed during the past two decades, with a regrettable lack of a better generalization and adaptation to the concrete involved primitives, especially to those modern stream ciphers based on word-based LFSRs. In this paper, we develop some necessary cryptanalytic tools to bridge this gap. First, a formal framework for fast correlation attacks over extension fields is constructed, under which the theoretical predictions of the computational complexities for both the offline and online/decoding phase can be reliably derived. Our decoding algorithm makes use of Fast Walsh Transform (FWT) to get a better performance. Second, an efficient algorithm to compute the large-unit distribution of a broad class of functions is proposed, which allows to find better linear approximations than the bitwise ones with low complexity in symmetric-key primitives. Last, we apply our methods to SNOW 2.0, an ISO/IEC 18033-4 standard stream cipher, which results in the significantly reduced complexities all below  $2^{164.15}$ . This attack is more than  $2^{49}$  times better than the best published result at Asiacrypt 2008. Our results have been verified by experiments on a small-scale version of SNOW 2.0.

**Keywords:** Stream ciphers · Cryptanalysis · Large-unit · SNOW 2.0 · Finite state machine (FSM) · Linear feedback shift register (LFSR)

## 1 Introduction

The design and analysis of any cipher in history have to match well with the computing technologies in a specified period. Fast correlation attacks, introduced by Meier and Staffelbach in 1989 [19], are commonly regarded as classical methods in the cryptanalysis of LFSR-based stream ciphers, which were usually implemented in hardware at that time. In general, fast correlation attacks have been constantly and steadily evolving [4, 5, 11], resulting in more and more powerful decoding methods dedicated to very large linear codes in the presence of a highly noisy channel.



On the other side, with the development of computing facilities, many word-oriented stream ciphers have been proposed, e.g., SNOW 2.0, SNOW 3G [6, 8] and Sosemanuk [2], aiming to combine the merits from the thoroughly studied LFSR theory with a fast implementation in software. Due to the complex form of the reduced LFSR recursion from the extension field to  $\text{GF}(2)$  (many taps and a large number of state variables), the previous bitwise fast correlation attacks do not work so well as expected in these cases. This motivates us to study the security of these word-oriented primitives against a new form of fast correlation attacks that works on some larger data unit.

**Our Contributions.** First, a formal framework for fast correlation attacks over extension fields is constructed, under which the theoretical predictions of the computational complexities for both the offline and online/decoding phase can be reliably derived. This gives an answer to the open problem of Meier in [18] at FSE 2011. We adapt the  $k$ -tree algorithm [24] to generate the desirable parity check equations in the pre-computation phase and propose a fast decoding algorithm for the online phase. Second, an efficient algorithm to compute the large-unit distributions of the generalized pseudo-linear functions modulo  $2^n$  (GPLFM), which includes all the previously studied relevant topics [17] in an unified framework, is proposed. This technique, serving as a basis to the first one, generalizes the algorithm in [22] and has the value in its own right. It can compute the noise distributions of the linear approximations of the GPLFM (including the addition modulo  $2^n$ ) in a larger alphabet of  $m$ -bit ( $m > 1$ ) size when  $m$  is divisible by  $n$  with a low complexity, e.g., for  $n = 32$ , the 2, 4, 8, 16-bit linear approximations can be found efficiently with a slice size depending on the structure of the primitive. Last, we apply our methods to SNOW 2.0, an ISO/IEC 18033-4 standard and a benchmark stream cipher in the European eSTREAM project. We build the byte-wise linear approximation of the FSM by further generalizing the GPLFM to include the S-boxes and restore the initial state of the LFSR (thus the key) with a fast correlation attack over  $\text{GF}(2^8)$ . The time/memory/data/pre-computation complexities of this attack are all below  $2^{186.95}$ . Then we further improve our attack by changing the linear mask from  $\text{GF}(2)$  to  $\text{GF}(2^8)$ , which results in the significantly reduced time/memory/data/pre-computation complexities all below  $2^{164.15}$ . This attack is more than  $2^{49}$  times better than the best published result at Asiacrypt 2008<sup>1</sup>. Table 1 presents a comparison of our attack on SNOW 2.0 with the best previous ones. Our results have been verified on a small-scale version of SNOW 2.0 with 16-bit word size in experiments.

**Table 1.** Comparison of the attacks on SNOW 2.0

	Type	Data	Time
[22]	Distinguishing attack	$2^{174}$	$2^{174}$
[13]	Key recovery attack	$2^{198.77}$	$2^{212.38}$
<b>This paper</b>	Key recovery attack	$2^{163.59}$	$2^{164.15}$

<sup>1</sup> Note that in the Asiacrypt 2008 paper [13], the complexity is written as  $2^{204.38}$  in the abstract, while from the formula in Sect. 6, this complexity is  $2^{212.38}$ .

**Outline.** We present some preliminaries relevant to our work in Sect. 2. In Sect. 3, the framework of fast correlation attacks over extension fields is established with detailed theoretical justifications. The new algorithm to accurately and efficiently compute the large-unit distribution of the GPLFM is provided in Sect. 4. The application of our approaches to SNOW 2.0 is given in Sect. 5. The improved attack using finite field linear masks is described in Sect. 6 with the experimental results. Finally, some conclusions are made and future work is pointed out in Sect. 7.

## 2 Preliminaries

In this section, some notations and basic definitions are presented. Denote the set of real numbers by  $\mathbf{R}$ . The binary field is denoted by  $\text{GF}(2)$  and the  $m$ -dimensional extension field of  $\text{GF}(2)$  is denoted by  $\text{GF}(2^m)$ . The modular addition is  $\boxplus$  and the usual xor operation is  $\oplus$ . The inner product of two  $n$ -dimensional vectors  $a$  and  $b$  over  $\text{GF}(2^m)$  is defined as  $\langle a, b \rangle = \langle (a_0, \dots, a_{n-1}), (b_0, \dots, b_{n-1}) \rangle := \bigoplus_{i=0}^{n-1} a_i b_i$ . As usual, a function  $f : \text{GF}(2^n) \rightarrow \text{GF}(2)$  is called a Boolean function and a function  $g = (g_1, \dots, g_m) : \text{GF}(2^n) \rightarrow \text{GF}(2^m)$  with each  $g_i$  ( $1 \leq i \leq m$ ) being a Boolean function is called a  $m$ -dimensional vectorial Boolean function.

**Definition 1.** Let  $X$  be a binary random variable, the correlation between  $X$  and zero is defined as  $c(X) = \Pr\{X = 0\} - \Pr\{X = 1\}$ . The correlation of a Boolean function  $f : \text{GF}(2^n) \rightarrow \text{GF}(2)$  to zero is defined as  $c(f) = \Pr\{f(X) = 0\} - \Pr\{f(X) = 1\}$ , where  $X \in \text{GF}(2^n)$  is an uniformly distributed random variable.

Given a vectorial Boolean function  $g : \text{GF}(2^n) \rightarrow \text{GF}(2^m)$ , define the distribution  $p_g$  of  $g(X)$  with  $X$  uniformly distributed as  $p_g(a) = \#\{X | g(X) = a\} / 2^n$  for all  $a \in \text{GF}(2^m)$ .

**Definition 2.** As in [1], the Squared Euclidean Imbalance (SEI) of  $p_g$  is  $\Delta(p_g) = 2^m \sum_{a \in \text{GF}(2^m)} (p_g(a) - \frac{1}{2^m})^2$ , which measures the distance between the target distribution and the uniform distribution.

SEI<sup>2</sup> is used to evaluate the efficiency of large-unit linear approximations in this paper. Here by *large-unit*, we refer to the linear approximation whose basic data unit is non-binary. The next definition introduces a powerful tool to compute the correlation of a nonlinear function and to reduce the complexity of the substitution step of a fast correlation attack [5].

**Definition 3.** Given a function  $f : \text{GF}(2^n) \rightarrow \mathbf{R}$ , for  $\omega \in \text{GF}(2^n)$ , the Walsh Transform of  $f$  at point  $\omega$  is defined as  $\hat{f}(\omega) = \sum_{x \in \text{GF}(2^n)} f(x) (-1)^{\langle \omega, x \rangle}$ .

The Walsh Transform of  $f$  can be computed efficiently with an algorithm called Fast Walsh Transform (FWT) [25] in  $n2^n$  time and  $2^n$  memory. The preparation of  $f$  takes  $2^n$  time, thus the total time complexity is  $2^n + n2^n$ , which is a large improvement compared to  $2^{2n}$ . The following fact [21] is used in our analysis.

<sup>2</sup> SEI is also referred to as capacity of the distribution in [9].

**Lemma 4.** *We consider a vectorial Boolean function  $g : GF(2^n) \rightarrow GF(2^m)$  with the probability distribution vector  $p_g$ . Then  $\Delta(p_g) = \sum_{a \in GF(2^m)} c^2(\langle a, g \rangle)$ , where  $c(\langle a, g \rangle)$  is the correlation of the Boolean function  $\langle a, g \rangle$ .*

Lemma 4 indicates that we can derive the SEI of distribution  $p_g$  with the correlations  $c(\langle a, g \rangle)$  for  $a \in GF(2^m)$ . Therefore, computing the SEI of the large data unit distribution can be reduced to the problem of looking for bitwise linear approximations with non-negligible correlations.

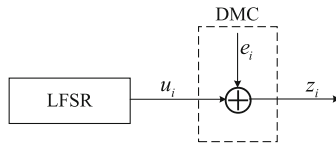
### 3 Fast Correlation Attacks Over Extension Fields

In this section, we will describe a formal framework for fast correlation attacks over  $GF(2^n)$ , which is the *first* comprehensive answer to the open problem how to amount fast correlation attack over the extension fields proposed in [3, 18]. Let us first define the notations used hereafter.

- $N$  is the number of available output words.
- $l$  is the word-length of the LFSR over  $GF(2^n)$ .
- $l'$  is the number of target words in decoding phase.
- $G$  is the  $l \times N$  generator matrix of a  $[N, l]$  linear code  $\mathcal{C}_1$  over  $GF(2^n)$ .
- $u_i \in GF(2^n)$  is the  $i$ -th output word of the LFSR.
- $z_i \in GF(2^n)$  is the  $i$ -th output word of the keystream generator.
- $e_i \in GF(2^n)$  is the  $i$ -th noisy variable of a Discrete Memoryless Channel (DMC).

#### 3.1 Model for Fast Correlation Attacks Over Extension Fields

The fast correlation attack over extension fields is also modelled as a decoding problem, i.e., the keystream segment  $\mathbf{z} = (z_1, z_2, \dots, z_N)$  can be seen as the transmission result of the LFSR sequence  $\mathbf{u} = (u_1, u_2, \dots, u_N)$  through a DMC with the noisy variables  $\mathbf{e} = (e_1, e_2, \dots, e_N)$ , as shown in Fig. 1. From this model, we can represent the received symbols  $z_i$  as  $z_i = u_i \oplus e_i$ , where the noise variable  $e_i$  is non-uniformly distributed for  $i = 1, \dots, N$ . The capacity of the DMC is  $C_{\text{DMC}} = \log(2^n) + \sum_{e \in GF(2^n)} \Pr\{e_i = e\} \cdot \log(\Pr\{e_i = e\})$ , where the maximum capacity is reached when  $\Pr\{e_i = e\} = 1/2^n$  for all  $e \in GF(2^n)$ . Then the above decoding problem is converted into decoding a  $[N, l]$  linear code  $\mathcal{C}_1$  over  $GF(2^n)$ , where  $N$  is the code length and  $l$  is the symbol-length of information, with the



**Fig. 1.** Model for fast correlation attacks over  $GF(2^n)$

code rate  $R = \log(2^n) \cdot l/N$ . Using Taylor series at order two, we achieve the following theorem, which theoretically connects the capacity of the DMC with the SEI of the noise distribution.

**Theorem 5.** *Let  $C_{\text{DMC}}$  be the capacity of a DMC over  $GF(2^n)$  and the noise variable  $e_i \in GF(2^n)$ , whose distribution is denoted by  $p_{e_i} = (\text{Pr}\{e_i = 0\}, \dots, \text{Pr}\{e_i = 2^n - 1\})$ . Then the theoretical relation between the capacity  $C_{\text{DMC}}$  and the SEI of  $p_{e_i}$ , i.e.,  $\Delta(p_{e_i})$ , is  $C_{\text{DMC}} \approx \frac{\Delta(p_{e_i})}{2 \ln(2)}$ .*

This theorem provides a tool for bridging the theory based on Shannon theory and that based on the SEI measure. Theorem 5 is the basis of our framework, enabling us to derive a lower bound on the keystream length required for a successful attack. Actually, a  $[N, l]$  linear code over  $GF(2^n)$  can be successfully decoded only if its code rate does not exceed the capacity of the transmission channel, pioneered in [23].

Theorem 5 and Shannon Theorem are combined together in our framework to give a theoretical analysis of the new fast correlation attacks over extension fields. Under this theoretical framework, we can assure that the fast correlation attack succeeds with a high probability, i.e.,  $0.5 < P_{\text{succ}} \leq 1$ , if  $R < C_{\text{DMC}}$ .

### 3.2 General Description of Fast Correlation Attacks Over Extension Fields

Our new algorithm is extracted from the previous work in [10, 12] by addressing some important unsolved problems therein. First, the pre-computation algorithm in [10, 12] uses the straight forward method to find all the possible collisions over extension fields, whose complexity is too high to be applied in cryptanalysis. Second, in Fig. 1, only a DMC with the following properties is considered, i.e., the distribution of the noise variable  $e_i$  satisfies  $\text{Pr}\{e_i = 0\} = 1/2^n + \delta$  and  $\text{Pr}\{e_i = e\} = 1/2^n - \delta/(2^n - 1), \forall e \in GF(2^n), e \neq 0$ , which is *not* the general case. Usually in the practice of correlation attacks, the distribution of noisy variable does not necessarily satisfy this condition. Third, the straightforward method is used to identify the correct key in the online phase, i.e., by evaluating parity-checks one by one for each possible codeword, which is inappropriate for cryptanalytic purposes. Last, a comprehensive theoretical justification is missing, which will assure the decoding reliability when simulations are infeasible.

**Preprocessing.** As in [4, 10, 12], we convert the original code  $\mathcal{C}_1$  directly derived from the primitive to a new code  $\mathcal{C}_2$ , which is expected to be easier to decode by some fast decoding algorithm later devised. Precisely, let the length of the LFSR be  $l$ -word. Then we have  $\mathbf{u} = (u_1, u_2, \dots, u_l) \cdot G$ , where  $(u_1, u_2, \dots, u_l)$  is the initial state of the LFSR. Let  $(\cdot, \dots, \cdot)^T$  be the transpose of a vector, we rewrite the matrix  $G$  in column vectors as  $G = (\mathbf{g}_1, \mathbf{g}_2, \dots, \mathbf{g}_N)$ , where  $\mathbf{g}_i = (g_i^1, g_i^2, \dots, g_i^l)^T$  ( $1 \leq i \leq N$ ) is the  $i$ -th column vector. In order to reduce the decoding complexity, we build a new code  $\mathcal{C}_2$  with a smaller number of information symbols  $\hat{\mathbf{u}} = (u_1, u_2, \dots, u_{l'})$  for a certain  $l' < l$  as follows. We first

look for some  $k$ -tuple column vectors  $(\mathbf{g}_{i_1}, \mathbf{g}_{i_2}, \dots, \mathbf{g}_{i_k})$  satisfying  $\mathbf{g}_{i_1} \oplus \mathbf{g}_{i_2} \oplus \dots \oplus \mathbf{g}_{i_k} = (c_1, c_2, \dots, c_{l'}, 0, \dots, 0)^T$ . For each  $k$ -tuple, we have

$$\bigoplus_{j=1}^k u_{i_j} = (u_1, u_2, \dots, u_l) \bigoplus_{j=1}^k \mathbf{g}_{i_j} = c_1 u_1 \oplus c_2 u_2 \oplus \dots \oplus c_{l'} u_{l'}. \tag{1}$$

This equation is called the *parity check* for  $u_1, \dots, u_{l'}$ . Since  $z_i = u_i \oplus e_i$ , we rewrite it as  $\bigoplus_{j=1}^k z_{i_j} = c_1 u_1 \oplus c_2 u_2 \oplus \dots \oplus c_{l'} u_{l'} \oplus \bigoplus_{j=1}^k e_{i_j}$ . Collect a desirable number of such  $k$ -tuples and denote the number of such derived equations by  $m_k$ . Denote the indices of  $t$ -th such tuple of columns by  $\{i_1^{(t)}, i_2^{(t)}, \dots, i_k^{(t)}\}$ . Let  $U_t = \bigoplus_{j=1}^k u_{i_j^{(t)}}$ ,  $1 \leq t \leq m_k$ . Thus we have constructed an  $[m_k, l']$ -code  $\mathcal{C}_2$ , i.e.,  $\mathbf{U} = (U_1, U_2, \dots, U_{m_k})$ .

**Processing.** Denote the received sequence by  $\mathbf{Z} = (Z_1, Z_2, \dots, Z_{m_k})$ , where  $Z_t = \bigoplus_{j=1}^k z_{i_j^{(t)}}$ . We first use the keystream words  $z_1, z_2, \dots, z_N$  to compute  $\mathbf{Z}$ . Then decode the code  $\mathcal{C}_2$  using the algorithm in the following subsection and output  $(u_1, u_2, \dots, u_{l'})$ . Using the DMC model and assuming that all the  $e_i$ s are independent random values over  $\text{GF}(2^n)$ , it is easy to see that the distribution of the folded noisy variable  $E_t = \bigoplus_{j=1}^k e_{i_j^{(t)}}$  can be computed by the convolution property via FWT. The new noise sequence can be represented as  $\mathbf{E} = (E_1, E_2, \dots, E_{m_k})$ .

### 3.3 Preprocessing Stage: Generating the Parity Checks

Now we present an algorithm to compute the desirable  $k$ -tuple parity checks with a relatively low complexity, while the straight forward method in [12] needs a complexity of  $O(N^k)$ . First look at the case of  $k = 2$ . Eq. (1) indicates that  $(g_{i_1}^{l'+1}, g_{i_1}^{l'+2}, \dots, g_{i_1}^l)^T \oplus (g_{i_2}^{l'+1}, g_{i_2}^{l'+2}, \dots, g_{i_2}^l)^T = (0, \dots, 0)^T$ . Thus the construction of parity checks is equivalent to the searching of  $n(l - l')$ -bit collision, i.e., just split  $(g_i^{l'+1}, g_i^{l'+2}, \dots, g_i^l)$  for  $i = 1, \dots, N$  into two lists  $L_1$  and  $L_2$ , and look for  $x_1 \in L_1, x_2 \in L_2$  such that  $x_1 \oplus x_2 = 0$ . Hence, by searching for collisions through these two lists, 2-tuple parity checks in our attack can be constructed.

Note that the crucial difference between  $\text{GF}(2^n)$  and  $\text{GF}(2)$  requires that the length of the partial collision positions cannot be arbitrary and should be a multiple of  $n$ . In general, we can split the truncated matrix columns of  $G$  into  $k$  lists and search for  $x_i \in L_i$  for  $1 \leq i \leq k$  such that  $\bigoplus_{i=1}^k x_i = 0$  holds for  $1 \leq i \leq k$ . This problem can be transformed into the well known  $k$ -sum problem. *Problem 1.* (The  $k$ -sum problem) Given  $k$  lists  $L_1, \dots, L_k$ , each of length  $\alpha$  and containing elements drawn uniformly and independently at random from  $\{0, 1\}^{n(l-l')}$ , find  $x_1 \in L_1, \dots, x_k \in L_k$  such that  $x_1 \oplus x_2 \oplus \dots \oplus x_k = 0$ .

Fortunately, this problem can be efficiently solved by the  $k$ -tree algorithm in [24]. It is shown that the  $k$ -tree algorithm requires  $O(k2^{n(l-l')/(1+\log k)})$  time and space and uses lists of size  $O(2^{n(l-l')/(1+\log k)})$ . The  $k$ -tree algorithm can

also find many solutions to the  $k$ -sum problem. It can find  $\beta^{1+\log k}$  solutions to the  $k$ -sum problem with  $\beta$  times as much work as finding a single solution, as long as  $\beta \leq 2^{n(l-l')/(l \log k(1+\log k))}$ . Thus the total time/space complexities are  $O(\beta k 2^{n(l-l')/(1+\log k)})$  and the size of each list is  $O(\beta 2^{n(l-l')/(1+\log k)})$ .

Now we show how to generate the  $m_k$   $k$ -tuple parity checks. Precisely, we denote the truncated partial vector of  $\mathbf{g}_i$  by  $\mathbf{x}_i = (g_i^{l'+1}, \dots, g_i^l)$  for  $i = 1, \dots, N$ . Then disjoint  $(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N)$  into  $k$  lists  $L_1, \dots, L_k$ , each of length  $\alpha = N/k$ . We want to find  $\mathbf{x}_1 \in L_1, \dots, \mathbf{x}_k \in L_k$  satisfying  $\mathbf{x}_1 \oplus \mathbf{x}_2 \oplus \dots \oplus \mathbf{x}_k = 0$ . This is exactly the same case as the  $k$ -sum problem, so we can adopt the  $k$ -tree algorithm in [24] to find the required number of desirable parity checks.

### 3.4 Processing Stage: Decoding the Code $\mathcal{C}_2$

It is well-known that decoding a random linear code over an extension field is a NP-hard problem. Here we present a fast decoding algorithm, which can be seen as a solution to this problem.

As shown in [5, 14], FWT can be used to accelerate the decoding process for the linear codes over GF(2). Here we derive a method based on Lemma 4 to exploit FWT for decoding linear codes over GF( $2^n$ ).

Let us denote the guessed value of the partial initial state  $\hat{\mathbf{u}} = (u_1, \dots, u_{l'})$  by  $\hat{\mathbf{u}}' = (u'_1, \dots, u'_{l'})$ . After pre-computation, we construct a distinguisher  $I(\hat{\mathbf{u}}') = c_1^{(t)}(u_1 \oplus u'_1) \oplus \dots \oplus c_{l'}^{(t)}(u_{l'} \oplus u'_{l'}) \oplus E_t = Z_t \oplus c_1^{(t)}u'_1 \oplus \dots \oplus c_{l'}^{(t)}u'_{l'}$ , to find the correct partial state  $\hat{\mathbf{u}}$ . If the guessed value  $\hat{\mathbf{u}}'$  is correct,  $I$  is expected to be biased; otherwise it approximates an uniform distribution.

Next, let us give a description on how to compute the SEI of  $I(\hat{\mathbf{u}}')$ , which is the crucial part of our algorithm. We need to substitute the  $z_i$ s into the parity check equations and evaluate the SEI of  $I$  for each possible  $\hat{\mathbf{u}}'$ . Combining Lemma 4 in Sect. 2.2 with FWT, we have the following method. Precisely, the SEI of  $I(\hat{\mathbf{u}}')$  can be computed by the correlations  $c(\langle \gamma, I \rangle)$ , where  $\langle \gamma, I \rangle$  is a boolean function and  $\gamma \in \text{GF}(2)^n$ . We can divide the vectorial boolean function  $I$  into  $n$  linearly independent boolean functions  $I_1, \dots, I_n$  and each boolean function can be expressed as  $I_i = \langle w_i, \hat{\mathbf{u}}' \rangle \oplus \langle v_i, Z_t \rangle$ , where  $w_i \in \text{GF}(2)^{n l'}$ ,  $v_i \in \text{GF}(2)^n$  are two binary coefficient vectors. Let  $Q = \text{span}\{I_1, \dots, I_n\}$  such that  $Q$  is a set of approximations generated by these  $n$  approximations  $I_i$ . Now the advantage is that FWT can be used to compute the correlation of each approximation  $I_i$  for  $i = 1, \dots, n$ , as described in [14].

Preciously, assume that we have  $m_k$   $n$ -bit parity checks over GF( $2^n$ ) with the same distribution. Then for each  $I_i$  there are  $m_k$  bitwise parity checks denoted by  $I_i^{(t)}$  for  $1 \leq t \leq m_k$ . In order to evaluate these  $m_k$  bitwise parity checks  $I_i^{(t)} = \langle w_i^{(t)}, \hat{\mathbf{u}}' \rangle \oplus \langle v_i^{(t)}, Z_t \rangle$  for each  $\hat{\mathbf{u}}'$ , we introduce an integer-valued function,

$$h(\hat{\mathbf{u}}') = \sum_{1 \leq t \leq m_k : \hat{\mathbf{u}}' = w_i^{(t)}} (-1)^{\langle v_i^{(t)}, Z_t \rangle},$$

for all  $\hat{\mathbf{u}}' \in \text{GF}(2^{n l'})$ . We compute the Walsh transform of  $h$  and then we can get an  $2^{n l'}$ -dimensional array storing the correlation  $c(I_i)$  indexed by  $\hat{\mathbf{u}}'$ . The total

time complexity for computing  $c(I_1), \dots, c(I_n)$  is  $O(n(m_k + l'n2^{l'n}))$  and memory complexity is  $O(n2^{l'n})$ . In addition, the correlations of the other  $2^n - n - 1$  linear approximations can be computed by the Piling-up Lemma [16]. Thus, we have got all the correlations for different guessed values of  $\hat{\mathbf{u}}$ . Again from Lemma 4, we can easily compute  $\Delta(I(\hat{\mathbf{u}}'))$  for each possible  $\hat{\mathbf{u}}'$ . Then, we can use a distinguisher described in [1] to recover the correct initial state. In total, the time complexity of decoding  $\mathcal{C}_2$  in such a way is  $O(n(m_k + l'n2^{l'n}) + 2^n2^{l'n})$ .

Now we give the theoretical justifications of our algorithm. Assume the noisy distribution of  $E_t$  over  $\text{GF}(2^n)$  is  $p_{E_t} = (\text{Pr}\{E_t = 0\}, \dots, \text{Pr}\{E_t = 2^n - 1\})$  and the code length of  $\mathcal{C}_2$  is  $m_k$ . According to the  $k$ -tree algorithm, using  $k$  lists, each of which has size of  $\alpha = \beta 2^{n(l-l')/(1+\log k)}$ , we can find  $\beta^{1+\log k}$  parity checks.

Since the number of parity checks pre-computed is  $m_k$ , thus we have  $m_k = \beta^{1+\log k}$ . Further, for the decoding to succeed, the code rate  $R = l' \cdot \log(2^n)/m_k$  of  $\mathcal{C}_2$  must satisfy  $R < C_{\text{DMC}}$ . Then by Theorem 5, the value of  $m_k$  can be calculated as  $m_k \approx (2^{l'n} \ln 2)/\Delta(p_{E_i})$ . The following theorem gives the required length  $N$  of the observed keystream segment for successfully decoding code  $\mathcal{C}_1$ .

**Theorem 6.** *Given a  $[N, l]$  linear code  $\mathcal{C}_1$  over  $\text{GF}(2^n)$ . After applying the pre-computation of our algorithm, we get a new  $[m_k, l']$  linear code  $\mathcal{C}_2$ , which is transmitted through a  $2^n$ -ary DMC with the noise distribution  $p_{E_i}$ . The required length  $N$  of the observed keystream segment for the algorithm to succeed is  $N \approx k 2^{\frac{n(l-l')}{\theta}} (2^{l'n} \ln 2)^{\frac{1}{\theta}} \Delta(p_{E_i})^{-\frac{1}{\theta}}$ , where  $\theta = 1 + \log k$ .*

## 4 Large-Unit Linear Approximation and Its Distribution

In this section, an efficient algorithm to accurately compute the large-unit distribution of the GPLFM is proposed. This is desired when the decoding algorithm is available.

### 4.1 Large-Unit Linear Approximations

Most of the previous work only study how to use the bitwise linear approximations to constitute a vector, here we directly focus on the non-binary linear approximations whose basic data unit is over  $\text{GF}(2^m)$  ( $m > 1$ ) and such non-binary unit linear approximations are called the *large-unit linear approximations* throughout this paper<sup>3</sup>. Let  $H(X_1, X_2, \dots, X_d)$  be a non-linear function, where the output and the input  $X_i$ s are all random variables over  $\text{GF}(2^n)$ . Our task is to accurately compute the  $m$ -bit large-unit distribution of some linear approximation of  $H$ . In practice, the choice of  $m$  cannot be arbitrary and is usually determined by the structure of the primitive and the underlying building blocks, e.g., the LFSR structure and the S-box size. When  $m$  is fixed, the output of  $H$  and each input  $X_i (1 \leq i \leq d)$  can all be regarded as some  $\frac{n}{m}$ -dimensional vectors over  $\text{GF}(2^m)$ . In this setting, the definition of a binary linear mask is as follows.

<sup>3</sup> As we can see, when  $m = 1$  it is just the bitwise approximation of  $F$ , while when  $m = n$  it becomes the  $n$ -bit linear approximation, discussed in [7, 17].

**Definition 7.** Let  $X \in GF(2^n)$  and  $\Omega = (\omega_{\frac{n}{m}}, \dots, \omega_2, \omega_1)$  be a  $\frac{n}{m}$ -dimensional binary vector, then  $X$  can be transformed to a  $\frac{n}{m}$ -dimensional vector  $X = (x_{\frac{n}{m}}, \dots, x_2, x_1)$  over  $GF(2^m)$  with  $x_i \in GF(2^m)$  for  $1 \leq i \leq \frac{n}{m}$ . The inner product between these two vectors is defined as  $\Omega \cdot X = \omega_{\frac{n}{m}} x_{\frac{n}{m}} \oplus \dots \oplus \omega_1 x_1 \in GF(2^m)$ , where  $\Omega$  is called the  $\frac{n}{m}$ -dimensional binary linear mask of  $X$  over  $GF(2^m)$ .

## 4.2 The Generalized Pseudo-Linear Function Modulo $2^n$

Now we first generalize the pseudo-linear function modulo  $2^n$  (PLFM) in [17] to GPLFM by introducing the binary mask with the inner product in Definition 7. Note that in [17], the distribution of some class of functions called PLFM over  $GF(2^n)$  is computed, here we consider similar problems of GPLFM in a smaller field  $GF(2^m)$  with  $m < n$ .

Assume the large-unit is of  $m$ -bit size. Let  $\mathcal{X} = \{X_1, X_2, \dots, X_d\}$  be a set of  $d$  uniformly distributed  $n$ -bit random variables with  $X_i \in GF(2^n)$  for  $1 \leq i \leq d$ ,  $\mathcal{C} = \{C_1, \dots, C_g\}$  be a set of  $n$ -bit constants and  $\mathcal{M}$  be a set of  $\frac{n}{m}$ -dimensional binary masks of  $\mathcal{X}$  and  $\mathcal{C}$ . Now each element in  $\mathcal{X}$  and  $\mathcal{C}$  can be regarded as a  $\frac{n}{m}$ -dimensional vector over  $GF(2^m)$ . We denote some symbol or expression on  $\mathcal{X}$  and  $\mathcal{C}$  by  $T_i$ . The following two definitions introduce the GPLFM, which generalizes the definition of PLFM in [17].

**Definition 8.** Given three sets  $\mathcal{X}$ ,  $\mathcal{C}$  and  $\mathcal{M}$ , we have:

1.  $\mathcal{A}$  is an arithmetic term<sup>4</sup>, if it has only the operation of arithmetic  $\boxplus$ , e.g.,  $\mathcal{A} = T_1 \boxplus T_2 \boxplus \dots$ .
2.  $\mathcal{B}$  is a Boolean term, if it only involves Boolean operations such as OR, AND, XOR, and others, e.g.,  $\mathcal{B} = (T_1 \oplus T_2) \& T_3$ .
3.  $\mathcal{S}$  is a simple term, if it is a symbol either from  $\mathcal{X}$  or  $\mathcal{C}$ .
4.  $\Omega \cdot X$  for  $X \in \{\mathcal{A}, \mathcal{B}, \mathcal{S}\}$  is the inner product result of the term  $X$  with the binary mask  $\Omega \in \mathcal{M}$ .

**Definition 9.**  $F(X_1, X_2, \dots, X_d)$  is called a generalized pseudo-linear function modulo  $2^n$  (GPLFM) on  $\mathcal{X}$ , if it can recursively be expressed in  $\Omega \cdot X$  for  $X \in \{\mathcal{A}, \mathcal{B}, \mathcal{S}\}$  combined by the Boolean operations.

It can be easily seen that the PLFM studied in [17] forms a subset of the GPLFM, which only satisfies the conditions 1 ~ 3 in Definition 8. In our large-unit linear approximation of SNOW 2.0 in Sects. 5 and 6, we actually further generalize the GPLFM functions by considering *parallel* boolean functions, i.e., the S-boxes and multiplication over finite fields are included in our framework.

## 4.3 Algorithm for Computing the Distribution of a GPLFM

Assume the basic large-unit is of  $m$ -bit size. Let  $F(X_1, \dots, X_d)$  be a GPLFM with  $\mathcal{X}$ ,  $\mathcal{C}$  and  $\Omega \in \mathcal{M}$ , where  $X_i \in GF(2^n)$  ( $1 \leq i \leq d$ ) and the binary masks

<sup>4</sup> An arithmetic subtraction  $\boxminus$  can be substituted by  $\boxplus$  using  $X \boxminus Y = X \boxplus (\bar{Y}) \boxplus 1 \pmod{2^n}$ , where  $\bar{\cdot}$  is the complement operation.



are  $\frac{n}{m}$ -dimensional vectors. We want to calculate the distribution of  $F$  in an efficient way for some large  $n$ . Note that if  $n \geq 32$  and  $d \geq 2$ , the distribution  $p_F$  is impossible to implement in practice with the straight forward method, which needs  $2^{nd}$  operations. Further, the algorithm in [17] cannot be applied to this problem due to the inner product operation inherent in the GPLFM over a smaller field  $\text{GF}(2^m)$ . Here we propose Algorithm 1 to fulfill this task.

Our basic idea is as follows. Since each coordinate of the binary mask can only take the value of 0 or 1, it actually selects which parts of the data arguments will take effect in the approximation. According to the binary mask  $\Omega$ , we can split each variable  $X_i \in \text{GF}(2^n)$  for  $i = 1, \dots, d$  into  $\frac{n}{m}$  blocks and each block has  $m$  bits, i.e.,  $X_i = (X_i^{\frac{n}{m}}, \dots, X_i^2, X_i^1)$ , where  $X_i^j \in \text{GF}(2^m)$  for  $1 \leq j \leq \frac{n}{m}$ . Since each block of the input variable is mutually independent, the function  $F$  can be split into  $\frac{n}{m}$  sub-functions  $F_i$  ( $1 \leq i \leq \frac{n}{m}$ ), which can be evaluated over a smaller space  $\text{GF}(2^m)$ . Each sub-function  $F_i$  can be seen as a PLFM over  $\text{GF}(2^m)$ , whose distribution can be efficiently calculated by the algorithm in [17].

---

**Algorithm 1.** Computing the  $m$ -dimensional distribution of a GPLFM over  $\text{GF}(2^n)$

---

**Parameters:**

$M_1$  and  $M_2$ : two consecutive connection matrices of size  $2^m \times |Cr_{\max}|$ ;

**Processing:**

- 1: Split the function  $F$  into  $\frac{n}{m}$  sub-functions  $F_i$  according to the binary masks
- 2:  $M_1 \leftarrow \text{ComputePLFM}(F_1(0, 0, X_1^1, \dots, X_d^1), M_1, 1)$ , shown in Appendix A
- 3: **for**  $i = 2, \dots, \frac{n}{m}$  **do**
- 4: Initialize  $M_2$  with zeros
- 5: **for**  $Cr_{i-1} = (0, 0, \dots, 0)$  **to**  $(d_1^+, d_2^+, \dots, d_s^+)$  **do**
- 6: **for**  $B_{i-1} = 0$  **to**  $2^m - 1$  **do**
- 7:  $M_2 \leftarrow \text{ComputePLFM}(F_i(B_{i-1}, Cr_{i-1}, X_1^i, \dots, X_d^i), M_2, M_1[B_{i-1}] [|Cr_{i-1}|])$ ;
- 8: **end for**
- 9: **end for**
- 10:  $M_1 \leftarrow M_2 / (2^m \cdot |Cr_{\max}|)$ ;
- 11: **end for**

**Output:**  $p_F(i) = M_1[i][0] + M_1[i][1] + \dots + M_1[i][|Cr_{\max}| - 1]$

---

On the other hand, the sub-function  $F_i$ s are connected with each other by the one direction information propagation from the least significant function  $F_1$  to the most significant  $F_{\frac{n}{m}}$ , caused by the carry bit introduced by  $\boxplus$  and the output of  $F_i$ , shown in Fig. 2. Therefore, we can use a connection matrix to characterize this propagation process.

Now, we compute the distribution  $p_F$  by calculating the  $F_i$ s one-by-one from 1 to  $\frac{n}{m}$ , as depicted in Fig. 2. Here  $B_{i-1} \in \text{GF}(2^m)$  is the output of sub-function  $F_{i-1}$  and  $Cr_{i-1}$  is the carry vector of  $F_{i-1}$  that will be propagated to  $F_i$ , generated by the arithmetic terms in  $F_{i-1}$ . If there are  $s$  arithmetic terms  $\mathcal{A}_j$  ( $1 \leq j \leq s$ ) in  $F$  (thus in each  $F_i$ ), then we have  $Cr_i = (cr_i^1, \dots, cr_i^s)$ , where each  $cr_i^j$  is the corresponding local carry value of the  $\mathcal{A}_j$  ( $1 \leq j \leq s$ ) when the inputs are truncated to the  $i$ th block. Note that though each block is  $m$ -bit size, the modular addition is still calculated bit-by-bit, thus the maximum local carry value is  $d_j^+$ , where  $d_j^+$  is the number of modular additions in  $\mathcal{A}_j$

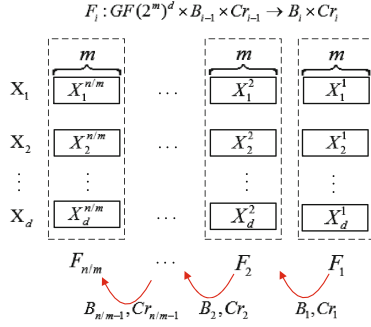


Fig. 2. The basic idea of our algorithm

( $1 \leq j \leq s$ ). Emphatically,  $Cr_i$  contains all the carry information of  $F_j$  for  $j < i$ , since the carry information is propagated from  $F_1$  to  $F_i$ . It is proved in [17] that for any arithmetic term  $\mathcal{A}_j$ , the maximum local carry value is  $d_j^+$  (the additions of carry value are not included). Similarly, denote the cardinality of  $Cr_i$  by  $|Cr_i| = ((cr_i^1 \cdot (d_2^+ + 1) + cr_i^2)(d_3^+ + 1) + cr_i^3) \cdot \dots$ , which is a one-to-one index mapping function from  $(cr_i^1, \dots, cr_i^s)$  to  $[0, \dots, |Cr_{\max}| - 1]$ , where  $|Cr_{\max}| = \prod_{j=1}^s (d_j^+ + 1)$  is the maximal possible cardinality of the carry vector  $Cr_i$ . We use a  $2^m \times |Cr_{\max}|$  matrix  $M_i$  to store the information of the  $F_j$ s ( $j \leq i$ ), where the matrix element  $M_i[B_i][|Cr_i|]$  for  $0 \leq B_i \leq 2^m - 1$  and  $0 \leq |Cr_i| \leq |Cr_{\max}| - 1$  represents the total number of the inputs  $(X_1^i, X_2^i, \dots, X_d^i)$  of  $F_i$  that result in the  $F_i$  output  $B_i$  and the carry vector  $Cr_i$ . Thus, the evaluation of  $F_i$  is converted into the computation of the matrix  $M_i$ .  $M_i$  stores all the output and carry information of  $F_i$ . Here we call it the **connection matrix**.

Now we need to evaluate the function  $F_i$  based on the connection matrix  $M_{i-1}$ , to obtain the next matrix  $M_i$ . It depends on the carry vector  $Cr_{i-1}$  and the output value  $B_{i-1}$  of  $F_{i-1}$ . For  $m > 1$ , since the sub-function  $F_i$  can be seen as a PLFM over  $GF(2^m)$ , which is recursively expressed in  $\mathcal{A}, \mathcal{B}, \mathcal{S}$ , we can use a sub-algorithm called **ComputePLFM** (Appendix A) to compute the matrix  $M_i$  ( $M_2$  in Algorithm 1) for all the possible values of  $B_{i-1}$  and  $Cr_{i-1}$ . Hereafter, when applying the Algorithm 1 we always assume that  $m > 1$ . The initial values are  $Cr_0 = (0, 0, \dots, 0)$  and  $B_0 = 0$ , i.e., the initial matrix  $M_0$  is set to be zero matrix. Our algorithm to compute the full  $m$ -dimensional distribution  $p_F = (p_F(0), p_F(1), \dots, p_F(2^m - 1))$  of a GPLFM  $F$  over  $GF(2^n)$  is shown in the Algorithm 1 diagram. Note that in Algorithm 1, only two connection matrices  $M_1$  and  $M_2$  are used to store the propagation information alternatively. The complexity analysis of Algorithm 1 is as follows. First look at the complexity of Algorithm 2 in Appendix A. Step 1 in Algorithm 2 needs a time complexity of  $O(m \cdot |Cr_{\max}| \cdot 2^d)$  from [17]. Step 2 to step 8 needs a complexity of  $O(2^m \cdot m \cdot |Cr_{\max}|)$ . Thus the complexity of Algorithm 2 is  $O(m \cdot |Cr_{\max}| \cdot (2^d + 2^m))$  and the total time complexity of our algorithm is  $O(n \cdot 2^m \cdot |Cr_{\max}|^2 \cdot (2^d + 2^m))$ .

### 5 A Key Recovery Attack on SNOW 2.0

In this section, we demonstrate a state recovery attack against SNOW 2.0. The description of SNOW 2.0 is detailed in [6]. Our new attack is based on the byte-wise linear approximation and utilizes the fast correlation attack over  $GF(2^8)$  to recover the correct initial state with much lower complexities.

#### 5.1 The Byte-Wise Linear Approximation of SNOW 2.0

In SNOW 2.0, denote the AES S-box and the Mixcolumn matrix in the  $S$  transform of FSM by  $S_R$  and  $M$  respectively. Since  $S_R$  is a 8-bit S-box, we let  $n = 32$  and  $m = 8$ . As SNOW 2.0 has two 32-bit memory registers  $R1$  and  $R2$  in the FSM, it is necessary to consider at least two consecutive steps of the FSM to eliminate these two registers in the approximation. Here we denote the two binary masks by  $\Gamma, \Lambda \in GF(2)^4$  respectively, thus the 32-bit word can be divided into 4 bytes and be regarded as a 4-dimensional vector over  $GF(2^8)$ . For example, let the binary mask  $\Gamma = (1, 0, 1, 0)$  and  $X = (x_4, x_3, x_2, x_1)$  be a 32-bit word of SNOW 2.0 in byte-wise form, thus  $\Gamma \cdot X = x_4 \oplus x_2$ . Applying  $\Gamma$  and  $\Lambda$  to  $z_t$  and  $z_{t+1}$  respectively, we have

$$\begin{aligned} \Gamma \cdot z_t &= \Gamma \cdot s_t \oplus \Gamma \cdot (R1_t \boxplus s_{t+15}) \oplus \Gamma \cdot R2_t, \\ \Lambda \cdot z_{t+1} &= \Lambda \cdot s_{t+1} \oplus \Lambda \cdot (s_{t+16} \boxplus R2_t \boxplus s_{t+5}) \oplus \Lambda \cdot S(R1_t). \end{aligned}$$

Let  $y_t = \text{Sbox}(R1_t) = (S_R(R1_t^4), S_R(R1_t^3), S_R(R1_t^2), S_R(R1_t^1))$  be the output of S-box. Since the Mixcolumn matrix  $M$  is a linear transformation over  $GF(2^8)$ , we have  $\Lambda \cdot S(R1_t) = \Lambda \cdot (My_t) = \Lambda' \cdot y_t$ . We can rewrite the above two equations as

$$\begin{aligned} \Gamma \cdot z_t &= \Gamma \cdot s_t \oplus \Gamma \cdot (\text{Sbox}^{-1}(y_t) \boxplus s_{t+15}) \oplus \Gamma \cdot R2_t, \\ \Lambda \cdot z_{t+1} &= \Lambda \cdot s_{t+1} \oplus \Lambda \cdot (s_{t+16} \boxplus R2_t \boxplus s_{t+5}) \oplus \Lambda' \cdot y_t. \end{aligned}$$

Now we have a new byte-wise linear approximation of SNOW 2.0, depicted in Fig. 3. Note that in Fig. 3, the  $S$  transform of the FSM is dissected to have an efficient approximation. Here we use two linear approximations

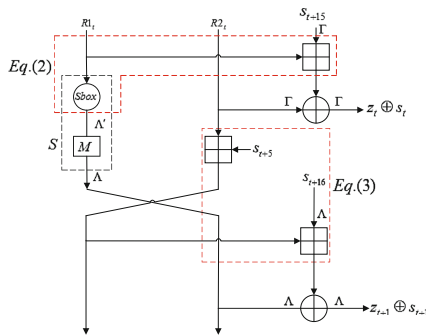


Fig. 3. The linear approximation of the FSM in SNOW 2.0

$$\Gamma \cdot (\text{Sbox}^{-1}(y_t) \boxplus s_{t+15}) = \Gamma \cdot s_{t+15} \oplus \Lambda' \cdot y_t \oplus \mathbf{N}_1(t), \tag{2}$$

$$\Lambda \cdot (s_{t+16} \boxplus s_{t+5} \boxplus R2_t) = \Lambda \cdot s_{t+16} \oplus \Lambda \cdot s_{t+5} \oplus \Lambda \cdot R2_t \oplus \mathbf{N}_2(t). \tag{3}$$

The linear approximation (3) is a GPLFM, thus we can adopt Algorithm 1 to compute the distribution of  $\mathbf{N}_2(t)$ . For the linear approximation (2), it is not a GPLFM in Definitions 8 and 9, thus we cannot use Algorithm 1 directly. But note that the four  $S_{RS}$  do not affect the independency among the bytes of  $y_t$ , thus we can revise Algorithm 1 to compute the distribution of  $\mathbf{N}_1(t)$  as shown in Algorithm 3.

---

**Algorithm 3.** Computing the Distribution in Eq. (2) over  $\text{GF}(2^8)$

---

**Parameters:**  
 $\Gamma = (\Gamma_4, \Gamma_3, \Gamma_2, \Gamma_1)$ ,  $s_t = (s_t^4, s_t^3, s_t^2, s_t^1)$ ,  $y_t = (y_t^4, y_t^3, y_t^2, y_t^1)$ ,  $\Lambda' = (\Lambda'_4, \Lambda'_3, \Lambda'_2, \Lambda'_1)$ ;

**Processing:**

- 1: Compute  $\Gamma_1(S_R^{-1}(y_t^1) + s_{t+15}^1) \oplus \Gamma_1 s_{t+15}^1 \oplus \Lambda'_1 y_t^1$  and store in  $M_1$
- 2: **for**  $i = 2, \dots, 4$  **do**
- 3:     Initialize  $M_2$  with zeros
- 4:     **for**  $y_t^i = 0, \dots, 255$  and  $s_{t+15}^i = 0, \dots, 255$  **do**
- 5:         **for**  $Cr_{i-1} = 0, 1$  **do**
- 6:             **for**  $B_{i-1} = 0, \dots, 255$  **do**
- 7:                  $B_i \leftarrow B_{i-1} \oplus \Gamma_i(S_R^{-1}(y_t^i) + s_{t+15}^i) \oplus \Gamma_i s_{t+15}^i \oplus \Lambda'_i y_t^i$ ;
- 8:                  $Cr_i \leftarrow (S_R(y_t^i) + s_{t+15}^i + Cr_{i-1})/2^8$ ;
- 9:                  $M_2[B_i][Cr_i] \leftarrow M_2[B_i][Cr_i] + M_1[B_{i-1}][Cr_{i-1}]$ ;
- 10:      $M_1 \leftarrow M_2/(2^8 \times 2)$ ;
- 11: **Output:** The distribution  $p_i = M_1[i][0] + M_2[i][1]$  for each  $0 \leq i \leq 255$

---

The time complexity of computing the distribution of  $\mathbf{N}_1(t)$  has dropped from  $2^{64}$  to  $2^{26.58}$ , which is a large improvement compared to the straightforward method. We have searched over all the different binary masks over  $\text{GF}(2)^4$  and found that when  $\Gamma = \Lambda$ , these two linear approximations will have larger SEIs. Thus the sum of  $\Gamma \cdot (z_t \oplus z_{t+1})$  can be expressed as

$$\Gamma \cdot (z_t \oplus z_{t+1}) = \Gamma \cdot s_t \oplus \Gamma \cdot s_{t+1} \oplus \Gamma \cdot s_{t+5} \oplus \Gamma \cdot s_{t+15} \oplus \Gamma \cdot s_{t+16} \oplus \mathbf{N}(t), \tag{4}$$

where  $\mathbf{N}(t) = \mathbf{N}_1(t) \oplus \mathbf{N}_2(t)$  is the folded noise variable introduced by the above two linear approximations, whose distribution can be computed by the convolution of the above two noise distributions. We have searched all the possible  $\Gamma$  and  $\Lambda$  and found that the *strongest* linear approximation of the FSM is as follows. When  $\Gamma = \Lambda = (1, 0, 1, 0)$ , the distribution of  $\mathbf{N}(t)$  has the value of SEI as  $\Delta(\mathbf{N}(t)) = 2^{-43.23}$ . Observe that given a noise distribution  $\text{Pr}\{\mathbf{N}(t)\}$ , the SEI can be *precisely* computed by Definition 2. Now, we have constructed the byte-wise linear approximation, i.e., Eq. (4), of SNOW 2.0. Next, we will use this linear approximation to recover the initial state of SNOW 2.0.

### 5.2 Fast Correlation Attack on SNOW 2.0

Now we apply the fast correlation attack over  $\text{GF}(2^8)$  to SNOW 2.0 to recover the initial state of the LFSR. Let the LFSR state be  $(s_{t+15}, \dots, s_t) \in \text{GF}(2^{32})^{16}$ , here

the LFSR is interpreted as a 64-byte LFSR over  $\text{GF}(2^8)$ , i.e.,  $(s_{t+15}^4, s_{t+15}^3, s_{t+15}^2, s_{t+15}^1, \dots, s_t^4, s_t^3, s_t^2, s_t^1) \in \text{GF}(2^8)^{64}$ . With the feedback polynomial we can express the linear approximation (4) in the initial state form as  $\Gamma \cdot (z_t \oplus z_{t+1}) = \Psi_t \cdot (s_{t+15}^4, s_{t+15}^3, s_{t+15}^2, s_{t+15}^1, \dots, s_t^4, s_t^3, s_t^2, s_t^1) \oplus N(t)$ , where  $\Psi_t \in \text{GF}(2^8)^{64}$  is the derived recursion of the LFSR.

For the decoding algorithm, we apply the precomputation algorithm in Sect. 3 to generate the parity checks with the parameters  $l = 64, l' = 17, k = 4$ , which are the best parameters we have found in terms of the total complexities. The distribution of the folded noise variables  $\mathbf{N}(t_{i_1}) \oplus \mathbf{N}(t_{i_2}) \oplus \mathbf{N}(t_{i_3}) \oplus \mathbf{N}(t_{i_4})$  can be computed by the applications of the convolutional operation twice. The SEI of this new distribution is found to be  $2^{-177.3}$ . Using 4 lists in the  $k$ -tree algorithm, we get about  $m_k = \beta^{1+\log k} = 2^{184.86}$  parity check equations. By Theorem 6, the data complexity is  $N = 2^{188.95}$  and the time/memory complexities of pre-processing stage are  $\beta k 2^{n(l-l')/(1+\log k)} = 2^{188.95}$ . Second, we perform the online decoding algorithm on the new code  $\mathcal{C}_2$  of the code length  $2^{188.95}$ . With a computational complexity of  $n(m_k + l'n2^{l'n}) + 2^n 2^{l'n} = 2^{187.86}$ , we can recover the  $17 \cdot 8 = 136$  bits of the initial state of LFSR, the other bits can be recovered with a much lower complexity.

Therefore, the final time/memory/data/pre-computation complexities are all upper bounded by  $2^{186.95}$ , which is more than  $2^{25}$  times better than the best previous result at Asiacrypt 2008 [13]. This obviously confirms the superiority of our new techniques.

## 6 An Improved Key Recovery Attack on SNOW 2.0

Recall in Sect. 4, we use the  $\frac{n}{m}$ -dimensional binary linear masks. Here we generalize this definition by making each component  $\omega_i \in \text{GF}(2^m)$  rather than  $\text{GF}(2)$ , i.e., changing the 0/1 coefficients to finite field coefficients, i.e., expressing  $X$  by  $X = (x_{\frac{n}{m}}, \dots, x_1) \in \text{GF}(2^m)^{\frac{n}{m}}$  with  $x_i \in \text{GF}(2^m)$  and denote the inner product by  $\Omega \cdot X = \omega_{\frac{n}{m}} x_{\frac{n}{m}} \oplus \dots \oplus \omega_1 x_1 \in \text{GF}(2^m)$ , where  $\omega_i x_i$  is the multiplication over  $\text{GF}(2^m)$ .  $\Omega$  is called the linear mask over  $\text{GF}(2^m)$  of  $X$ . Now these new nonlinear functions are not GPLFM in Definitions 8 and 9, for we have changed the linear mask from  $\text{GF}(2)$  to  $\text{GF}(2^m)$ . Thus we cannot apply the Algorithm 1 to compute the distributions of these new functions directly. Instead, we further revise Algorithm 1 to efficiently compute the distributions of such functions in the following analysis of SNOW 2.0.

### 6.1 Linear Approximations of SNOW 2.0 Over $\text{GF}(2^8)$

The process of finding the linear approximations of SNOW 2.0 is nearly the same as in Sect. 5. In order to find the best linear masks over  $\text{GF}(2^8)$ , let us take a closer look at the details of the  $S$  permutation in FSM. Let  $A' = (A'_4, A'_3, A'_2, A'_1)$  denote the linear mask over  $\text{GF}(2^8)$  of the 4 byte outputs of the Sbox, where the multiplication is computed in  $\text{GF}(2^8)$  defined by the AES Mixcolumn. Then, we can express  $A \cdot S(\omega)$  as

$$\underbrace{(A_1, A_2, A_3, A_4)}_{\left( \begin{array}{cccc} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{array} \right)} \left( \begin{array}{c} S_R(\omega_1) \\ S_R(\omega_2) \\ S_R(\omega_3) \\ S_R(\omega_4) \end{array} \right) = (A'_1, A'_2, A'_3, A'_4) \left( \begin{array}{c} S_R(\omega_1) \\ S_R(\omega_2) \\ S_R(\omega_3) \\ S_R(\omega_4) \end{array} \right),$$

where  $A_i, A'_i \in \text{GF}(2^8)$  for  $1 \leq i \leq 4$ . We adopt the field  $\text{GF}(2^8)$  as that defined by the AES Mixcolumn and assume that the linear masks over  $\text{GF}(2^8)$  are also defined in this field. Here we still use the two linear approximations over  $\text{GF}(2^8)$ , i.e., Eqs. (2) and (3), but the linear masks  $\Gamma, \Lambda$  are 4-dimensional vectors over  $\text{GF}(2^8)$ . The algorithm to compute the distribution of Eq. (2) is similar as before, except that  $\Gamma = (\Gamma_4, \Gamma_3, \Gamma_2, \Gamma_1) \in \text{GF}(2^8)^4$  rather than  $\text{GF}(2)^4$ , shown in Algorithm 3. The distribution of  $\mathbf{N}_2(t)$  with the linear mask  $\Lambda \in \text{GF}(2^8)^4$  can be computed by Algorithm 4 in Appendix B. The time complexity is  $3 \cdot (2^8)^3 \cdot 2^8 \approx 2^{33.58}$ , while the straightforward method needs a complexity of  $2^{96}$ .

Now we describe how to find the linear masks  $\Lambda, \Gamma$  that satisfy Eqs. (2) and (3) with large SEIs. Our strategy is to limit the Hamming weights of the linear masks  $\Lambda$  and  $\Lambda'$  over  $\text{GF}(2^8)$ . Denote the Hamming weight of a vector by  $wt(\cdot)$ , thus  $wt(\Lambda')$  determines the number of active S-boxes in the S-box ensemble  $S$ . In the experiments, we found that the SEI of  $\mathbf{N}_2(t)$  is dependent on  $wt(\Lambda)$ . The lower value of  $wt(\Lambda)$ , the larger  $\Delta(\mathbf{N}_2(t))$ . We have searched all the linear masks  $\Lambda, \Lambda'$  with  $wt(\Lambda) \leq 3$  and  $wt(\Lambda') \leq 3$  and found 255 different linear masks having the same largest value of  $\Delta(\mathbf{N}(t))$ . For example, when  $\Gamma = \Lambda = (0x00, 0x01, 0x00, 0x03)$ , we get the best linear approximation with the noise distribution  $\mathbf{N}(t)$  having a SEI of  $\Delta(\mathbf{N}(t)) = 2^{-29.23}$ .

Please see Appendix C for unifying the two fields before decoding. Then we launch the fast correlation attack over  $\text{GF}(2^8)$  with the parameters  $n = 32, l = 64, l' = 19, k = 4$ . The data complexity is  $N \approx 2^{163.59}$ , while the time/memory complexities of the pre-computation is  $2^{163.59}$ . After pre-computation, we can acquire about  $m_k = 2^{124.79}$  parity checks. For the online decoding algorithm, the time complexity is  $2^{162.52}$  with the above parameter configuration. Note that here all the complexities are below  $2^{164.15} \approx 2^{162.52} + 2^{163.59}$ , which are considerably reduced compared to the binary mask case. The reason is that the linear masks with the finite field coefficients greatly extend the searching space and can well exploit the algebraic structure of the two finite fields (one defined in a tower manner in the LFSR and the other in the Mixcolumn) inherent in SNOW 2.0.

## 6.2 Experimental Results

We have verified each step of our new techniques in simulations to support the theoretical analysis. We have used the GNU Multiple Precision Arithmetic Library in Linux system to verify the exact distribution of each linear approximation that has been found, thus the SEI of our large-unit linear approximation is precisely evaluated *without* any precision error. Then we have run extensive experiments on a small-scale version of SNOW 2.0, called s-SNOW 2.0 described in Appendix D, that have verified our approach.

We have computed the 4-bit linear approximation of the s-SNOW 2.0 with Algorithm 1 in theory and verified it in practice. Then we executed the experiments on the decoding algorithm in Sect. 3.4. We randomly fixed the values of 60 bits of the initial state of the LFSR and tried to recover the remaining 20-bit by our method. The chosen parameters are  $l' = 20, m_k = 2^{15.39}$ . We first use s-SNOW 2.0 to generate  $2^{17}$  keystream bits  $z_t$  for a randomly generated 80-bit initial state. Then we store  $z_t$  and  $s_t$  in two arrays for  $t = 1, \dots, 2^{17}$ . Thus we can construct  $2^{17}$  parity checks  $I^{(t)} = \Gamma \cdot (z_t \oplus z_{t+1}) \oplus \Gamma \cdot s_t \oplus \Gamma \cdot s_{t+1} \oplus \Gamma \cdot s_{t+3} \oplus \Gamma \cdot s_{t+4} \oplus \Gamma \cdot s_t$ , for  $t = 0, \dots, 2^{17} - 1$ . Second, for each parity check  $I^t$ , we use the LFSR feedback polynomial to express each  $s_t$  for  $t > 4$  as a linear combination of the LFSR initial state variables. Now we get  $2^{17}$  parity checks only containing  $(s_0, s_1, s_2, s_3, s_4)$  after fixing 60-bit in the state. Third, we divide the 4-bit linear approximation  $I^{(t)}$  into four bitwise linear approximations, i.e.,  $I_1^{(t)} = \langle (0, 0, 0, 1), I^t \rangle, I_2^{(t)} = \langle (0, 0, 1, 0), I^t \rangle, I_3^{(t)} = \langle (0, 1, 0, 0), I^t \rangle, I_4^{(t)} = \langle (1, 0, 0, 0), I^t \rangle$ . For each possible 20-bit initial state, we use FWT to compute the correlations  $c(I_i^{(t)})$  for  $i = 1, \dots, 4$ . Fourth, we apply Lemma 4 to compute the SEI of  $p_I$  for each possible initial state. Then we use the SEI to distinguish the correct initial state. We ran the experiments for randomly generated values 100 times with *different* fixed values at *different* positions in the LFSR state, and we found that the correct key always ranks in the top 10 in the candidates list. These 10 candidates have  $\Delta(p_I)$  around  $2^{-10.6}$ , which verified the theoretical analysis.

Therefore, the experimental results have provided a solid support to our decoding algorithm and we can get reliable predictions from our theoretical analysis when the simulation is infeasible to perform. Further, our decoding method is essentially the LLR method in linear cryptanalysis, whose validity can be guaranteed by the theory of linear cryptanalysis.

## 7 Conclusions

In this paper, we have developed two new cryptanalytic tools to bridge the gap between the widely used primitives employing word-based LFSRs and the current mainstream bitwise fast correlation attacks. The first one, a formal framework for fast correlation attacks over extension fields with a thorough theoretical analysis, is the *first* comprehensive answer to the corresponding open problem in the field of correlation attacks. The second technique, serving as a basis to the first one, allows to efficiently compute the bias distributions of large-unit linear approximations of the flexibly derived GPLFM, which includes all the previously studied topics in the open literature in an unified framework. The size of the data unit is usually chosen according to the structure of the underlying primitive and the building blocks, which greatly extends the freedom of the adversary in the cryptanalysis of many symmetric-key primitives. As an application, we adapted these two techniques to SNOW 2.0, an ISO/IEC 18033-4 standard and a benchmark stream cipher in the European eSTREAM project, and achieved the best key recovery attacks known so far. The new methods are

generic and are applicable to other symmetric-key primitives as well, e.g., SNOW 3G, Sosemanuk, Dragon, and some CAESAR candidates. It is our future work to study the large-unit linear approximations of these primitives and launch various attacks accordingly.

**Acknowledgments.** This work is supported by the National Grand Fundamental Research 973 Program of China (Grant No. 2013CB338002), and the programs of the National Natural Science Foundation of China (Grant No. 60833008, 60603018, 61173134, 91118006, 61272476). The third author was supported in part by the Research Council KU Leuven: a senior postdoctoral scholarship SF/14/010 linked to the GOA TENSE (GOA/11/007).

## A Diagrams of the Invoked Algorithm 2

---

**Algorithm 2.** ComputePLFM( $F_i, M, D_{i-1}$ )

---

**Parameters:**  $D_{i-1}$ : a probability of  $F_{i-1}$  that results in  $B_{i-1}, Cr_{i-1}$

**Temporary Variable:**  $B_i, Cr_i$ : the output and carry vector of  $F_i$

$\mathbf{v} = (v_0, \dots, v_{|Cr_{\max}|-1})$ : a  $|Cr_{\max}|$ -dimensional vector

**Processing:**

- 1: Submit  $F_i$  to the precomputed algorithm in [17] and get  $2m$  matrices  $M_{(B_i)_t|t}$ ;
- 2: **for**  $B_i = 0$  **to**  $2^m - 1$  **do**
- 3:  $\mathbf{v} = (\prod_{t=m-1}^0 M_{(B_i)_t|t}) \times (1, 0, \dots, 0)^T$ ;
- 4: **for**  $j = 0$  **to**  $|Cr_{\max}| - 1$  **do**
- 5:  $M[B_i][j] = M[B_i][j] + (v_j/2^{md})D_{i-1}$ ;

**Return:**  $M$

---

In the diagram of Algorithm 2,  $(B_i)_t$  is the  $t$ -th bit of  $B_i$  and the matrices in [17] have the similar meaning with our connection matrix, which store the carry information for each bit.

## B Computing the Distribution in Eq. (3) over GF( $2^8$ )

---

**Algorithm 4.** Computing the distribution of  $N_2(t)$

---

**Parameters:**

$\Lambda = (\Lambda_4, \Lambda_3, \Lambda_2, \Lambda_1) \in \text{GF}(2^8)^4$ ,  $s_t = (s_t^4, s_t^3, s_t^2, s_t^1)$ ,  $R_2t = (R_2t^4, R_2t^3, R_2t^2, R_2t^1)$ ;

**Processing:**

- 1: Compute  $A_1(s_{t+16}^1 \boxplus s_{t+5}^1 \boxplus R_2t^1) \oplus A_1s_{t+16}^1 \oplus A_1s_{t+5}^1 \oplus A_1R_2t^1$  and store in  $M_1$
  - 2: **for**  $i = 2, \dots, 4$  **do**
  - 3: Initialize  $M_2$  with zeros
  - 4: **for**  $s_{t+16}^i = 0, \dots, 255$  and  $s_{t+5}^i = 0, \dots, 255$  and  $R_2t^i = 0, \dots, 255$  **do**
  - 5: **for**  $Cr_{i-1} = 0, 1, 2$  **do**
  - 6: **for**  $B_{i-1} = 0, \dots, 255$  **do**
  - 7:  $B_i \leftarrow B_{i-1} \oplus A_i(s_{t+16}^i + s_{t+5}^i + R_2t^i) \oplus A_i s_{t+16}^i \oplus A_i s_{t+5}^i \oplus A_i R_2t^i$ ;
  - 8:  $Cr_i \leftarrow (s_{t+16}^i + s_{t+5}^i + R_2t^i + Cr_{i-1})/2^8$ ;
  - 9:  $M_2[B_i][|Cr_i|] \leftarrow M_2[B_i][|Cr_i|] + M_1[B_{i-1}][|Cr_{i-1}|]$ ;
  - 10:  $M_1 \leftarrow M_2/(2^8 \times 3)$ ;
  - 15: **Output:**  $p_i = M_1[i][0] + M_2[i][1] + M_2[i][2]$  for each  $0 \leq i \leq 255$
-



### C Unifying the Two Fields

Note that in Eq. (4), the mask  $\Gamma = (0x00, 0x01, 0x00, 0x03)$  is defined over the Mixcolumn field  $GF(2^8)$ , which is different from the corresponding field of the LFSR. We need to first unify the two fields for an efficient decoding, otherwise there will be the folded noise introduced by whether xoring the two field constants or not. Here we adopt the following routine to solve this problem. To facilitate the decoding phase, we first find an equivalent representation of the LFSR part theoretically so that it is defined over the new  $GF(2^{32})$  field, which is derived as follows.

We first substitute the low-level  $GF(2^8)$  field of the LFSR defined by  $x^8 + x^7 + x^5 + x^3 + 1$  (field constant  $0xa9$ ) with the  $GF(2^8)$  field defined in Mixcolumn by  $x^8 + x^4 + x^3 + x + 1$  (field constant  $0x1b$ ), and then randomly select a primitive polynomial of degree 4 over this new field to construct the new  $GF(2^{32})$  field. Let  $\{s_i\}_{i=0}^\infty$  be the sequence generated by the LFSR defined over the original  $GF(2^{32})$  field in SNOW 2.0, our observation is that the sequence itself is just a string of bits and is independent of the definition of the underlying field, thus once one segment of sufficient length of the sequence is produced from a LFSR over the field associated with one definition, we can use the classical Berlekamp-Massey algorithm [15] over the field with another definition to reconstruct the LFSR feedback polynomial over the latter field and as a by product, the equivalent state conversion relation between the two field definitions can be obtained. Note that the LFSR sequence  $\{s_i\}_{i=0}^\infty$  in SNOW 2.0 is primitive, thus the new generated LFSR over the new  $GF(2^{32})$  field is also of length 16. Compared with the other parts of our attack, the complexity of computing the equivalent representation of the LFSR part defined in the new field is negligible. The overall complexity of our attack is dominated by the complexity of the decoding phase.

### D A Small Scale Version of SNOW 2.0

The LFSR consists of 5 units and each unit is a 16-bit word in  $GF(2^{16})$ . The feedback polynomial is  $\pi(x) = \alpha x^5 + \alpha^{-1}x^3 + x^2 + 1 \in GF(2^{16})[x]$ , where  $\alpha$  is a root of  $x^4 + \beta^{10}x^3 + \beta^6x^2 + x + \beta^{11}$ , and  $\beta$  is a root of  $x^4 + x + 1 \in GF(2)[x]$ . The FSM has two 16-bit registers  $R1$  and  $R2$  updated by  $R1_{t+1} = (s_{t+3} \boxplus R2_t) \bmod 2^{16}$  and  $R2_{t+1} = S(R1_t)$ . The function  $S$  is composed of four parallel Nibble S-boxes followed by the following MixColumn.

$$S(s_i) = \begin{pmatrix} 1 & 4 \\ 4 & 1 \end{pmatrix} \begin{pmatrix} S_R(s_i^1) & S_R(s_i^3) \\ S_R(s_i^2) & S_R(s_i^4) \end{pmatrix},$$

where  $S_R$  is the Nibble S-box in Small AES [20]. The output of FSM is  $F_t = (s_{t+4} \boxplus R1_t) \oplus R2_t$ . The generated keystream is  $z_t = F_t \oplus s_t$ .

### References

1. Baignères, T., Junod, P., Vaudenay, S.: How far can we go beyond linear cryptanalysis? In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 432–450. Springer, Heidelberg (2004)

2. Berbain, C., et al.: SOSEMANUK, a fast software-oriented stream cipher. In: Robshaw, M., Billet, O. (eds.) *New Stream Cipher Designs*. LNCS, vol. 4986, pp. 98–118. Springer, Heidelberg (2008)
3. Canteaut, A.: Fast correlation attacks against stream ciphers and related open problems. In: *2005 IEEE Information Theory Workshop on Theory and Practice in Information-Theoretic Security*, pp. 49–54 (2005)
4. Chepyzhov, V.V., Johansson, T., Smeets, B.: A simple algorithm for fast correlation attacks on stream ciphers. In: Schneier, B. (ed.) *FSE 2000*. LNCS, vol. 1978, pp. 181–195. Springer, Heidelberg (2001)
5. 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)
6. Ekdahl, P., Johansson, T.: A new version of the stream cipher SNOW. In: Nyberg, K., Heys, H. (eds.) *SAC 2002*. LNCS, vol. 2595, pp. 47–61. Springer, Berlin (2003)
7. Englund, H., Maximov, A.: Attack the dragon. In: Maitra, S., Veni Madhavan, C.E., Venkatesan, R. (eds.) *INDOCRYPT 2005*. LNCS, vol. 3797, pp. 130–142. Springer, Heidelberg (2005)
8. ETSI/SAGE. Specification of the 3GPP confidentiality and integrity algorithms uea2 & uia2. In: *Document 2: SNOW 3G Specification, version 1.1, September 2006*. <http://www.3gpp.org/ftp/>
9. Hermelin, M., Cho, J.Y., Nyberg, K.: Multidimensional extension of Matsui’s algorithm 2. In: Dunkelman, O. (ed.) *FSE 2009*. LNCS, vol. 5665, pp. 209–227. Springer, Heidelberg (2009)
10. Jönsson, F., Johansson, T.: Correlation attacks on stream ciphers over  $GF(2^n)$ . In: *2001 IEEE International Symposium on Information Theory-ISIT 2001*, p. 140 (2001)
11. Johansson, T., Jönsson, F.: Fast correlation attacks through reconstruction of linear polynomials. In: Bellare, M. (ed.) *CRYPTO 2000*. LNCS, vol. 1880, pp. 300–315. Springer, Heidelberg (2000)
12. Jönsson, F.: Some results on fast correlation attacks. Ph.D. thesis, Lund University, Sweden (2002)
13. Lee, J.-K., Lee, D.-H., Park, S.: Cryptanalysis of SOSEMANUK and SNOW 2.0 using linear masks. In: Pieprzyk, J. (ed.) *ASIACRYPT 2008*. LNCS, vol. 5350, pp. 524–538. Springer, Heidelberg (2008)
14. Lu, Y., Vaudenay, S.: Faster correlation attack on Bluetooth keystream generator E0. In: Franklin, M. (ed.) *CRYPTO 2004*. LNCS, vol. 3152, pp. 407–425. Springer, Heidelberg (2004)
15. Massey, J.L.: Shift-register synthesis and BCH decoding. *IEEE Trans. Inf. Theor.* **IT-15**(1), 122–127 (1969)
16. Matsui, M.: Linear cryptanalysis method for DES cipher. In: Helleseth, T. (ed.) *EUROCRYPT 1993*. LNCS, vol. 765, pp. 386–397. Springer, Heidelberg (1994)
17. Maximov, A., Johansson, T.: Fast computation of large distributions and its cryptographic applications. In: Roy, B. (ed.) *ASIACRYPT 2005*. LNCS, vol. 3788, pp. 313–332. Springer, Heidelberg (2005)
18. Meier, W.: Fast correlation attacks: methods and countermeasures. In: Joux, A. (ed.) *FSE 2011*. LNCS, vol. 6733, pp. 55–67. Springer, Heidelberg (2011)
19. Meier, W., Staffelbach, O.: Fast correlation attacks on certain stream ciphers. *J. Cryptology* **1**, 159–176 (1989)
20. Musa, M.A., Schaefer, E.F., Wedig, S.: A simplified AES algorithm and its linear and differential cryptanalyses. *Cryptologia* **27**(2), 148–177 (2003)

21. Nyberg, K., Hermelin, M.: Multidimensional Walsh transform and a characterization of bent functions. In: 2007 IEEE Information Theory Workshop on Information Theory for Wireless Networks, pp. 1–4 (2007)
22. Nyberg, K., Wallén, J.: Improved linear distinguishers for SNOW 2.0. In: Robshaw, M. (ed.) FSE 2006. LNCS, vol. 4047, pp. 144–162. Springer, Heidelberg (2006)
23. Shannon, C.E.: A mathematical theory of communication. *ACM Sigmobility Mob. Comput. Commun. Rev.* **5**(1), 3–55 (2001)
24. Wagner, D.: A generalized birthday problem. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 288–304. Springer, Heidelberg (2002)
25. Yarlagadda, R.K.R., Hershey, J.E.: *Hadamard Matrix Analysis and Synthesis with Applications to Communications and Signal/Image Processing*. Kluwer Academic Publishers, Boston (1997)

# Cryptanalysis of Full Sprout

Virginie Lallemand and María Naya-Plasencia<sup>(✉)</sup>

Inria, Bordeaux, France  
maria.naya.plasencia@gmail.com

**Abstract.** A new method for reducing the internal state size of stream cipher registers has been proposed in FSE 2015, allowing to reduce the area in hardware implementations. Along with it, an instantiated proposal of a cipher was also proposed: Sprout. In this paper, we analyze the security of Sprout, and we propose an attack that recovers the whole key more than  $2^{10}$  times faster than exhaustive search and has very low data complexity. The attack can be seen as a divide-and-conquer evolved technique, that exploits the non-linear influence of the key bits on the update function. We have implemented the attack on a toy version of Sprout, that conserves the main properties exploited in the attack. The attack completely matches the expected complexities predicted by our theoretical cryptanalysis, which proves its validity. We believe that our attack shows that a more careful analysis should be done in order to instantiate the proposed design method.

**Keywords:** Stream cipher · Cryptanalysis · Lightweight · Sprout

## 1 Introduction

The need of low-cost cryptosystems for several emerging applications like RFID tags and sensor networks has drawn considerable attention to the area of lightweight primitives over the last years. Indeed, those new applications have very limited resources and necessitate specific algorithms that ensure a perfect balance between security, power consumption, area size and memory needed. The strong demand from the community (for instance, [5]) and from the industry has led to the design of an enormous amount of promising such primitives, with different implementation features. Some examples are PRESENT [6], CLEFIA [26], KATAN/KTANTAN [11], LBlock [28], TWINE [27], LED [17], PRINCE [7], KLEIN [16], Trivium [10] and Grain [18].

The need for clearly recommended lightweight ciphers requires that the large number of these potential candidates be narrowed down. In this context, the need for a significant cryptanalysis effort is obvious. This has been proved by the big number of security analyses of the previous primitives that has appeared (to cite a few: [1, 13, 15, 19–21, 24, 25]).

---

Partially supported by the French Agence Nationale de la Recherche through the BLOC project under Contract ANR-11-INS-011.

Stream ciphers are good candidates for lightweight applications. One of the most important limitations to their lightweight properties is the fact that to resist time-memory-data trade-off attacks, the size of their internal state must be at least twice the security parameter.

In FSE 2015, Armknecht et al. proposed [3,4] a new construction for stream ciphers designed to scale down the area required in hardware. The main intention of their paper is to revisit the common rule to resist against time-memory-data trade-off attacks, and reduce the minimal internal state of stream ciphers. To achieve this goal, the authors decided to involve the secret key not only in the initialization process but also in the keystream generation phase. To support this idea, an instance of this new stream cipher design is specified. This instance is based on the well studied stream cipher Grain128a [2] and as such has been named **Sprout**. In this paper we analyze the security of this cipher, and present an attack on the full version that allows the attacker to recover the whole 80-bit key with a time complexity of  $2^{69.39}$ , that is  $2^{10}$  times faster than exhaustive search and needs very few bits of keystream. Our attack exploits an evolved divide-and-conquer idea.

In order to verify our theoretical estimation of the attack, we have implemented it on a toy version of Sprout that maintains all the properties that we exploit during the attack, and we have corroborated our predicted complexities, being able then to validate our cryptanalysis.

This paper is organised as follows: we first recall the specifications of the stream cipher Sprout in Sect. 2, and then describe our attack in Sect. 3. We provide the details of the implementation that has verified the validity of our attack in Sect. 4. Section 5 provides a discussion on how the attack affects the particular instantiation and the general idea.

## 2 Description of Sprout

In [3] the authors aim at reducing the size of the internal state used in stream ciphers while resisting to time-data-memory trade-off (TMDTO) attacks. They propose to this purpose a new design principle for stream ciphers such that the design paradigm of long states can be avoided. This is done by introducing a state update function that depends on a fixed secret key. The designers expect a minimum time effort equivalent to an exhaustive search of the key for an attacker to lead an attack, since she has to determine the key prior to realise the TMDTO.

**Sprout** is the concrete instantiation of this new type of stream ciphers developed in [3]. It has an IV and a key size of 80 bits. Based on Grain128a, this keystream generator is composed of two feedback shift registers of 40 bits, one linear (the LFSR) and one non-linear (the NLFSR), an initialization function and an update function, both key-dependent, and of an output function that produces the keystream (see Fig. 1). The maximal keystream length that can be produced under the same IV is  $2^{40}$ .

We first recall some notations that will be used in the following:

- $t$  clock-cycle number
- $L^t = (l_0^t, l_1^t, \dots, l_{39}^t)$  state of the LFSR at clock  $t$

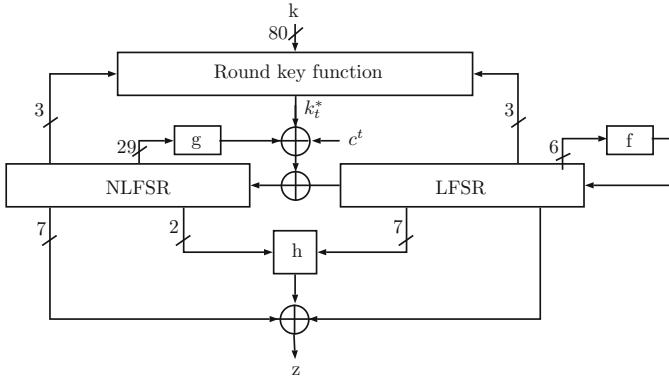


Fig. 1. Sprout KeyStream generation

- $N^t = (n_0^t, n_1^t, \dots, n_{39}^t)$  state of the NLFSR at clock  $t$
- $iv = (iv_0, iv_1, \dots, iv_{69})$  initialisation vector
- $k = (k_0, k_1, \dots, k_{79})$  secret key
- $k_t^*$  round key bit generated during the clock-cycle  $t$
- $z_t$  keystream bit generated during the clock-cycle  $t$
- $c_t$  round constant at clock  $t$  (generated by a counter).

A **counter** is set to determine the key bit to use at each clock and also to update the non linear register. More specifically, the counter is made up of 9 bits that count until 320 in the initialisation phase, and then count in loop from 0 to 79 in the keystream generation phase. The fourth bit ( $c_4^t$ ) is used in the feedback bit computation of the NLFSR.

The 40-bit **LFSR** uses the following retroaction function, that ensures maximal period:  $l_{39}^{t+1} = f(L^t) = l_0^t + l_5^t + l_{15}^t + l_{20}^t + l_{25}^t + l_{34}^t$ .

The remaining state is updated as  $l_i^{t+1} = l_{i+1}^t$  for  $i$  from 0 to 38.

The **NLFSR** is also 40-bit long and uses a feedback computed by:

$$\begin{aligned}
 n_{39}^{t+1} &= g(N^t) + k_t^* + l_0^t + c^t \\
 &= k_t^* + l_0^t + c^t + n_0^t + n_{13}^t + n_{19}^t + n_{35}^t + n_{39}^t + n_2^t n_{25}^t + n_3^t n_5^t + n_7^t n_8^t + n_{14}^t n_{21}^t \\
 &\quad + n_{16}^t n_{18}^t + n_{22}^t n_{24}^t + n_{26}^t n_{32}^t + n_{33}^t n_{36}^t n_{37}^t n_{38}^t + n_{10}^t n_{11}^t n_{12}^t + n_{27}^t n_{30}^t n_{31}^t,
 \end{aligned}$$

where  $k_t^*$  is defined as:

$$\begin{aligned}
 k_t^* &= k_t, 0 \leq t \leq 79 \\
 k_t^* &= (k_{t \bmod 80}) \times (l_4^t + l_{21}^t + l_{37}^t + n_9^t + n_{20}^t + n_{29}^t), t \geq 80
 \end{aligned}$$

The remaining state is updated as  $n_i^{t+1} = n_{i+1}^t$  for  $i$  from 0 to 38.

In the following, we name by  $\sum l$  the sum of the LFSR bits that intervene in  $k_t^*$  when  $t \geq 80$  (i.e.  $\sum l \triangleq l_4^t + l_{21}^t + l_{37}^t$ ) and by  $\sum n \triangleq n_9^t + n_{20}^t + n_{29}^t$  its NLFSR counterpart, leading to the following equivalent definition of  $k_t^*$  when  $t \geq 80$ :

$$k_t^* = (k_{t \bmod 80}) \times (\sum l + \sum n)$$

**Update and Output Function.-** The output of the stream cipher is a boolean function computed from bits of the LFSR and of the NLFSR. The nonlinear part of it is defined as:

$$h(x) = n_4^t l_6^t + l_8^t l_{10}^t + l_{32}^t l_{17}^t + l_{19}^t l_{23}^t + n_4^t n_{38}^t l_{32}^t$$

And the output bit is given by:

$$z_t = n_4^t l_6^t + l_8^t l_{10}^t + l_{32}^t l_{17}^t + l_{19}^t l_{23}^t + n_4^t n_{38}^t l_{32}^t + l_{30}^t + \sum_{j \in B} n_j^t$$

with  $B = \{1, 6, 15, 17, 23, 28, 34\}$ . Each time a keystream bit is generated, both feedback registers are updated by their retroaction functions.

**Initialization.-** The IV is loaded in the initial state in the following way:  $n_i^0 = iv_i, 0 \leq i \leq 39, l_i = iv_{i+40}, 0 \leq i \leq 29$  and  $l_i^0 = 1, 30 \leq i \leq 38, l_{39}^0 = 0$ . The cipher is then clocked 320 times; instead of outputting the keystream bits, these bits are used as feedback in the FSRs:

$$l_{39}^{t+1} = z_t + f(L^t)$$

$$n_{39}^{t+1} = z_t + k_t^* + l^t + c_4^t + g(N^t)$$

**Keystream Generation.-** After the 320 initialisation clocks, the keystream starts being generated according to the previously defined output function; one keystream bit per state update.

### 3 Key-Recovery Attack on Full Sprout

The attack described in this section and that has allowed us to attack the full version of Sprout, exploits the short sizes of the registers, the little dependency between them when generating the keystream and the non-linear influence of the keybits in the update function. We use an evolved divide-and-conquer attack, combined with a guess-and-determine technique for recovering the key bits, that resembles the analysis applied to the hash function Shabal from [9, 22]. It recovers the whole key much faster than an exhaustive search and needs very little data.

Our attack is composed of three steps: in the first one, the attacker builds and arranges two independent lists of possible internal states for the LFSR and for the NLFSR at an instant  $r' = 320 + r$ . For now on, we will refer to time with respect to the state after initialization, being  $t = 0$  the instant where the first keystream bit is output. During the second step, we merge the previous lists with the help of some bits from the keystream that will allow to perform a sieving in order to exclusively keep as candidates the pairs of states that could have generated the known keystream bits. Finally, once a reduced set of possible internal states is kept, we will recover the whole key by using some additional keystream bits. Through all the attack, we consider  $r + \#z$  keystream bits as known  $(z_0, \dots, z_{r+\#z-1})$ . The last  $1 + \#z$  bits are used in the second step of the

attack, for reducing the number of state candidates. The first  $r-1$  bits are used in the last step of the attack, for recovering the only one correct state and the whole key. We will use these bits in our attack, and therefore they represent the data complexity. As we show in the following, the parameters  $r$  and  $\#z$  are the ones we adapt to optimize the attack, and in order to mount the best possible attacks, we always have  $\#z \geq 6$  and  $r \geq 1$ .

We first describe some useful preliminary remarks. Next we describe the three steps of the attack, and finally we provide a summary of the full attack along with the detailed complexities of each step.

### 3.1 Preliminary Remarks

We present in this subsection some observations on Sprout, that we use in the following sections for mounting our attack.

Let us consider the internal state of the cipher at time  $t$ . If we guessed<sup>1</sup> both registers at time  $t$ , how could we discard some incorrect guesses by using some known keystream bits?

*Linear Register.* - First of all, let us remark that the linear register state is totally independent from the rest during the keystream generation phase. Then, once its 40-bit value at time  $t$  are guessed, we can compute all of its future and past states during the keystream generation, including all its bits involved in the keystream generation.

We describe now the four sievings that can be performed in order to reduce the set of possible states with the help of the conditions imposed by the keystream bits.

**Type I: Direct Sieving of 2 Bits.**- From Sect. 2 we know that the keystream bit at clock cycle  $t$  is given by:

$$z_t = n_4^t l_6^t + l_8^t l_{10}^t + l_{32}^t l_{17}^t + l_{19}^t l_{23}^t + n_4^t n_{38}^t l_{32}^t + l_{30}^t + \sum_{j \in B} n_j^t$$

with  $B = \{1, 6, 15, 17, 23, 28, 34\}$ . We can see that 9 bits of the NLFSR intervene in the keystream bit computation, 7 linearly and 2 as part of terms of degree 2 and 3, as depicted on Fig. 2 (in this figure, instant  $r$  corresponds to the generic instant  $t$  that we consider in this section). The first observation we can make is that if we know the 80 bits of the internal state at clock  $t$ , then we can directly compute the impact of the LFSR and of the NLFSR in the value of  $z_t$  and of  $z_{t+1}$  (see  $r$  and  $r+1$  on Fig. 2), which will potentially give us a sieving of two bits: as  $z_t$  and  $z_{t+1}$  are known, the computed values should collide with the known ones. The number of state candidates will then be reduced by a factor of  $2^{-2}$ . For instants positioned after  $t+1$ , the bit  $n_{38}^t$  turns unknown so we cannot exploit the same direct sieving. In the full version of the attack, this sieving involves keystream bits  $z_r$  and  $z_{r+1}$ .

<sup>1</sup> Which cannot be done as it contains  $2^{80}$  possible values and therefore exceeds the exhaustive search complexity.





the relation with  $z_{t+i}$ . In this case not only we reduce the number of possible states, but we also recover some associated key bit candidates 2 out of 3 times, as we show in details in Sect. 3.3. For each bit that we need to guess ( $\times 2$ ) we obtain a sieving of  $2^{-1}$ , which compensate. The total number of state candidates, when considering the positions that need a bit guessing and the ones that do not, is reduced by a factor of  $(3/4) \approx 2^{-0.415}$  per keystream bit considered with the type III conditions. For our full attack this gives  $2^{-0.415 \times (\#z - 2 - 4)}$ , as  $\#z$  is the number of bits considered during conditions of type I, III and IV (the one bit used during type 2 is not included in  $\#z$ ). As sieving of type I always uses 2 bits, and conditions of type IV, as we see next, always use 4 bits, sieving of type III remains with  $\#z - 2 - 4$  keystream bits. In the full version of the attack, this sieving involves keystream bits  $z_{t+i}$  for  $i$  from 2 to  $(\#z - 5)$ .

**Type IV: Probabilistic Sieving.-** In the full version of the attack, this sieving involves keystream bits  $z_{t+i}$  for  $i$  from  $\#z - 4$  to  $(\#z - 1)$ . Now, we do not guess bits anymore, but instead we analyse what more we can say about the states, *i.e.* whether we can reduce the amount of candidates any further. We point out that  $n_{38}^{t+i}$  only appears in one term from  $h$ . What happens if we consider also the next 4 keystream bits? What information can the next keystream bits provide? In fact, as represented in Fig. 2, the next four keystream bits could be computed without any additional guesses with each considered pair of states, but for the bit  $n_{38}^{t+i}$ , that is not known. But if we have a look carefully, this bit only affects the corresponding keystream bit one time out of three. Indeed, the partial expression given by  $h$ :

$$n_4^{t+i} n_{38}^{t+i} l_{32}^{t+i}$$

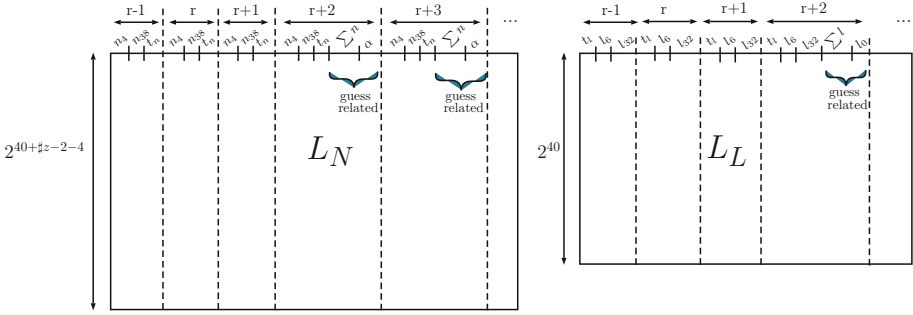
is only affected by  $n_{38}^{t+i}$  for 3/4 of the values the other two related variables,  $n_4^{t+i}$  and  $l_{32}^{t+i}$ , can take. Therefore, even without knowing  $n_{38}^{t+i}$ , we can perform a sieving of one bit 3/4 of the times. On average, as this can be done up to considering four more keystream bits, marked in Fig. 2 with 3/4, we will obtain an additional sieving of  $4 \times 3/4 = 3$  bits, *i.e.* the number of state candidates will be additionally reduced by  $2^{-3}$ .

We can now start describing our attack.

### 3.2 Building the Lists $L_L$ and $L_N$

We pointed out in the previous section that guessing the whole internal state at once (80 bits) would already be as expensive as the exhaustive key search. Therefore, we start our attack by guessing separately the states of both the NLSFR and the LFSR registers at instant  $r$ . For each register we build a list, obtaining two independent lists  $L_L$  and  $L_N$ , which contain respectively the possible state bit values of the internal states of the LFSR, and respectively of the NLFSR, at a certain clock-cycle  $r' = 320 + r$ , *i.e.*  $r$  rounds after the first keystream bit is generated.

More precisely,  $L_L$  is filled with the  $2^{40}$  possibilities for the 40 bits of the LFSR at time  $r$  (which we denoted by  $l_0$  to  $l_{39}$ ).  $L_N$  is a bigger list that contains



**Fig. 3.** Lists  $L_L$  and  $L_N$  before starting the attack. All the values used for the sorting can be computed from the original states, and the  $\alpha_{r+i}$  in the case of  $L_N$

$2^{40+\#z-2-4} = 2^{34+\#z}$  elements<sup>2</sup>, corresponding to the 40-bit state of the NLFSR (denoted by  $n_0$  to  $n_{39}$ ), each coupled to the  $2^{\#z-2-4}$  possible values for  $\alpha_r = k_r^* + l_0^r + c_4^r$  to  $\alpha_{r+\#z-6} = k_{r+\#z-6}^* + l_0^{r+\#z-6} + c_4^{r+\#z-6}$ . See Fig. 4 for a better description of  $\alpha$ .

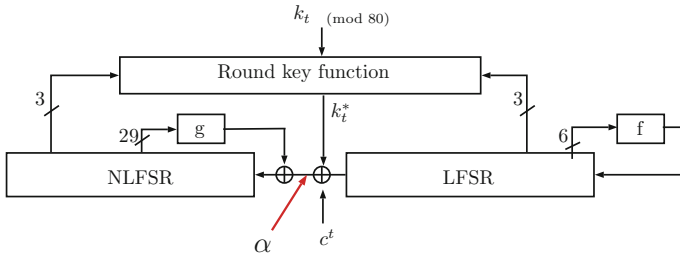
As detailed next, we also store additional bits deduced from the previous ones to speed up the attack. In  $L_N$ , we store for certain instants of time the bits  $n_4, n_{38}, t_n \triangleq \sum_{j \in B} n_j$  (the linear contribution of the NLFSR to the output bit  $z$ ) and  $\sum n = n_9 + n_{20} + n_{29}$  (the sum of the NLFSR bits that appear in the key selection process) while in  $L_L$  it is  $l_6, l_{32}, t_l \triangleq l_{30} + l_8 l_{10} + l_{19} l_{23} + l_{17} l_{32} + z_t$  and  $\sum l = l_4 + l_{21} + l_{37}$ . These bits are arranged as shown in Fig. 3.

### 3.3 Reducing the Set of Possible States

The main aim of this step is to use the precomputed lists  $L_L$  and  $L_N$  to combine them and keep only the subset of the crossproduct that corresponds to a full internal state for the registers and that could generate the keystream bits considered. It is easy to see that this problem perfectly corresponds to *merging lists* with respect to a relation, introduced in [23]. Therefore, we will use the algorithms proposed to solve it in [12, 14, 23] in order to efficiently find the remaining candidate pairs. Let us point out here that in the complexities we take into account for applying these algorithms, we not only take into account the candidates kept on the lists, but also the cost of sorting and comparing the lists.

Of course, our aim is to make the number of remaining state candidates shorter than the trivial amount of  $2^{80}$  (the total number of possible internal states for the registers). To achieve this, we use the sieves described in Sect. 3.1 as the relations to consider during the merging of the lists. The sieves were deduced from relations that the known keystream bits and the state bits at time  $r$  must satisfy.

<sup>2</sup> In the next section we describe how to reduce the state candidates step by step, so if only conditions of type I and II are considered, no guesses are needed and  $L_N$  is of size  $2^{40}$ . When sieving conditions of type III are considered, but not of type IV, as in Table 2, the size of  $L_N$  is  $2^{40+\#z-2}$  instead, i.e. the size of the list is  $2^{40+\#z-2-\#IV}$ , where  $\#IV$  are the conditions of type IV considered.



**Fig. 4.** Position of the additional guesses stored in list  $L_N$

For the sake of simplicity, we start by presenting an attack that only uses the sievings of type I and II. Next we will show how to also take into consideration the sieving of type III, and finally we will show how to also take into account the sieving of type IV, and therefore the 4 sievings at once for obtaining a reduced set of possible initial states.

**Sievings of Type I and II with  $z_{r-1}, z_r$  and  $z_{r+1}$ .** Exceptionally, in this simplified version of the attack we consider  $\sharp z = 2$ , and  $t$  is at least one. We therefore know at least three keystream bits:  $z_{t-1}, z_t$  and  $z_{t+1}$ , that we use for reducing the size of the set of possible internal states at instant  $t$ .

We consider the previously built lists  $L_L$  and  $L_N$  both of size  $2^{40}$  (no guesses are performed for this sievings) and are sorted as follows (see the three first columns of lists in Fig. 3):

- $L_L$  is sorted according to  $t_l^t = l_{30}^t + l_8^t l_{10}^t + l_{19}^t l_{23}^t + l_{17}^t l_{32}^t + z_t, l_6^t$  and  $l_{32}^t$  at instants  $r - 1, r$  and  $r + 1$ .
- $L_N$  is sorted according to  $n_4^t, n_{38}^t$  and finally  $t_n^t = \sum_{j \in B} n_j^t$  at time  $r - 1, r$  and  $r + 1$ .

Given our new notations, we can rewrite the equation expressing  $z_t$ , as:

$$t_l^t + t_n^t + n_4^t(n_{38}^t l_{32}^t + l_6^t) = 0$$

We will use it for  $t$  from  $r - 1$  to  $r + 1$ . The idea is then to use the relations implied by these three equations to deduce the possible initial state values of the LFSR and of the NLFSR in a guess and determine way.

For instance, if we first consider the situations in which the bits  $n_4^t$  and  $n_{38}^t$  are null, we know that the relation  $t_l^t + t_n^t = 0$  must be satisfied so that we can only combine one eighth of  $L_N$  ( $n_4^t = 0, n_{38}^t = 0$  and  $t_n^t = 0$ , or respectively  $n_4 = 0, n_{38} = 0$  and  $t_n = 1$ ) with one half of  $L_L$  (in which  $t_l = 0$ , respectively  $t_l = 1$ ). The same way, fixing other values for  $n_4, n_{38}$  and  $t_n$  we obtain other restricted number of possibilities for the values of  $t_l^t, l_6^t$  and  $l_{32}^t$ . We reduce the total number of candidate states by  $2^{-1}$  per keystream bit considered. When considering the equations from the three keystream bits  $z_{t-1}, z_t$  and  $z_{t+1}$ , we therefore obtain  $2^{77}$  possible combinations instead of  $2^{80}$ .

This is a direct application of the gradual matching algorithm from [23], and we provide a detailed description of how the algorithm works and should be implemented in Sect. 4.2.

**Additional Sieving of Type III with  $z_{r+2}, \dots, z_{r+\#z-1}$ .**<sup>3</sup> We can easily improve the previous result by taking into account the sieving of type III presented in the previous section. List  $L_N$  will have, in this case, a size of  $2^{40+\#z-2}$ , where  $\#z-2$  is the number of keystream bits that will be treated with sieving of type III, and therefore, the number of  $\alpha_{t+i}$  bits that will be guessed (for  $i$  from 0 to  $\#z-2-1$ ). The attacker is given  $(1+\#z)$  bits of keystream  $(z_{r-1}, \dots, z_{r+\#z-1})$ , and she can directly exploit  $z_{r-1}, z_r$  and  $z_{r+1}$  with sieving conditions of type I and II. Next arranging the table as showed in Fig. 3 will help exploiting the conditions derived from keystream bits  $z_{r+2}, \dots, z_{r+\#z-1}$ .

To explain in more detail the sieving probability deduced in Sect. 3.1 with respect to one condition of type III, we refer to Table 1 where in 1 case out of 4 the cohabitation of a fixed value of bits of  $L_L$  and  $L_N$  is impossible, which indicates to the attacker that the internal state is not possible, retaining a proportion of 3/4 of the considered states.

We recall that, so far (as we have not discussed yet the application of sieving conditions of type IV), the number of keystream bits treated by type III of conditions is  $\#z-2$ . We have one additional sieving condition of type III per each one of these  $\#z-2$  bits of the keystream. Each additional condition to test reduces the number of possible combinations of sublists by a factor of  $\frac{3}{4} = 2^{-0.4150}$ , as we have just seen. By repeating this process  $\#z-2$  times, we finally obtain a number  $2^{80-3-0.415*(\#z-2)}$  of possible internal states. Let us detail the cost of obtaining this reduced set of possible states. The process of the attack considering sievings of type I, II and III simultaneously, which is done using a gradual matching technique as described in [23], can be broadly summarized as follows and can be visualized in Table 2.

1. Consider the two precomputed lists  $L_N$  and  $L_L$  of respective sizes  $2^{40+\#z-2}$  and  $2^{40}$ , containing all the possibilities for the 40-bit long internal states of the NLFSR and the  $\#z-2$  additional guesses and respectively the 40-bit long possible internal states of the LFSR.
2. For  $i$  from 0 to  $\#z$ , consider keystream bit  $z_{r+i}$ , and:
  - (a) if  $i \leq 2$ , divide the current (sub)list from  $L_N$  in  $2^3$  sublists according to the values of  $n_4, n_{38}$  and  $t_n$  at time  $r+i-1$  and divide the current (sub)list from  $L_L$  into  $2^3$  sublists according to the values of  $t_l, l_6$  and  $l_{32}$  also at time  $r+i-1$ . According to the previous discussion, we know that only  $2^{3+3-1} = 2^5$  combinations of sublists are possible (for sievings of type I and II). For each one of the  $2^5$  possible combinations, consider the next value for  $i$ .

---

<sup>3</sup> In the full attack, the last keystream bit considered here is  $z_{r+\#z-1-4}$ , as  $\#z$  is four units bigger when considering sieving conditions of type IV.

**Table 1.** Restrictions obtained from the additional guess, deduced from the formula of  $n_{39}^{t+1}$

guess	$\sum n$	$l_0$	$\sum l$	information
0	0	0	0	none
			1	$k = 0$
		1	0	impossible
			1	$k = 1$
	1	0	0	$k = 0$
			1	none
		1	0	$k = 1$
			1	impossible
1	0	0	0	impossible
			1	$k = 1$
		1	0	none
			1	$k = 0$
	1	0	0	$k = 1$
			1	impossible
		1	0	$k = 0$
			1	none

(b) if  $i > 2$ , divide further the current sublist from  $L_N$  in  $2^5$  sublists according to the values of the 5 bits  $n_4, n_{38}, t_n, \sum n$  and  $\alpha_{r+i-1-2} = (k_{r+i-1-2}^* + l_{r+i-1-2})$  (the additional guess) at time  $r + i - 1$  and divide the current sublist from  $L_L$  in  $2^5$  sublists according to the values of the 5 bits  $t_l, l_6, l_{32}, \sum l$  and  $l_0$  at time  $r + i - 1$ . According to the previous discussion, we know that only  $2^{5+5-1-0.415} = 2^{8.585}$  combinations of those sublists are possible. For each one of the  $2^{8.585}$  possible combinations, consider the next value for  $i$ .

For a given value of  $\#z$ , the log of the complexity of recursively obtaining the reduced possibilities for the internal state by this method could be computed as the sum of the right most column of Table 2, as this represents the total number of possible sublist combinations to take into account plus the sum of this column and the log of the *relative sizes* in both remaining sublists, which are given in the last line considered, as, for each possible combination of the sublists, we have to try all the elements remaining in one list with all the elements in the other. In the cases where the log is negative ( $-h$ ), we only check the combinations with the other sublists when we find a non empty one, which happens with probability  $2^{-h}$ , and this also corresponds to the described complexity.

Let us consider  $\#z = 8$ . The total time complexity<sup>4</sup> will be

$$2^{3*5+6*8.585} + 2^{3*5+6*8.585+8-1+1} \approx 2^{74.51}$$

<sup>4</sup> We are not giving here the complexity yet in number of encryptions, which will reduce it when comparing with an exhaustive search.

**Table 2.** .

i	$L_N$ sublists size (log)	$L_L$ sublists size (log)	matching pairs at this step (log)
	$40 + \#z - 2$	40	
0	$35 + \#z$	37	5
1	$32 + \#z$	34	5
2	$29 + \#z$	31	5
3	$24 + \#z$	26	8.585
4	$19 + \#z$	21	8.585
5	$14 + \#z$	16	8.585
6	$9 + \#z$	11	8.585
7	$4 + \#z$	6	8.585
8	$\#z - 1$	1	8.585
9	$\#z - 6$	'-4'	8.585
10	$\#z - 11$	'-9'	8.585

If we considered for instance  $\#z = 9$ , we obtain for  $i = 9$  a number of possible combinations of  $2^{3 \cdot 5 + 7 \cdot 8.585} \approx 2^{75.095}$  for checking if the corresponding sublist is empty or not, and so the attack will be more expensive than when considering  $\#z = 8$ , which seems optimal (without including conditions of type IV).

To compare with exhaustive search (so to give the time complexity on encryption functions), we have to multiply  $2^{74.51}$  by  $\frac{8}{(320)}$ , where  $\frac{8}{(320)} = 2^{-5.32}$  is the term comparing our computations with one encryption, i.e. 320 initialization rounds, and we do not take into account the following 80 rounds for recovering one unique key, as with early abort techniques one or two rounds should be enough. This gives  $2^{69.19}$  as time complexity, for recovering  $2^{74.5}$  possible states.

We can still improve this, by using the sieving of type 4, as we show in the next section.

**Additional Sieving of Type IV with  $z_{r+2}, \dots, z_{r+\#z-1}$ .**- Applying the type IV sieving is quite straight forward, as no additional guesses are needed: It just means that on average, we have an additional extra sieving of  $2^{-3}$  per possible state found after the sievings of type I, II and III. In the end, when considering all the sievings, we recover  $2^{71.5}$  possible states with a time complexity determined by the previous step (applying sieving of type III which is the bottleneck) of  $2^{69.19}$  encryption calls.

As previously we have determined that the optimal value for  $\#z$  when considering sieving conditions of type I, II and III is 8, now, as we consider 4 additional keystream bits, the optimal value is  $\#z = 8 + 4 = 12$ .

The question now is: how to determine, from the  $2^{71.5}$  possible states, which one is correct, and whether it is possible or not to recover the whole key. We will see how both things are possible with negligible additional cost.

### 3.4 Full Key Recovery Attack: Guessing a Middle State

The main idea that allows us to recover the whole master key with a negligible extra complexity is considering the guessed states of the registers as not the first initial one, obtained right after initialization and generation of  $z_0$ , but instead, guessing the state after having generated  $r$  keystream bits, with  $r > 0$  (for instance, values of  $r$  that we will consider are around 100). The data needs will be  $r + \#z$  keystream bits, which is more than reasonably low (the keystream generation limit provided by the authors is  $2^{40}$  bits). We recall here that the optimal value for  $\#z$  is 12.

With a complexity equivalent to  $2^{69.19}$  encryptions, we have recovered  $2^{71.5}$  possible internal states at time  $r$  using  $\#z + 1 = 13$  keystream bits, reducing the initial total amount by  $2^{8.5}$ . The question now is: how to find the only correct one, out of these  $2^{71.5}$  possible states? And can we recover the 80-bit master key? We recall that, on average, we have already recovered  $(\#z - 2 - 4) * 2/3 = 4$  keybits during the type III procedure described in Sect. 3.3. For the sake of simplicity, and as the final complexity won't be modified (it might be slightly better for the attacker if we consider them in some cases), we will forget about these 4 keybits.

*Inverting one Round for Free.*- Using Fig. 2, we will describe how to recover the whole key and the correct internal state with a negligible cost. This can be done with a technique inspired by the one for inverting the round function of the Shabal [8] hash function, proposed in [9, 22]. The keystream bit from column  $z$ , marked with a 1 (at round  $(r - 2)$ ) represents  $z_{r-2}$ , and implies the value of  $n_1^{r-2}$  at this same round<sup>5</sup>, which implies the value of  $n_0^{r-1}$ , one round later. This last value also completely determines the value of the guessed bit in round  $r - 1$  ( $\alpha_{r-1}$ ), which determines the value of this same round  $k_{r-1}^*$ , which, with a probability of 1/2, will determine the corresponding key bit and with probability of 1/4 won't be a valid state, corresponding to the case of  $k_{r-1}^* = 1$  and  $(l_4^{r-1} + l_{21}^{r-1} + l_{37}^{r-1} + n_9^{r-1} + n_{20}^{r-1} + n_{29}^{r-1}) = 0$ , producing a sieving of 3/4 (we only keep 3/4 of the states on average).

*Inverting Many Rounds for Free.*- We can repeat the exact same procedure considering also the keystream bits marked with 2 and 3 ( $z_{r-3}$  and  $z_{r-4}$  respectively). When we arrive backwards at round  $(r - 5)$ , we are considering the keystream bit marked with 4, that is actually  $z_{r-5}$ , and the bit  $n_4^{r-5}$  needed for checking the output equations that wasn't known before, is now known as it is  $n_1^{r-2}$ , that was determined when considering the keystream bit  $z_{r-2}$ . We can therefore repeat the procedure for keystream bits 4, 5, 6... and so on. Indeed, in the same way, we can repeat this for as many rounds as we want, with a negligible cost (but for the constant represented by the number of rounds).

*Choosing the Optimal Value for  $r$ .*- As we have seen, going backwards  $r$  rounds (so up to the initialisation state) will determine on average  $r/2$  key bits, and for

<sup>5</sup> This result comes from the expression of  $z_{r-2}$  that linearly involves  $n_1^{r-2}$  while all the other involved terms are known.



each keystream bit considered we have a probability of  $3/4$  of keeping the state as candidate, so we will keep a proportion of  $(3/4)^{r-1}$  state candidates.

Additionally, if  $r > 80$ , because of the definition of  $k^*$ , the master key involved bits will start repeating<sup>6</sup>. For the kept state candidates, we have an additional probability of around  $2/3 \times 2/3 = 2^{-2}$  of having determined the bit at one round as well as exactly 80 rounds before. The  $2/3$  comes from the fact that, for having one key bit at an instant  $t$  determined we need  $(l_4^t + l_{21}^t + l_{37}^t + n_9^t + n_{20}^t + n_{29}^t) = 1$ , and as the case  $(l_4^t + l_{21}^t + l_{37}^t + n_9^t + n_{20}^t + n_{29}^t) = 0$  with  $k_t^* = 1$  has been eliminated by discarding states, we have that 2 out of the three remaining cases will determine a key bit. Therefore, when this happens, we need the bits to collide in order to keep the tested state as a candidate. This happens with an additional probability of  $1/2$  per bit.

We first provide here the equations considering  $r \leq 80$ . Given  $2^{71.5}$  possible states obtained during the second step, the average number of states that we will keep as candidates after inverting  $r$  rounds ( $\#s$ ) is  $\#s = 2^{71.5} \times (3/4)^r$ . Each one has  $\#K = r \times 2/3$  determined key bits on average.

For  $160 > r > 80$ , the average number of states that we will keep as candidates is

$$\#s = 2^{71.5} \times (3/4)^r \times 2^{-(r-80) \times (2/3)^2}.$$

Each one has  $\#K = r \times 2/3 - (r - 80) \times (2/3)^2$  determined key bits on average.

For any  $r$ , as we can gradually eliminate the candidate states on the fly, we do not need to compute backwards all the 100 bits but for very few of them. The complexity of testing the kept states in encryption function calls in the worst case will be

$$2^{71.5} \times \frac{1}{320} + 2^{71.5-1 \times 0.41} \times \frac{2}{320} + \dots + 2^{71.5-(r-1) \times 0.41} \times \frac{r}{320},$$

we can upper bound this complexity by  $10 \times 2^{71.5} \times \frac{1}{320} \approx 2^{66.5}$ , which is lower than the complexity to perform the previous step, described in Sect. 3.3, so won't be the bottleneck.

As for each final kept state, we have to try all the possibilities for the remaining  $80 - \#K$  key bits, we can conclude that the final complexity of this last part of the attack in number of encryptions is

$$\#s \times 2^{80-\#K}, \tag{1}$$

Which will be negligible most of the times (as a small increase in  $r$  means a big reduction of this complexity).

The optimal case is obtained for values of  $r$  close to 100, so we won't provide the equations when  $r > 160$ .

For our attack, it would seem enough to choose  $r = 80$  in order to have this last step less expensive than the previous one, and therefore, in order not to

---

<sup>6</sup> As previously said, for the sake of simplicity we do not take into account the  $\#z$  bits computed from  $r$  forward, and we discuss in the next section on implementation, the very little this changes in the final complexity (any way, it could only help the attacker, so the attack is as least as "good" as explained in our analysis).

increase the time complexity. We can choose  $r = 100$  so that we are sure that things will behave correctly and the remaining possible key candidates can be very efficiently tested. We recall that the optimal value for  $\#z$  was  $8 + 4$ , which means that the data complexity of our attack is  $r + \#z = 112$  bits of keystream, which is very small. We have  $\#s = 2^{21.11}$  and  $\#K = 57.2$ . The complexity of this step is therefore  $2^{21.11} \times 2^{80-57.2} = 2^{43.91}$ , which is much lower than the complexity of the previous steps.

### 3.5 Full Attack Summary

We consider  $r = 100$  and  $\#z = 12$ . The data complexity of the attack is therefore 112 bits.

First, we have precomputed and carefully arranged the two lists  $L_L$  and  $L_N$ , of size  $2^{40}$  and  $2^{40+12-4-2} = 2^{46}$ , and  $2^{46}$  will be the memory needed to perform the attack, as all the remaining steps can be performed on the fly. Next, we merged both lists with respect to the sieving conditions of type I, II, III and IV, obtaining  $2^{71.5}$  state candidates with a complexity of  $2^{69.19}$  encryptions. For each candidate state, we compute some clocks backwards, in order to perform an additional sieving and to recover some key bits. This can be done with a complexity of  $2^{66.5}$ . The kept states and associated key bits are tested by completing the remaining key bits, and we only keep the correct one. This is done with a cost of  $2^{43.91}$ . We recover then the whole master key with a time complexity of  $2^{69.39}$  encryptions, *i.e.* around  $2^{10}$  times faster than an exhaustive key search. In the next section we implement the attack on a reduced version of the cipher, being able to prove the validity of our theoretical analysis, and verifying the attack.

## 4 Implementation and Verification of the Attack

To prove the validity of our attack, we experimentally test it on a shrunk cipher with similar structure and properties. More specifically, we built a small stream cipher according to the design principles used for Sprout but with a key of 22 bits and two states of 11 bits. We then implemented our attack and checked the returned complexities.

### 4.1 Toy Cipher Used

The toy cipher we built is the one represented in Fig. 5. It follows the same structure as Sprout but its registers are around 4 times smaller. We have chosen the functions so that the sieving conditions behaved similarly as in our full round attack. We keep the same initialisation principle and set the number of initialisation rounds to  $22 \times 4 = 88$  (in Sprout there are  $80 \times 4 = 320$  initialisation rounds).

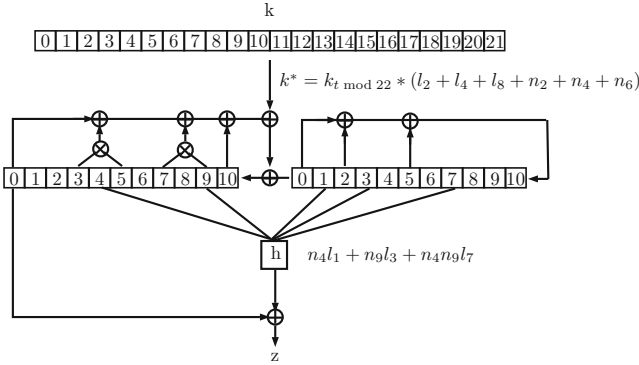


Fig. 5. Toy cipher

### 4.2 Algorithm Implemented

#### Steps 1 and 2 of the Attack.-

1. Ask for  $r + \#z = r + 3$  keystream bits generated from time  $t = 0$  to  $t = r + 3 - 1$ , that we denote by  $z^0, z^1, \dots, z^{r+2}$
2. Build a list  $L_L$  of size  $2^{11}$  containing all the possible values for the 11 bits of the linear register at time  $t = r$ , sorted according to:
  - $l_1^r, l_3^r$  and  $l_7^r$  at time  $t = r$ ,
  - $l_1^{r+1}, l_3^{r+1}$  and  $l_7^{r+1}$  at time  $t = r + 1$ ,
  - $l_3^{r+2}$  and  $l_7^{r+2}$  at time  $t = r + 2$  and finally
  - $l_0^r, l_2^r + l_4^r + l_8^r$  at time  $t = r$
3. Build a list  $L_N$  of size  $2^{11+1} = 2^{12}$  that contains all the possible state values of the non-linear register at time  $t = r$  plus the value of an additional guess and sort it according to:
  - $n_0^r + z^r, n_4^r$  and  $n_9^r$  at time  $t = r$ ,
  - $n_0^{r+1} + z^{r+1}, n_4^{r+1}$  and  $n_9^{r+1}$  at time  $t = r + 1$ ,
  - $n_0^{r+2} + z^{r+2}, n_4^{r+2}$  and  $n_9^{r+2}$  at time  $t = r + 2$  and finally
  - $\alpha^r$  (the guessed bit) at time  $t = r$
4. Create a new list  $M$  containing the possible value of  $L_L$  and  $L_N$  together:
  - (a) Consider the states of  $L_L$  and  $L_N$  for which the first indexes ( $l_1^r, l_3^r$  and  $l_7^r$  in  $L_L$  and  $n_0^r + z^r, n_4^r$  and  $n_9^r$  in  $L_N$ ) verify the equation given by the keystream bit at time  $t = r$ :

$$z^r = n_4^r l_1^r + n_9^r l_3^r + n_4 n_9 l_7^r + n_0^r$$

- i. Apply a second filter given by the second indexes ( $l_1^{r+1}, l_3^{r+1}$  and  $l_7^{r+1}$  in  $L$  and  $n_0^{r+1} + z^{r+1}, n_4^{r+1}$  and  $n_9^{r+1}$  in  $G$ ) by checking if the equation given by the keystream bit at time  $t = r + 1$  holds:

$$z^{r+1} = n_4^{r+1} l_1^{r+1} + n_9^{r+1} l_3^{r+1} + n_4^{r+1} n_9^{r+1} l_7^{r+1} + n_0^{r+1}$$

- A. Similarly, apply a sieving according to the third indexes. Remark here that  $l_1$  at time  $t = r + 2$  is equal to the already fixed bit  $l_3$  at time  $t = r$ . Finally, use the additional information deduced from  $\alpha$  at time  $t = r$  that must verify

$$\alpha^r = k^r \cdot (l_2^r + l_4^r + l_8^r + n_2^r + n_4^r + n_6^r)$$

so that it implies a contradiction if  $l_2^r + l_4^r + l_8^r = n_2^r + n_4^r + n_6^r$  and  $\alpha^r \neq l_0$  at the same time.

As discussed in Sect. 3.3, the resulting filter on the cardinal product of the list is of  $2^{-1-1-1-0.415}$  so  $2^{23-3.415} = 2^{19.585}$  possible states remain at this point.

**Step 3 of the Attack.-**

1. For each of the  $2^{19.585}$  possible states at time  $t = r$ , create a vector of 22 bits  $\tilde{K}$  for the possible value of the key associated to it:
  - (a) For time  $t = r - 1$  to  $t = 0$ :
    - i. Deduce the values of  $n_i^t$ ,  $i = 1 \dots 10$  and of  $l_i^t$ ,  $i = 1 \dots 10$  from the state at time  $t + 1$
    - ii. Compute the value of  $n_0^t$  given by the keystream bit equation as:

$$n_0^t = z^t + n_4^t l_1^t + n_9^t l_3^t + n_4^t n_9^t l_7^t$$

and of  $l_0^t$  given by the LFSR retroaction equation as:

$$l_0^t = l_2^t + l_5^t + l_{10}^{t+1}$$

and deduce from it the value of

$$k^{*t} = n_0^t + n_3^t n_5^t + n_7^t n_9^t + n_{10}^t + l_0 + n_{10}^{t+1}$$

(given by the NLFSR retroaction equation)

- iii. Compute the value of  $l_2^t + l_4^t + l_8^t + n_2^t + n_4^t + n_6^t$  and combine it with the value of  $k^{*t}$  obtained in the previous step:
  - A. If  $l_2^t + l_4^t + l_8^t + n_2^t + n_4^t + n_6^t = 0$  and  $k^{*t} = 1$ , there is a contradiction so discard the state and try another one by going back to Step 1.
  - B. If  $l_2^t + l_4^t + l_8^t + n_2^t + n_4^t + n_6^t = 1$  and  $k^{*t} = 0$  check if the bit has already been set in  $\tilde{K}$ . If no, set it to 0. Else, if there is a contradiction, discard the state and try another one by going back to Step 1.
  - C. If  $l_2^t + l_4^t + l_8^t + n_2^t + n_4^t + n_6^t = 1$  and  $k^{*t} = 1$  check if the bit has already been set in  $\tilde{K}$ . If no, set it to 1. Else, if there is a contradiction, discard the state and try another one by going back to Step 1.

### 4.3 Results

The previous algorithm has been implemented and tested for various values of  $r$ . At the end of step 2 we recovered indeed  $2^{19.5}$  state candidates. In all the cases, the pair formed by the correct internal state and the partial right key were included amongst the candidates at the end of step 3. The results are displayed in Table 3, together with the values predicted by theory. We recall here that the expected number of states at the end of the key recovery is given by the formula in Sect. 3.4 which in this case can be simplified by:

$$2^{19.5} \times (3/4)^r = 2^{19.5-0.415r} \quad \text{when } r < |k| \text{ and by}$$

$$2^{19.5} \times (3/4)^r \times 2^{-(r-|k|) \times (2/3)^2} = 2^{29.35-0.859r} \quad \text{when } r \geq |k|.$$

In the same way, we expect the following amount of bits to be determined:

$$r \times (2/3) \quad \text{when } r < |k| \text{ and}$$

$$r \times (2/3) - (r - |k|) \times (2/3)^2 \quad \text{when } r \geq |k|.$$

This leads to the comparison given in Table 3 in which we can remark that theory and practice meet quite well.

Note that given the implementation results, a sensible choice would be to consider a value of  $r$  around 26. Indeed,  $r = 26$  means that the attacker has to consider all the  $2^{7.32}$  states at the end of the key recovery part and for each of them has to exhaust on average the 6.67 unknown bits, leading to an additional complexity of  $2^{13.99}$ . This number has to be compared to the time complexity of the previous operation. The time complexity for recovering the  $2^{19.585}$  candidates at the end of step 2 is the bottleneck of the time complexity. According to Sect. 3.3, this term can be approximated by  $2^{19.585} \times \frac{3}{88} \simeq 2^{14.71}$  encryptions. So recovering the full key is of negligible complexity in comparison, and  $r = 26$  leads to an attack of time complexity smaller than  $2^{15}$  encryptions, coinciding with our theoretical complexity.

**Table 3.** Experimental results obtained on average on 300 random states and keys

r	20	21	22	23	24	25	26	27	28	29	30
log of number of states remaining at the end of the key recovery	11.28	10.85	10.47	9.68	8.95	8.01	7.32	6.63	5.75	5.17	4.42
theory	11.3	10.9	10.5	9.6	8.8	7.9	7.0	6.2	5.3	4.4	3.6
unknown bits	8.68	8.02	7.30	7.12	6.96	6.77	6.67	6.32	6.29	6.03	5.94
theory	8.7	8.0	7.3	7.1	6.9	6.7	6.4	6.2	6.0	5.8	5.6

## 5 Conclusion

In this paper we present a key-recovery attack on the stream cipher Sprout, proposed at FSE 2015, that allows to recover the whole key more than  $2^{10}$  times faster than exhaustive search. We have implemented our attack on a toy version of the cipher. This implemented attack behaves as predicted, and, therefore, we have been able to verify the correctness of our approach. Our attack exploits the small size of the registers and the non-linear influence of the key in the update function. It shows a security issue on Sprout and suggests that a more careful analysis should be done in order to instantiate the proposed design method.

An interesting direction to look at for repairing this weakness would be to consider the key influence on the update function as linear.

## References

1. Abdelraheem, M.A., Blondeau, C., Naya-Plasencia, M., Videau, M., Zenner, E.: Cryptanalysis of ARMADILLO2. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 308–326. Springer, Heidelberg (2011)
2. Ågren, M., Hell, M., Johansson, T., Meier, W.: Grain-128a: a new version of grain-128 with optional authentication. *IJWMC* **5**(1), 48–59 (2011)
3. Armknecht, F., Mikhalev, V.: On Lightweight stream ciphers with shorter internal states. In: FSE 2015. LNCS. Springer (2015, to appear)
4. Armknecht, F., Mikhalev, V.: On Lightweight Stream Ciphers with Shorter Internal States. Cryptology ePrint Archive, Report 2015/131 (2015). <http://eprint.iacr.org/2015/131>
5. Bogdanov, A., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y.: Hash functions and RFID tags: mind the gap. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 283–299. Springer, Heidelberg (2008)
6. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsoe, C.: PRESENT: an ultra-lightweight block cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 450–466. Springer, Heidelberg (2007)
7. Borghoff, J., et al.: PRINCE – a low-latency block cipher for pervasive computing applications. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 208–225. Springer, Heidelberg (2012)
8. Bresson, E., Canteaut, A., Chevallier-Mames, B., Clavier, C., Fuhr, T., Gouget, A., Icart, T., J. Misarsky, Naya-Plasencia, M., Paillier, P., Pornin, T., Reinhard, J., Thuillet, C., Videau, M.: Shabal. In: The first SHA-3 Candidate Conference, Leuven, Belgium (2009)
9. Bresson, E., Canteaut, A., Chevallier-Mames, B., Clavier, C., Fuhr, T., Gouget, A., Icart, T., Misarsky, J.F., Naya-Plasencia, M., Paillier, P., Pornin, T., Reinhard, J.R., Thuillet, C., Videau, M.: Indifferentiability with Distinguishers: Why Shabal Does Not Require Ideal Ciphers. Cryptology ePrint Archive, Report 2009/199 (2009). <http://eprint.iacr.org/2009/199>
10. De Cannière, C.: TRIVIUM: a stream cipher construction inspired by block cipher design principles. In: Katsikas, S.K., López, J., Backes, M., Gritzalis, S., Preneel, B. (eds.) ISC 2006. LNCS, vol. 4176, pp. 171–186. Springer, Heidelberg (2006)

11. De Cannière, C., Dunkelman, O., Knežević, M.: KATAN and KTANTAN — a family of small and efficient hardware-oriented block ciphers. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 272–288. Springer, Heidelberg (2009)
12. Canteaut, A., Naya-Plasencia, M., Vayssière, B.: Sieve-in-the-middle: improved MITM attacks. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 222–240. Springer, Heidelberg (2013)
13. Collard, B., Standaert, F.-X.: A statistical saturation attack against the block cipher PRESENT. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 195–210. Springer, Heidelberg (2009)
14. Dinur, I., Dunkelman, O., Keller, N., Shamir, A.: Efficient dissection of composite problems, with applications to cryptanalysis, knapsacks, and combinatorial search problems. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 719–740. Springer, Heidelberg (2012)
15. Dinur, I., Shamir, A.: Cube attacks on tweakable black box polynomials. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 278–299. Springer, Heidelberg (2009)
16. Gong, Z., Nikova, S., Law, Y.W.: KLEIN: a new family of lightweight block ciphers. In: Juels, A., Paar, C. (eds.) RFIDSec 2011. LNCS, vol. 7055, pp. 1–18. Springer, Heidelberg (2012)
17. Guo, J., Peyrin, T., Poschmann, A., Robshaw, M.J.B.: The LED block cipher. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 326–341. Springer, Heidelberg (2011)
18. Hell, M., Johansson, T., Meier, W.: Grain: a stream cipher for constrained environments. *IJWMC* **2**(1), 86–93 (2007)
19. Lallemand, V., Naya-Plasencia, M.: Cryptanalysis of KLEIN. In: Cid, C., Rechberger, C. (eds.) FSE 2014. LNCS, vol. 8540, pp. 451–470. Springer, Heidelberg (2015)
20. Leander, G., Abdelraheem, M.A., AlKhzaimi, H., Zenner, E.: A cryptanalysis of PRINTCIPHER: the invariant subspace attack. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 206–221. Springer, Heidelberg (2011)
21. Mendel, F., Rijmen, V., Toz, D., Varıcı, K.: Differential analysis of the LED block cipher. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 190–207. Springer, Heidelberg (2012)
22. Naya-Plasencia, M.: Chiffrements à flot et fonctions de hachage : conception et cryptanalyse. INRIA Paris-Rocquencourt, Project SECRET et Université Pierre et Marie Curie, France, Thèse (2009)
23. Naya-Plasencia, M.: How to improve rebound attacks. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 188–205. Springer, Heidelberg (2011)
24. Naya-Plasencia, M., Peyrin, T.: Practical cryptanalysis of ARMADILLO2. In: Canteaut, A. (ed.) FSE 2012. LNCS, vol. 7549, pp. 146–162. Springer, Heidelberg (2012)
25. Nikolić, I., Wang, L., Wu, S.: Cryptanalysis of round-reduced LED. In: Moriai, S. (ed.) FSE 2013. LNCS, vol. 8424, pp. 112–130. Springer, Heidelberg (2014)
26. Shirai, T., Shibutani, K., Akishita, T., Moriai, S., Iwata, T.: The 128-bit blockcipher CLEFIA (extended abstract). In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 181–195. Springer, Heidelberg (2007)
27. Suzuki, T., Minematsu, K., Morioka, S., Kobayashi, E.: TWINE: a lightweight block cipher for multiple platforms. In: Knudsen, L.R., Wu, H. (eds.) SAC 2012. LNCS, vol. 7707, pp. 339–354. Springer, Heidelberg (2013)
28. Wu, W., Zhang, L.: LBlock: a lightweight block cipher. In: Lopez, J., Tsudik, G. (eds.) ACNS 2011. LNCS, vol. 6715, pp. 327–344. Springer, Heidelberg (2011)

# Higher-Order Differential Meet-in-the-middle Preimage Attacks on SHA-1 and BLAKE

Thomas Espitau<sup>1,2</sup>, Pierre-Alain Fouque<sup>3,4</sup>(✉), and Pierre Karpman<sup>2,5</sup>

<sup>1</sup> École normale supérieure de Cachan, Cachan, France  
tespita@ens-cachan.fr

<sup>2</sup> Inria, Villeurbanne, France  
pierre.karpman@inria.fr

<sup>3</sup> Université de Rennes 1, Rennes, France

<sup>4</sup> Institut Universitaire de France, Paris, France  
pa.fouque@gmail.com

<sup>5</sup> Nanyang Technological University, Singapore, Singapore

**Abstract.** At CRYPTO 2012, Knellwolf and Khovratovich presented a differential formulation of advanced meet-in-the-middle techniques for preimage attacks on hash functions. They demonstrated the usefulness of their approach by significantly improving the previously best known attacks on SHA-1 from CRYPTO 2009, increasing the number of attacked rounds from a 48-round *one-block preimage without padding* and a 48-round *two-block preimage without padding* to a 57-round *one-block preimage without padding* and a 57-round *two-block preimage with padding*, out of 80 rounds for the full function. In this work, we exploit further the differential view of meet-in-the-middle techniques and generalize it to higher-order differentials. Despite being an important technique dating from the mid-90's, this is the first time higher-order differentials have been applied to meet-in-the-middle preimages. We show that doing so may lead to significant improvements to preimage attacks on hash functions with a simple linear message expansion. We extend the number of attacked rounds on SHA-1 to give a 62-round *one-block preimage without padding*, a 56-round *one-block preimage with padding*, and a 62-round *two-block preimage with padding*. We also apply our framework to the more recent SHA-3 finalist BLAKE and its newer variant BLAKE2, and give an attack for a 2.75-round *preimage with padding*, and a 7.5-round *pseudo-preimage on the compression function*.

**Keywords:** Hash function · Preimage attack · Higher-order differential meet-in-the-middle · SHA-1 · BLAKE · BLAKE2

## 1 Introduction

A hash function is a cryptographic primitive that is used to compress any binary string of arbitrary length to one of a fixed predetermined length:  $H : \{0, 1\}^* \rightarrow$

---

P. Karpman—Partially supported by the Direction Générale de l'Armement and by the Singapore National Research Foundation Fellowship 2012 (NRF-NRFF2012-06).



$\{0, 1\}^n$ . Hash functions hold a special role among cryptographic primitives, as operating without a key. This makes the analysis of their security somewhat harder than for most other primitives, but three notions are commonly used for that purpose: *collision resistance*, means that it is hard for an attacker to find two distinct strings (or messages)  $m$  and  $m'$  such that  $H(m) = H(m')$ ; *second preimage resistance* means that it is hard given a predetermined message  $m$  to find a distinct message  $m'$  such that  $H(m) = H(m')$ ; and *preimage resistance* means that it is hard given a target  $t$  to find a message  $m$  such that  $H(m) = t$ . The hardness level associated with these three notions depends on the length of the output of  $H$ , and is  $\mathcal{O}(2^{\frac{n}{2}})$  for collision resistance, and  $\mathcal{O}(2^n)$  for (second) preimage resistance. In addition to these notions, it is also common to evaluate the security of hash function through the one of its building blocks.

In this work, we give a framework that can be used to attack the preimage resistance of hash functions designed around certain principles. We show the usefulness of our approach by improving the best known attacks on two popular hash functions: the first is the NIST standard SHA-1 [16], which is a widely used function originally designed by the NSA in the 1990's; the second is the SHA-3 finalist BLAKE [3], which along with its updated version BLAKE2 is increasingly being used in modern applications.

Our starting point is the meet-in-the-middle technique, which was first used in cryptography by Diffie and Hellman in 1977 to attack double-encryption [8]. Its use for preimage attack is much more recent and is due to Aoki and Sasaki, who used it as a framework to attack various hash functions, including for instance SHA-0 and SHA-1 [2]. The basic principle behind a meet-in-the-middle technique is to exploit the fact that some value can be computed in two different ways involving different parts of a secret, which can then be sampled independently of each other. In the case of hash function cryptanalysis, there is no actual secret to consider, but a similar technique can nonetheless be exploited in certain cases; we show in more details how to do so in the preliminaries of Sect. 2.

At CRYPTO 2012, Knellwolf and Khovratovich introduced a differential formulation of the meet-in-the-middle framework of Aoki and Sasaki, which they used to improve the best attacks on SHA-1. One of the main interests of their approach is that it simplifies the formulation of several advanced extensions of the meet-in-the-middle technique, and thereby facilitates the search for attack parameters (in the case of meet-in-the-middle attacks, this roughly corresponds to finding good partitions for the ‘secret’).

In this work, we further exploit this differential formulation and generalize it to use higher-order differentials, which were introduced in cryptography by Lai in 1994 [13]. The essence of this technique is to consider ‘standard’ differential cryptanalysis as exploiting properties of the first-order derivative of the function one wishes to analyze; it is then somehow natural to generalize the idea and to consider higher-order derivatives as well. Let us illustrate this with a small example using XOR ‘ $\oplus$ ’ differences: consider a function  $f$  and assume the differential  $\Delta_\alpha f \triangleq f(x) \oplus f(x \oplus \alpha) = A$  holds with a good probability; this is the same as saying that the derivative of  $f$  in  $\alpha$  is biased towards  $A$ . In particular, if  $f$  is linear, this is

equal to a constant value  $f(\alpha)$ , though this is obviously not true in general. Now consider the value  $\Delta_\alpha f(x) \oplus \Delta_\alpha f(x \oplus \beta) = f(x) \oplus f(x \oplus \alpha) \oplus f(x \oplus \beta) \oplus f(x \oplus \alpha \oplus \beta)$ , which corresponds to taking the derivative of  $f$  twice, first in  $\alpha$ , and then in  $\beta$ . The nice point about doing this is that this function may be more biased than  $\Delta_\alpha f$  was, for instance by being constant when  $\Delta_\alpha f$  is linear. This process can be iterated at will, each time decreasing the algebraic degree of the resulting function until it reaches zero.

As higher-order differentials are obviously best formulated in differential form, they combine neatly with the differential view of the framework of Knellwolf and Khovratovich, whereas using such a technique independently of any differential formulation would probably prove to be much more difficult.

**Previous and New Results on SHA-1 and BLAKE(2).** The first preimage attacks on SHA-1 were due to De Cannière and Rechberger [7], who used a system-based approach that in particular allows to compute practical preimages for a non-trivial number of steps. In order to attack more steps, Aoki and Sasaki later used a MiTM approach [2]. This was subsequently improved by Knellwolf and Khovratovich [12], who attacked the highest number of rounds so far. To be more precise, they attack reduced versions of the function up to 52 steps for *one-block preimages with padding*, 57 steps for *one-block preimages without padding*, and 60 steps for *one-block pseudo-preimages with padding*. The latter two attacks can be combined to give 57 steps *two-block preimages with padding*. In this work, we present *one-block preimages with padding* up to 56 steps, *one-block preimages without padding* up to 62 steps, *one-block pseudo preimages with padding* up to 64 steps, resulting in *two-block preimages with padding* up to 62 steps.

The previous best known result for the BLAKE hash function, as far as preimages are concerned, is a 2.5-round attack by Li and Xu [14]. In a compression function model, the previous best attack reached 4 rounds [19]. For BLAKE2, the only known result is a pseudo-preimage attack on the full compression function targeting a small class of weak preimages of a certain form [10]. In this paper, we give a 2.75-round (resp. 3-round) *preimage attack* on BLAKE-512 and BLAKE2b, and a 7.5-round (resp. 6.75) pseudo-preimage on the compression functions of the larger (resp. smaller) variants of BLAKE and BLAKE2.

We give a summary of these results in Table 1.

## 2 Meet-in-the-middle Attacks and the Differential Framework from CRYPTO 2012

As a preliminary, we give a description of the meet-in-the-middle framework for preimage attacks on hash functions, and in particular of the differential formulation of Knellwolf and Khovratovich from CRYPTO 2012 [12].

The relevance of meet-in-the-middle for preimage attacks comes from the fact that many hash functions are built from a compression function which is an *ad hoc* block cipher used in one of the PGV modes [17]. One such popular mode is the so-called *Davies-Meyer*, where a compression function  $h : \{0, 1\}^v \times$

**Table 1.** Existing and new results on SHA-1 and BLAKE(2) (the complexity is given in base-2 logarithm).

Function	# blocks	# rounds	Complexity	Ref.
SHA-1	1	52	158.4	[12]
	1	52	156.7	Sect. 4.3
	1	56	159.4	Sect. 4.3
	2	57	158.8	[12]
	2	58	157.9	Sect. 4.4
	2	62	159.3	Sect. 4.4
SHA-1, without padding	1	57	158.7	[12]
	1	58	157.4	Sect. 4.2
	1	62	159	Sect. 4.2
SHA-1, pseudo-preimage	1	60	157.4	[12]
	1	61	156.4	Sect. 4.4
	1	64	158.7	Sect. 4.4
BLAKE-512	1	2.5	481	[14]
	1	2.75	510.3	Sect. 5.3
BLAKE2b	1	2.75	511	Sect. 5.3
BLAKE-256 c.f., pseudo-preimage	1	6.75	253.9	Sect. 5.2
BLAKE-512 c.f., pseudo-preimage	1	7.5	510.3	Sect. 5.2
BLAKE2s c.f., pseudo-preimage	1	6.75	253.8	Sect. 5.2
BLAKE2b c.f., pseudo-preimage	1	12	0 (weak class)	[10]
	1	7.5	510.3	Sect. 5.2

$\{0, 1\}^n \rightarrow \{0, 1\}^v$  compressing a chaining value  $c$  with a message  $m$  to form the updated chaining value  $c' = h(c, m)$  is defined as  $h(c, m) \triangleq f(m, c) + c$ , with  $f : \{0, 1\}^n \times \{0, 1\}^v \rightarrow \{0, 1\}^v$  a block cipher of key-length and message-length  $n$  and  $v$  respectively.

Given such a compression function, the problem of finding a preimage of  $h$  is equivalent to finding a key  $m$  for  $f$  such that  $f(m, p) = c$  for a pair  $(p, c)$ , with  $c = t - p$ . Additional constraints can also be put on  $p$ , such as prescribing it to a fixed initialization value  $[iv]$ .

In its most basic form, a meet-in-the-middle attack can speed-up the search for a preimage if the block cipher  $f$  can equivalently be described as the composition  $f_2 \circ f_1$  of two block ciphers  $f_1 : \mathcal{K}_1 \times \{0, 1\}^v \rightarrow \{0, 1\}^v$  and  $f_2 : \mathcal{K}_2 \times \{0, 1\}^v \rightarrow \{0, 1\}^v$  with independent key spaces  $\mathcal{K}_1, \mathcal{K}_2 \subset \{0, 1\}^n$ . Indeed, if this is the case, an attacker can select a subset  $\{k_i^1, i = 1 \dots N_1\}$  (resp.  $\{k_j^2, j = 1 \dots N_2\}$ ) of keys of  $\mathcal{K}_1$  (resp.  $\mathcal{K}_2$ ), which together suggest  $N \triangleq N_1 \cdot N_2$  candidate keys  $k_{ij}^{12} \triangleq (k_i^1, k_j^2)$  for  $f$  by setting  $f(k_{ij}^{12}, \cdot) = f_2(k_j^2, \cdot) \circ f_1(k_i^1, \cdot)$ .

Since the two sets  $\{f_1(k_i^1, p), i = 1 \dots N_1\}$  and  $\{f_2^{-1}(k_j^2, c), j = 1 \dots N_2\}$  can be computed independently, the complexity of testing  $f(k_{ij}^{12}, p) = c$  for  $N$  keys is only of  $\mathcal{O}(\max(N_1, N_2))$  time and  $\mathcal{O}(\min(N_1, N_2))$  memory, which is less than  $N$  and can be as low as  $\sqrt{N}$  when  $N_1 = N_2$ .

## 2.1 Formalizing Meet-in-the-middle Attacks with Related-Key Differentials

Let us denote by  $(\alpha, \beta) \xrightarrow[p]{f} \gamma$  the fact that  $\Pr_{(x,y)} [f(x \oplus \alpha, y \oplus \beta) = f(x, y) \oplus \gamma] = p$ , meaning that  $(\alpha, \beta)$  is a related-key differential for  $f$  that holds with probability  $p$ . The goal of an attacker is now to find two linear sub-spaces  $D_1$  and  $D_2$  of  $\{0, 1\}^m$  such that:

$$D_1 \cap D_2 = \{0\} \quad (1)$$

$$\forall \delta_1 \in D_1 \exists \Delta_1 \in \{0, 1\}^v \text{ s.t. } (\delta_1, 0) \xrightarrow[1]{f_1} \Delta_1 \quad (2)$$

$$\forall \delta_2 \in D_2 \exists \Delta_2 \in \{0, 1\}^v \text{ s.t. } (\delta_2, 0) \xrightarrow[1]{f_2^{-1}} \Delta_2. \quad (3)$$

Let  $d_1$  and  $d_2$  be the dimension of  $D_1$  and  $D_2$  respectively. Then for a set  $M$  of messages  $\mu_i \in \{0, 1\}^m$  (or more precisely the quotient space of  $\{0, 1\}^m$  by  $D_1 \oplus D_2$ ), one can define  $\#M$  distinct sets  $\mu_i \oplus D_1 \oplus D_2$  of dimension  $d_1 + d_2$  (and size  $2^{d_1+d_2}$ ), which can be tested for a preimage with a complexity of only  $\mathcal{O}(\max(2^{d_1}, 2^{d_2}))$  time and  $\mathcal{O}(\min(2^{d_1}, 2^{d_2}))$  memory. We recall the procedure to do so in Algorithm 1.

---

### Algorithm 1. Testing $\mu \oplus D_1 \oplus D_2$ for a preimage [12]

---

**Input:**  $D_1, D_2 \subset \{0, 1\}^m, \mu \in \{0, 1\}^m, p, c$

**Output:** A preimage of  $c + p$  if there is one in  $\mu \oplus D_1 \oplus D_2$ ,  $\perp$  otherwise

**Data:** Two lists  $L_1, L_2$  indexed by  $\delta_2, \delta_1$  respectively

```

1 forall the  $\delta_2 \in D_2$  do
2    $L_1[\delta_2] \leftarrow f_1(\mu \oplus \delta_2, p) \oplus \Delta_2$ 
3 forall the  $\delta_1 \in D_1$  do
4    $L_2[\delta_1] \leftarrow f_2^{-1}(\mu \oplus \delta_1, c) \oplus \Delta_1$ 
5 forall the  $(\delta_1, \delta_2) \in D_1 \times D_2$  do
6   if  $L_1[\delta_2] = L_2[\delta_1]$  then
7     return  $\mu \oplus \delta_1 \oplus \delta_2$ 
8 return  $\perp$ 

```

---

**Analysis of Algorithm 1.** For the sake of simplicity we assume that  $d_1 = d_2 \triangleq d < \frac{v}{2}$ . The running time of every loop of Algorithm 1 is therefore  $\mathcal{O}(2^d)$  (assuming efficient data structures and equality testing for the lists), and  $\mathcal{O}(2^d)$  memory is necessary for storing  $L_1$  and  $L_2$ . It is also clear that if the condition  $L_1[\delta_2] = L_2[\delta_1]$  is met, then  $\mu \oplus \delta_1 \oplus \delta_2$  is a preimage of  $c + p$ . Indeed, this translates

to  $f_1(\mu \oplus \delta_2, p) \oplus \Delta_2 = f_2^{-1}(\mu \oplus \delta_1, c) \oplus \Delta_1$ , and using the differential properties of  $D_1$  and  $D_2$  for  $f_1$  and  $f_2$ , we have that  $f_1(\mu \oplus \delta_1 \oplus \delta_2, p) = f_1(\mu \oplus \delta_2, p) \oplus \Delta_1$  and  $f_2^{-1}(\mu \oplus \delta_1 \oplus \delta_2, c) = f_2^{-1}(\mu \oplus \delta_1, c) \oplus \Delta_2$ . Hence,  $f_1(\mu \oplus \delta_1 \oplus \delta_2, p) = f_2^{-1}(\mu \oplus \delta_1 \oplus \delta_2, c)$ , and  $f(\mu \oplus \delta_1 \oplus \delta_2, p) = c$ . This algorithm therefore allows to search through  $2^{2d}$  candidate preimages with a complexity of  $\mathcal{O}(2^d)$ , and thus gives a speed-up of  $2^d$ . The complexity of a full attack is hence  $\mathcal{O}(2^{v-d})$ .

**Comparison with Basic Meet-in-the-middle.** When setting  $\Delta_1 = \Delta_2 = 0$ , this differential variant of the meet-in-the-middle technique becomes a special case of the general formulation of the basic technique given above: the key spaces  $\mathcal{K}_1$  and  $\mathcal{K}_2$  now possess a structure of affine spaces. The advantage of this restriction comes from the fact that it gives a practical way of searching for the key spaces, as differential path search is a well-studied area of symmetric cryptanalysis. Another major advantage is that it makes the formulation of several extensions to this basic attack very natural, without compromising the ease of the search for the key spaces. One such immediate extension is obviously to consider non-zero values for  $\Delta_1$  and  $\Delta_2$ . As noted by Knellwolf and Khovratovich, this already corresponds to an advanced technique of *indirect matching* in the original framework of Aoki and Sasaki. Further extensions are detailed next.

### 2.2 Probabilistic Truncated Differential Meet-in-the-middle

There are two natural ways to generalize the differential formulation of the meet-in-the-middle, which both correspond to relaxing one of the conditions from above. First, one can consider differentials of probability less than one (though a high probability is still usually needed); second, one can consider truncated differentials by using an equivalence relation ‘ $\equiv$ ’ instead of the equality (usually taken as a truncated equality:  $a \equiv b[m] \Leftrightarrow a \wedge m = b \wedge m$  for  $a, b, m \in \{0, 1\}^v$ ), denoting by  $(\alpha, \beta) \xrightarrow[p]{f} \gamma$  the fact that  $\Pr_{(x,y)} [f(x \oplus \alpha, y \oplus \beta) \equiv f(x, y) \oplus \gamma] = p$ . Hence Eq. 2 becomes:

$$\forall \delta_1 \in D_1 \exists \Delta_1 \in \{0, 1\}^v \text{ s.t. } (\delta_1, 0) \xrightarrow[p_1]{f_1} \Delta_1, \tag{4}$$

for some probability  $p_1$  and relation  $\equiv$ , and similarly for Eq. 3.

Again, these generalizations correspond to advanced techniques of Aoki and Sasaki’s framework, which find here a concise and efficient description.

The only change to Algorithm 1 needed to accommodate these extensions is to replace the equality by the appropriate equivalence relation on line 6. However, the fact that this equivalence holds no longer ensures that  $\mu \oplus \delta_1 \oplus \delta_2$  is a preimage, which implies an increased complexity: firstly, even when it *is* a preimage, the relation on line 6 might not hold with probability  $1 - p_1 p_2$ , meaning that on average one needs to test  $1/p_1 p_2$  times more candidates in order to account for the false negatives; secondly, if we denote by  $s$  the average size of the equivalence classes under  $\equiv$  (when using truncation as above, this is equal to  $2^{v-r}$  with  $r$

the Hamming weight of  $m$ ), then on average one needs to check  $s$  potential preimages as returned on line 6 before finding a valid one, in order to account for the false positives. The total complexity of an attack with the modified algorithm is therefore  $\mathcal{O}((2^{v-d} + s)/\tilde{p}_1\tilde{p}_2)$ , where  $\tilde{p}_1$  and  $\tilde{p}_2$  are the respective average probabilities for  $p_1$  and  $p_2$  over the spaces  $D_1$  and  $D_2$ .

### 2.3 Splice-and-cut, Initial Structures and Bicliques

These two techniques are older than the framework of [12], but are fully compatible with its differential approach.

Splice-and-cut was introduced by Aoki and Sasaki in 2008 [1]. Its idea is to use the feedforward of the compression function so as to be able to start the computation of  $f_1$  and  $f_2^{-1}$  not from  $p$  and  $c$  but from an intermediate value from the middle of the computation of  $f$ . If one sets  $f = f_3 \circ f_2 \circ f_1$  and calls  $s$  the intermediate value  $f_3^{-1}(c)$  (or equivalently  $f_2 \circ f_1(p)$ ), an attacker may now sample the functions  $f_1(t - f_3(s))$  and  $f_2^{-1}(s)$  on their respective (as always independent) key-spaces when searching a preimage for  $t$ . By giving more possible choices for the decomposition of  $f$ , one can hope for better attacks. This however comes at the cost that they are now pseudo-preimage attacks, as one does not control the value of the IV anymore which is now equal to  $t - f_3(s)$ .

A possible improvement to a splice-and-cut decomposition is the use of *initial structures* [18], which were later reformulated as *bicliques* [11]. Instead of starting the computations in the middle from an intermediate value  $s$ , the idea is now to start from a set of multiple values possessing a special structure that spans several rounds. If the cost of constructing such sets is negligible w.r.t the rest of the computations, the rounds spanned by the structure actually come for free. In more details, a biclique, say for  $f_3$  in the above decomposition of  $f$ , is a set  $\{m, D_1, D_2, Q_1, Q_2\}$  where  $m$  is a message,  $D_1$  and  $D_2$  are linear spaces of dimension  $d$ , and  $Q_1$  (resp.  $Q_2$ ) is a list of  $2^d$  values indexed by the differences  $\delta_1$  of  $D_1$  (resp.  $D_2$ ) s.t.  $\forall(\delta_1, \delta_2) \in D_1 \times D_2 \quad Q_2[\delta_2] = f_3(m \oplus \delta_1 \oplus \delta_2, Q_1[\delta_1])$ . This allows to search the message space  $m \oplus D_1 \oplus D_2$  in  $\mathcal{O}(2^d)$  with a meet-in-the-middle approach that does not need any call to  $f_3$ , essentially bypassing this part of the decomposition.

## 3 Higher-Order Differential Meet-in-the-middle

We now describe how to modify the framework of Sect. 2 to use higher-order differentials. Let us denote by  $(\{\alpha_1, \alpha_2\}, \{\beta_1, \beta_2\}) \xrightarrow[p]{f} \gamma$  the fact that  $\Pr_{(x,y)} [f(x \oplus \alpha_1 \oplus \alpha_2, y \oplus \beta_1 \oplus \beta_2) \oplus f(x \oplus \alpha_1, y \oplus \beta_1) \oplus f(x \oplus \alpha_2, y \oplus \beta_2) = f(x, y) \oplus \gamma] = p$ , meaning that  $(\{\alpha_1, \alpha_2\}, \{\beta_1, \beta_2\})$  is a related-key order-2 differential for  $f$  that holds with probability  $p$ .

Similarly as in Sect. 2, the goal of the attacker is to find four linear subspaces  $D_{1,1}, D_{1,2}, D_{2,1}, D_{2,2}$  of  $\{0, 1\}^m$  in direct sum (cf. Eq. (5)) such that:

$$D_{1,1} \oplus D_{1,2} \oplus D_{2,1} \oplus D_{2,2} \quad (5)$$

$$\forall \delta_{1,1}, \delta_{1,2} \in D_{1,1} \times D_{1,2} \exists \Delta_1 \in \{0, 1\}^v \text{ s.t. } (\{\delta_{1,1}, \delta_{1,2}\}, \{0, 0\}) \xrightarrow{f_1} \Delta_1 \quad (6)$$

$$\forall \delta_{2,1}, \delta_{2,2} \in D_{2,1} \times D_{2,2} \exists \Delta_2 \in \{0, 1\}^v \text{ s.t. } (\{\delta_{2,1}, \delta_{2,2}\}, \{0, 0\}) \xrightarrow{f_2^{-1}} \Delta_2. \quad (7)$$

Then  $M \oplus \delta_{1,1} \oplus \delta_{1,2} \oplus \delta_{2,1} \oplus \delta_{2,2}$  is a *preimage* of  $c + p$  if and only if  $f_1(\mu \oplus \delta_{1,1} \oplus \delta_{1,2} \oplus \delta_{2,1} \oplus \delta_{2,2}, c) = f_2^{-1}(\mu \oplus \delta_{1,1} \oplus \delta_{1,2} \oplus \delta_{2,1} \oplus \delta_{2,2}, p)$  which is equivalent by the Eqs. (6) and (7) to the equality:

$$\begin{aligned} f_1(\mu \oplus \delta_{1,1} \oplus \delta_{2,1} \oplus \delta_{2,2}, p) \oplus & f_2^{-1}(\mu \oplus \delta_{2,1}, \oplus \delta_{1,1} \oplus \delta_{1,2}, c) \oplus \\ f_1(\mu \oplus \delta_{1,2} \oplus \delta_{2,1} \oplus \delta_{2,2}, p) \oplus & = f_2^{-1}(\mu \oplus \delta_{2,2}, \oplus \delta_{1,1} \oplus \delta_{1,2}, c) \oplus \quad (8) \\ f_1(\mu \oplus \delta_{2,1} \oplus \delta_{2,2}, p) \oplus \Delta_1 & f_2^{-1}(\mu \oplus \delta_{1,1} \oplus \delta_{1,2}, c) \oplus \Delta_2. \end{aligned}$$

We denote by  $d_{i,j}$  the dimension of the sub-space  $D_{i,j}$  for  $i, j = 1, 2$ . Then for a set  $M$  of messages  $\mu \in \{0, 1\}^m$  one can define  $\#M$  affine sub-sets  $\mu_i \oplus D_{1,1} \oplus D_{1,2} \oplus D_{2,1} \oplus D_{2,2}$  of dimension  $d_{1,1} + d_{1,2} + d_{2,1} + d_{2,2}$  (since the sub-spaces  $D_{i,j}$  are in direct sum by hypothesis), which can be tested for a preimage using (8). This can be done efficiently by a modification of Algorithm 1 into the following Algorithm 2.

**Analysis of Algorithm 2.** If we denote by  $\Gamma_1$  and  $\Gamma_2$  the cost of evaluating of  $f_1$  and  $f_2^{-1}$  and  $\Gamma_{match}$  the cost of the test on line 14, then the algorithm allows to test  $2^{d_{1,1}+d_{1,2}+d_{2,1}+d_{2,2}}$  messages with a complexity of  $2^{d_{1,2}+d_{2,1}+d_{2,2}}\Gamma_2 + 2^{d_{1,1}+d_{2,1}+d_{2,2}}\Gamma_2 + 2^{d_{1,1}+d_{1,2}+d_{2,1}}\Gamma_1 + 2^{d_{1,1}+d_{1,2}+d_{2,1}}\Gamma_1 + 2^{d_{1,1}+d_{1,2}}\Gamma_2 + 2^{d_{2,1}+d_{2,2}}\Gamma_1 + \Gamma_{match}$ . The algorithm must then be run  $2^{n-(d_{1,1}+d_{1,2}+d_{2,1}+d_{2,2})}$  times in order to test  $2^n$  messages. In the special case where all the linear spaces have the same dimension  $d$  and if we consider that  $\Gamma_{match}$  is negligible with respect to the total complexity, the total complexity of an attack is then of:  $2^{n-4d} \cdot (2^{3d} \cdot (2\Gamma_1 + 2\Gamma_2) + 2^{2d} \cdot (\Gamma_1 + \Gamma_2)) = 2^{n-d+1}\Gamma + 2^{n-2d}\Gamma = \mathcal{O}(2^{n-d})$  where  $\Gamma$  is the cost of the evaluation of the total compression function  $f$ . We think that the assumption on the cost of  $\Gamma_{match}$  to be reasonable given the small size of  $d$  in actual attacks and the fact that performing a single match is much faster than computing  $f$ .

The factor that is gained from a brute-force search of complexity  $\mathcal{O}(2^n)$  is hence of  $2^d$ , which is the same as for Algorithm 1. However, one now needs four spaces of differences of size  $2^d$  instead of only two, which might look like a setback. Indeed the real interest of this method does not lie in a simpler attack but in the fact that using higher-order differentials may now allow to attack functions for which no good-enough order-1 differentials are available.

**Using Probabilistic Truncated Differentials.** Similarly as in Sect. 2, Algorithm 2 can be modified in order to use probabilistic truncated differentials instead of probability-1 differentials on the full state. The changes to the algorithm and the complexity evaluation are identical to the ones described in Sect. 2.2, which we refer to for a description.

---

**Algorithm 2.** Testing  $\mu \oplus D_{1,1} \oplus D_{1,2} \oplus D_{2,1} \oplus D_{2,2}$  for a preimage

---

**Input:**  $D_{1,1}, D_{1,2}, D_{2,1}, D_{2,2} \subset \{0, 1\}^m$ ,  $\mu \in \{0, 1\}^m$ ,  $p, c$ 
**Output:** A preimage of  $c + p$  if there is one in  $\mu \oplus D_{1,1} \oplus D_{1,2} \oplus D_{2,1} \oplus D_{2,2}$ ,  
 $\perp$  otherwise

**Data:** Six lists:

 $L_{1,1}$  indexed by  $\delta_{1,2}, \delta_{2,1}, \delta_{2,2}$ 
 $L_{1,2}$  indexed by  $\delta_{1,1}, \delta_{2,1}, \delta_{2,2}$ 
 $L_{2,1}$  indexed by  $\delta_{1,1}, \delta_{1,2}, \delta_{2,2}$ 
 $L_{2,2}$  indexed by  $\delta_{1,1}, \delta_{1,2}, \delta_{2,1}$ 
 $L_1$  indexed by  $\delta_{2,2}, \delta_{2,1}$ 
 $L_2$  indexed by  $\delta_{1,1}, \delta_{1,2}$ 

```

1 forall the  $\delta_{1,2}, \delta_{2,1}, \delta_{2,2} \in D_{1,2} \times D_{2,1} \times D_{2,2}$  do
2    $L_{1,1}[\delta_{1,2}, \delta_{2,1}, \delta_{2,2}] \leftarrow f_2^{-1}(\mu \oplus \delta_{1,2} \oplus \delta_{2,1} \oplus \delta_{2,2}, c)$  ;
3 forall the  $\delta_{1,1}, \delta_{2,1}, \delta_{2,2} \in D_{1,1} \times D_{2,1} \times D_{2,2}$  do
4    $L_{1,2}[\delta_{1,1}, \delta_{2,1}, \delta_{2,2}] \leftarrow f_2^{-1}(\mu \oplus \delta_{1,1} \oplus \delta_{2,1} \oplus \delta_{2,2}, c)$  ;
5 forall the  $\delta_{1,1}, \delta_{1,2}, \delta_{2,2} \in D_{1,1} \times D_{1,2} \times D_{2,2}$  do
6    $L_{2,1}[\delta_{1,1}, \delta_{1,2}, \delta_{2,2}] \leftarrow f_1(\mu \oplus \delta_{1,1} \oplus \delta_{1,2} \oplus \delta_{2,2}, p)$  ;
7 forall the  $\delta_{1,1}, \delta_{1,2}, \delta_{2,1} \in D_{1,1} \times D_{1,2} \times D_{2,1}$  do
8    $L_{2,2}[\delta_{1,1}, \delta_{1,2}, \delta_{2,1}] \leftarrow f_1(\mu \oplus \delta_{1,1} \oplus \delta_{1,2} \oplus \delta_{2,1}, p)$  ;
9 forall the  $\delta_{1,1}, \delta_{1,2} \in D_{1,1} \times D_{1,2}$  do
10   $L_2[\delta_{1,1}, \delta_{1,2}] \leftarrow f_2^{-1}(\mu \oplus \delta_{1,1} \oplus \delta_{1,2}, c) \oplus \Delta_1$  ;
11 forall the  $\delta_{2,1}, \delta_{2,2} \in D_{2,1} \times D_{2,2}$  do
12   $L_1[\delta_{2,1}, \delta_{2,2}] \leftarrow f_1(\mu \oplus \delta_{2,1} \oplus \delta_{2,2}, p) \oplus \Delta_2$  ;
13 forall the  $\delta_{1,1}, \delta_{1,2}, \delta_{2,1}, \delta_{2,2} \in D_{1,1} \times D_{1,2} \times D_{2,1} \times D_{2,2}$  do
14   if  $L_{1,1}[\delta_{1,2}, \delta_{2,1}, \delta_{2,2}] \oplus L_{1,2}[\delta_{1,1}, \delta_{2,1}, \delta_{2,2}] \oplus L_1[\delta_{2,1}, \delta_{2,2}] =$   

 $L_{2,1}[\delta_{1,1}, \delta_{1,2}, \delta_{2,2}] \oplus L_{2,2}[\delta_{1,1}, \delta_{1,2}, \delta_{2,1}] \oplus L_2[\delta_{1,1}, \delta_{1,2}]$  then
15      $\perp$  return  $\mu \oplus \delta_{1,1} \oplus \delta_{1,2} \oplus \delta_{2,1} \oplus \delta_{2,2}$ 
16 return  $\perp$ 

```

---

## 4 Applications to SHA-1

### 4.1 Description of SHA-1

SHA-1 is an NSA-designed hash function standardized by the NIST [16]. It combines a compression function which is a block cipher with 512-bit keys and 160-bit messages used in Davies-Meyer mode with a Merkle-Damgård mode of operation [15, Chap. 9]. Thus, the initial vector (IV) as well as the final hash are 160-bit values, and messages are processed in 512-bit blocks. The underlying block cipher of the compression function can be described as follows: let us denote by  $m_0, \dots, m_{15}$  the 512-bit key as 16 32-bit words. The *expanded* key  $w_0, \dots, w_{79}$  is defined as:

$$w_i = \begin{cases} m_i & \text{if } i < 16 \\ (w_{i-3} \oplus w_{i-8} \oplus w_{i-14} \oplus w_{i-16}) \lll 1 & \text{otherwise.} \end{cases}$$



Then, if we denote by  $a, b, c, d, e$  a 160-bit state made of 5 32-bit words and initialized with the plaintext, the ciphertext is the value held in this state after iterating the following procedure (parametered by the round number  $i$ ) 80 times:

$$\begin{aligned} t &\leftarrow (a \lll 5) + \Phi_{i \div 20}(b, c, d) + e + k_{i \div 20} + w_i \\ e &\leftarrow d \\ d &\leftarrow c \\ c &\leftarrow b \lll 30 \\ b &\leftarrow a \\ a &\leftarrow t, \end{aligned}$$

where ‘ $\div$ ’ denotes the integer division,  $\Phi_{0\dots3}$  are four bitwise Boolean functions, and  $k_{0\dots3}$  are four constants (we refer to [16] for their definition).

Importantly, before being hashed, a message is *always* padded with at least 65 bits, made of a ‘1’ bit, a (possibly zero) number of ‘0’ bits, and the length of the message in bits as a 64-bit integer. This padding places an additional constraint on the attacker as it means that even a preimage for the compression function with a valid IV is not necessarily a preimage for the hash function.

## 4.2 One-Block Preimages without Padding

We apply the framework of Sect. 3 to mount attacks on SHA-1 for *one-block preimages without padding*. These are rather direct applications of the framework, the only difference being the fact that we use *sets* of differentials instead of *linear spaces*. This has no impact on Algorithm 2, but makes the description of the attack parameters less compact.

As was noted in [12], the message expansion of SHA-1 being linear, it is possible to attack 15 steps both in the forward and backward direction (for a total of 30) without advanced matching techniques: it is sufficient to use a message difference in the kernel of the 15 first steps of the message expansion. When applying our framework to attack more steps (say 55 to 62), we have observed experimentally that splitting the forward and backward parts around steps 22 to 27 seems to give the best results. A similar behaviour was observed by Knellwolf and Khovratovich in their attacks, and this can be explained by the fact that the SHA-1 step function has a somewhat weaker diffusion when computed backward compared to forward.

We use Algorithm 3 to construct a suitable set of differences in the preparation of an attack. This algorithm was run on input differences of low Hamming weight; these are kept only when they result in output differences with truncation masks that are long enough and with good overall probabilities. The sampling parameter  $Q$  that we used was  $2^{15}$ ; the threshold value  $t$  was subjected to a tradeoff: the larger it is, the less bits are chosen in the truncation mask, but the better the probability of the resulting differential. In practice, we used values between 2 and 5, depending on the differential considered.

Once input and output differences have been chosen, we use an adapted version of Algorithm 2 from [12] given in Algorithm 4 to compute suitable truncation masks.

---

**Algorithm 3.** Computing a suitable output difference for a given input difference

---

**Input:** A chunk  $f_i$  of the compression function,  $\delta_{i,1}, \delta_{i,2} \in \{0, 1\}^m$ , a threshold value  $t$ , a sample size  $Q$ , an internal state  $c$ .

**Output:** An output difference  $\mathcal{S}$ , and a mask  $T_{\mathcal{S}}$  for the differential  $((\delta_{i,1}, \delta_{i,2}), 0) \xrightarrow{f_i} \mathcal{S}$

**Data:** An array  $d$  of  $n$  counters initially set to 0.

```

1 for  $q = 0$  to  $Q$  do
2   Choose  $\mu \in \{0, 1\}^m$  at random ;
3    $\Delta \leftarrow f_i(\mu \oplus \delta_{i,1} \oplus \delta_{i,2}, c) \oplus f_i(\mu \oplus \delta_{i,1}, c) \oplus f_i(\mu \oplus \delta_{i,2}, c) \oplus f_i(\mu, c)$ ;
4   for  $i = 0$  to  $n - 1$  do
5     if the  $i^{\text{th}}$  bit of  $\Delta$  is 1 then
6        $d[i] \leftarrow d[i] + 1$ ;
7 for  $i = 0$  to  $n - 1$  do
8   if  $d[i] \geq t$  then
9     Set the  $i$ -th bit of the output difference  $\mathcal{S}$  to 1;
```

---

The choice of the size of the truncation mask  $d$  in this algorithm offers a tradeoff between the probability one can hope to achieve for the resulting truncated differential and how efficient a filtering of “bad” messages it will offer. In our applications to SHA-1, we chose masks of size about  $\min(\log_2(|D_{1,1}|), \log_2(|D_{1,2}|), \log_2(|D_{2,1}|), \log_2(|D_{2,2}|))$ , which is consistent with taking masks of size the dimension of the affine spaces as is done in [12].

We similarly adapt Algorithm 3 from [12] as Algorithm 5 in order to estimate the average false negative probability associated with the final truncated differential.

We conclude this section by giving the statistics for the best attacks that we found for various reduced versions of SHA-1 in Table 2, the highest number of attacked rounds being 62. Because the difference spaces are no longer affine, they do not lend themselves to a compact description and their size is not necessarily a power of 2 anymore. The ones we use do not have many elements, however, which makes them easy to enumerate.

### 4.3 One-Block Preimages with Padding

If we want the message to be properly padded, 65 out of the 512 bits of the last message blocks need to be fixed according to the padding rules, and this naturally restricts the positions of where one can now use message differences. This has in particular an adverse effect on the differences in the backward step, which Hamming weight increases because of some features of SHA-1’s message expansion algorithm. The overall process of finding attack parameters is otherwise unchanged from the non-padded case. We give statistics for the best attacks

---

**Algorithm 4.** Find truncation mask T for matching

---

**Input:**  $D_{1,1}, D_{1,2}, D_{2,1}, D_{2,2} \subset \{0, 1\}^m$ , a sample size  $Q$ , a mask size  $d$ .

**Output:** A truncation mask  $T \in \{0, 1\}^n$  of Hamming weight  $d$ .

**Data:** An array  $k$  of  $n$  counters initially set to 0.

```

1 for  $q = 0$  to  $Q$  do
2   Choose  $\mu \in \{0, 1\}^m$  at random ;
3    $c \leftarrow f(\mu, [iv])$ ;
4   Choose  $(\delta_{1,1}, \delta_{1,2}, \delta_{2,1}, \delta_{2,2}) \in D_{1,1} \times D_{1,2} \times D_{2,1} \times D_{2,2}$  at random;
5    $\Delta \leftarrow f_1(\mu \oplus \delta_{1,1} \oplus \delta_{1,2}, c) \oplus f_1(\mu \oplus \delta_{1,1}, c) \oplus f_1(\mu \oplus \delta_{1,2}, c)$ ;
6    $\Delta \leftarrow \Delta \oplus f_2^{-1}(\mu \oplus \delta_{2,1} \oplus \delta_{2,2}, c) \oplus f_2^{-1}(\mu \oplus \delta_{2,2}, c) \oplus f_2^{-1}(\mu \oplus \delta_{2,2}, c)$ ;
7   for  $i = 0$  to  $n - 1$  do
8     if the  $i^{th}$  bit of  $\Delta$  is 1 then
9        $k[i] \leftarrow k[i] + 1$ ;
10 Set the  $d$  bits of lowest counter value in  $k$  to 1 in T.

```

---



---

**Algorithm 5.** Estimate the average false negative probability

---

**Input:**  $D_{1,1}, D_{1,2}, D_{2,1}, D_{2,2} \subset \{0, 1\}^m, T \in \{0, 1\}^n$ , a sample size  $Q$

**Output:** Average false negative probability  $\alpha$ .

**Data:** A counter  $k$  initially set to 0.

```

1 for  $q = 0$  to  $Q$  do
2   Choose  $\mu \in \{0, 1\}^m$  at random ;
3    $c \leftarrow f(\mu, [iv])$ ;
4   Choose  $(\delta_{1,1}, \delta_{1,2}, \delta_{2,1}, \delta_{2,2}) \in D_{1,1} \times D_{1,2} \times D_{2,1} \times D_{2,2}$  at random;
5    $\Delta \leftarrow f_1(\mu \oplus \delta_{1,1} \oplus \delta_{1,2}, c) \oplus f_1(\mu \oplus \delta_{1,1}, c) \oplus f_1(\mu \oplus \delta_{1,2}, c)$ ;
6    $\Delta \leftarrow \Delta \oplus f_2^{-1}(\mu \oplus \delta_{2,1} \oplus \delta_{2,2}, c) \oplus f_2^{-1}(\mu \oplus \delta_{2,2}, c) \oplus f_2^{-1}(\mu \oplus \delta_{2,2}, c)$ ;
7   for  $i = 0$  to  $n - 1$  do
8     if  $\Delta \not\equiv_T 0^n$  then
9        $k \leftarrow k + 1$ ;
10 return  $k/Q$ 

```

---

that we found in Table 3. One will note that the highest number of attacked rounds dropped from 62 to 56 when compared to Table 2.

#### 4.4 Two-Block Preimages with Padding

We can increase the number of rounds for which we can find a preimage with a properly padded message at the cost of using a slightly longer message of two blocks: if we are able to find *one-block pseudo preimages with padding* on enough rounds, we can then use the *one-block preimage without padding* to bridge the former to the IV. Indeed, in a pseudo-preimage setting, the additional freedom degrees gained from removing any constraint on the IV more than compensate for the ones added by the padding. We first describe how to compute such pseudo-preimages.

**Table 2.** One block preimage without padding.  $N$  is the number of attacked steps,  $Split$  is the separation step between the forward and the backward chunk,  $d_{i,j}$  is the  $\log_2$  of the cardinal of  $D_{i,j}$  and  $\alpha$  is the estimate for the false negative probability. The complexity is computed as described in Sect. 3.

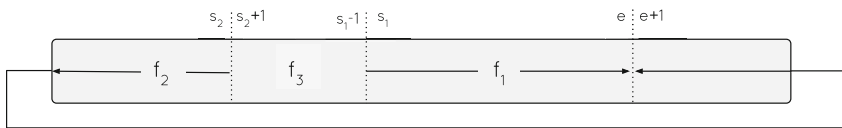
$N$	$Split$	$d_{1,1}$	$d_{1,2}$	$d_{2,1}$	$d_{2,2}$	$\alpha$	Complexity
58	25	7.6	9.0	9.2	9.0	0.73	157.4
59	25	7.6	9.0	6.7	6.7	0.69	157.7
60	26	6.5	6.0	6.7	6.0	0.60	158.0
61	27	4.7	4.8	5.7	5.8	0.51	158.5
62	27	4.7	4.8	4.3	4.6	0.63	159.0

**Table 3.** One block preimage without padding.  $N$  is the number of attacked steps,  $Split$  is the separation step between the forward and the backward chunk,  $d_{i,j}$  is the  $\log_2$  of the cardinal of  $D_{i,j}$  and  $\alpha$  is the estimation for false negative probability. The complexity is computed as described in Sect. 3.

$N$	$Split$	$d_{1,1}$	$d_{1,2}$	$d_{2,1}$	$d_{2,2}$	$\alpha$	Complexity
51	23	8.7	8.7	8.7	8.7	0.72	155.6
52	23	9.1	9.1	8.2	8.2	0.61	156.7
53	23	9.1	9.1	3.5	3.5	0.61	157.7
55	25	6.5	6.5	5.9	5.7	0.52	158.2
56	25	6	6.2	7.2	7.2	0.6	159.4

**One-Block Pseudo-preimages.** If we relax the conditions on the IV and do not impose anymore that it is fixed to the one of the specifications, it becomes possible to use a splice-and-cut decomposition of the function, as well as short (properly padded) bicliques.

The (reduced) compression function of SHA-1  $f$  is now decomposed into three smaller functions as  $f = f_2^t \circ f_1 \circ f_3 \circ f_2^b$ ,  $f_3$  being the rounds covered by the biclique. The function  $f_1$  covers the steps  $s_1$  to  $e$ ,  $f_2 = f_2^t \circ f_2^b$  covers  $s_2$  to  $e + 1$  through the feedforward, and  $f_3$  covers  $s_2 + 1$  to  $s_1 - 1$ , as shown in Fig. 1.



**Fig. 1.** A splice-and-cut decomposition with biclique.

Finding the parameters is done in the exact same way as for the one-block preimage attacks. Since the bicliques only cover 7 steps, one can generate many

of them from a single one by modifying some of the remaining message words outside of the biclique proper. Therefore, the amortized cost of their construction is small and considered negligible w.r.t. the rest of the attack. The resulting attacks are shown in Table 4.

**Table 4.** One block pseudo-preimage with padding.  $N$  is the number of attacked steps,  $d_{i,j}$  is the  $\log_2$  of the cardinal of the set  $D_{1,2}$  and  $\alpha$  is the estimation for false negative probability. The various splits are labeled as in Fig. 1. The complexity is computed as described in Sect. 3.

$N$	$s_1$	$e$	$s_2$	$d_{1,1}$	$d_{1,2}$	$d_{2,1}$	$d_{2,2}$	$\alpha$	Complexity
61	27	49	20	7.0	7.0	7.5	7.5	0.56	156.4
62	27	50	20	5.8	5.7	7.2	7.2	0.57	157.0
63	27	50	20	6.7	6.7	7.7	7.7	0.57	157.6
64	27	50	20	3	3	4.5	4.7	0.69	158.7

**Complexity of the Two-Block Attacks.** Using both one-block attacks, it is simple to mount a two-block attack at the combined cost of each of them. For a given target  $c$ , one:

1. uses a properly-padded pseudo-preimage attack, yielding the second message block  $\mu_2$  and an IV  $[iv]'$ ;
2. uses a non-padded preimage attack with target  $[iv]'$ , yielding a first message block  $\mu_2$ .

From the Merkle-Damgård structure of SHA-1, it follows that the two-block message  $(\mu_1, \mu_2)$  is a preimage of  $c$ .

For attacks up to 60 rounds, we can use the pseudo-preimage attacks of [12]; for 61 and 62 rounds, we use the ones of this section. This results in attacks of complexities as shown in Table 5.

## 5 Applications to BLAKE and BLAKE2

### 5.1 Description of BLAKE

The hash function BLAKE [3] was a candidate and one of the five finalists of the SHA-3 competition, that ended in November 2012. Although it was not selected as the winner, no weaknesses were found in BLAKE and it is accepted as being a very secure and efficient hash function [6]. More recently, a faster variant BLAKE2 has been designed [5]; both functions come in two variants with a chaining value of 256 bits (BLAKE-256 and BLAKE2s) and 512 bits (BLAKE-512 and BLAKE2b). The design of BLAKE is somewhat similar to the one of SHA-1, as being built around a compression function in Merkle-Damgård mode.

**Table 5.** Two-block preimage attacks on SHA-1 reduced to  $N$  steps. The pseudo-preimage attacks followed by ‘★’ come from [12].

$N$	Second block complexity	First block complexity	Total complexity
58	156.3★	157.4	157.9
59	156.7★	157.7	158.3
60	157.5★	158.0	158.7
61	156.4	158.5	158.8
62	157.0	159.0	159.3

It does however feature a few notable differences: first, the compression function takes two additional inputs to the message  $m$  and the previous chaining value  $c$ , in the form of a user-defined salt  $s$  and a block counter  $t$ . The new chaining value  $c'$  is thus defined as  $c' \triangleq h(c, m, s, t)$ ; second, the compression function follows the local wide-pipe paradigm which was introduced by BLAKE’s predecessor LAKE [4], meaning that the state size of the compression function  $h$  is larger than the size of the chaining value  $c$ . In particular, this implies that  $c$  is first expanded to form the input to  $h$ , and that the output of the latter is compressed in order to give  $c'$ . This feature has some important consequences when analyzing the function and makes some types of attacks harder to perform, as we will see later. We describe BLAKE-512 in more details, and refer to the specification document [3] for a full description of the function and of its variants. Similarly, the changes from BLAKE to BLAKE2 having no impact on our overall attack strategy, we refer the reader to the specifications of BLAKE2 for more details [5].

**Initialization and Finalization of the Compression Function.** The state of the compression function is logically seen as a  $4 \times 4$  array of 64-bit words  $v_{0\dots 15}$ , making 1024 bits in total. It is initialized from 8 words of incoming chaining value  $c_{0\dots 7}$ , 4 words of salt  $s_{0\dots 4}$ , 2 words of counter  $t_{0,1}$  and 8 words of constant  $k_{0\dots 7}$ , as shown below:

$$\begin{pmatrix} v_0 & v_1 & v_2 & v_3 \\ v_4 & v_5 & v_6 & v_7 \\ v_8 & v_9 & v_{10} & v_{11} \\ v_{12} & v_{13} & v_{14} & v_{15} \end{pmatrix} \leftarrow \begin{pmatrix} c_0 & c_1 & c_2 & c_3 \\ c_4 & c_5 & c_6 & c_7 \\ s_0 \oplus k_0 & s_1 \oplus k_1 & s_2 \oplus k_2 & s_3 \oplus k_3 \\ t_0 \oplus k_4 & t_0 \oplus k_5 & t_1 \oplus k_6 & t_1 \oplus k_7 \end{pmatrix}.$$

The outgoing chaining value  $c'_{0\dots 7}$  is defined from the final value of the state  $v'_{0\dots 15}$ , the initial value of the chaining value and the salt as:

$$\begin{aligned} c'_0 &\leftarrow c_0 \oplus s_0 \oplus v'_0 \oplus v'_8 & c'_4 &\leftarrow c_4 \oplus s_0 \oplus v'_4 \oplus v'_{12} \\ c'_1 &\leftarrow c_1 \oplus s_1 \oplus v'_1 \oplus v'_9 & c'_5 &\leftarrow c_5 \oplus s_1 \oplus v'_5 \oplus v'_{13} \\ c'_2 &\leftarrow c_2 \oplus s_2 \oplus v'_2 \oplus v'_{10} & c'_6 &\leftarrow c_6 \oplus s_2 \oplus v'_6 \oplus v'_{14} \\ c'_3 &\leftarrow c_3 \oplus s_3 \oplus v'_3 \oplus v'_{11} & c'_7 &\leftarrow c_7 \oplus s_3 \oplus v'_7 \oplus v'_{15}. \end{aligned}$$

**Round Function.** One round of BLAKE is made of eight calls to a ‘quarter-round’ function  $G_i$  on part of the state:

$$\begin{array}{cccc} G_0(v_0, v_4, v_8, v_{12}) & G_1(v_1, v_5, v_9, v_{13}) & G_2(v_2, v_6, v_{10}, v_{14}) & G_3(v_3, v_7, v_{11}, v_{15}) \\ G_4(v_0, v_5, v_{10}, v_{15}) & G_5(v_1, v_6, v_{11}, v_{12}) & G_6(v_2, v_7, v_8, v_{13}) & G_7(v_3, v_4, v_9, v_{14}). \end{array}$$

There are 16 such rounds for BLAKE-512<sup>1</sup>. Furthermore, because the whole state is updated twice during one round (once by  $G_{0\dots3}$  and once by  $G_{4\dots7}$ ), one such update will be called a half-round. The function  $G_i(a, b, c, d)$  is defined for round  $r$  as:

$$\begin{array}{ll} 1 : a \leftarrow a + b + (m_{\sigma_r(2i)} \oplus k_{\sigma_r(2i+1)}) & 5 : a \leftarrow a + b + (m_{\sigma_r(2i+1)} + k_{\sigma_r(2i)}) \\ 2 : d \leftarrow (d \oplus a) \ggg 32 & 6 : d \leftarrow (d \oplus a) \ggg 16 \\ 3 : c \leftarrow c + d & 7 : c \leftarrow c + d \\ 4 : b \leftarrow (b \oplus c) \ggg 25 & 8 : b \leftarrow (b \oplus c) \ggg 11, \end{array}$$

with  $\sigma$  a round-dependent permutation of  $\{0 \dots 15\}$ . The padding is nearly the same as for SHA-1. The only difference is that a ‘1’ bit is again systematically appended after the ‘0’ bits (if any). Hence, there are at least 66 bits of padding.

**Terminology.** As can be seen from the above description, there is an additional initialization phase for the compression function in BLAKE when compared to most hash functions and SHA-1 in particular. We choose to call *pseudo-preimage on the compression function* a preimage attack that bypasses this initialization and requires complete freedom for the initial 16-word state, yet that does respect the finalization (and thence forms an attack for the same level of security than the compression function, *i.e.* 512 bits for BLAKE-512). This is a more restrictive model than attacking the underlying block cipher of the compression function in a PGV mode, which would in itself be a significant attack on a building block of the hash function.

### 5.2 Pseudo-preimage on the Compression Function

If we relax the conditions on the initialization of the compression function, we can use a splice-and-cut approach to mount a pseudo-preimage attack in a straight application of the framework, which we did on the BLAKE(2) family. We note that because of the use of a local-wide-pipe construction, matching in the middle of the compression function has a complexity the square of the actual security level for preimages. Therefore we perform the matching phase right on the output of the compression function.

We mount an attack on the compression function reduced to 7.5 rounds of BLAKE-512 and BLAKE 2b attacking round 0.5 (resp. 0.25) to round 8 (resp. 7.75), and 6.75 rounds of BLAKE-256 and BLAKE2s attacking round 0.75 to round 7.5. We decompose this reduced compression function  $f$  as  $f_1 \circ f_3 \circ f_2$ . The function  $f_1$  starts at round 5.25 and ends at round 8 (resp. 7.75 for BLAKE2b

<sup>1</sup> There are 14 for BLAKE-256, 12 for BLAKE2b and 10 for BLAKE2s.

**Table 6.** One block pseudo-preimage without padding on BLAKE-512 and BLAKE2b;  $d_{i,j}$  is the  $\log_2$  of the cardinal of the set  $D_{i,j}$  and  $\alpha$  is the estimation for false negative probability. The complexity is computed as described in Sect. 3. The various splits are labeled as in Fig. 1.

Function	Start	$s_1$	$e$	$s_2$	$d_{1,1}$	$d_{1,2}$	$d_{2,1}$	$d_{2,2}$	$\alpha$	Complexity
BLAKE-512	0.5	5.25	8	4	3.0	9.3	4.0	9.5	0.49	510.3
BLAKE2b	0.25	5.25	7.75	4	4.5	8.0	3.9	3.9	0.41	510.9
BLAKE-256	0.75	5.25	7	4	4.1	7.2	9.0	9.0	0.64	253.9
BLAKE2s	0.75	5.25	7	4	4.1	7.2	9.0	9.0	0.68	253.8

and 7.5 for BLAKE2s and BLAKE-256),  $f_2^{-1}$  covers rounds 4 to 0.5 (resp. 0.25 and 0.75), and  $f_3$  is a biclique covering the 0.75 remaining rounds. Finding the parameters of the attack is done similarly as in Sect. 4. Since the biclique covers less than one round, it leaves some of the message words free, which can then be used to generate many similar bicliques. Therefore, the amortized cost of their construction is small and considered negligible w.r.t. the rest of the attack. The only message words with differences are  $m_2$  &  $m_8$  in the forward computation and  $m_3$  &  $m_{11}$  in the backward computation. As a consequence, the whole message can easily be properly padded. The statistics of the resulting attacks are shown in Table 6.

### 5.3 Preimage on the Hash Function

We now adapt the framework to mount a preimage attack for the larger variants of BLAKE and BLAKE2. Because of the quick diffusion of BLAKE’s round function (notably due to the fact that every message word is used in every round), we were unsuccessful when searching for difference spaces resulting in a good meet-in-the-middle decomposition that simultaneously preserves the initialization of the compression function.

To overcome this problem, we use a single difference space, in the forward direction only. The use of order-2 differentials proves to be critical at this point, as no gain could be easily obtained otherwise. More precisely, we use differences of the type  $(\{\alpha_1, \alpha_2\}, \{0, 0\}) \xrightarrow[p]{f} 0$ , meaning that with probability  $p$  over the messages  $m$ ,  $f(m) \oplus f(m \oplus \alpha_1) \oplus f(m \oplus \alpha_2) \equiv f(m \oplus \alpha_1 \oplus \alpha_2)$  for a truncation mask  $T$ . This equality can be used with Algorithm 2 modified as Algorithm 6 in an attack. The basic idea at play here is that after computing  $f(m)$ ,  $f(m \oplus \alpha_1)$  and  $f(m \oplus \alpha_2)$ , one can test the values of  $f(m \oplus \alpha_1 \oplus \alpha_2)$  for essentially no additional cost. We give an illustration of this process in the full version of the paper [9].

**Analysis of Algorithm 6.** The test on line 6 can be performed efficiently by using an appropriate data structure (typically a hash table), resulting in overall



---

**Algorithm 6.** Testing  $\mu \oplus D_{1,1} \oplus D_{1,2}$  for a preimage

---

**Input:**  $D_{1,1}, D_{1,2} \subset \{0, 1\}^m, \mu \in \{0, 1\}^m, p, c$   
**Output:** A preimage if there is one in  $\mu \oplus D_{1,1} \oplus D_{1,2}, \perp$  otherwise  
**Data:** Two lists  $L_1, L_2$  indexed by  $\delta_{1,2}, \delta_{1,1}$  respectively

```

1 forall the  $\delta_{1,2} \in D_{1,2}$  do
2    $L_1[\delta_{1,2}] \leftarrow f_1(\mu \oplus \delta_{1,2}, p)$ 
3 forall the  $\delta_{1,1} \in D_{1,1}$  do
4    $L_2[\delta_{1,1}] \leftarrow f_1(\mu \oplus \delta_{1,1}, c)$ 
5 forall the  $(\delta_{1,1}, \delta_{1,2}) \in D_{1,1} \times D_{1,2}$  do
6   if  $L_1[\delta_{1,2}] \oplus L_2[\delta_{1,1}] \equiv f_1(\mu, c) \oplus c \oplus p$  then
7     return  $\mu \oplus \delta_{1,1} \oplus \delta_{1,2}$ 
8 return  $\perp$ 

```

---

linear time for the loop on line 5. In line with Sect. 2.2, the total complexity of an attack based on this algorithm thus becomes  $(2^{n-d_1-d_2}(2_1^d + 2_2^d)\Gamma + s)/\tilde{p}$ , where  $d_1$  (resp.  $d_2$ ) is the dimension of  $D_{1,1}$  (resp.  $D_{1,2}$ ),  $\Gamma$  is the cost of calling  $f_1$ ,  $s$  the complexity of retesting and  $\tilde{p}$  is the average success probability of the order-2 differentials.

**Table 7.** The preimage attacks on BLAKE(2);  $d_{i,j}$  is the  $\log_2$  of the cardinal of  $D_{i,j}$  and  $\alpha$  is the estimate for the false negative probability.

Function	#rounds	$d_{1,1}$	$d_{1,2}$	$\alpha$	Complexity
BLAKE-512	2.75	4.0	9.5	0.6	510.3
BLAKE2b	2.75	3.1	6.9	0.4	511.0

**Application to BLAKE(2).** We introduce the differences at round 5.25 and let them propagate with good probability for 2.75 rounds for BLAKE-512 and BLAKE2b. Like in Sect. 5.3 the only message words with differences are  $m_2 \mathcal{E} m_8$  in the forward computation and  $m_3 \mathcal{E} m_{11}$  in the backward computation. As a consequence, the whole message can easily be properly padded. The attack parameters are found as before and lead to the attacks in Table 7.

## References

1. Aoki, K., Sasaki, Y.: Preimage attacks on one-block MD4, 63-step MD5 and more. In: Avanzi, R.M., Keliher, L., Sica, F. (eds.) SAC 2008. LNCS, vol. 5381, pp. 103–119. Springer, Heidelberg (2009)
2. 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)

3. Aumasson, J.P., Henzen, L., Meier, W., Phan, R.C.W.: SHA-3 proposal BLAKE, version 1.3 (2008). Available online at <https://131002.net/blake/>
4. Aumasson, J.-P., Meier, W., Phan, R.C.-W.: The hash function family LAKE. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 36–53. Springer, Heidelberg (2008)
5. Aumasson, J.-P., Neves, S., Wilcox-O’Hearn, Z., Winnerlein, C.: BLAKE2: simpler, smaller, fast as MD5. In: Jacobson, M., Locasto, M., Mohassel, P., Safavi-Naini, R. (eds.) ACNS 2013. LNCS, vol. 7954, pp. 119–135. Springer, Heidelberg (2013)
6. Chang, S.j., Perlner, R., Burr, W.E., Turan, M.S., Kelsey, J.M., Paul, S., Bassham, L.E.: Third-Round Report of the SHA-3 Cryptographic Hash Algorithm Competition. NIST Interagency Report 7896 (2012)
7. De Cannière, C., Rechberger, C.: Preimages for reduced SHA-0 and SHA-1. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 179–202. Springer, Heidelberg (2008)
8. Diffie, W., Hellman, M.E.: Special feature exhaustive cryptanalysis of the NBS data encryption standard. *Computer* **10**, 74–84 (1977)
9. Espitau, T., Fouque, P.A., Karpman, P.: Higher-Order Differential Meet-in-The-Middle Preimage Attacks on SHA-1 and BLAKE. IACR Cryptology ePrint Archive 2015, 515 (2015). <https://eprint.iacr.org/2015/515>
10. Guo, J., Karpman, P., Nikolić, I., Wang, L.: Analysis of BLAKE2. In: Benaloh, J. (ed.) CT-RSA 2014. LNCS, vol. 8366, pp. 402–423. Springer, Heidelberg (2014). [https://dx.doi.org/10.1007/978-3-319-04852-9\\_21](https://dx.doi.org/10.1007/978-3-319-04852-9_21)
11. Khovratovich, D., Rechberger, C., Savelieva, A.: Bicliques for preimages: attacks on Skein-512 and the SHA-2 family. In: Canteaut, A. (ed.) FSE 2012. LNCS, vol. 7549, pp. 244–263. Springer, Heidelberg (2012)
12. Knellwolf, S., Khovratovich, D.: New preimage attacks against reduced SHA-1. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 367–383. Springer, Heidelberg (2012)
13. Lai, X.: Higher order derivatives and differential cryptanalysis. In: Blahut, R.E., Costello Jr., D.J., Maurer, U., Mittelholzer, T. (eds.) Communications and Cryptography, pp. 227–233. Springer, USA (1994)
14. Li, J., Xu, L.: Attacks on Round-Reduced BLAKE. IACR Cryptology ePrint Archive 2009, p. 238 (2009). <https://eprint.iacr.org/2009/238>
15. Menezes, A., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press, Boca Raton (1996)
16. National Institute of Standards and Technology: FIPS 180–4: Secure Hash Standard (SHS), March 2012
17. Preneel, B., Govaerts, R., Vandewalle, J.: Hash functions based on block ciphers: a synthetic approach. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 368–378. Springer, Heidelberg (1994). [https://dx.doi.org/10.1007/3-540-48329-2\\_31](https://dx.doi.org/10.1007/3-540-48329-2_31)
18. 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)
19. Wang, L., Ohta, K., Sakiyama, K.: Free-start preimages of round-reduced BLAKE compression function. ASIACRYPT rump session (2009). <https://www.iacr.org/conferences/asiacrypt2009//rump/slides/11.pdf>

# **Implementations**

# Decaf: Eliminating Cofactors Through Point Compression

Mike Hamburg<sup>(✉)</sup>

Rambus Cryptography Research, San Francisco, USA  
mhamburg@cryptography.com

**Abstract.** We propose a new unified point compression format for Edwards, Twisted Edwards and Montgomery curves over large-characteristic fields, which effectively divides the curve’s cofactor by 4 at very little cost to performance. This allows cofactor-4 curves to efficiently implement prime-order groups.

## 1 Introduction

“Let  $\mathbb{G}$  be a group of prime order  $q$ .” This defines the requirements for the main group in many cryptographic systems [1, 9, 16, 18, 19], most often with the intention that  $\mathbb{G}$  will be the group of points on an elliptic curve. However, practical implementations usually do not quite deliver a group of prime order  $q$ , at least not without significant caveats. Implementations of prime-order curves usually have incomplete or variable-time addition formulas. For example, OpenSSL 1.0.1f, LibTomCrypt 1.17, PolarSSL 1.3.9 and Crypto++ 5.6.2 all use a branch to decide whether the inputs to their point-addition functions are equal, so that they can call the doubling function instead. Some of these libraries also have branches to detect cases where two points add to the identity point, or where one of them is the identity point. Even if this does not introduce timing variations in ECDH or ECDSA signing, it may introduce timing variations in other systems. Furthermore, there is a special case when encoding and decoding the identity point, which is at infinity in the Weierstrass model.

This problem can be mitigated by using complete addition laws. While such laws exist for prime-order curves [6, 11], they are faster and much simpler for other elliptic curves such as (twisted) Edwards curves [4, 5, 14], Hessian curves [15], Jacobi quartics [8] or Jacobi intersections [20, 26]. These curves have a *cofactor*, denoted  $h$ , where the order of the curve is  $h \cdot q$  for some large prime  $q$ . The cofactor  $h$  is always divisible by 3 for Hessian curves, and by 4 for the other models.

### 1.1 Pitfalls of a Cofactor

Many authors consider the advantages of a non-prime-order group, such the points on an Edwards curve, to outweigh the disadvantages. But the disadvantages are not negligible. There are several pitfalls which appear specifically for  $h > 1$ :

*Small-Subgroup Attacks.* Here an attacker sends a point whose order divides  $h$ , and a hapless user multiplies it by some scalar and uses the result. This will either result in a point known to the attacker (if the scalar is known to be divisible by  $h$ ), or worse it may give the attacker information about the scalar. If the scalar is a private key, then leaking a few bits is a minor problem, though it is devastating to password-authenticated key exchange (PAKE) protocols [13].

Leaking a few bits of a scalar is a much more serious problem if the scalar is arithmetically related to a private key. Menezes and Ustaoglu used scalar leaks through the cofactor in their attack [27] on MQV [25] and HMQV [24]. HMQV was designed to avoid this weakness, but not successfully.

A related attack is to replace a point  $P$  with  $P + T$ , where  $T$  lies in a small subgroup. If the user multiplies by a scalar  $s$ , they will get  $sP + sT$  instead of  $sP$ , where the difference  $sT$  gives away the low-order bits of  $s$ . Therefore, it isn't always enough to reject points in the small subgroup.

The usual defense against these attacks is to multiply certain points by  $h$ , and possibly to abort the protocol if the result is the identity. But one must decide which points to take these steps on, and the extra factor of  $h$  can complicate the arithmetic. In a prime-order group, this attack is easier to mitigate: at most, one must check for the identity point in the proper places.

*Non-injective Behavior.* Multiplication by a scalar is a 1-to-1 function if the scalar is relatively prime to the group order. In a prime-order group, this is any scalar in  $[1, q - 1]$ , and is true of a random scalar with high probability. The same is not true in a composite-order group. This means that adding a small-order element to e.g. a public key can produce the same result, possibly resulting in identity misbinding. This can be mitigated by making scalars relatively prime to  $h$  — exactly the opposite of techniques which clear the cofactor.

*Covert Channels.* Non-injective behavior may make it easier to exfiltrate data through the cofactor component, even in protocols where behavior is otherwise deterministic.

*Implementation-Defined Behavior.* Some systems, such as Ed25519 [7], do not specify behavior when the inputs have a nonzero  $h$ -torsion component. In particular, Ed25519 signature verification can be different for batched vs singleton signatures, or between different implementations. This can cause disruption in protocols where all parties must agree on whether a signature is valid, such as a blockchain or Byzantine agreement. In other cases, it may make it easier to fingerprint implementations.

*Nontrivial Modifications.* If a system or protocol is specified and proved secure on a prime-order group, then both the system and the proof may need to be changed for a group with a cofactor  $h > 1$ . Usually the modification is small. Often it is enough simply to multiply the outputs by  $h$ . However, only an expert will be able to tell exactly what modification is required. Cryptographic proofs are difficult, and this may represent enough work to prevent adoption.

## 1.2 Our Contribution

The cofactor pitfalls can be avoided by using a related group of prime order  $q$ . The most obvious choice is the order- $q$  subgroup of the elliptic curve. But validating membership in that subgroup is slow and complex, requiring either an extra scalar multiplication, checking for roots of division polynomials, or inverting multiple isogenies.

We propose two new ways to build a group of prime order  $q$  based on an Edwards or twisted Edwards curve  $\mathcal{E}$  of order  $4q$ , thus eliminating the cofactor. In the first proposal of this paper, the group is  $\mathcal{E}/\mathcal{E}[4]$ . That is, two points on the curve  $\mathcal{E}$  are considered equal if their difference has small order (dividing 4). This requires three changes:

- The function which checks equality of group elements must be modified.
- The function which encodes points before sending them must encode these equal points as equal sequences of bits.
- The function which decodes points must be designed to accept only the possible outputs of the encoding function.

Our second, improved proposal uses a different group  $\psi(\mathcal{E})$ , which in the usual case will be  $2\mathcal{E}/\mathcal{E}[2]$ . That is, only the even ( $2q$ -order) points of  $\mathcal{E}$  are used, and two points are considered the same if they differ by a point of order 2. This requires the same three changes.

The difficult parts are the encoding and decoding routines, which are the main contributions of this paper. We describe the encoding algorithm as “compression” because its output is an element of the underlying field  $\mathbb{F}$  rather than the usual two elements. In fact, it will be a “non-negative” element of  $\mathbb{F}$ , which allows us to save an additional bit.

It is important to note that internally, the points used in the first proposal can be any points on  $\mathcal{E}$ , and in the second proposal they can be any even points on  $\mathcal{E}$ . Points which differ by a point of order 4 (resp. 2) are considered equal, and will be encoded to binary strings in the same way. This is similar to using projective coordinates: two values in memory may be considered same point and encode to the same binary string, even though the  $X, Y$  and  $Z$  coordinates are different. This is how using a prime-order  $\psi(\mathcal{E})$  instead of  $\mathcal{E}$  mitigates small-subgroup attacks. Points of small order can appear internally, but they are considered equal to the identity element. Likewise  $P + T$  can appear internally with  $T$  in a small-order subgroup, but it is considered equal to  $P$  and is encoded in the same way.

With the combination of the complete Edwards group law and our point encoding, protocols can gain the simplicity, security and speed benefits of (twisted) Edwards curves without any cofactor-related difficulty. The cost is a small increase in code complexity in the point encoding and decoding functions. On balance, we believe that our encoding can make the design of the entire system simpler. In terms of overhead, our encoding and decoding perform as well as existing point compression and decompression algorithms.

Designers often use untwisted Edwards, twisted Edwards or Montgomery curves. Montgomery curves give simple Diffie-Hellman protocols, and twisted Edwards curves give a speed boost but have incomplete formulas in fields of order 3 (mod 4). Our second proposal adds flexibility for curve choice. The same wire format can be used for a Montgomery curve as for its 4-isogenous Edwards and twisted Edwards curves. Furthermore, for twisted Edwards curves of cofactor 4, the subgroup we use avoids the incomplete cases in the addition laws.

Our group  $\psi(\mathcal{E})$  and encoding algorithm can be used on curves with cofactor greater than 4. It still divides the cofactor by 4, so  $\psi(\mathcal{E})$  will not have prime order. Additionally,  $\psi(\mathcal{E})$  is not of the form  $2\mathcal{E}/\mathcal{E}[2]$  if  $\mathcal{E}$  has full 2-torsion.

We call this technique “Decaf” after the procedure which divides the effect of coffee by 4. We have built reference and optimized implementations of Decaf, and have posted them online at <http://sourceforge.net/p/ed448goldilocks/code/ci/decaf/tree/>. Our code carries out essentially all the operations described in this paper and appendices on the curve `Ed448-Goldilocks`, reducing the cofactor from 4 to 1.

## 2 Definitions and Notation

*Finite Field.* Let  $\mathbb{F}$  be a finite field whose characteristic is neither 2 nor 3.

*Even Elements.* An element  $g$  of an Abelian group  $\mathbb{G}$  is said to be *even* if  $g = 2h$  for some  $h \in \mathbb{G}$ . The even elements form a subgroup denoted  $2\mathbb{G}$ .

*Torsion Elements.* An element  $g$  of a group  $\mathbb{G}$  is a  $k$ -torsion element if  $k \cdot g = 0_{\mathbb{G}}$ . The  $k$ -torsion elements of an Abelian group form a subgroup usually denoted  $\mathbb{G}[k]$ . The  $k$ -torsion subgroup of an elliptic curve over a finite field has order dividing  $k^2$ ; in particular, the 2-torsion subgroup has size 1, 2 or 4.

*Projective Space.* Denote by  $\mathbb{P}^n(\mathbb{F})$  the  $n$ -dimensional projective space over  $\mathbb{F}$ . Its elements are written as ratios  $(X : Y : Z : \dots)$ , usually in upper-case. As a traditional short-cut, we usually write the elements of  $\mathbb{P}^2(\mathbb{F})$  as a lower-case tuple  $(x, y)$  equivalent to  $(x : y : 1)$ , with the understanding that the equations involving these points may have been extended to cover “points at infinity” of the form  $(X : Y : 0)$ .

*Twisted Edwards Curves.* Twisted Edwards curves have two parameters,  $a$  and  $d$ . They are specified as

$$\mathcal{E}_{a,d} := \{(x, y) \in \mathbb{P}^2(\mathbb{F}) : a \cdot x^2 + y^2 = 1 + d \cdot x^2 \cdot y^2\}$$

Another form, extended homogeneous coordinates [22], is used for high performance and simpler formulas:

$$\mathcal{E}_{a,d} := \{(X : Y : Z : T) \in \mathbb{P}^3(\mathbb{F}) : XY = ZT \text{ and } a \cdot X^2 + Y^2 = Z^2 + d \cdot T^2\}$$

We will use “untwisted” to mean  $a = 1$ . “Twisted” is the general case, which we sometimes narrow to  $a = -1$ . The identity point of any Edwards curve is  $(0, 1) = (0 : 1 : 1 : 0)$ .

An Edwards curve is called “complete” if  $d$  and  $ad$  are nonsquare in  $\mathbb{F}$ , which also implies that  $a$  is square. A complete Edwards curve has no points at infinity, and supports fast addition formulas which are complete in that they compute the correct answer for any two input points [5].

*Montgomery Curves.* A Montgomery curve has two parameters, called  $A$  and  $B$ . It has the form

$$\mathcal{M}_{B,A} := \{(u, v) \in \mathbb{P}^2(\mathbb{F}) : Bv^2 = u \cdot (u^2 + Au + 1)\}$$

The identity point of this curve is a point at infinity, namely  $(0 : 1 : 0)$ . The curve is “untwisted” if  $B = 1$ . Over  $3 \pmod{4}$  fields, any twisted Montgomery curve can be put into a form with  $B = 1$ , but over  $1 \pmod{4}$  fields, this is not true. In particular,  $B \neq 1$  is potentially useful to handle the twist of Curve25519, which has cofactor 4.

*Jacobi Quartic Curves.* A Jacobi quartic curve has two parameter, called  $A$  and  $e$ , and is defined by

$$\mathcal{J}_{e,A} := \{(s, t) \in \mathbb{P}^2(\mathbb{F}) : t^2 = es^4 + 2As^2 + 1\}$$

with an identity point at  $(0, 1)$ . The curve is “untwisted” if  $e = 1$ . We will only consider curves with  $e = a^2$  in this paper; such curves always have full 2-torsion.

*The Curve Parameters.* As a corollary of Ahmadi and Granger’s work [2], for any  $a, d \in \mathbb{F} \setminus \{0, 1\}$ , the following curves are isogenous:

$$\mathcal{E}_{a,d}; \mathcal{E}_{-a,d-a}; \mathcal{M}_{a,2-4d/a}; \mathcal{J}_{a^2,a-2d}$$

Specifically, the Edwards, twisted Edwards and Montgomery curves are all 2-isogenous to the Jacobi quartic, and thus 4-isogenous to each other. We will write the 2-isogenies explicitly in Sects. 4.1 and 5. Since our point encoding works on this family of isogenous curves, we will consider these specific curves parameterized by  $a$  and  $d$ .

We will write  $\mathcal{E}$  as a shorthand for  $\mathcal{E}_{a,d}$ ,  $\mathcal{J}$  for  $\mathcal{J}_{a^2,a-2d}$ , and  $\mathcal{M}$  for  $\mathcal{M}_{a,2-4d/a}$ .

*Coset.* In an Abelian group  $\mathbb{G}$ , the coset of a subgroup  $H \subset \mathbb{G}$  with respect to an element  $g \in \mathbb{G}$  is  $H + g := \{h + g : h \in H\}$ .

*Non-negative Field Elements.* Let  $p > 2$  be prime. Define a residue  $x \in \mathbb{F} = \mathbb{Z}/p\mathbb{Z}$  to be “non-negative” if the least absolute residue for  $x$  is in  $[0, (p - 1)/2]$ , and “negative” otherwise. This definition can be generalized (easily but non-canonically) to extension fields. Define  $|x|$  to be  $x$  or  $-x$ , whichever is non-negative. Define  $\sqrt{x}$  to be an arbitrary square root of  $x$ , not necessarily the non-negative one.

We chose this definition of non-negative because it is easy to evaluate, and it works over every odd-characteristic field. Alternative choices would be to distinguish by the low bit, or for fields  $3 \pmod{4}$ , by the Legendre symbol. We avoided the Legendre symbol because it restricts field choices and is somewhat expensive to compute.



*Encoding.* For sets  $S$  and  $T$ , and encoding from  $S$  to  $T$  is an efficient function  $\text{enc} : S \rightarrow T$  with efficient left-inverse  $\text{dec} : T \rightarrow S \uplus \{\perp\}$ , which fails by returning  $\perp$  on every element of  $T \setminus \text{enc}[S]$ . We are interested in an encoding from an elliptic curve  $\mathcal{E}$  over the field  $\mathbb{F}$  to a binary set  $\{0, 1\}^n$  for some fixed  $n$ . We assume that the implementer has already chosen an encoding from  $\mathbb{F}$  to binary. Since encodings can be composed and distributed over products, it suffices to encode to a set such as  $\mathbb{F}$ ,  $\mathbb{F}^2$  or  $\mathbb{F} \times \{0, 1\}$  which has a natural encoding to binary.

*Compression.* Since most elliptic curve forms are defined as subsets of  $\mathbb{P}^2(\mathbb{F})$ , they admit a straightforward encoding to  $\mathbb{F}^2$  (and thence to binary) with a finite number of special cases corresponding to points at infinity. We call an encoding “point compression” or simply “compression” if its codomain is smaller than  $\mathbb{F}^2$  when naturally encoded to binary. Most of the encoding algorithms in this paper map to the set  $\mathbb{F}$  or to its non-negative elements, and so are point compression functions. The set of non-negative elements of  $\mathbb{F}$  generally requires one fewer bit to encode than  $\mathbb{F}$  itself.

### 3 An Edwards-Only Solution

There is a simple way to remove the a cofactor of 4 from an untwisted Edwards curve. A complete Edwards curve  $\mathcal{E}_{a,d}$  has a 4-torsion subgroup of size exactly 4, whose coset with respect to  $P = (x, y)$  is

$$\mathcal{E}[4] + P = \{(x, y); (y/\sqrt{a}, -x\sqrt{a}); (-x, -y); (-y/\sqrt{a}, x\sqrt{a})\}$$

Of this coset, there is exactly one representative point such that  $y$  and  $xy$  are both non-negative, and  $x$  is nonzero.<sup>1</sup> We can define the encoding of  $P$  to be the  $y$ -value of this representative. Note that the representation of the identity point is  $(0, -1)$ , so the identity point encodes to  $0 \in \mathbb{F}$ .

Similar solutions apply to incomplete Edwards curves. For curves whose 4-torsion group is  $Z_4$ , there is exactly one representative with  $y$  and  $y/x$  both finite and non-negative. For curves with full 2-torsion, there is exactly one representative with  $x$  finite and both  $y$  and  $(y^2 + ax^2)/xy$  non-negative.

The usual addition formulas for incomplete Edwards curves produce the wrong answer  $(0/0)$  for operations involving points at infinity, but are otherwise complete. Therefore, if the decoding operation chooses a coset representative in a subgroup that contains no points at infinity (e.g. in the prime-order subgroup), then it is safe to use these curves. However, there is not an obvious way to make this section’s decoding formulas restrict to a subgroup.

Furthermore, this format is not compatible with the fast, simple Montgomery ladder on Montgomery curves. We will remedy these problems using a slightly more complex encoding.

---

<sup>1</sup> When  $a = 1$ , there is also exactly one representative where  $x$  and  $y$  are both non-negative, and  $x$  is nonzero, which could make for a simpler encoding.

## 4 A Solution from the Jacobi Quartic

On the Jacobi quartic  $\mathcal{J}_{a^2, a-2d}$ , the coset of the 2-torsion group with respect to  $P = (s, t)$  is exactly

$$\mathcal{J}[2] + P = \{(s, t); (-s, -t); (1/as, -t/as^2); (-1/as, t/as^2)\}$$

So a similar solution applies on  $\mathcal{J}$  modulo its 2-torsion: we can encode a point  $P$  by the  $s$ -coordinate of the coset representative  $(s, t)$ , where  $s$  is non-negative and finite, and  $t/s$  is non-negative or infinite<sup>2</sup>. Call this encoding  $\text{enc}_{\mathcal{J}}(P)$ , and call the corresponding decoding algorithm  $\text{dec}_{\mathcal{J}}$ . Note that the identity point encodes to  $0 \in \mathbb{F}$ .

### 4.1 From the Jacobi Quartic to Edwards Curves

The curves  $\mathcal{E}_{a,d}$  and  $\mathcal{J}_{a^2, a-2d}$  are isogenous by the map

$$\phi_a(s, t) = \left( \frac{2s}{1+as^2}, \frac{1-as^2}{t} \right) \text{ with dual } \bar{\phi}_a(x, y) = \left( \frac{x}{y}, \frac{2-y^2-ax^2}{y^2} \right)$$

Note that swapping  $(a, d)$  with  $(-a, d-a)$  results in the same curve  $\mathcal{J}_{a^2, a-2d}$ , and gives an isogeny  $\phi_{-a}$  to the curve  $\mathcal{E}_{-a, d-a}$ .

We will need the following lemma, whose trivial proof is omitted:

**Lemma 1.** *Let  $\phi$  be a homomorphism from an abelian group  $\mathbb{G}$  to another abelian group  $\mathbb{H}$ , and let  $\mathbb{G}'$  be a subgroup of  $\mathbb{G}$ . Then  $\phi$  acts as a well-defined homomorphism from  $\mathbb{G}/\mathbb{G}'$  to  $\phi[\mathbb{G}]/\phi[\mathbb{G}']$  which is a subgroup of  $\mathbb{H}/\phi[\mathbb{G}']$ . Furthermore, if  $\ker \phi \subseteq \mathbb{G}'$ , then  $\phi$  acts as an isomorphism between these groups.*

Since the isogeny  $\phi_a$  is a group homomorphism whose kernel is in  $\mathcal{J}[2]$ , we can extend the encoding on  $\mathcal{J}/\mathcal{J}[2]$  to an encoding on  $\phi_a[\mathcal{J}]/\phi_a[\mathcal{J}[2]]$ :

$$\text{enc}(P) := \text{enc}_{\mathcal{J}}(\phi_a^{-1}(P)) \text{ with } \text{dec}(b) := \phi_a(\text{dec}_{\mathcal{J}}(b))$$

The lemma shows that both encoding and decoding are well-defined. In particular,  $P$  has two preimages under  $\phi_a$ , but they represent the same element of  $\mathcal{J}/\mathcal{J}[2]$  and have the same encoding under  $\text{enc}_{\mathcal{J}}$ .

Let  $\psi(\mathcal{E})$  denote the group  $\phi_a[\mathcal{J}]/\phi_a[\mathcal{J}[2]]$ . If the 4-torsion group of  $\mathcal{E}$  is cyclic, then  $\psi(\mathcal{E})$  is more simply expressed as  $2\mathcal{E}/\mathcal{E}[2]$ .

### 4.2 Encoding

When encoding from  $\psi(\mathcal{E})$ , we are given a point  $P = (x, y)$  in the image of  $\phi_{aq}$  on  $\mathcal{E}$ . We need to efficiently compute  $s$  where  $(s, t) = \phi_a^{-1}(x, y)$ . We know that

$$\begin{aligned} x &= 2s/(1+as^2) \\ \text{so } s &= (1 \pm \sqrt{1-ax^2})/ax \end{aligned}$$

<sup>2</sup> Checking the sign of  $s/t$  works about as well as  $t/s$  in our formulas; the choice of  $t/s$  is more or less arbitrary.

Also,

$$\begin{aligned}
 y &= (1 - as^2)/t \\
 \text{so } t/s &= (1 - as^2)/sy \\
 &= \mp 2\sqrt{1 - ax^2}/xy
 \end{aligned}$$

It turns out to be particularly straightforward to compute this encoding from the popular extended homogeneous coordinates. Explicit formulas are given in Appendix A.1.

### 4.3 Decoding

To decode, we are given  $s$  and must compute

$$(x, y) = \left( \frac{2s}{1 + as^2}, \frac{1 - as^2}{\sqrt{a^2s^4 + (2a - 4d)s^2 + 1}} \right)$$

with the square root  $t$  taken so that  $t/s$  is non-negative. This requires the “inverse square root trick” to compute  $1/s$  and  $t$  at the same time, with care to avoid division by 0. The exact formulas are given in Appendix A.2. The input must be rejected if  $s$  is negative or if it is not a field element (eg. if it is the binary encoding of a number  $\geq p$ ), or if the square root doesn’t exist.

It is simplest to decode to projective form, so that the denominators need not be cleared. It is also relatively easy to decode to affine form by batching a computation of  $1/(1 + as^2)$  with the square root. Decoded points always have a well-defined affine form on curves with cofactor exactly 4, because those curves have no points at infinity in the image of  $\phi_a$ .

### 4.4 Completeness

Importantly, if the cofactor of  $\mathcal{J}$  is exactly 4, then the image  $\psi(\mathcal{E})$  contains no points at infinity. An easy way to see this is that if  $\phi_a(s, t)$  were at infinity, then  $\bar{\phi}_a(\phi_a(s, t))$  would be either at infinity or at  $(0, -1)$ . In either case, it would be a nontrivial 2-torsion point [21]. But it cannot be a 2-torsion point, because  $\bar{\phi}_a \circ \phi_a$  is the doubling map on  $\mathcal{J}$  (by definition of an isogeny), and its image is exactly the subgroup of order  $q$ .

### 4.5 Equality

Ordinarily, testing for equality in a quotient group  $\mathbb{G}/\mathbb{H}$  requires testing whether  $P = Q + H$  for each  $H \in \mathbb{H}$ . But if the cofactor is exactly 4, then equality testing is actually easier on  $\psi(\mathcal{E})$  than on  $\mathcal{E}$ . In this case, two points  $(X_1 : Y_1 : Z_1 : T_1)$  and  $(X_2 : Y_2 : Z_2 : T_2)$  are equal if and only if

$$X_1 \cdot Y_2 = X_2 \cdot Y_1$$

This is because  $X/Y$  is the  $s$ -coordinate of the image  $Q$  of  $\bar{\phi}_a(X : Y : Z : T)$  on  $\mathcal{J}$ . The only other point with that  $s$ -coordinate has a nontrivial 2-torsion component (it is  $(0, -1) - Q$ ), but the image  $(\bar{\phi}_a \circ \phi_a)[\mathcal{J}]$  is the prime-order subgroup  $\mathcal{J}[q]$ .

In particular, for a curve of cofactor exactly 4, a point  $(X : Y : Z : T)$  is equal to the identity precisely when  $X = 0$ .

## 4.6 Security

Using Decaf gives the security benefits of a prime-order group without weakening well-studied cryptographic assumptions. In particular:

- The discrete logarithm problem is equivalent on  $\mathcal{E}, \mathcal{J}$  and  $\psi(\mathcal{E})$ . The same is true for computational Diffie-Hellman, gap DH, static DH, strong DH, and should hold for similar computation problems.
- If the Decaf group  $\psi(\mathcal{E})$  has prime order  $q$ , then the DDH problem is equivalent on  $\mathcal{E}[q], \mathcal{J}[q]$  and  $\psi(\mathcal{E})$ . The same is true for decision linear, and should hold for similar decision problems. These decision problems are easy on groups with a small cofactor, such as  $\mathcal{E}$  itself.

The straightforward proofs of these reductions are omitted.

## 4.7 Batch Encoding

On a server which needs to generate signatures and/or ephemeral keys at prodigious rates, it may be advantageous to batch the point encoding algorithm.

The encoding algorithm listed above cannot be batched easily because of the inverse square root computation. However, the square root can be avoided if we wish to compress  $2P$  instead of  $P$ , that is, if  $P$  is computed as  $(k/2 \bmod q) \cdot B$  instead of  $k \cdot B$ . In this case, we can simply evaluate the dual 2-isogeny  $\bar{\phi}$  from  $\mathcal{E}$  to  $\mathcal{J}$ :

- Compute  $1/(xy)$  and  $t/s = (2 - y^2 - ax^2)/xy$ .
- If  $t/s$  is non-negative, then output  $|s| = |x/y| = |x^2/xy|$ .
- Otherwise output  $|1/s| = |y/x| = |y^2/xy|$ .

The computation of  $1/(xy)$  can be batched over multiple points using Montgomery's trick.

## 4.8 Performance

Overall, Decaf's performance is very similar to a traditional point compression scheme. Encoding and decoding take one field exponentiation each.

A comparison to existing point encoding algorithms is shown in Fig. 1. It shows:

- The encoding and decoding costs.
- The cost to clear the cofactor if one remains.

Encoding	enc cost	dec cost	clear $h$	order	factor	size
$(x, y)$	$1I + 2M$	$3M$	$12M$	$4q \rightarrow q$	4	$2\lceil \lg  \mathbb{F}  \rceil$
$(x, \text{sign } y)$	$1I + 2M$	$1I_2 + 3M$	$12M$	$4q \rightarrow q$	4	$\lceil \lg  \mathbb{F}  \rceil + 1$
$(x, \text{sign } y)$ , check	$1I + 2M$	$> 2I_2 + L$	0	$q$	1	$\lceil \lg  \mathbb{F}  \rceil + 1$
First proposal	$1I + 2M$	$1I_2 + 3M$	0	$4q$	1	$\lceil \lg  \mathbb{F}  \rceil - 1$
Second proposal	$1I_2 + 7M$	$1I_2 + 10M$	0	$2q$	1	$\lceil \lg  \mathbb{F}  \rceil - 1$
Batchable	$1I + 6M$	$1I_2 + 10M$	0	$2q$	2	$\lceil \lg  \mathbb{F}  \rceil - 1$

**Fig. 1.** Cost of encoding and decoding algorithms.  $M$  = multiply,  $I$  = inversion,  $I_2$  = inverse square root,  $L$  = Legendre symbol. Squarings are treated as  $0.8M$  and multiplies by constants as  $0.2M$ , but columns are rounded to the nearest  $M$ .

- The order of the resulting points on the curve, with  $4q \rightarrow q$  meaning a cofactor that will most likely be cleared.
- The extra factor induced by encoding and cofactor clearing.
- The size in bits of the encoding’s codomain.

If inversion  $I$  is implemented using Fermat’s little theorem, it is likely to be slightly more expensive than an inverse square root  $I_2$ . In practice, implementations that need both  $I$  and  $I_2$  with  $|\mathbb{F}| \equiv 3 \pmod{4}$  often implement inversion as  $x/(\pm\sqrt{x^2})^2$ , costing  $M + 2S$  more, and this is usually close to optimal anyway.<sup>3</sup>

The  $(x, y)$  method is uncompressed, and  $(x, \text{sign } y)$  is classically compressed. These methods do not remove the cofactor, so many protocols will remove it at the cost of two doublings  $\approx 12M$ . This changes the order of the internal points from  $4q$  to  $q$ . The third row is compression with order checking. The order checking can be accomplished by inverting a 2-isogeny twice: the first inversion requires an inverse square root, but the second requires only checking that the root exists, i.e. computing a Legendre symbol.

The first proposal (Sect. 3) is a quotient group on an untwisted Edwards curve. It is slightly more expensive on a twisted Edwards curve, and is dangerous for such curves when  $|\mathbb{F}| \equiv 3 \pmod{4}$  because the internal points can have order 4. The second proposal (Sect. 4) avoids this problem, and gives an encoding compatible with several curve models, but at the cost of about 8 extra field multiplications and correspondingly higher complexity.

A downside of methods which include an inverse square root  $I_2$  is that they cannot use an EGCD-based inversion method. They also cannot be batched using Montgomery’s batch inversion trick, which accomplishes  $N$  inversions using one inversion and  $3(N - 1)$  multiplications. The batchable encoding method (Sect. 4.7) replaces the inverse square root in encoding with an inversion but multiplies by an extra factor of 2.

<sup>3</sup> We tested a simple dynamic program which computes 3-register powering ladders, and it gave this technique for NIST P-192, P-256, P-384, P-521, and also for  $2^{414} - 17$  (Curve41417) and  $2^{448} - 2^{224} - 1$  (Ed448-Goldilocks).

It is seen that our methods cost less in total than point compression plus clearing the cofactor. Even for operations which do not need to clear the cofactor (eg. key generation), the overhead from our encoding is relatively small. Fast key generation operations cost on the order of 3M per bit of the curve’s order, so the difference in encoding costs is well under 1% for cryptographically useful curves.

## 5 Compatibility with Montgomery Curves

The Montgomery ladder on Montgomery curves is a very simple and fast way to implement scalar multiplication for Diffie-Hellman (DH) key exchange. In its simplest form, the ladder discards sign information, making it inherently incompatible with any point encoding format that conveys sign information. Furthermore, it does not distinguish between the curve and its quadratic twist, necessitating the use of twist-safe curves [3]. However, we would like to interoperate with the Montgomery ladder with minimal changes. For example, if a protocol uses a  $(u, \text{sign } v)$  format, then the ladder can be modified to compute the sign, or the protocol can be changed to discard the sign bit for DH outputs.

We will show how to use Decaf with the Montgomery ladder on the curve

$$\mathcal{M}_{a,2-4d/a} : av^2 = u \cdot (u^2 + (2 - 4d/a) \cdot u + 1)$$

where conveniently the value of  $(A + 2)/4$  is  $1 - d/a$ . The curve  $\mathcal{M}_{a,2-4d/a}$  is isogenous to  $\mathcal{J}_{a^2,a-2d}$  by the maps

$$\phi(s, t) = \left( \frac{1}{as^2}, -\frac{t}{as^3} \right) \text{ and } \bar{\phi}(u, v) = \left( \frac{1 - u^2}{2av}, \frac{a(u + 1)^4 + 8du(u^2 + 1)}{4a^2v^2} \right)$$

More simply,  $\phi(s, t) = (as^2, ts) + T_2$ , where the 2-torsion point  $T_2$  can be ignored due to the quotient. This means that Montgomery ladder implementations can take input in Decaf format, simply by starting the ladder at  $u = as^2$ .

When the ladder finishes, it is possible to efficiently encode the output point in the Decaf point format, including the correct sign information for  $v$ . However, recovering the sign information is complicated. Furthermore, it is possible to reject elements on the twist rather than on the curve, which the usual Montgomery ladder does not do, and it is possible to do all of this with only one field exponentiation (an inverse square root). This means that the Montgomery ladder will behave exactly the same as a standard decoding, scalar multiplication and encoding. We give the full details of how to do this in Appendix B. Some of the formulas in that section may be of independent interest.

It is also possible (and complicated) to do these things with existing point formats such as  $(u, \text{sign } v)$ , but almost no implementations do. Instead, since the Montgomery ladder is used almost exclusively for Diffie-Hellman, most implementations clear the cofactor and output only  $u$ , losing the information about  $v$ . This leaves the Montgomery ladder code very simple. It is also easy to do this

with the Decaf encoding, by clearing the cofactor<sup>4</sup> and outputting  $|1/\sqrt{au}|$ . The implementation should abort on  $u = 0$  and  $u = \infty$ , which lie in a small subgroup. This will also reject points on the twist, because even points on the twist have either  $u = 0, u = \infty$  or  $au$  nonsquare.

An Edwards or twisted Edwards implementation can interoperate with this simpler behavior simply by computing  $|s| = |x/y|$ , instead of encoding any sign information.

## 6 Hashing to the Curve

Some protocols require a map from  $\mathbb{F}$  to a curve [10, 19, 23], either to build a hash function which is either indifferentiable from a random oracle, or at least suitable for encoding computational Diffie-Hellman (CDH) challenges in the random oracle model. We could do this by using Elligator 1 or 2 on  $\mathcal{E}$  or  $\mathcal{M}$ . Since Decaf only operates subgroups on these curves, we would need to double the output of the map to make sure it is in the subgroup.

However, there is a better solution. We can instead use Elligator 2 on the Jacobi quartic  $\mathcal{J}$ , since it has a point of order 2. Then we can translate this point to the Edwards and Montgomery curves using the isogeny. That way, the groups and maps implemented by these curves are all compatible. The formulas for Elligator 2 are found in Appendix C. It is important to note that Elligator 2 provides a 1:1 map to a group of order  $h \cdot q$ , not of order  $(h/4) \cdot q$ . Therefore, the map be up to 4:1 once the isogeny and quotient are applied.

This map is suitable for deriving CDH challenges from a random oracle. That is, it is still suitable for use in derivatives of BLS [10], SPAKE2 [1]<sup>5</sup>, SPEKE [23] and possibly Dragonfly [19]<sup>6</sup>. These protocols do not require a random oracle map to  $\mathbb{G}$ . They only require a map from strings to the curve which is at most  $k$ -to-1 for small  $k$ , hits at least a  $1/\ell$  fraction of the points for small  $\ell$ , and whose inverse is efficiently sampleable. When a full random oracle map to  $\mathbb{G}$  is required, Brier et al.’s result [12] shows that mapping two independently chosen field elements and adding them is sufficient.

It is still possible to use Elligator 2 as a partial steganographic encoding for public keys, as in EKE. One may invert the isogeny to obtain a point on  $\mathcal{J}$ , randomize its 2-torsion components, and apply the inverse map defined by Elligator 2. Unfortunately, this requires an extra randomization step and an extra inverse-square-root operation compared to the original Elligator 2.

<sup>4</sup> Clearing the cofactor takes one doubling on a cofactor-4 curve, because of the isogeny. Another option would be to quotient out the cofactor, by choosing the lexicographically greater of  $|\sqrt{u/a}|$  and  $|1/\sqrt{au}|$ . But this is more complex and doesn’t reject points on the twist.

<sup>5</sup> Replacing SPAKE2’s  $H(\text{password}) \cdot (M, N)$  with an Elligator-like map results in a protocol with the same properties but no static-CDH assumption.

<sup>6</sup> Dragonfly lacks a security proof, so we cannot actually be sure that any such map is suitable. But given the similarity to SPEKE this seems likely.

## 7 Future Work

We do not believe that Decaf is the last word in cofactor-reducing compression algorithms. It would be useful, for example, to eliminate the cofactor of 8 in Curve25519. Additionally, an improved encoding scheme with simpler formulas would make this technique more compelling.

## 8 Conclusion

We have shown a straightforward way to implement a prime-order group  $\mathbb{G}$  using Edwards, twisted Edwards, Jacobi quartic and Montgomery curves. All four curve shapes implement the same group and so are compatible, except that as usual it is complicated to make the Montgomery ladder retain sign information. Our technique is otherwise similar in complexity and performance to traditional point compression techniques, though it may improve performance by making faster curves safe. Furthermore, we have shown how to implement an Elligator-like map from  $\mathbb{F}$  to  $\mathbb{G}$ , which is also compatible with all 4 models.

**Acknowledgements.** The author thanks Mark Marson, Steven Galbraith and the anonymous reviewers for their editorial suggestions.

## A Explicit Formulas for Encoding and Decoding

### A.1 Encoding

We wish to compute

$$s = (1 \pm \sqrt{1 - ax^2})/ax \quad \text{and} \quad t/s = \mp 2\sqrt{1 - ax^2}/xy$$

We know from the curve equation that

$$(1 - ax^2) \cdot (1 - y^2) = 1 + ax^2y^2 - (y^2 + ax^2) = (a - d)x^2y^2$$

so that

$$\sqrt{1 - ax^2}/xy = \pm \sqrt{(a - d)/(1 - y^2)}$$

Observe that in extended homogeneous coordinates,

$$1/x^2 = (a - dy^2)/(1 - y^2) = (aZ^2 - dY^2)/(Z^2 - Y^2)$$

and therefore

$$1/x = (aZX - dYT)/(Z^2 - Y^2)$$

This leads to a relatively simple encoding formula given an inverse-square-root algorithm:

$$\begin{aligned} r &\leftarrow 1/\sqrt{(a - d) \cdot (Z + Y) \cdot (Z - Y)} \\ u &\leftarrow (a - d) \cdot r \\ r &\leftarrow -r \text{ if } -2 \cdot u \cdot Z \text{ is negative} \\ s &\leftarrow |u \cdot (r \cdot (aZ \cdot X - d \cdot Y \cdot T) + Y)/a| \end{aligned}$$



In theory, this formula has an exceptional case because it divides by 0 when  $y = \pm 1$ . But if the inverse square root function returns  $r = 0$  in this case, the correct answer  $s = 0$  will emerge.

### A.2 Decoding

To decode to  $\mathcal{E}_{a,d}$ , we must recover

$$(x, y) = \left( \frac{2s}{1 + as^2}, \frac{1 - as^2}{\sqrt{a^2s^4 + (2a - 4d)s^2 + 1}} \right)$$

where  $\sqrt{a^2s^4 + (2a - 4d)s^2 + 1}/s$  is non-negative. We can compute this as:

$$\begin{aligned} & \text{Reject unless } s = |s| \\ X & \leftarrow 2 \cdot s \\ Z & \leftarrow 1 + a \cdot s^2 \\ u & \leftarrow Z^2 - 4d \cdot s^2 \\ v & \leftarrow \begin{cases} 1/\sqrt{u \cdot s^2} & \text{if } u \cdot s^2 \text{ is square and nonzero} \\ 0 & \text{if } u \cdot s^2 = 0 \\ \text{[reject]} & \text{if } u \cdot s^2 \text{ is not square} \end{cases} \\ v & \leftarrow -v \text{ if } u \cdot v \text{ is negative} \\ w & \leftarrow v \cdot s \cdot (2 - Z) \\ w & \leftarrow w + 1 \text{ if } s = 0 \\ Y & \leftarrow w \cdot Z \\ T & \leftarrow w \cdot X \\ P & \leftarrow (X : Y : Z : T) \end{aligned}$$

The special case for  $s = 0$  is required to decode  $s = 0$  to the identity point instead of the point  $(0, 0)$ , which isn't on the curve.

## B The Modified Montgomery Ladder

We use a modified version of the traditional Montgomery ladder on Montgomery curves [28]. This uses our knowledge of  $s_0$  such that the initial state  $u_0 = as_0^2$ . It converts

$$(s_0, U_1 : Z_1, U_2 : Z_2) \text{ where } (s_0 : v_0 : 1) + (U_1 : V_1 : Z_1) - (U_2 : V_2 : Z_2) = 0_{\mathcal{M}}$$

for some unknown  $v_0, V_1, V_2$ , to

$$(s_0, U_3 : Z_3, U_4 : Z_4)$$

where

$$(U_4 : V_4 : Z_4) = 2(U_2 : V_2 : Z_2) \text{ and } (U_3 : V_3 : Z_3) = (U_1 : V_1 : Z_1) + (U_2 : V_2 : Z_2)$$

again with the  $V$  components unknown. Note that on  $\mathcal{M}_{a,2-4d/a}$ , the coefficient  $(A + 2)/4 = 1 - d/a$ . The ladder works as follows:

$$\begin{aligned} E &\leftarrow U_2 + Z_2 \\ F &\leftarrow U_2 - Z_2 \\ G &\leftarrow U_1 + Z_1 \\ H &\leftarrow U_1 - Z_1 \\ K &\leftarrow E^2 - F^2 \\ U_3 &\leftarrow (EH + FG)^2 \\ Z_3 &\leftarrow a \cdot (s_0 \cdot (EH - FG))^2 \\ U_4 &\leftarrow E^2 \cdot F^2 \\ Z_4 &\leftarrow K \cdot (F^2 + (1 - d/a) \cdot K) \end{aligned}$$

The computation of the common subexpressions  $EH, FG, E^2$  and  $F^2$  has been removed for brevity. The salient feature of this ladder is that it computes as intermediate results  $S_3$  and  $W_3$  which are square roots of  $X_3$  and  $Z_3/a$  respectively. Our implementation allocates its temporary variables in a way that prevents these intermediates from being overwritten until the next iteration of the ladder. This means that they will be available to the encoding algorithm.

### B.1 Encoding

At the end of the Montgomery ladder, we need to encode the point stored in  $U_4 : Z_4$ . Recall that a point on  $\mathcal{M}_{a,2-4d/a}$  is the image of an isogeny from  $\mathcal{J}_{a^2, a-2d}$ :

$$\phi((s, t)) = (as^2, st) + T_2 \text{ with inverse } \phi^{-1}((u, v) + T_2) = (\sqrt{u/a}, v/\sqrt{u/a})$$

Because the encoding is the same modulo 2-torsion components, we can ignore  $T_2$  completely. The principal difficulty of encoding is to simultaneously determine  $\sqrt{u_2}$  and its inverse from the ladder state, and whether  $t_2/s_2 = av_2/u_2$  is positive given that  $t_0/s_0 = av_0/u_0$  is known to be positive. Fortunately, the invariants of the ladder make this possible. From [17], Appendices A.1 and A.2, we know that if

$$(u_0, v_0) + (u_1, v_1) + (u_2, v_2) = 0 \text{ on } \mathcal{M}$$

that is, if these points lie on a line, then

$$2av_1v_2 = (u_1u_2 + 1)(u_1 + u_2) - u_0(u_1 - u_2)^2 + 2Au_1u_2 \tag{1}$$

and symmetrically, and also

$$4(u_0 + u_1 + u_2 + A)(u_0u_1u_2) = (1 - u_0u_1 - u_1u_2 - u_0u_2)^2 \tag{2}$$

Adding (2) to  $2u_0 \cdot$  (1), we have

$$4au_0v_1v_2 = (1 - u_1u_2 - u_0u_2 + u_0u_1) \cdot (1 - u_1u_2 + u_0u_2 - u_0u_1) \tag{3}$$

and symmetrically. Multiplying and dividing symmetric copies of this equation, we obtain

$$4au_0u_1v_2^2/u_2 = (1 - u_1u_2 - u_0u_2 + u_0u_1)^2$$

so that

$$\frac{v_2}{u_2} = \frac{1 - u_1u_2 - u_0u_2 + u_0u_1}{\pm 2\sqrt{au_0u_1u_2}}$$

This equation cannot determine the sign of the square root, but (3) shows it to be consistent for all three points on the line.

This means that it is enough to compute  $\pm 1/\sqrt{au_0u_1u_2}$ . This will allow us to determine  $av_0/u_0$  to adjust the sign of the square root. It will allow us to check whether  $av_2/u_2$  is negative, in which case we should output  $1/\sqrt{au_2}$  instead of  $\sqrt{u_2/a}$ . Furthermore, the input point  $s_0$  is  $\sqrt{u_0/a}$ , and the modified Montgomery ladder state contains either  $\sqrt{au_1}$  or  $\sqrt{au_2}$ , depending on the last bit of the ladder. This allows us to compute  $\sqrt{u_2/a}$  or its inverse from the ladder state and  $1/\sqrt{au_0u_1u_2}$  with no additional field exponents. In the actual computation,  $u_1$  and  $u_2$  are given in projective form, but this does not greatly complicate matters because the equations are nearly homogeneous.

*Special Cases.* If  $u_2$  is zero or infinite, then it is a 2-torsion point and the output is zero. Likewise if  $u_0 = 0$ , then the base point is the identity and again the output will always be zero. If  $u_1$  is zero or infinite, then the output is either the initial point or its inverse, depending on whether the last step in the ladder swapped  $u_1$  and  $u_2$ . So the output should be either  $s_0$  or  $|1/as_0|$ .

*Twist Rejection.* The above procedure will reject points which are on the twist of  $\mathcal{M}$  instead of on  $\mathcal{M}$  itself, because  $au_0u_1u_2$  will not be square for such points. However, there is one sticking point: if the secret key is 0 or  $\pm 1$  modulo the twist’s group order  $q'$ , then  $u_1$  or  $u_2$  may be 0 even if  $u_0 \neq 0$ . This would make  $au_0u_1u_2 = 0$  a square number. This quirk is unlikely to apply in a real system, but it is still worth avoiding.

To fix this issue, we can test whether  $au_0u_1u_2 = 0$ ; if so, the special case above is used to determine the sign, so the output of the square root doesn’t matter. Therefore, the square root can be changed to  $\sqrt{u_0^2 + Au_0 + 1}$  to determine whether the input point was on the curve or not.

## C Elligator 2

The Jacobi quartic  $\mathcal{J}_{a^2, a-2d}$  is birationally equivalent to the Legendre curve

$$ay^2 = x(x-1)(x-d/a)$$

by the maps

$$(s, t) = \left( \frac{ax-d}{a^2y}, \frac{ax^2-2dx+d}{ax(x-1)} \right); \quad (x, y) = \left( \frac{as^2+t+1}{2as^2}, \frac{as^2+t+1-2ds^2}{2a^2s^3} \right)$$

Since the Legendre curve has 3 points of order 2, we have a choice of which to use as the point of order 2 in Elligator 2. It turns out to be best for symmetry if we use  $(d/a, 0)$ .

The Elligator 2 map  $E$  to  $\mathcal{J}_{a^2, a-2d}$  starts with a fixed quadratic nonresidue  $n$  and an input  $r_0 \in \mathbb{F}$ , and begins by computing a probable nonresidue  $r := nr_0^2$ . It then returns

$$(s, t) = \left( +\sqrt{\frac{(r+1)(a-2d)}{(dr+a-d)(dr-ar-d)}}, \frac{-(r-1)(a-2d)^2}{(dr+a-d)(dr-ar-d)} - 1 \right)$$

or

$$(s, t) = \left( -\sqrt{\frac{r(r+1)(a-2d)}{(dr+a-d)(dr-ar-d)}}, \frac{r(r-1)(a-2d)^2}{(dr+a-d)(dr-ar-d)} - 1 \right)$$

for whichever case the square root is defined, prioritizing the second case if  $r = 0$  and both are square. Here we take  $+\sqrt{\cdot}$  to mean the non-negative square root, and  $-\sqrt{\cdot}$  to mean its negation. This formulation seems to have two advantages over other ways to formulate Elligator 2 for this curve. First, the two cases are very similar, requiring only very small adjustments for the square vs. nonsquare cases. Second, the formula is invariant under the parameter involution  $(a, d) \leftrightarrow (-a, d - a)$  which preserves  $\mathcal{J}$  but swaps the untwisted and twisted Edwards curves. This means that if one implementor uses  $(a, d)$  and another chooses  $(-a, d - a)$ , then even their Elligator 2 implementations will remain compatible.

The map to  $\mathcal{J}$  can easily be computed as follows:

$$\begin{aligned} r &\leftarrow nr_0^2 \\ D &\leftarrow (dr + a - d) \cdot (dr - ar - d) \\ N &\leftarrow (r + 1) \cdot (a - 2d) \\ c, e &\leftarrow \begin{cases} +1, & 1/\sqrt{ND} \text{ if } ND \text{ is square} \\ -1, nr_0/\sqrt{nND} & \text{otherwise} \end{cases} \\ s &\leftarrow c \cdot |N \cdot e| \\ t &\leftarrow -c \cdot N \cdot (r - 1) \cdot ((a - 2d) \cdot e)^2 - 1 \end{aligned}$$

Note that if  $D$  or  $N$  is 0, the result should be some 2-torsion point ether at  $(0, \pm 1)$  or at infinity. In fact, a naïve inverse square root algorithm will return  $e = 0$  in this case, resulting in the point  $(0, -1)$  which is indeed a 2-torsion point. Since we are quotienting out the 2-torsion group, this result is satisfactory.

To map to the Edwards or Montgomery curves, one simply applies the isogeny from  $\mathcal{J}$ .

To invert the Elligator 2 map, let  $c = \text{sign } s$  and note that

$$nr_0^2 = r = \frac{(2d - a)s^2 + c(t + 1)}{(2d - a)s^2 - c(t + 1)}$$

If  $\sqrt{r/n}$  exists, then it is the inverse; otherwise, there is no inverse. Since the group in question isn't  $\mathcal{J}$  but rather  $\mathcal{J}/\mathcal{J}[2]$ , to ensure a random sample  $E^{-1}(P)$ , one must add a random 2-torsion element to  $P$ .

## References

1. Abdalla, M., Pointcheval, D.: Simple password-based encrypted key exchange protocols. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 191–208. Springer, Heidelberg (2005)
2. Ahmadi, O., Granger, R.: On isogeny classes of edwards curves over finite fields. Cryptology ePrint Archive, Report 2011/135 (2011). <http://eprint.iacr.org/2011/135>
3. Bernstein, D.J.: Curve25519: new Diffie-Hellman speed records. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.) PKC 2006. LNCS, vol. 3958, pp. 207–228. Springer, Heidelberg (2006)
4. Bernstein, D.J., Birkner, P., Joye, M., Lange, T., Peters, C.: Twisted edwards curves. In: Vaudenay, S. (ed.) AFRICACRYPT 2008. LNCS, vol. 5023, pp. 389–405. Springer, Heidelberg (2008)
5. Bernstein, D.J., Lange, T.: Faster addition and doubling on elliptic curves. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 29–50. Springer, Heidelberg (2007)
6. Bernstein, D., Lange, T.: Complete addition laws for elliptic curves (2009). <http://cr.yp.to/talks/2009.04.17/slides.pdf>
7. Bernstein, D.J., Duif, N., Lange, T., Schwabe, P., Yang, B.-Y.: High-speed high-security signatures. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 124–142. Springer, Heidelberg (2011)
8. Billet, O., Joye, M.: The Jacobi model of an elliptic curve and side-channel analysis. In: Fossorier, M.P.C., Høholdt, T., Poli, A. (eds.) AAECC 2003. LNCS, vol. 2643, pp. 34–42. Springer, Heidelberg (2003)
9. Boneh, D., Halevi, S., Hamburg, M., Ostrovsky, R.: Circular-secure encryption from decision Diffie-Hellman. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 108–125. Springer, Heidelberg (2008)
10. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the weil pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, p. 514. Springer, Heidelberg (2001)
11. Bos, J.W., Costello, C., Longa, P., Naehrig, M.: Selecting elliptic curves for cryptography: an efficiency and security analysis. Cryptology ePrint Archive, Report 2014/130 (2014). <http://eprint.iacr.org/>
12. Brier, E., Coron, J.-S., Icart, T., Madore, D., Randriam, H., Tibouchi, M.: Efficient indifferentiable hashing into ordinary elliptic curves. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 237–254. Springer, Heidelberg (2010)
13. Clarke, D., Hao, F.: Cryptanalysis of the dragonfly key exchange protocol. IET Inf. Secur. **8**(6), 283–289 (2014)
14. Edwards, H.M.: A normal form for elliptic curves. Bull.-Am. Math. Soc. **44**(3), 393 (2007)
15. Farashahi, R.R., Joye, M.: Efficient arithmetic on hessian curves. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 243–260. Springer, Heidelberg (2010)
16. Goh, E.-J., Jarecki, S., Katz, J., Wang, N.: Efficient signature schemes with tight reductions to the Diffie-Hellman problems. J. Crypt. **20**(4), 493–514 (2007)
17. Hamburg, M.: Fast and compact elliptic-curve cryptography. Cryptology ePrint Archive, Report 2012/309 (2012). <http://eprint.iacr.org/2012/309>
18. Hao, F., Ryan, P.: J-PAKE: authenticated key exchange without PKI. Cryptology ePrint Archive, Report 2010/190 (2010). <http://eprint.iacr.org/>

19. Harkins, D.: Dragonfly: a pake scheme (2012). <http://www.ietf.org/proceedings/83/slides/slides-83-cfrg-0.pdf>
20. Hisil, H., Wong, K., Carter, G., Dawson, E.: Faster group operations on elliptic curves. Cryptology ePrint Archive, Report 2007/441 (2007). <http://eprint.iacr.org/>
21. Hisil, H., Wong, K.K.-H., Carter, G., Dawson, E.: Jacobi quartic curves revisited. Cryptology ePrint Archive, Report 2009/312 (2009). <http://eprint.iacr.org/2009/312>
22. Hisil, H., Wong, K.K.-H., Carter, G., Dawson, E.: Twisted edwards curves revisited. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 326–343. Springer, Heidelberg (2008)
23. Jablon, D.P.: Strong password-only authenticated key exchange. ACM SIGCOMM Comput. Commun. Rev. **26**(5), 5–26 (1996)
24. Krawczyk, H.: HMQV: a high-performance secure Diffie-Hellman protocol. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 546–566. Springer, Heidelberg (2005)
25. Law, L., Menezes, A., Minghua, Q., Solinas, J., Vanstone, S.: An efficient protocol for authenticated key agreement. Des. Codes Crypt. **28**(2), 119–134 (2003)
26. Liardet, P., Smart, N.P.: Preventing SPA/DPA in ECC systems using the Jacobi form. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, p. 391. Springer, Heidelberg (2001)
27. Menezes, A., Ustaoglu, B.: On the importance of public-key validation in the MQV and HMQV key agreement protocols. In: Barua, R., Lange, T. (eds.) INDOCRYPT 2006. LNCS, vol. 4329, pp. 133–147. Springer, Heidelberg (2006)
28. Montgomery, P.: Speeding the pollard and elliptic curve methods of factorization. Math. Comput. **48**(177), 243–264 (1987)

# Actively Secure OT Extension with Optimal Overhead

Marcel Keller<sup>(✉)</sup>, Emmanuela Orsini, and Peter Scholl

Department of Computer Science, University of Bristol, Bristol, UK  
{m.keller, emmanuela.orsini, peter.scholl}@bristol.ac.uk

**Abstract.** We describe an actively secure OT extension protocol in the random oracle model with efficiency very close to the passively secure IKNP protocol of Ishai et al. (Crypto 2003). For computational security parameter  $\kappa$ , our protocol requires  $\kappa$  base OTs, and is the first practical, actively secure protocol to match the cost of the passive IKNP extension in this regard. The added communication cost is only additive in  $O(\kappa)$ , independent of the number of OTs being created, while the computation cost is essentially two finite field operations per extended OT. We present implementation results that show our protocol takes no more than 5% more time than the passively secure IKNP extension, in both LAN and WAN environments, and thus is essentially optimal with respect to the passive protocol.

## 1 Introduction

Oblivious transfer (OT) is a fundamental primitive in cryptography, used in the construction of a range of protocols. In particular, OT is sufficient and necessary for secure multi-party computation [10, 15, 25], and is also often used in special-purpose protocols for tasks such as private set intersection [22]. Due to a result of Impagliazzo and Rudich [11] it is very unlikely that OT is possible without the use of public-key cryptography, so all OT constructions have quite a high cost when used in a practical context.

**OT Extension.** Since OT requires public key machinery, it is natural to wonder whether OT can be efficiently ‘extended’. That is, starting with a small number of ‘base OTs’, create many more OTs with only symmetric primitives, somewhat analogous to the use of hybrid encryption to extend public key encryption. Beaver [3] first showed how, starting with  $\kappa$  base OTs, one could create  $\text{poly}(\kappa)$  additional OTs using only symmetric primitives, with computational security  $\kappa$ . Beaver’s protocol is very elegant, but requires evaluation of pseudo-random generators within Yao’s garbled circuits; therefore it is highly impractical. In 2003, Ishai, Kilian, Nissim and Petrank [12] presented a very efficient protocol for extending OTs, requiring only black-box use of symmetric primitives and  $\kappa$  base OTs. Concretely, the main cost of their basic protocol is computing and sending just *two* hash function values per OT. Asharov et al. [1] gave several algorithmic

optimizations to the IKNP protocol, reducing the communication down to one hash value for the *random OT* variant, where the sender’s messages are sampled at random, and using a cache-oblivious algorithm for matrix transposition, which turned out to be the computational bottleneck when implemented naively. By analysing an implementation, they suggest that the actual bottleneck of the IKNP protocol is communication, particularly in wide area networks (WANs) with high latency and low bandwidth.

The above protocols are only secure against passive adversaries, who are trusted to strictly follow the protocol. For the case of actively secure protocols, which remain secure against arbitrary deviations from the protocol, typically most cryptographic protocols have a much greater cost than their passive counterparts. The actively secure OT extension of Ishai et al. [12] uses an expensive *cut-and-choose* technique, where  $s$  runs of the protocol are done in parallel to achieve  $O(s)$  bits of statistical security. In recent years, the cost of actively secure OT extension has improved greatly, down to just constant overhead. The TinyOT protocol for secure two-party computation [20] is based on a very efficient OT extension where the total cost is roughly  $\frac{8}{3}$  times the passively secure IKNP extension – this applies to communication and computation, as well as the number of base OTs required to run the protocol. Very recently, in an independent and concurrent work, Asharov et al. [2] gave a protocol reducing the overhead even further: their protocol requires roughly  $\kappa + s$  base OTs for computational security  $\kappa$  and statistical security  $s$ , which in practice reduces the constant from  $\frac{8}{3} \approx 2.7$  down to  $\approx 1.4$ , plus an additive overhead in  $O(\kappa)$ .

*Applications of OT Extension.* As we mentioned before, OT extension has been getting a lot of attention recently, because the efficiency of this procedure plays a decisive role in the overall efficiency of a number of protocols for secure computations where the number of OTs needed is very large, for example in the two-party and multiparty TinyOT protocols [5, 18, 20], in the MiniMAC protocol of Damgård et al. [7] and in private set intersection protocols [8, 22].

**Our Contributions.** In this paper we give the first practical, actively secure OT extension protocol requiring only  $\kappa$  base OTs, matching the efficiency of the passively secure IKNP extension in this respect. For communication and computation costs, the overhead on top of IKNP is negligible: our protocol requires 2 finite field (of size  $\kappa$ ) operations per extended OT, plus a small communication overhead of  $O(\kappa)$  bits in a constant number of rounds, independent of the number of OTs being performed, which amortizes away when creating many OTs. We give extensive benchmarks (in both LAN and WAN settings) showing that the practical cost of our protocol for performing 10 million OTs is less than 6% more than the IKNP extension, and so is almost optimal. In contrast, the protocol of Asharov et al. [2] takes at least 80% more time than the passive protocol in the WAN setting and over 20% more in the LAN setting for  $2^{23}$  OTs, according to their implementation figures.

The comparison table below shows the concrete efficiency of various other OT extension protocols, in terms of the number of base OTs required and the



total communication and computation cost for creating  $\ell$  OTs. Our protocol is more efficient than all previous protocols in all of these measures. Note that these comparisons are for OT extensions on strings of length at least  $\kappa$  bits. For shorter strings, the passively secure protocol of Kolesnikov and Kumaresan [16] is more efficient, but it does not seem straightforward to apply our techniques to obtain an actively secure protocol in that setting. We also omit the protocol of Ishai et al. [13], since although asymptotically this has only a constant overhead, the protocol is based on the ‘MPC-in-the-head’ technique, which has not been shown to be practical.

**Table 1.** Concrete cost of OT extension protocols for producing  $\ell$  OTs of 128-bit strings with 128-bit computational and 40-bit statistical security parameters.

Protocol	Seed OTs	Comms.	Comp.	Security
[12]	128	$\ell \cdot 128$ bits	$2\ell$ hashes	passive, CRF
[12]	$> 5000$	$O(\ell \cdot \kappa \cdot s)$	$O(\ell \cdot s)$ hashing	active, CRF
[17]	323	$O(\ell \cdot \kappa^2)$	$O(\ell \cdot \kappa^2)$ XOR	active, CRF
[20]	342	$\ell \cdot 342$ bits + 43KB	$O(\ell)$ hashing	active, RO
[2]	170	$\ell \cdot 175$ bits + 22KB	$O(\ell)$ hashing	active, CRF
<b>This work</b>	128	$\ell \cdot 128$ bits + 10KB	$2\ell + 336$ hashes	active, RO

Our protocol is similar in structure to previous protocols [2, 20], in that we carry out one run of the passively secure IKNP extension, and then use a *correlation check* to enforce correct behaviour. As in the previous protocols, it is possible that a cheating receiver may pass our check, in which case some information on the sender’s secret is leaked. However, the leakage is such that we still only need  $\kappa$  base OTs, and then must sacrifice  $\kappa + s$  of the extended OTs produced from the IKNP extension, where  $s$  is a statistical security parameter, to ensure security. The check itself is extremely simple and only requires a constant number of hash computations on a fixed input length, unlike previous checks where the amount of data being hashed increases with the number of extended OTs (Table 1).

**Random Oracle Usage.** We prove our OT extension protocol secure in the random oracle model, used for a functionality  $\mathcal{F}_{\text{Rand}}$ , which securely generates random values, and the hash function  $H$  used to randomize the correlated OT outputs. For the function  $H$ , Ishai et al. [12] prove security of their protocol in the standard model, under the assumption that  $H$  is a *correlation robust function*. The protocol of Asharov et al. [2] is proven secure with the additional requirement that  $H$  satisfies some kind of leakage resilience, and it is conjectured that the protocol of Nielsen et al. [20] is also secure in this model.

Note that in the case of random OT, where the sender’s outputs are defined as randomly chosen by the functionality, the security of the protocol (using

the optimization of Asharov et al. [1], which cuts the communication in half) has only ever been proven in the random oracle model, because of the need for the simulator to program the receiver’s outputs from  $H$  to be as defined by the functionality. Random OT can be used for an offline/online scenario where random OTs are generated in advance of the inputs being known, and is also often used in practical protocols (e.g. [20,22]), so we take the pragmatic approach of using random oracles for our security proofs, which also simplifies the exposition. However, due to the similarities between our protocol and previous ones [2,20], we believe it is likely that our (non-random) OT extension protocol can also be proven secure under a form of correlation robustness for  $H$ .

## 2 Preliminaries

### 2.1 Notation

We denote by  $\kappa$  the computational security parameter and by  $s$  the statistical security parameter. We let  $\text{negl}(\kappa)$  denote some unspecified function  $f(\kappa)$ , such that  $f = o(\kappa^{-c})$  for every fixed constant  $c$ , saying that such a function is *negligible* in  $\kappa$ . We say that a probability is *overwhelming* in  $\kappa$  if it is  $1 - \text{negl}(\kappa)$ . We denote by  $a \stackrel{\$}{\leftarrow} A$  the random sampling of  $a$  from a distribution  $A$ , and by  $[d]$  the set of elements  $\{1, \dots, d\}$ .

Throughout the proofs we will often identify  $\mathbb{F}_2^\kappa$  with the finite field  $\mathbb{F}_{2^\kappa}$ . Addition is the same in both; we will use “ $\cdot$ ” for multiplication in  $\mathbb{F}_{2^\kappa}$  and “ $*$ ” for the componentwise product in  $\mathbb{F}_2^\kappa$ . We use lower case letters to denote elements in  $\mathbb{F}_2$  and bold lower case letters for vectors in  $\mathbb{F}_2^\kappa$  and elements in  $\mathbb{F}_{2^\kappa}$ . We will use the notation  $\mathbf{v}[i]$  to denote the  $i$ -th entry of  $\mathbf{v}$ .

Given a matrix  $A$ , we denote its rows by subindices  $\mathbf{a}_i$  and its columns by superindices  $\mathbf{a}^k$ . Given a vector  $\mathbf{v} \in \mathbb{F}_2^\kappa$ , we denote by  $\bar{\mathbf{v}}$  the vector in  $\mathbb{F}_2^\kappa$  such that  $\mathbf{v} + \bar{\mathbf{v}} = \mathbf{1}$ . We say that a vector  $\mathbf{v} \in \mathbb{F}_2^\kappa$  is *monochrome* if  $v[i] = v[j]$ , for each  $i, j \in [\kappa]$ ; otherwise we say it is *polychrome*.

In our proofs we often use the notion of affine space. We recall that an affine space is a set  $X$  that admits a free transitive action of a vector space  $V$ .

### 2.2 Oblivious Transfer and OT Extension

Oblivious transfer (OT) [4,9,23,24] is a two-party protocol between a *sender*  $S$  and a *receiver*  $R$ . The sender transmits part of its input to  $R$ , in such a way that  $S$  remains oblivious as what part of its input was transmitted and  $R$  does not obtain more information than it is entitled.

We use three main oblivious transfer functionalities. We denote by  $\mathcal{F}_{\text{OT}}$  the standard  $\binom{2}{1}$ -OT functionality, where the sender  $S$  inputs two messages  $\mathbf{v}_0, \mathbf{v}_1 \in \mathbb{F}_2^\kappa$ , and the receiver inputs a choice bit  $x$ , and at the end of the protocol  $R$  learns only the selected message  $\mathbf{v}_x$ . We use the notation  $\mathcal{F}_{\text{OT}}^{\kappa,\ell}$  to denote the functionality that provides  $\ell$   $\binom{2}{1}$ -OTs of messages in  $\mathbb{F}_2^\kappa$  (see Fig. 1 for a formal definition). Another variant of OT is correlated OT, where the sender’s messages

**Functionality  $\mathcal{F}_{\text{OT}}^{\kappa, \ell}$**

$\mathcal{F}$  running with  $R, S$  and an adversary  $\mathcal{S}$  proceeds as follows:

- The functionality waits for input  $(\mathbf{v}_{0,i}, \mathbf{v}_{1,i}) \in \mathbb{F}_2^\kappa \times \mathbb{F}_2^\kappa$ ,  $i \in [\ell]$ , from  $S$  and  $x_1, \dots, x_\ell$ , with  $x_i \in \mathbb{F}_2$ , from  $R$ .
- It outputs  $\mathbf{v}_{x_i,i}$ ,  $i \in [\ell]$ , to  $R$ .

**Fig. 1.** The OT functionality

**Functionality  $\mathcal{F}_{\text{COTe}}^{\kappa, \ell}$**

The functionality is parametrized by the number  $\ell$  of resulting OTs and by the key length  $\kappa$ .

Running with parties  $S, R$ , and an ideal adversary  $\mathcal{S}$  it operates as follows.

**Initialize:** Upon receiving  $\Delta$  from  $S$ , where  $\Delta \in \mathbb{F}_{2^\kappa}$ , the functionality stores  $\Delta$ .

**Extend:**

- Upon receiving  $(\mathbf{x}_1, \dots, \mathbf{x}_\ell)$  from  $R$ , where  $\mathbf{x}_i \in \mathbb{F}_{2^\kappa}$ , sample  $\mathbf{t}_j \in \mathbb{F}_{2^\kappa}$ ,  $j = 1, \dots, \ell$ , and output them to  $R$ . Compute  $\mathbf{q}_j = \mathbf{t}_j + \mathbf{x}_j * \Delta$ ,  $j = 1, \dots, \ell$ , and output them to  $S$ .
- If  $R$  is corrupt, wait for  $S$  to input  $\mathbf{t}_j$  and output as before.

**Fig. 2.** Correlated OT with errors functionality  $\mathcal{F}_{\text{COTe}}$

are correlated, i.e.  $\mathbf{v}_0 + \mathbf{v}_1 = \Delta$  for a fixed  $\Delta \in \mathbb{F}_2^\kappa$ ; in Fig. 3 we give a version of this functionality which allows “errors”. Finally, in the random OT functionality,  $\mathcal{F}_{\text{ROT}}$ , the messages  $\mathbf{v}_0, \mathbf{v}_1$  are sampled uniformly at random by the functionality (Fig. 7).

**IKNP Protocol Augmented with Errors.** In Fig. 2, we model the IKNP extension as a separate functionality,  $\mathcal{F}_{\text{COTe}}$  that incorporates a cheating receiver’s behavior, and call this *correlated OT with errors*. Figure 3 gives the implementation of this functionality: after the first phase and the local expansion of the seeds through a pseudorandom generator PRG,  $R$  holds two  $\ell \times \kappa$  matrices  $\{\mathbf{t}_0^i\}_{i \in [\kappa]}, \{\mathbf{t}_1^i\}_{i \in [\kappa]}$ , while  $S$  holds the vector  $\Delta \in \mathbb{F}_2^\kappa$  and the matrix  $\{\mathbf{t}_{\Delta_i}^i\}_{i \in [\kappa]}$ . In the extension phase, we allow a cheating receiver  $R$  to input vectors  $\mathbf{x}_1, \dots, \mathbf{x}_\ell \in \mathbb{F}_2^\kappa$ , instead of inputting bits  $x_1, \dots, x_\ell$ . To better understand this situation we can imagine  $R$  inputting an  $\ell \times \kappa$  matrix  $X$ , having  $\mathbf{x}_1, \dots, \mathbf{x}_\ell \in \mathbb{F}_2^\kappa$  as rows and  $\mathbf{x}^1, \dots, \mathbf{x}^\kappa \in \mathbb{F}_2^\ell$  as columns. If  $R$  is honest then  $\mathbf{x}^1 = \dots = \mathbf{x}^\kappa$  and the rows  $\mathbf{x}_j$  are “monochrome” vectors, i.e. consisting either of all 0’s or all 1’s. At this point the receiver computes  $\mathbf{u}^i = \mathbf{t}_0^i + \mathbf{t}_1^i + \mathbf{x}^i$ , for each  $i \in [\kappa]$ . Clearly, if  $R$  is honest, they send the same vector  $\mathbf{x}^i$  for each  $i$ . After this step  $S$  computes  $\mathbf{q}^i = \mathbf{t}_{\Delta_i}^i + \mathbf{u}^i + \mathbf{u}^i \cdot \Delta_i = \mathbf{t}_0^i + \mathbf{x}^i \cdot \Delta_i$ , obtaining the  $\ell \times \kappa$  matrix  $Q$ , having  $\mathbf{q}^i$  as columns and  $\mathbf{q}_j = \mathbf{t}_{0,j} + \mathbf{x}_j * \Delta$  as rows. If  $\mathbf{x}_j$  is monochrome, i.e.  $\mathbf{x}_j = x_j \cdot (1, \dots, 1)$ ,

then  $\mathbf{q}_j = \mathbf{t}_{0,j} + x_j \cdot \Delta$ , otherwise, rewriting  $\mathbf{x}_j$  as  $\mathbf{x}_j = x_j \cdot (1, \dots, 1) + \mathbf{e}_j$ , we get  $\mathbf{q}_j = \mathbf{t}_{0,j} + x_j \cdot \Delta + \mathbf{e}_j * \Delta$ , where  $\mathbf{e}_j$  is an “error” vector counting the number of positions in which  $R$  cheated.

Notice that, compared with the original IKNP protocol, the protocol COTe stops before hashing the output with the random oracle to break the correlation and performing the final round of communication. It is easy to see (and was shown e.g. by Nielsen [19]) that the protocol for COTe (given in Fig. 3) securely implements this functionality.

**Protocol for COTe <sup>$\kappa, \ell$</sup>**

**Initialize:** This is independent of inputs and only needs to be done once.

1.  $R$  samples  $\kappa$  pairs of  $\kappa$ -bit seeds,  $\{(\mathbf{k}_0^i, \mathbf{k}_1^i)\}_{i=1}^\kappa$ .
2.  $S$  samples a random  $\kappa$ -bit string  $\Delta$ .
3. The parties call  $\kappa \times \text{OT}_\kappa$  with inputs  $\Delta$  and  $\mathbf{k}_0, \mathbf{k}_1$ .
4.  $S$  receives  $\mathbf{k}_{\Delta_i}^i$  for  $i = 1, \dots, \kappa$ .

**Extend:** This creates  $\ell$  extended C-OTs. Note that this phase can be iterated, as done by Asharov et al. [1].

1.  $R$  inputs monochrome vectors  $\mathbf{x}_1, \dots, \mathbf{x}_\ell$ . Let  $x_1, \dots, x_\ell$  be the bits of the vectors for the case when  $R$  is honest.
2. Expand  $\mathbf{k}_0^i$  and  $\mathbf{k}_1^i$  using a pseudo random generator (PRG), letting
 
$$\mathbf{t}_0^i = \text{PRG}(\mathbf{k}_0^i) \in \mathbb{F}_2^\ell \quad \text{and} \quad \mathbf{t}_1^i = \text{PRG}(\mathbf{k}_1^i) \in \mathbb{F}_2^\ell, \quad i = 1, \dots, \kappa.$$
 so  $R$  knows  $(\mathbf{t}_0^i, \mathbf{t}_1^i)$  and  $S$  knows  $\mathbf{t}_{\Delta_i}^i$  for  $i = 1, \dots, \kappa$ .
3.  $R$  computes
 
$$\mathbf{u}^i = \mathbf{t}_0^i + \mathbf{t}_1^i + \mathbf{x}^i \in \mathbb{F}_2^\ell, \quad i = 1, \dots, \kappa,$$
 where  $\mathbf{x}^i = (x_1, \dots, x_\ell) \in \mathbb{F}_2^\ell$  and sends them to  $S$ . Here we are creating the keys correlation that permits to extend OTs, inverting the role of sender and receiver.
4.  $S$  computes
 
$$\mathbf{q}^i = \Delta_i \cdot \mathbf{u}^i + \mathbf{t}_{\Delta_i}^i \in \mathbb{F}_2^\ell.$$
 Notice that  $\mathbf{q}^i = \mathbf{t}_0^i + \Delta_i \cdot \mathbf{x}^i$ , for  $i = 1, \dots, \kappa$ .
5. Let  $\mathbf{q}_j$  denote the  $j$ -th row of the  $\ell \times \kappa$  bit matrix  $Q = [\mathbf{q}^1 | \dots | \mathbf{q}^\kappa]$ , and similarly let  $\mathbf{t}_j$  be the  $j$ -th row of  $[\mathbf{t}_0^1 | \dots | \mathbf{t}_0^\kappa]$ . Note that
 
$$\mathbf{q}_j = \mathbf{t}_j + \mathbf{x}_j * \Delta, \quad j = 1, \dots, \ell.$$

Output:  $R$  outputs  $\mathbf{t}_j$ ,  $S$  outputs  $\mathbf{q}_j$  and  $\Delta$ .

**Fig. 3.** Protocol for correlated OT with errors between  $S$  and  $R$ .

### 3 Our Actively Secure OT Extension Protocol

In this section we describe our protocol for actively secure OT extension based on the passive IKNP functionality,  $\mathcal{F}_{\text{COTe}}$ . We recall that to deal with malicious adversaries, all the known actively secure OT extension protocols add a

consistency check to the passive secure IKNP protocol to ensure that  $R$  inputs consistent values.

For example, in previous works [2, 20] this check is added before the “extension” phase, i.e. before the sender  $S$  “reverses” the base OTs and breaks the correlation, effectively checking on the OT seeds. In our construction we check the correlation for consistency *after* the extension step, precisely after the execution of COTe, actually checking the extended OTs.

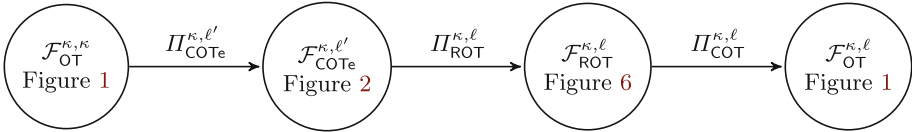


Fig. 4. Relationship between the different functionalities used to go from  $\mathcal{F}_{OT}^{\kappa, \kappa}$  to  $\mathcal{F}_{OT}^{\kappa, \ell}$ .

**Functionality  $\mathcal{F}_{Rand}^{\overline{r}}$**

**Random sample:** Upon receiving  $(rand; u)$  from all parties, it samples a uniform  $r \in \mathbb{F}$  and outputs  $(rand, r)$  to all parties.

Fig. 5. Functionality  $\mathcal{F}_{Rand}^{\overline{r}}$

The high level idea of our protocol in Fig. 7 is to perform a simple correlation check to ensure that the receiver used the same vector  $\mathbf{x}^i$  for each  $\mathbf{u}^i$  sent in Step 3 of the IKNP extension. If the check passes, then the correlated OTs are hashed to obtain random OTs. This check requires sacrificing  $\kappa + s$  extended OTs to ensure security, so we obtain a reduction from  $\mathcal{F}_{ROT}^{\kappa, \ell}$  to  $\mathcal{F}_{COTe}^{\kappa, \ell'}$ , with  $\ell' = \ell + (\kappa + s)$ .

The intuition in this reduction is that, if the check passes, the adversary can only learn few bits of the correlation vector  $\Delta$ , and hence the values  $H(\mathbf{q}_j + \Delta)$  are actually random except with negligible probability. Finally, if required, the random OTs obtained from ROT can be derandomized with an additional set of messages from the sender, using the standard reduction from  $\mathcal{F}_{OT}^{\kappa, \ell}$  to  $\mathcal{F}_{ROT}^{\kappa, \ell}$ .

The relationship between all the functionalities used are described in Fig. 4. The first stage to  $\mathcal{F}_{COTe}$  essentially consists of the IKNP OT extension protocol (with some modifications from the protocol by Asharov et al. [1]) that we have seen in the previous section.

### 3.1 Protocol from COTe to ROT

Here we describe the protocol implementing the  $\mathcal{F}_{ROT}^{\kappa, \ell}$  functionality in Fig. 6. The main idea of our construction is to use a variant of the MAC check protocol from SPDZ [6], adapted for two parties where one party holds the MAC key,

**Functionality  $\mathcal{F}_{\text{ROT}}^{\kappa, \ell}$**

The functionality is parametrized by the number  $\ell$  of resulting OTs and by the length of the OT strings  $\kappa$ .

Running with parties  $S$ ,  $R$  and an ideal adversary denoted by  $\mathcal{S}$ , it operates as follows.

- Upon receiving  $(R, (x_1, \dots, x_\ell))$  from  $R$ , where  $x_j \in \mathbb{F}_2$ , the functionality samples random  $(\mathbf{v}_{0,j}, \mathbf{v}_{1,j}) \in \mathbb{F}_2^{2\kappa}$ , for  $j \in [\ell]$ . Then it sends  $(\mathbf{v}_{0,j}, \mathbf{v}_{1,j})$  to  $S$  and  $\mathbf{v}_{x_j, j}$  to  $R$ .
- If  $R$  is corrupt: if  $\mathcal{S}$  inputs **Abort**,  $\mathcal{F}$  sends **Abort** to  $S$  and it halts. Otherwise it waits for  $\mathcal{S}$  to input  $x_j$  for all  $j \in [\ell]$ . Then it samples random  $(\mathbf{v}_{0,j}, \mathbf{v}_{1,j})$ ,  $j \in [\ell]$  and outputs them to  $S$ . It also sends  $\mathbf{v}_{x_j, j}$  to  $S$  for all  $j \in [\ell]$ .
- If  $S$  is corrupt it waits for  $\mathcal{S}$  to input  $(\mathbf{v}_{0,j}, \mathbf{v}_{1,j})$ ,  $j \in [\ell]$ , and then outputs as above using these values.

**Fig. 6.** Functionality  $\mathcal{F}_{\text{ROT}}^{\kappa, \ell}$

to check the correlation is consistent. The correlation check is performed on the  $\ell'$  correlated OTs of length  $\kappa$  output by  $\mathcal{F}_{\text{COTE}}^{\kappa, \ell'}$ , i.e. after the vectors have been transposed. Recall that after running  $\mathcal{F}_{\text{COTE}}$ , the sender  $S$  has  $\Delta, \mathbf{q}_1, \dots, \mathbf{q}_{\ell'} \in \mathbb{F}_2^\kappa$  and the receiver  $R$  has  $\mathbf{x}_1, \dots, \mathbf{x}_{\ell'}, \mathbf{t}_1, \dots, \mathbf{t}_{\ell'} \in \mathbb{F}_2^\kappa$  such that  $\mathbf{q}_j = \mathbf{t}_j + \mathbf{x}_j * \Delta$  for  $j \in [\ell']$ . If  $R$  was honest then every  $\mathbf{x}_j$  is monochrome, so  $\mathbf{q}_j = \mathbf{t}_j + x_j \cdot \Delta$  for bits  $x_1, \dots, x_{\ell'}$ .

To carry out the check, both parties first securely generate  $\ell'$  random weights  $\chi_1, \dots, \chi_{\ell'} \in \mathbb{F}_2^\kappa$  (using Fig. 5), and then compute weighted sums of their outputs from  $\mathcal{F}_{\text{COTE}}$ . Then  $R$  sends these values to  $S$  to check consistency with  $S$ 's output. So,  $R$  computes  $x = \sum_{j=1}^{\ell'} x_j \cdot \chi_j$ ,  $t = \sum_{j=1}^{\ell'} \mathbf{t}_j \cdot \chi_j$  and  $S$  computes  $q = \sum_{j=1}^{\ell'} \mathbf{q}_j \cdot \chi_j$ , where the vectors  $\mathbf{t}_j, \mathbf{q}_j, \chi_j$  are viewed as elements of  $\mathbb{F}_2^\kappa$  and multiplications are performed in this finite field.  $S$  then checks that  $q = t + x \cdot \Delta$ .

Clearly, by linearity of the correlated OT output, the check will always pass for an honest receiver. If  $R$  is corrupted then it is possible they may pass the check despite having used polychromatic  $\mathbf{x}_j$  vectors; in this case they will learn some information about  $\Delta$ . We show that this leakage is *optimal*, in the sense that a cheating receiver can learn  $c$  bits of information on  $\Delta$  with at most probability  $2^{-c}$ , and the possible errors in the resulting OTs do not provide the adversary with any further useful information. Looking ahead to the proof, the success probability of a receiver who passes the check in breaking the resulting OTs with  $q = \text{poly}(\kappa)$  queries to  $H$  will therefore be  $q/2^{\kappa-c}$ , giving an overall success probability of  $q/2^\kappa$ . This implies that  $\kappa$  base OTs suffice for computational security  $\kappa$ .

On the other hand, if the sender is corrupted, our correlation check introduces the possibility that the values of  $x$  and  $t$  could leak information about  $R$ 's input bits  $x_1, \dots, x_{\ell'}$ . However, we show that it suffices to perform  $\kappa + s$  additional OTs with random choice bits to counter against this leakage, for statistical security  $s$ .

Overall, this means our protocol requires only  $\kappa$  base OTs, which is optimal with respect to the IKNP extension, and an additive overhead of  $s + \kappa$  extended OTs, regardless of the number  $\ell$  of OTs required, as well as just  $O(\kappa)$  additional communication in a constant number of rounds.

### 3.2 Analysis of the Correlation Check

**Corrupt Sender.** To ensure that the correlation check step is secure against a corrupt sender we must carefully choose the parameter  $\ell'$ , which determines the size of the batch each check is performed on. Recall that the elements in the field  $\mathbb{F}$  are  $\kappa$  bits long; if  $\ell' \leq \kappa$  then it is likely that the secret bits  $x_j$  will be uniquely determined given  $\chi_j$  and  $x$ , so an adversary could attempt to solve the corresponding knapsack problem to recover these. As we will see in the proof in Theorem 1, to thwart this attack, we use a technical lemma giving a bound on the rank of a random binary matrix. This is also the reason why we do not let the sender sample  $\{\chi_j\}_{j=1}^{\ell}$ .

**Corrupt Receiver.** The case of a corrupt receiver is much more involved. We now investigate a cheating receiver's success probability in the correlation check stage of the ROT protocol in Fig. 7. Let  $\mathbf{x}_1, \dots, \mathbf{x}_{\ell'}$  be the vectors in  $\mathbb{F}_2^\kappa$  input by  $R$  during the protocol. Taking these to be the rows of a  $\ell' \times \kappa$  matrix, let  $\mathbf{x}^1, \dots, \mathbf{x}^\kappa$  be the *columns* of the same matrix, in  $\mathbb{F}_2^{\ell'}$ . If  $R$  was honest then  $\{\mathbf{x}_j\}_{j \in [\ell']}$  are all monochrome and  $\{\mathbf{x}^i\}_{i \in [\kappa]}$  are all equal. The following Lemma gives the main properties needed from our correlation check.

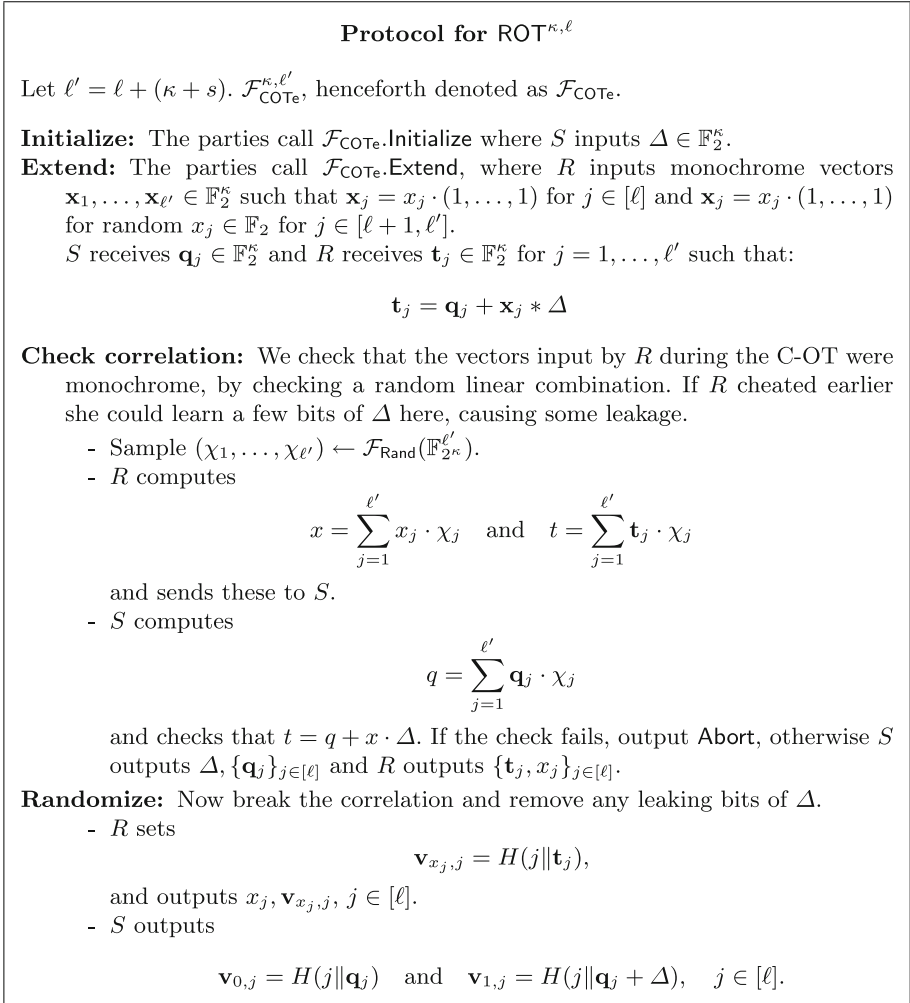
**Lemma 1.** *Let  $S_\Delta \subseteq \mathbb{F}_2^\kappa$  be the set of all  $\Delta$  for which the correlation check passes, given the view of the receiver. Except with probability  $2^{-\kappa}$ , there exists  $k \in \mathbb{N}$  such that*

1.  $|S_\Delta| = 2^k$ .
2. For every  $\mathbf{s} \in \{\mathbf{x}^i\}_{i \in [\kappa]}$ , let  $H_{\mathbf{s}} = \{i \in [\kappa] \mid \mathbf{s} = \mathbf{x}^i\}$ . Then one of the following holds:
  - For all  $i \in H_{\mathbf{s}}$  and any  $\Delta^{(1)}, \Delta^{(2)} \in S_\Delta$ ,  $\Delta_i^{(1)} = \Delta_i^{(2)}$ .
  - $k \leq |H_{\mathbf{s}}|$ , and  $|\{\Delta_{H_{\mathbf{s}}}\}_{\Delta \in S_\Delta}| = 2^k$ , where  $\Delta_{H_{\mathbf{s}}}$  denotes the vector consisting of the bits  $\{\Delta_i\}_{i \in H_{\mathbf{s}}}$ . In other words,  $S_\Delta$  restricted to the bits corresponding to  $H_{\mathbf{s}}$  has entropy at least  $k$ .

Furthermore, there exists  $\hat{\mathbf{s}}$  such that  $k \leq |H_{\hat{\mathbf{s}}}|$ .

*Proof.* See full version [14].

We now give some intuition about the meaning of this statement. The set  $S_\Delta$  is the set of all possible values of  $\Delta$  with which the correlation check could pass – note that since  $\Delta$  is uniformly random to the receiver, their probability of passing the check is therefore  $|S_\Delta|/2^\kappa$ . For some vector  $\mathbf{s} \in \{\mathbf{x}^i\}_{i \in [\kappa]}$ , the set  $H_{\mathbf{s}}$  represents indices of all of the vectors equal to  $\mathbf{s}$ . Clearly, for an honest receiver,  $H_{\mathbf{s}}$  is always just the set  $\{1, \dots, \kappa\}$ , and so the size of  $H_{\mathbf{s}}$  measures the



**Fig. 7.** Random OT extension protocol from correlated OT with errors.

amount of deviation in the protocol for a given  $\mathbf{s}$ . The precise indices in  $H_{\mathbf{s}}$  are also important, as they correspond to a subset of the bits of the secret  $\Delta$ , which could be learnt using  $2^{|H_{\mathbf{s}}|}$  queries to the hash function (causing the simulation to abort in our security proof).

The second part of the lemma implies that *for any*  $\mathbf{s}$ , either the bits of  $\Delta$  corresponding to the indices in  $H_{\mathbf{s}}$  are constant for all possible  $\Delta \in S_{\Delta}$ , or, the size of  $H_{\mathbf{s}}$  is at least  $k$ , which means the corresponding abort in the simulation occurs with probability at least  $1 - 2^{-k+\kappa}$ . Clearly in the first case, the adversary gains no new information, but in the second case we have a bound on the amount of information an adversary can learn, which directly corresponds to the size of



the set  $S_\Delta$ , and hence also the success probability in the correlation check. The final part of the Lemma, concerning  $\hat{\mathbf{s}}$ , simply states that there is always a vector  $\mathbf{s}$  that satisfies the second condition, so at least one block of  $k$  bits of  $\Delta$  remains hidden. A careful analysis of these possible deviations allows us to show that  $\kappa$  base OTs suffice for our protocol.

### 3.3 Proof of Security

**Theorem 1.** *The protocol in Fig. 7 securely implements the  $\mathcal{F}_{\text{ROT}}^{\kappa,\ell}$  functionality in the  $(\mathcal{F}_{\text{COTe}}, \mathcal{F}_{\text{Rand}}, \mathcal{F}_{\text{RO}})$ -hybrid model with computational security parameter  $\kappa$ .*

The computational security parameter  $\kappa$  manifests itself in that the adversary is only allowed  $\text{poly}(\kappa)$  calls of the random oracle in the proof. Other than that, the simulation is statistically indistinguishable.

*Proof.* We construct a simulator  $\mathcal{S}$  that has access to  $\mathcal{F}_{\text{ROT}}$ , and show that no environment  $\mathcal{Z}$  can distinguish between an interaction with  $\mathcal{S}$  and  $\mathcal{F}_{\text{ROT}}$  and an interaction with the real adversary  $\mathcal{A}$  and the real parties. To simulate a real world execution of the protocol,  $\mathcal{S}$  starts an internal copy of  $\mathcal{A}$  and runs an internal copy of the protocol with dummy parties  $\pi_S$  and  $\pi_R$ , as shown in Fig. 8.

First we deal with the (simpler) case of a corrupt sender. Since the simulator gets the sender’s secret  $\Delta$ , it is straightforward to construct  $x$  and  $t$  that will pass the check. All we need to do is argue indistinguishability from the real world execution. We need the following lemma.

**Lemma 2.** *Let  $A$  be a random  $(\kappa + m) \times \kappa$  matrix over  $\mathbb{F}_2$ , where  $m > 0$ . Then  $A$  has rank  $\kappa$  except with probability less than  $2^{-m}$ .*

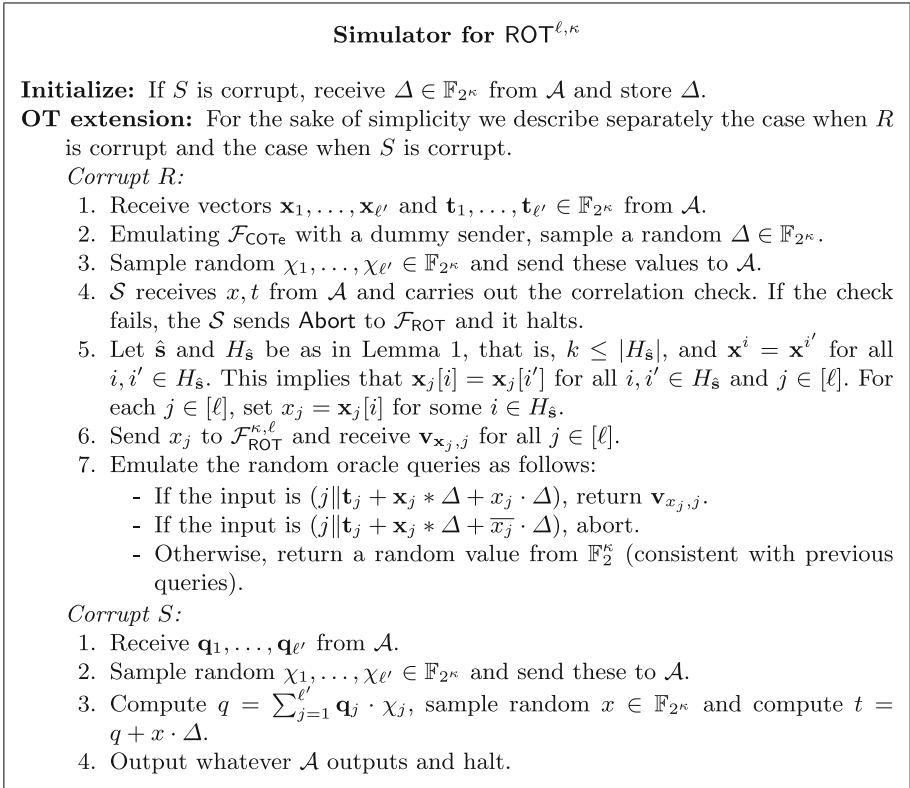
*Proof.* See full version [14].

Recall that in the real world the sender receives

$$x = \sum_{j=1}^{\ell'} x_j \cdot \chi_j = \sum_{j=1}^{\ell} x_j \cdot \chi_j + \sum_{j=\ell+1}^{\ell'} x_j \cdot \chi_j.$$

The second summation corresponds to the image of a linear map from  $\mathbb{F}_2^{\ell'-\ell} = \mathbb{F}_2^{\kappa+s}$  to  $\mathbb{F}_2^\kappa$ . From Lemma 2, it follows that this map has full rank with probability  $1 - 2^{-s}$ . In this case, the second summation is uniformly random in  $\mathbb{F}_2^\kappa$  because  $(x_{\ell+1}, \dots, x_{\ell'})$  were chosen uniformly at random by  $R$ , and so indistinguishable from the simulated random  $x$ . Finally,  $t$  has the same distribution in both worlds because there is only one  $t$  fulfilling the equation  $q = t + x \cdot \Delta$ .

We now consider the case of  $R$  being corrupted. In steps 1–4,  $\mathcal{S}$  simply emulates  $\mathcal{F}_{\text{COTe}}$  and the correlation check, choosing random values for the dummy sender’s input. Lemma 1 states that (except with negligible probability)  $|S_\Delta| = 2^k$  for some  $k \in \mathbb{N}$ . For every  $\mathbf{s} \in \{\mathbf{x}^i\}_{i \in [\kappa]}$ , let  $H_{\mathbf{s}} = \{i \in [\kappa] \mid \mathbf{s} = \mathbf{x}^i\}$  and  $\hat{\mathbf{s}}$  as in Lemma 1. Recall that the adversary knows  $(\mathbf{t}_j, \mathbf{x}_j)$  such that  $\mathbf{t}_j = \mathbf{q}_j + \mathbf{x}_j * \Delta$ .



**Fig. 8.** Simulator for random OT extension

If  $x_1, \dots, x_\ell$  are the bits of  $\hat{s}$  then this can be expressed as  $\mathbf{t}_j = \mathbf{q}_j + x_j \cdot \Delta + \mathbf{e}_j * \Delta$ , where  $\mathbf{e}_j = (x_j, \dots, x_j) + \mathbf{x}_j$  is an adversarially chosen error vector. By definition,  $\mathbf{e}_j[i] = \mathbf{e}_j[i']$  for all  $i, i' \in H_{\hat{s}}$ , for any  $\mathbf{s} \in \{\mathbf{x}^i\}_{i \in [\kappa]}$  and  $j \in [\ell]$ .

In step 7, the simulator responds to the adversary's random oracle queries. Notice that it is the queries  $\mathbf{q}_j = \mathbf{t}_j + \mathbf{x}_j * \Delta$  and  $\mathbf{q}_j + \Delta = \mathbf{t}_j + \overline{x_j} * \Delta$  that require the reply conforming to the output of  $\mathcal{F}_{\text{ROT}}^{\kappa, \ell}$ . The simulator knows  $\mathbf{v}_{x_j, j}$ , which is the output of  $H(j \| \mathbf{q}_j + x_j \cdot \Delta)$  in the real-world protocol. On the other hand, if the adversary queries  $(j \| \mathbf{q}_j + \overline{x_j} \cdot \Delta)$ , the simulator cannot give the right output and thus aborts.

We now investigate the probability that this happens, given that the correlation check has passed. It holds that

$$\mathbf{q}_j + \overline{x_j} \cdot \Delta = \mathbf{t}_j + \mathbf{x}_j * \Delta + \overline{x_j} \cdot \Delta = \mathbf{t}_j + (\mathbf{x}_j + (\overline{x_j}, \dots, \overline{x_j})) * \Delta.$$

For  $i \in H_{\hat{s}}$ ,  $\mathbf{x}_j[i] = x_j$  and thus  $(\mathbf{x}_j + (\overline{x_j}, \dots, \overline{x_j}))[i] = 1$ . By Lemma 1, there are  $|S_\Delta| = 2^k$  possibilities for  $(\mathbf{x}_j + (\overline{x_j}, \dots, \overline{x_j})) * \Delta$  and hence  $\mathbf{q}_j + \overline{x_j} \cdot \Delta$ , given  $\Delta \in S_\Delta$ . Therefore, the probability of one such query is  $2^{-k}$ .

However, we must also show that the environment cannot learn any additional information from previous queries. For example, when  $R$  queries  $\mathbf{q}_j + x_j \cdot \Delta$  to get their correct OT output, the environment (who sees the honest sender output so can verify this has occurred) can learn  $\mathbf{e}_j * \Delta$  by computing  $\mathbf{q}_j + x_j \cdot \Delta + \mathbf{t}_j$ . By definition, the bits of  $\mathbf{e}_j$  corresponding to any index set  $H_s$  are constant. Furthermore, Lemma 1 states that either  $\Delta_i^{(1)} = \Delta_i^{(2)}$  for all  $\Delta^{(1)}, \Delta^{(2)} \in S_\Delta$  and  $i \in H_s$  or  $|H_s| \geq k$ . In the first case,  $e_j[i] \cdot \Delta_i$  is known by the fact that  $\Delta \in S_\Delta$ . In the second case, consider that  $e_j[i]$  is the same for all  $i \in H_s$ . If  $e_j[i] = 0$  for all  $i \in H_s$ , then  $e_j[i] \cdot \Delta_i = 0$  for all  $i \in H_s$ . On the other hand, if  $e_j[i] = 1$ , there are  $2^k$  possibilities for  $\mathbf{e}_j * \Delta$  (given  $\Delta \in S_\Delta$  as above) and thus for  $\mathbf{q}_j + x_j \cdot \Delta$ . Hence, either the latter is known to the adversary already or the probability of querying it is  $2^{-k}$  per query.

It follows that the probability the simulation aborts after the correlation check has passed is at most  $q \cdot 2^{-k}$ , where  $q$  is the number of queries made by the environment. Now taking into account the fact that the check passes with probability  $|S_\Delta| \cdot 2^{-\kappa} + 2^{-\kappa} = 2^{-\kappa} \cdot (2^k + 1)$ , the overall success probability of distinguishing is at most  $q \cdot 2^{-\kappa} \cdot (1 + 2^{-k})$ , which is negligible in  $\kappa$ .  $\square$

### 3.4 From ROT to OT

Finally we show how to reduce  $\mathcal{F}_{OT}^{\kappa, \ell}$  to  $\mathcal{F}_{ROT}^{\kappa, \ell}$ .

**Lemma 3.** *The protocol in Fig. 9 securely implements the  $\mathcal{F}_{OT}^{\kappa, \ell}$  functionality in the  $\mathcal{F}_{ROT}^{\kappa, \ell}$ -hybrid model.*

*Proof.* It is easy to describe a simulator for a corrupt  $R$ .  $\mathcal{S}$  runs a copy of  $\mathcal{A}$  setting dummy parties  $\pi_R$  and  $\pi_S$  and then simulates for them a real execution of DeROT, running an internal copy of  $\mathcal{F}_{ROT}$ .

We just need to show indistinguishability of the transcripts and of the outputs. In both worlds,  $\{\mathbf{d}_{x_i, i}\}_{i \in [\ell]}$  and  $\{\mathbf{v}_{x_i, i}\}_{i \in [\ell]}$  are distributed uniformly subject to the condition  $\mathbf{d}_{x_i, i} + \mathbf{v}_{x_i, i} = \mathbf{y}_{x_i, i}$  for all  $i \in [\ell]$ , as the pads  $\mathbf{v}_{0, i}$  and  $\mathbf{v}_{1, i}$  provided by  $\mathcal{F}_{ROT}$  are random and independent of  $R$ 's view, except with negligible probability.

**Protocol DeROT $^{\kappa, \ell}$**

1. The parties run ROT $^{\kappa, \ell}$  with  $R$  inputting  $\mathbf{x} = (\mathbf{x}_1, \dots, \mathbf{x}_{\ell'})$ ,  $\mathbf{x}_i \in \mathbb{F}_2^\kappa$ ,  $S$  receives  $\{(\mathbf{v}_{0, i}, \mathbf{v}_{1, i})\}_{i \in [\ell]} \in \mathbb{F}_2^\kappa \times \mathbb{F}_2^\kappa$ , and  $R$  receives  $\{\mathbf{v}_{x_i, i}\}_{i \in [\ell]}$ .
2.  $S$  sends  $\{(\mathbf{d}_{0, i}, \mathbf{d}_{1, i})\}_{i \in [\ell]} = \{(\mathbf{v}_{0, i} + \mathbf{y}_{0, i}, \mathbf{v}_{1, i} + \mathbf{y}_{1, i})\}_{i \in [\ell]} \in \mathbb{F}_2^\kappa \times \mathbb{F}_2^\kappa$  to  $R$ .
3.  $R$  outputs  $\{\mathbf{y}_{x_i, i}\}_{i \in [\ell]} = \{\mathbf{v}_{x_i, i} + \mathbf{d}_{x_i, i}\}_{i \in [\ell]}$ .

**Fig. 9.** Derandomization protocol for random OT.

## 4 Implementation

In this section, we evaluate the efficiency of our random OT extension protocol. As was done in previous works [1, 2], we tested the protocol in a standard LAN setting and a simulated WAN environment, using the Linux `tc` tool to create an average round-trip-time of 100 ms (with standard deviation 1 ms) and limit bandwidth to 50 Mbps (comparable with the setting reported by Asharov et al. [2]). We used computational security parameter  $\kappa = 128$  and statistical security parameter  $s = 64$  throughout, and instantiated the PRG with AES-128 in counter mode (using Intel AES-NI) and the random oracle  $H$  with SHA-1.  $\mathcal{F}_{\text{Rand}}$  is implemented using a standard hash-based commitment scheme, where both parties commit to and then open a seed, then the XOR of the two values is used to seed a PRG, which is UC-secure in the random oracle model.

Our implementation was written in C++ using the Miracl library for elliptic curve arithmetic in the base OTs, which were executed using the actively secure protocol of Peikert et al. [21]. All benchmarks were taken as an average of 20 runs on Intel Core i7-3770S 3.1 GHz processors with 8 cores and 32 GB of memory.

**Implementation Optimizations.** The correlation check stage of our protocol requires computing values of the form  $\sum_i x_i \cdot y_i$  where  $x_i, y_i \in \mathbb{F}_{2^\kappa}$ . We used Intel PCLMUL instructions to efficiently compute carryless multiplications and then performed summations and the check itself in the polynomial ring (of length  $2\kappa - 1$ ) to avoid having to do expensive reduction by the finite field polynomial.

As was done by Asharov et al. [1], we use Eklundh’s algorithm for transposing the matrices  $T$  and  $Q$  during the COTe protocol in a cache-friendly manner, which makes the time spent in this stage less than 3% of the total runtime. Our implementation also supports multi-threading, making use of the 8 cores available on our test machines.

### 4.1 Comparison of Protocols

Table 2 shows the time taken for our implementation to compute 10 million OT extensions (excluding the base OTs) in a variety of settings. The one-directional setting is a traditional OT between a sender and a receiver, whilst in the bi-directional times, both parties perform both roles simultaneously (for a total of 20 million OTs). The bi-directional variant is often required for secure two-party and multi-party computation protocols, and over a LAN is much more efficient than performing the one-directional protocol twice, but less so in the WAN setting where communication is the bottleneck.

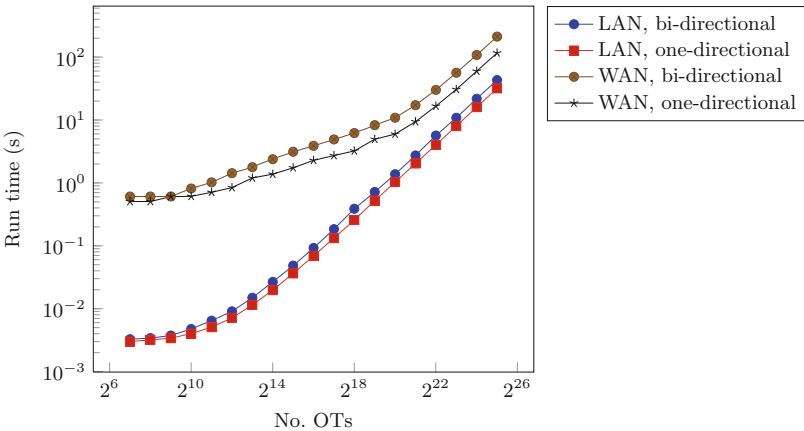
The passive protocol is just the standard IKNP extension (with the random OT communication optimization of Asharov et al. [1]), which is essentially our protocol without the correlation check. In the LAN setting, the time difference between the active and passive protocols is less than 5%. The WAN times for the passive and active protocols are very similar, however it should be noted that there was more variation in our WAN experiments – computing 95% confidence

**Table 2.** Random OT extension runtimes in seconds, using either 1 or 8 threads. The one-directional time is for 10 million OTs between a sender and receiver, whilst for the bi-directional time both parties are playing each role for a total of 20 million OTs.

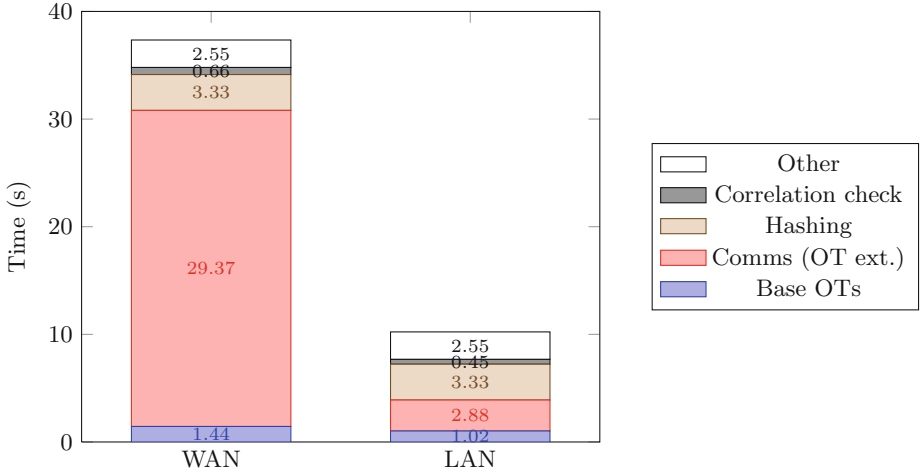
Protocol	Comms. (MB)	LAN time (s)		WAN time (s)	
		One-dir.	Bi-dir.	One-dir.	Bi-dir.
Passive, IKNP (1T)	160MB	9.1037	12.5148	36.2319	66.2692
Passive, IKNP (8T)		3.3258	4.0827	28.4410	53.3977
Active, ours (1T)	160MB	9.5589	12.9461	36.2653	66.6558
Active, ours (8T)		3.3516	4.2020	28.4569	54.1157

intervals for these means in the table gives a variation of up to  $\pm 3\%$ . This is probably mostly due to network variation and can be taken as evidence that our protocol has roughly the same performance as the passive IKNP extension. The total amount of data sent (in all protocols) is almost identical, due to the very low overhead of our correlation check. Compared with the reported timings for the protocol of Asharov et al. [2], our runtimes are much improved: their actively secure times are between 40% and 80% higher than their passive implementation, whilst ours almost match the efficiency of the passive protocol. (We do not directly compare figures due to the different benchmarking environments involved.)

Figure 10 illustrates the performance of our protocol as the number of OTs computed varies, in both the WAN and LAN settings, tested in the one-directional and bi-directional modes of OT operation.



**Fig. 10.** Performance of our OT extension protocol for various numbers of OTs. Times exclude the base OTs.



**Fig. 11.** Profiling results for running 10 million OTs in a single thread.

## 4.2 Profiling

Figure 11 presents profiling results for the main components of our protocol, run in a single thread creating 10 million OTs. It clearly demonstrates that the bottleneck of our protocol is communication from the IKNP extension phase, as was reported for the passive secure implementation of Asharov et al. [1]. The correlation check that we require for active security has a negligible impact on the runtime; the best way to further optimize our implementation in the LAN setting would be to target the hash function computations. The ‘Other’ section includes overhead from PRG computations, matrix transposition and allocating memory, which could also potentially be reduced a small amount.

**Acknowledgements.** We would like to thank Claudio Orlandi and Morten Bech for pointing out minor errors in the proof of Theorem 1, and the anonymous reviewers whose comments helped to improve the paper. This work has been supported in part by EPSRC via grant EP/I03126X.

## References

1. Asharov, G., Lindell, Y., Schneider, T., Zohner, M.: More efficient oblivious transfer and extensions for faster secure computation. In: Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security, pp. 535–548. ACM (2013)
2. Asharov, G., Lindell, Y., Schneider, T., Zohner, M.: More efficient oblivious transfer extensions with security for malicious adversaries. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 673–701. Springer, Heidelberg (2015)

3. Beaver, D.: Correlated pseudorandomness and the complexity of private computations. In: Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, pp. 479–488. ACM (1996)
4. Brassard, G., Crepeau, 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)
5. Burra, S.S., Larraia, E., Nielsen, J.B., Nordholt, P.S., Orlandi, C., Orsini, E., Scholl, P., Smart, N.P.: High performance multi-party computation for binary circuits based on oblivious transfer. Cryptology ePrint Archive, Report 2015/472 (2015). <http://eprint.iacr.org/>
6. Damgård, I., Keller, M., Larraia, E., Pastro, V., Scholl, P., Smart, N.P.: Practical covertly secure MPC for dishonest majority – or: breaking the SPDZ limits. In: Crampton, J., Jajodia, S., Mayes, K. (eds.) ESORICS 2013. LNCS, vol. 8134, pp. 1–18. Springer, Heidelberg (2013)
7. Damgård, I., Lauritsen, R., Toft, T.: An empirical study and some improvements of the minimac protocol for secure computation. In: Proceedings of the Security and Cryptography for Networks - 9th International Conference, SCN 2014, 3–5 September 2014, Amalfi, Italy, pp. 398–415 (2014)
8. Dong, C., Chen, L., Wen, Z.: When private set intersection meets big data: an efficient and scalable protocol. In: 2013 ACM SIGSAC Conference on Computer and Communications Security, CCS 2013, 4–8 November 2013, Berlin, Germany, pp. 789–800 (2013)
9. Even, S., Goldreich, O., Lempel, A.: A randomized protocol for signing contracts. Commun. ACM **28**(6), 637–647 (1985)
10. Goldreich, O., Vainish, R.: How to solve any protocol problem - an efficiency improvement. In: Pomerance, C. (ed.) CRYPTO 1987. LNCS, vol. 293, pp. 73–86. Springer, Heidelberg (1988)
11. Impagliazzo, R., Rudich, S.: Limits on the provable consequences of one-way permutations. In: Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing, pp. 44–61. ACM (1989)
12. 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)
13. 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)
14. Keller, M., Orsini, E., Scholl, P.: Actively secure OT extension with optimal overhead. Cryptology ePrint Archive (2015, to appear). <http://eprint.iacr.org/>
15. Kilian, J.: Founding cryptography on oblivious transfer. In: Proceedings of the 20th Annual ACM Symposium on Theory of Computing, 2–4 May 1988, Chicago, Illinois, USA, pp. 20–31 (1988)
16. Kolesnikov, V., Kumaresan, R.: Improved OT extension for transferring short secrets. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 54–70. Springer, Heidelberg (2013)
17. Larraia, E.: Extending oblivious transfer efficiently. In: Aranha, D.F., Menezes, A. (eds.) LATINCRYPT 2014. LNCS, vol. 8895, pp. 368–386. Springer, Heidelberg (2015)
18. Larraia, E., Orsini, E., Smart, N.P.: Dishonest majority multi-party computation for binary circuits. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part II. LNCS, vol. 8617, pp. 495–512. Springer, Heidelberg (2014)

19. Nielsen, J.B.: Extending oblivious transfers efficiently - how to get robustness almost for free. *IACR Cryptology ePrint Arch.* **2007**, 215 (2007)
20. Nielsen, J.B., Nordholt, P.S., Orlandi, C., Burra, S.S.: A new approach to practical active-secure two-party computation. In: Safavi-Naini, R., Canetti, R. (eds.) *CRYPTO 2012*. LNCS, vol. 7417, pp. 681–700. Springer, Heidelberg (2012)
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)
22. Pinkas, B., Schneider, T., Zohner, M.: Faster private set intersection based on OT extension. In: *Proceedings of the 23rd USENIX Security Symposium, 20–22 August 2014, San Diego, CA, USA*, pp. 797–812 (2014)
23. Rabin, M.O.: How to exchange secrets with oblivious transfer, (1981). Harvard University Technical report 81
24. Wiesner, S.: *SIGACT News*. Conjugate Coding **15**(1), 78–88 (1983)
25. Yao, A.C.: How to generate and exchange secrets (extended abstract). In: *27th Annual Symposium on Foundations of Computer Science, 27–29 October 1986, Toronto, Canada*, pp. 162–167 (1986)



# Algebraic Decomposition for Probing Security

Claude Carlet<sup>1</sup>, Emmanuel Prouff<sup>2</sup>, Matthieu Rivain<sup>3</sup>(✉), and Thomas Roche<sup>2</sup>

<sup>1</sup> LAGA, UMR 7539, CNRS, Department of Mathematics,  
University of Paris XIII and University of Paris VIII, Paris, France  
claude.carlet@univ-paris8.fr

<sup>2</sup> ANSSI, Paris, France

{emmanuel.prouff,thomas.roche}@ssi.gouv.fr

<sup>3</sup> CryptoExperts, Paris, France

matthieu.rivain@cryptoexperts.com

**Abstract.** The probing security model is very popular to prove the side-channel security of cryptographic implementations protected by masking. A common approach to secure nonlinear functions in this model is to represent them as polynomials over a binary field and to secure their nonlinear multiplications thanks to a method introduced by Ishai, Sahai and Wagner at Crypto 2003. Several schemes based on this approach have been published, leading to the recent proposal of Coron, Roy and Vivek which is currently the best known method when no particular assumption is made on the algebraic structure of the function. In the present paper, we revisit this idea by trading nonlinear multiplications for low-degree functions. Specifically, we introduce an algebraic decomposition approach in which a nonlinear function is represented as a sequence of functions with low algebraic degrees. We therefore focus on the probing-secure evaluation of such low-degree functions and we introduce three novel methods to tackle this particular issue. The paper concludes with a comparative analysis of the proposals, which shows that our algebraic decomposition method outperforms the method of Coron, Roy and Vivek in several realistic contexts.

## 1 Introduction

Since their introduction in [15, 16], side-channel attacks are known to be a serious threat against implementations of cryptographic algorithms. Among the existing strategies to secure an implementation, one of the most widely used relies on *secret sharing* (aka *masking*). Using secret sharing at the implementation level enables to achieve provable security in the so-called *probing security model* [13]. In this model, it is assumed that an adversary can recover information on a limited number of intermediate variables of the computation. This model has been argued to be practically relevant to address so-called *higher-order* side-channel attacks and it has been the basis of several efficient schemes to protect block ciphers [1, 6, 11, 12, 23, 24]. More recently, it has been shown in [10] that the probing security of an implementation actually implies its security in the more realistic *noisy leakage model* introduced in [22]. This makes probing security a

very appealing feature for the design of secure implementations against side-channel attacks.

The first generic probing secure scheme, here called the ISW scheme, was designed by Ishai, Sahai and Wagner in [13]. It was later used by Rivain and Prouff to design an efficient probing-secure implementation of AES.<sup>1</sup> Several works then followed to improve this approach and to extend it to other SPN block ciphers [6, 8, 9, 14, 25]. In a nutshell, these methods consist in representing the nonlinear functions of such ciphers (the so-called *s-boxes*) as polynomials over a binary field. Evaluating such polynomials then involves some *linear* operations (e.g. squaring, addition, multiplication by constants) which are secured with straightforward and efficient methods, and some so-called *nonlinear multiplications* which are secured using ISW. The proposed polynomial evaluation methods then aim at minimizing the number of nonlinear multiplications. The method recently introduced by Coron, Roy and Vivek in [9] (called CRV in the following) is currently the most efficient scheme for the probing-secure evaluation of nonlinear functions without particular algebraic structure.

In this paper we introduce a new approach based on *algebraic decomposition* for the probing-secure evaluation of nonlinear functions. More precisely, we propose a method — inspired from [9] — to efficiently compute a function from the evaluation of several other functions having a low algebraic degree. We subsequently study the problem of designing efficient probing-secure evaluation methods for low-degree functions. Specifically, we introduce three different methods to tackle this issue with different and complementary assets. The first one relies on a recursive higher-order derivation of the target function. Remarkably, it enables to get a  $t$ -probing secure evaluation of a function of algebraic degree  $s$  (for arbitrary order  $t$ ) from several  $s$ -probing secure evaluations of the function. Namely, for degree- $s$  functions, it reduces  $t$ -probing security to  $s$ -probing security. This method has a complexity exponential in  $s$  and is hence only interesting for low-degree functions. In particular, for the case  $s = 2$ , we show how to specialize it into an efficient algorithm which happens to be a generalization of a previous method proposed in [8] to secure a peculiar type of multiplications (specifically  $x \mapsto x \cdot \ell(x)$  where  $\ell$  is linear). Our generalization of this algorithm can be applied to secure *any* quadratic function more efficiently than a *single* multiplication with the ISW scheme. Our second approach also applies a recursive derivation technique, but in a more direct way which allows us to design a stand-alone probing-secure evaluation method. Interestingly, this method does not rely on the polynomial representation of the target function  $h$ , and its processing only involves additions and evaluations of  $h$ . When the latter evaluation can be tabulated (which is often the case in the context of *s-boxes*), the proposed algorithm can be a valuable alternative to the state-of-the-art solutions based on field multiplications. However, this method also has a com-

---

<sup>1</sup> The original proposal of [24] actually involved a weak mask refreshing algorithm and the weakness was exploited in [8] to exhibit a flaw in the *s-box* processing. The authors of [8] also proposed a correction, which was recently verified for  $d \leq 4$  using program verification techniques [2].

plexity exponential in the algebraic degree of  $h$ , which makes it only interesting for low degree functions. Eventually, the third method consists in tweaking the CRV method for the case of low-degree functions. The tweak is shown to substantially reduce the number of nonlinear multiplications for functions of a given (low) algebraic degree.

The theoretical analysis of our proposals is completed by an extensive comparative analysis considering several ratios for the cost of a field multiplication over the cost of other elementary operations. The results show that for functions of algebraic degree 2 our generalization of [8] is always the most efficient. On the other hand, for functions of algebraic degree 3, the tweaked CRV method achieves the best performances except for small probing orders for which our second method takes the advantage. For high-degree functions, our algebraic decomposition method reaches its best performances when it is combined with our generalization of [8]. For some multiplication cost ratios, our method is more efficient than CRV, which makes it the currently best known method for the probing-secure evaluation of nonlinear functions in these scenarios. As an example, for functions of dimension  $n = 8$ , our method outperforms CRV whenever a field multiplication takes more than 5 elementary operations (which we believe to be a realistic scenario).

From a general point of view, this paper shows that the issue of designing efficient probing-secure schemes for nonlinear functions may be reduced to the secure evaluation of functions of small algebraic degrees. This approach, and the first encouraging results reported in this paper, opens a promising avenue for further research on this topic. Our work also has strong connections with the field of *threshold implementations* in which secret sharing techniques are used to design hardware masking schemes resistant to glitches [19,20]. An efficient threshold implementation is indeed usually obtained by decomposing the target nonlinear function into lower-degree functions (ideally quadratic functions). In this context, some decompositions have been proposed for specific functions, such as the PRESENT s-box [21], the AES s-box [18], and the DES s-boxes [3]. Some works have also followed an exhaustive approach to provide decompositions for small-size s-boxes, specifically all bijective 3-bit and 4-bit s-boxes [3,17]. Certain 5-bit and 6-bit s-boxes have also been considered in [4].

## 2 Preliminaries

### 2.1 Functions in Finite Fields and Derivation

Along this paper, we shall denote by  $\llbracket i, j \rrbracket$  the integer interval  $[i, j] \cap \mathbb{Z}$  for any pair of integers  $(i, j)$  with  $i \leq j$ , and we shall denote by  $\mathbb{F}_{2^n}$  the finite field with  $2^n$  elements for any integer  $n \geq 1$ . Choosing a basis of  $\mathbb{F}_{2^n}$  over  $\mathbb{F}_2$  enables to represent elements of  $\mathbb{F}_{2^n}$  as elements of the vector space  $\mathbb{F}_2^n$  and *vice versa*. In the following, we assume that the same basis is always used to represent elements of  $\mathbb{F}_{2^n}$  over  $\mathbb{F}_2$ . For any positive integers  $m \leq n$ , a function from  $\mathbb{F}_2^n$  to  $\mathbb{F}_2^m$  is called an  $(n, m)$ -*function* (the dimensions could be omitted if they are clear from the context). Nonlinear functions used in block-ciphers are usually called *s-boxes*.

The set of  $(n, m)$ -functions is denoted by  $\mathcal{B}_{n,m}$ . When  $m$  divides  $n$ , any function  $h \in \mathcal{B}_{n,m}$  can be represented by a polynomial function  $x \in \mathbb{F}_{2^n} \mapsto \sum_{i=0}^{2^n-1} a_i x^i$  where the  $a_i$  are constant coefficients in  $\mathbb{F}_{2^n}$  that can be obtained by applying Lagrange’s Interpolation. When  $m$  does not divide  $n$ , the  $m$ -bit outputs of  $h$  can be embedded into  $\mathbb{F}_{2^n}$  by padding them to  $n$ -bit outputs (e.g. by setting the most significant bits to 0). This ensures that any function  $h \in \mathcal{B}_{n,m}$  can be evaluated as a polynomial over  $\mathbb{F}_{2^n}$ . If padding has been used, then it can be removed after the polynomial evaluation by mapping the output from  $\mathbb{F}_{2^n}$  to  $\mathbb{F}_2^n$ . The *algebraic degree* of a function  $h \in \mathcal{B}_{n,m}$  is the integer value  $\max_{a_i \neq 0}(\text{HW}(i))$  where the  $a_i$  are the coefficients of the polynomial representation of  $h$  and where  $\text{HW}(i)$  denotes the Hamming weight of  $i$  (see e.g. [5] for more details about this notion). The algebraic degree must not be confused with the classical notion of polynomial degree which is the integer value  $\max_{a_i \neq 0}(i)$ . A function of algebraic degree  $s$  will sometimes be called a *degree- $s$  function*.

This paper intensively uses the notion of *higher-order derivative* of a  $(n, m)$ -function. It is recalled hereafter.

**Definition 1 (Higher-Order Derivative).** *Let  $n$  and  $m$  be two positive integers such that  $m \leq n$  and let  $h \in \mathcal{B}_{n,m}$ . For any positive integer  $t$  and any  $t$ -tuple  $(a_1, a_2, \dots, a_t) \in (\mathbb{F}_2^n)^t$ , the  $(n, m)$ -function  $D_{a_1, a_2, \dots, a_t} h$  which maps any  $x \in \mathbb{F}_{2^n}$  to  $\sum_{I \subseteq [1, t]} h(x + \sum_{i \in I} a_i)$  is called the  $t^{\text{th}}$ -order derivative of  $h$  with respect to  $(a_1, a_2, \dots, a_t)$ .*

Any  $t^{\text{th}}$ -order derivative of a degree- $s$  function has an algebraic degree bounded above by  $s - t$ . In the following we shall denote by  $\varphi_h^{(t)}$  the  $(n, m)$ -function defined by

$$\varphi_h^{(t)} : (a_1, a_2, \dots, a_t) \mapsto D_{a_1, a_2, \dots, a_t} h(0). \tag{1}$$

This function has some well-known properties recalled hereafter.

**Proposition 1.** *Let  $h \in \mathcal{B}_{n,m}$  be a degree- $s$  function. Then, the function  $\varphi_h^{(s)}$  is  $s$ -linear symmetric and equals zero for any family of  $a_i$  linearly dependent over  $\mathbb{F}_2$ . Conversely, if  $\varphi_h^{(s)}$  is  $s$ -linear, then the algebraic degree of  $h$  is at most  $s$ .*

### 2.2 Sharing and Probing Security

For two positive integers  $k$  and  $d$ , a  $(k, d)$ -sharing of a variable  $x$  defined over some finite field  $\mathbb{K}$  is a random vector  $(x_1, x_2, \dots, x_k)$  over  $\mathbb{K}^k$  such that  $x = \sum_{i=1}^k x_i$  holds (*completeness equality*) and any tuple of  $d-1$  shares  $x_i$  is a uniform random vector over  $\mathbb{K}^{d-1}$ . The variable  $x$  is also called the *plain value* of the sharing  $(x_i)_i$ . If  $k = d$ , the terminology simplifies to  $d$ -sharing.

An algorithm with domain  $\mathbb{K}^d$  is said  $t$ -probing secure if on input a  $d$ -sharing  $(x_1, x_2, \dots, x_d)$  of some variable  $x$ , it admits no tuple of  $t$  or less intermediate variables that depends on  $x$ . An algorithm achieving  $t$ -probing security is resistant to the class of  $t^{\text{th}}$ -order side-channel attacks (see for instance [7, 22, 24]). In this paper we will further use the following non-standard notion.

**Definition 2 (Perfect  $t$ -probing Security).** *An algorithm is said to achieve perfect  $t$ -probing security if every  $\ell$ -tuple of its intermediate variables can be perfectly simulated from an  $\ell$ -tuple of its input shares for every  $\ell \leq t$ .*

It can be shown that for the tight security case  $t = d - 1$ , the notion above is equivalent to the usual  $t$ -probing security.

**Lemma 1.** *An algorithm taking a  $d$ -sharing as input achieves  $(d - 1)$ -probing security if and only if it achieves perfect  $(d - 1)$ -probing security.*

We shall simply say that an algorithm taking a  $d$ -sharing as input is probing secure when it achieves (perfect)  $(d - 1)$ -probing security. We will also say that an algorithm admits a  $t$ -order flaw when  $t$  of its intermediate variables jointly depend on the plain input (hence contradicting  $t$ -probing security).

It is worth noticing that achieving (perfect) probing security for the evaluation of a linear function  $g$  is pretty simple. Computing a  $d$ -sharing of  $g(x)$  from a  $d$ -sharing  $(x_1, x_2, \dots, x_d)$  of  $x$  can indeed be done by applying  $g$  to every share. The process is clearly (perfectly)  $t$ -probing secure with  $t = d - 1$  and the completeness holds by linearity of  $g$  since we have  $g(x) = g(x_1) + g(x_2) + \dots + g(x_d)$ . The same holds for an affine function but the constant term  $g(0)$  must be added to the right side if and only if  $d$  is odd (without impacting the probing security).

Probing security is more tricky to achieve for nonlinear functions. In [13], Ishai, Sahai, and Wagner tackled this issue by introducing the first generic  $t$ -probing secure scheme for the multiplication over  $\mathbb{F}_2$  which can be easily extended to the multiplication over any finite field. Let  $(x_i)_i$  and  $(y_i)_i$  be the  $d$ -sharings of two variables  $x$  and  $y$  over some binary field  $\mathbb{K}$ , the ISW scheme proceeds as follows:

1. for every  $1 \leq i < j \leq d$ , sample a random value  $r_{i,j}$  over  $\mathbb{K}$ ,
2. for every  $1 \leq i < j \leq d$ , compute  $r_{j,i} = (r_{i,j} + a_i \cdot b_j) + a_j \cdot b_i$ ,
3. for every  $1 \leq i \leq d$ , compute  $c_i = a_i \cdot b_i + \sum_{j \neq i} r_{i,j}$ .

The completeness holds from  $\sum_i c_i = \sum_{i,j} a_i \cdot b_j = (\sum_i a_i)(\sum_j b_j)$  since every random value  $r_{i,j}$  appears exactly twice in the sum and hence vanishes.<sup>2</sup> The above multiplication procedure was proved  $t$ -probing secure with  $t \leq (d - 1)/2$  in [13]. This was later improved in [24] to a tight  $t$ -probing security with  $t \leq d - 1$ .

### 2.3 Secure Evaluation of Nonlinear Functions

In the original scheme of Ishai *et al.*, a computation is represented as a Boolean circuit composed of logical operations NOT and AND. In [24], Rivain and Prouff generalized their approach to larger fields which allowed them to design an efficient probing-secure computation of the AES cipher. The main issue in securing SPN ciphers like AES lies in the s-box computations (the rest of the cipher being

---

<sup>2</sup> This is true since  $\mathbb{K}$  is a binary field, but the method can easily be extended to any field by defining  $r_{j,i} = (a_i \cdot b_j - r_{i,j}) + a_j \cdot b_i$  in Step 2.

purely linear operations). The Rivain-Prouff scheme computes the AES s-box using 4 multiplications over  $\mathbb{F}_{2^8}$  (secured with ISW) plus some linear operations (specifically squaring over  $\mathbb{F}_{2^8}$ , and the initial affine transformation). This approach was then extended in [6] to secure any s-box in  $\mathcal{B}_{n,m}$ . The principle is to represent the s-box as a polynomial  $\sum_i a_i \cdot x^i$  in  $\mathbb{F}_{2^n}[x]$  whose evaluation is then expressed as a sequence of linear functions (*e.g.* squaring over  $\mathbb{F}_{2^n}$ , additions, multiplications by coefficients) and *nonlinear multiplications* over  $\mathbb{F}_{2^n}$ . The former operations can be simply turned into probing-secure operations of complexity  $O(d)$  as described in the previous section, whereas the latter operations are secured using ISW with complexity  $O(d^2)$ . The total complexity is hence mainly impacted by the number of nonlinear multiplications involved in the underlying polynomial evaluation method. This observation led to a series of publications aiming at developing polynomial evaluation methods with least possible costs in terms of nonlinear multiplications. Some heuristics in this context were proposed by Carlet *et al.* in [6], and then improved by Roy and Vivek in [25] and by Coron, Roy and Vivek in [9]. In the latter reference, a method, that we shall call the CRV method, is introduced. It is currently the most efficient one for probing-secure evaluation of nonlinear functions.

**Tabulated Multiplications.** Further works have shown that secure evaluation methods could be improved by relying on specific kind of nonlinear multiplications. Assume for instance that the dimension  $n$  is such that a lookup table with  $2^{2n}$  entries is too big for some given architecture whereas a lookup table with  $2^n$  entries fits (*e.g.* for  $n = 8$  one needs 64 kilobytes for the former and 256 bytes for the latter). This means that a multiplication over  $\mathbb{F}_{2^{n/2}}$  can be computed using a single lookup, whereas a multiplication over  $\mathbb{F}_{2^n}$  must rely on more costly arithmetic (*e.g.* schoolbook method, log-exp tables, Karatsuba, etc.). This observation was used by Kim, Hong, and Lim in [14] to design an efficient alternative to the Rivain-Prouff scheme based on the so-called *tower-field representation* of the AES s-box [26]. Using this representation, the AES s-box can be computed based on 5 multiplications over  $\mathbb{F}_{2^4}$  instead of 4 multiplications over  $\mathbb{F}_{2^8}$ , which results in a significant gain of performance when the former are tabulated while the latter are computed using log-exp tables (although one more multiplication is involved). Another approach, proposed in [8] by Coron, Prouff, Rivain, and Roche, is to consider the multiplications of the form  $x \cdot \ell(x)$  where  $\ell$  is a linear function. Such multiplications can be secured using a variant of ISW relying on a table for the function  $x \mapsto x \cdot \ell(x)$ . This variant that we shall call the CPRR scheme, allowed the authors to design a faster probing-secure scheme for the AES s-box. It was further used in [12] to reduce the complexity of the probing-secure evaluation of power functions in  $\mathbb{F}_{2^n}$  with  $n \leq 8$ .

### 3 The Algebraic Decomposition Method

In this section, we introduce a new algebraic decomposition method that split the evaluation of a nonlinear function with arbitrary algebraic degree into several evaluations of functions with given (low) algebraic degree  $s$ . Our proposed

method is inspired from the CRV method but relies on low-degree polynomial evaluations instead of nonlinear multiplications.

We consider a function  $h \in \mathcal{B}_{n,m}$  which is seen as a polynomial  $h(x) = \sum_{j=0}^{2^n-1} a_j x^j$  over  $\mathbb{F}_{2^n}[x]$ . We start by deriving a family of generators  $(g_i)_i$  as follows:

$$\begin{cases} g_1(x) = f_1(x) \\ g_i(x) = f_i(g_{i-1}(x)), \end{cases} \tag{2}$$

where the  $f_i$  are random polynomials of given algebraic degree  $s$ . Then we randomly generate  $t$  polynomials  $(q_i)_i$  over the subspace of polynomials  $\sum_{j=1}^r \ell_j \circ g_j$  where the  $\ell_j$  are linearized polynomials (*i.e.* polynomials of algebraic degree 1). This is done by sampling random linearized polynomials  $(\ell_{i,j})_{i,j}$  and by computing

$$q_i(x) = \sum_{j=1}^r \ell_{i,j}(g_j(x)) + \ell_{i,0}(x). \tag{3}$$

Eventually, we search for  $t$  polynomials  $p_i$  of algebraic degree  $s$  and for  $r + 1$  linearized polynomials  $\ell_i$  such that

$$h(x) = \sum_{i=1}^t p_i(q_i(x)) + \sum_{i=1}^r \ell_i(g_i(x)) + \ell_0(x). \tag{4}$$

From such polynomials, we get a method to compute  $h(x)$  by subsequently evaluating (2), (3) and (4). This method involves  $r + t$  evaluations of degree- $s$  polynomials (the  $f_i$  and the  $p_i$ ), plus some linear operations. The following table gives the exact operation counts (where “#eval deg- $s$ ” denotes the number of evaluations of degree- $s$  functions, “#eval LP” denotes the number of evaluations of linearized polynomials, and “#add” denotes the number of additions).

#eval deg- $s$	#eval LP	#add
$r + t$	$(t + 1)(r + 1)$	$r \cdot t + t + r$

The complexity of the resulting  $d$ -probing secure evaluation can be obtained by multiplying by  $d$  the number of additions and the number of evaluations of linearized polynomials (which might be tabulated) and by adding  $r + t$  secure evaluations of degree- $s$  functions.

*Remark 1.* The generation step of our method can be generalized as follows:

$$\begin{cases} g_1(x) = f_1(x) \\ g_i(x) = f_i\left(\sum_{j=1}^{i-1} \ell'_{i,j}(g_j(x))\right) \end{cases} \tag{5}$$

where the  $f_i$  are random polynomials of given algebraic degree  $s$  and the  $\ell'_{i,j}$  are random linearized polynomials. This generalization has no impact when  $r \leq 2$ . In particular, it has no impact on our experimental results for  $s \in \{2, 3\}$  and  $n \in \llbracket 4, 8 \rrbracket$  (indeed, the best counts are always obtained with  $r \leq 2$  since we stop at  $g_2(x) = f'_2 \circ f_1(x)$  where  $f'_2 = f_2 \circ \ell_{2,1}$  is of degree  $s_1$  – see hereafter –). However, (5) might give better results than (2) for higher values of  $n$  and/or  $s$ .

As in the CRV method, the search of polynomials  $(p_i)_i$  and  $(\ell_i)_i$  satisfying (4) for given polynomials  $(g_i)_i$  and  $(q_i)_i$  amounts to solve a system of linear equations over  $\mathbb{F}_{2^n}$ :

$$A \cdot \mathbf{b} = \mathbf{c}. \quad (6)$$

The target vector  $\mathbf{c}$  has  $2^n$  coordinates which are the values taken by  $h(x)$  for all  $x$  over  $\mathbb{F}_{2^n}$ , that is

$$\mathbf{c} = (h(e_1), h(e_2), h(e_3), \dots, h(e_{2^n})),$$

where  $\{e_1, e_2, \dots, e_{2^n}\} = \mathbb{F}_{2^n}$ . The coordinates of the vector  $\mathbf{b}$  are the variables of the system that represents the solutions for the coefficients of the polynomials  $(p_i)_i$  and  $(\ell_i)_i$ . The matrix  $A$  is then defined as the concatenation of several submatrices:

$$A = (\mathbf{1} \mid A_{p_1} \mid A_{p_2} \mid \dots \mid A_{p_t} \mid A_{\ell_0} \mid A_{\ell_1} \mid \dots \mid A_{\ell_r}),$$

where  $\mathbf{1}$  is the  $2^n$ -dimensional column vector with all coordinates equal to 1  $\in \mathbb{F}_{2^n}$ , where the  $A_{p_i}$  are  $2^n \times N_s$  submatrices, with  $N_s = \sum_{d=1}^s \binom{n}{d}$ , and where the  $A_{\ell_i}$  are  $2^n \times n$  submatrices. For every  $i \in \llbracket 1, t \rrbracket$ , the  $2^n \times N_s$  matrix  $A_{p_i}$  is defined as:

$$A_{p_i} = \begin{pmatrix} q_i(e_1)^{\beta_1} & q_i(e_1)^{\beta_2} & \dots & q_i(e_1)^{\beta_{N_s}} \\ q_i(e_2)^{\beta_1} & q_i(e_2)^{\beta_2} & \dots & q_i(e_2)^{\beta_{N_s}} \\ \vdots & \vdots & \ddots & \vdots \\ q_i(e_{2^n})^{\beta_1} & q_i(e_{2^n})^{\beta_2} & \dots & q_i(e_{2^n})^{\beta_{N_s}} \end{pmatrix},$$

where  $\{\beta_i\} = \{\beta \mid 1 \leq \text{HW}(\beta) \leq s\} \subseteq [0; 2^n - 1]$  (*i.e.* the  $\beta_i$  are the powers with non-zero coefficients in a degree- $s$  polynomial). On the other hand,  $A_{\ell_i}$  is defined as the  $2^n \times n$  matrix:

$$A_{\ell_0} = \begin{pmatrix} e_1^{\alpha_1} & e_1^{\alpha_2} & \dots & e_1^{\alpha_n} \\ e_2^{\alpha_1} & e_2^{\alpha_2} & \dots & e_2^{\alpha_n} \\ \vdots & \vdots & \ddots & \vdots \\ e_{2^n}^{\alpha_1} & e_{2^n}^{\alpha_2} & \dots & e_{2^n}^{\alpha_n} \end{pmatrix} \quad \text{and} \quad A_{\ell_i} = \begin{pmatrix} g_i(e_1)^{\alpha_1} & g_i(e_1)^{\alpha_2} & \dots & g_i(e_1)^{\alpha_n} \\ g_i(e_2)^{\alpha_1} & g_i(e_2)^{\alpha_2} & \dots & g_i(e_2)^{\alpha_n} \\ \vdots & \vdots & \ddots & \vdots \\ g_i(e_{2^n})^{\alpha_1} & g_i(e_{2^n})^{\alpha_2} & \dots & g_i(e_{2^n})^{\alpha_n} \end{pmatrix}$$

for  $i \geq 1$ , where  $\{\alpha_i\} = \{2^i \mid 0 \leq i \leq n-1\}$  (*i.e.* the  $\alpha_i$  are the powers with non-zero coefficients in a linearized polynomial).

System (6) has  $2^n$  equations and  $t \cdot N_s + (r+1)n$  unknowns. Such a system admits a solution for every choice of  $h(x)$  if and only if its rank is  $2^n$ , which implies the following inequality as necessary condition:

$$t \cdot N_s + (r+1)n \geq 2^n \Leftrightarrow t \geq \frac{2^n - (r+1)n}{N_s}. \quad (7)$$

In other words our method requires at least  $(2^n - (r+1)n)/N_s$  secure evaluations of degree- $s$  polynomials. Another necessary condition is that the algebraic degree of the  $p_i \circ q_i$  reaches  $n$ , that is  $r$  verifies  $s^{r+1} \geq n$ . This constraint becomes  $s^{r+1} \geq n-1$  if we only focus on bijective functions (since their algebraic degree is at most  $n-1$ ).



*Remark 2.* We stress that once a full-rank system has been found for some given parameters  $r$  and  $t$ , it can be applied to get a decomposition for *every*  $n$ -bit non-linear function (the considered function being the target vector of the system). Note however that for some specific function it might be possible to find a decomposition involving less than  $r + t$  degree- $s$  polynomials. For instance, the 4-bit s-box of PRESENT can be decomposed into 2 quadratic functions [21], whereas we need 3 quadratic functions to decompose any 4-bit s-box.

**Experimental Results.** We provide hereafter some experimental results for our algebraic decomposition method. We experimented our method for  $n \in \llbracket 4, 8 \rrbracket$  and  $s \in \{2, 3\}$ . The following table summarizes the best results that we obtained and compares them to the lower bound resulting from the above constraints.

	$n = 4$	$n = 5$	$n = 6$	$n = 7$	$n = 8$
#eval-2 (achieved)	<b>3</b>	<b>4</b>	<b>5</b>	<b>8</b>	<b>11</b>
#eval-2 (lower bound)	2	4	5	6	9
#eval-3 (achieved)	<b>2</b>	<b>3</b>	<b>3</b>	<b>4</b>	<b>4</b>
#eval-3 (lower bound)	2	2	3	3	4

Note that our method can be generalized to involve the evaluation of functions with different (low) algebraic degrees. In particular, in our experiments, we considered the hybrid case where the  $g_i$  are of degree  $s_1$  and the  $p_i$  are of degree  $s_2$ . In that case, the constraint on  $r$  becomes  $s_1^r \cdot s_2 \geq n$  (resp.  $s_1^r \cdot s_2 \geq n - 1$  for bijective functions), and the constraint on  $t$  remains as in (7) with  $s_2$  instead of  $s$ . The following table summarizes the best results for the hybrid case  $(s_1, s_2) = (2, 3)$ . This case was always more favorable than the case  $(s_1, s_2) = (3, 2)$ .

	$n = 4$	$n = 5$	$n = 6$	$n = 7$	$n = 8$
#eval-2 + #eval-3 (achieved)	<b>1+1</b>	<b>2+1</b>	<b>1+2</b>	<b>2+2</b>	<b>2+3</b>
#eval-2 + #eval-3 (lower bound)	1 + 1	1 + 1	1 + 2	2 + 2	2 + 3

The following table gives the exact parameters  $(s_1, s_2)$ ,  $r$ , and  $t$ , that we achieved for every  $n \in \llbracket 4, 8 \rrbracket$ . Note that the entries  $4^*$ ,  $5^*$ , and  $7^*$  stand for bijective functions of size  $n \in \{4, 5, 7\}$ , which enjoy more efficient decompositions. For the other considered values of  $n$ , the bijective property did not enable any improvement.

*Remark 3.* For the case  $n = 4$ , it was shown in [3] that every cubic bijective function in  $\mathcal{B}_{4,4}$  can be either decomposed as  $h(\cdot) = f_1 \circ f_2(\cdot)$  or as  $h(\cdot) = f_1 \circ f_2 \circ f_3(\cdot)$ , where the  $f_i$  are quadratic functions, if and only if it belongs to the alternating group of permutations in  $\mathcal{B}_{4,4}$ . It was then shown in [17] that the

$n$	#eval-2	#eval-3	$(s_1, s_2)$	$r$	$t$
4	3	–	(2, 2)	1	2
	3	–	(2, 2)	2	1
	1	1	(2, 3)	1	1
	–	2	(3, 3)	1	1
4*	–	1	(1, 3)	0	1
5	4	–	(2, 2)	2	2
	2	1	(2, 3)	2	1
	–	3	(3, 3)	1	2
5*	3	–	(2, 2)	1	2
6	5	–	(2, 2)	2	3
	1	2	(2, 3)	1	2
	–	3	(3, 3)	1	2
7	8	–	(2, 2)	2	6
	2	2	(2, 3)	2	2
	–	4	(3, 3)	1	3
	–	4	(3, 3)	2	2
7*	1	2	(2, 3)	1	2
8	11	–	(2, 2)	2	9
	11	–	(2, 2)	3	8
	2	3	(2, 3)	2	3
	–	4	(3, 3)	1	3

so-called *optimal s-boxes* in  $\mathcal{B}_{4,4}$  can be decomposed as  $h(\cdot) = f_1(\cdot) + f_2 \circ f_3(\cdot)$ . The authors also suggested to use a decomposition of the form  $h(\cdot) = f_1(\cdot) + f_2 \circ f_3(\cdot) + f_4 \circ f_5(\cdot) + \dots$  for other functions in  $\mathcal{B}_{4,4}$ . Our results demonstrate that *every* function in  $\mathcal{B}_{4,4}$  can be decomposed using 3 quadratic functions plus some linear functions. Moreover, we know from [3] that there exist functions in  $\mathcal{B}_{4,4}$  that cannot be decomposed using only two quadratic functions. This shows the optimality of our method for the case  $n = 4$  and  $s = 2$ . This also suggests that the lower bound (7) might not be tight.

### 4 Reducing the Probing Order Down to the Algebraic Degree

In this section we start by showing that arbitrary  $t$ -probing security can always be reduced to  $s$ -probing security where  $s$  is the algebraic degree of the function to protect. Specifically, we give a method to construct a  $t$ -probing secure processing of a degree- $s$  function from its  $s$ -probing secure processing. Then, we apply our method for the case  $s = 2$ , where a simple 2-probing secure processing can be turned into an efficient  $t$ -probing secure evaluation of any (algebraically) quadratic function.

### 4.1 General Case

Let  $n$  and  $m$  be two positive integers such that  $m \leq n$  and let  $h \in \mathcal{B}_{n,m}$  be the vectorial function whose processing must be secured at the order  $t = d - 1$  for some  $d$ . By definition of  $\varphi_h^{(s)}$ , for every tuple  $(a_1, \dots, a_s) \in (\mathbb{F}_2^n)^s$  we have:

$$h\left(\sum_{i=1}^s a_i\right) = \varphi_h^{(s)}(a_1, a_2, \dots, a_s) + \sum_{I \subsetneq [1,s]} h\left(\sum_{i \in I} a_i\right). \tag{8}$$

Iterating (8) we obtain the following theorem where, by convention,  $h(\sum_{i \in \emptyset} a_i)$  equals  $h(0)$ . The proof is given in the full version of the paper.

**Theorem 1.** *Let  $h \in \mathcal{B}_{n,m}$  be a vectorial function of algebraic degree at most  $s$ . Then, for every  $d \geq s$  we have:*

$$h\left(\sum_{i=1}^d a_i\right) = \sum_{1 \leq i_1 < \dots < i_s \leq d} \varphi_h^{(s)}(a_{i_1}, \dots, a_{i_s}) + \sum_{j=0}^{s-1} \eta_{d,s}(j) \sum_{\substack{I \subseteq [1,d] \\ |I|=j}} h\left(\sum_{i \in I} a_i\right),$$

where  $\eta_{d,s}(j) = \binom{d-j-1}{s-j-1} \pmod 2$  for every  $j \leq s - 1$ .

From Theorem 1 we get the following corollary.

**Corollary 1.** *Let  $h \in \mathcal{B}_{n,m}$  be a vectorial function of algebraic degree at most  $s$ . Then, for every  $d > s$  we have:*

$$h\left(\sum_{i=1}^d a_i\right) = \sum_{j=0}^s \mu_{d,s}(j) \sum_{\substack{I \subseteq [1,d] \\ |I|=j}} h\left(\sum_{i \in I} a_i\right),$$

where  $\mu_{d,s}(j) = \binom{d-j-1}{s-j} \pmod 2$  for every  $j \leq s$ .

Corollary 1 states that, for any  $d$ , the evaluation of a degree- $s$  function  $h \in \mathcal{B}_{n,m}$  on the sum of  $d$  shares can be expressed as several evaluations of  $h$  on sums  $\sum_{i \in I} a_i$  with  $|I| \leq s$ . We can then reduce a  $(d - 1)$ -probing secure evaluation of  $h$  to several  $(j - 1)$ -probing secure evaluations of  $h$  where  $j = |I| \leq s$ . Doing so, each evaluation takes  $j = |I|$  shares  $(a_i)_{i \in I}$  and computes a  $j$ -sharing of  $h(\sum_{i \in I} a_i)$ . Afterwards, one needs a secure scheme to combine the obtained shares of all the  $h(\sum_{i \in I} a_i)$ , with  $I \subseteq [1, d]$  such that  $|I| \leq s$  and  $\mu_{d,s}(|I|) = 1$ , into a  $d$ -sharing of  $h(a)$ .

The overall process is summarized in the following algorithm, where `SecureEval` is a primitive that performs a  $(j - 1)$ -probing secure evaluation of  $h$  on a  $j$ -sharing input for any  $j \leq s$ , and where `SharingCompress` is a primitive that on input  $(x_i)_{i \in [1,k]}$  produces a  $d$ -sharing of  $\sum_{i=1}^k x_i$  for any  $k \leq d$ . The full version of this paper includes the description of such `SharingCompress` algorithm which achieves perfect  $t$ -probing security, and provide a security proof for the overall method.

---

**Algorithm 1.** Secure evaluation of a degree- $s$  function

---

**Input** : a  $d$ -sharing  $(x_1, x_2, \dots, x_d)$  of  $x \in \mathbb{F}_{2^n}$   
**Output**: a  $d$ -sharing  $(y_1, y_2, \dots, y_d)$  of  $y = h(x)$

- 1 **for**  $I \subseteq [1, d]$  **with**  $|I| \leq s$  **and**  $\mu_{d,s}(|I|) = 1$  **do**
- 2    $(r_{I,k})_{k \leq |I|} \leftarrow \text{SecureEval}(h, (x_i)_{i \in I})$
- 3    $(y_1, y_2, \dots, y_d) \leftarrow \text{SharingCompress}((r_{I,k})_{k \leq |I|, I \subseteq [1, d], \mu_{d,s}(|I|=1})$
- 4 **return**  $(y_0, y_1, \dots, y_d)$

---

**Complexity.** For every  $s$  and every  $d > s$ , the term  $\mu_{d,s}(j)$  always equals 1 when  $j = s$ . On the other hand, whenever  $d \equiv s \pmod{2^\ell}$  with  $\ell = \lfloor \log_2 s \rfloor + 1$ , the term  $\mu_{d,s}(j)$  equals 0 for every  $j < s$ . For the sake of efficiency, we shall assume that such a value of  $d$  is always chosen for our method.<sup>3</sup> Under this assumption, the complexity of Algorithm 1 is of  $\binom{d}{s}$  calls to **SecureEval** with  $s$  shares and one call to **SharingCompress** from  $k$  shares to  $d$  shares where  $k = s \binom{d}{s}$ . From the complexity of the sharing compression method, we obtain the following operation count (where “#add” and “#rand” respectively denote the number of additions and the number of sampled random values in the sharing compression).

	#SecureEval	#add	#rand
Exact count	$\binom{d}{s}$	$(s \binom{d}{s} - d)(d + 1)$	$\frac{1}{2}(s \binom{d}{s} - d)(d - 1)$
Approximation	$(\frac{1}{s!})d^s$	$(\frac{1}{(s-1)!})d^{s+1}$	$(\frac{1}{2(s-1)!})d^{s+1}$

### 4.2 Quadratic Case

For degree-2 (aka quadratic) functions  $h$ , it may be observed that  $\varphi_h^{(2)}(x_i, x_j)$  equals  $h(x_i + x_j + r) + h(x_i + r) + h(x_j + r) + h(r)$  whatever  $(x_i, x_j, r) \in (\mathbb{F}_{2^n})^3$  (this holds since  $\varphi_h^{(3)}(x_i, x_j, r) = 0$  for quadratic functions). This observation and Theorem 1 imply the following equality for any integer  $d \geq s$  and any  $r_{i,j}$  values in  $\mathbb{F}_{2^n}$ :

$$\begin{aligned}
 h\left(\sum_{i \in [1, d]} x_i\right) &= \sum_{1 \leq i < j \leq d} (h(x_i + x_j + r_{i,j}) + h(x_j + r_{i,j}) + h(x_i + r_{i,j}) + h(r_{i,j})) \\
 &+ \sum_{i \in [1, d]} h(x_i) + ((d + 1) \bmod 2) \cdot h(0).
 \end{aligned}
 \tag{9}$$

Equality (9) shows that the evaluation of a quadratic function in a sum of  $d$  shares can be split into a sum of terms depending on at most two shares  $x_i$  and  $x_j$ . In this particular case it is then possible to use an improved sharing compression inspired from ISW, which gives<sup>4</sup> Algorithm 2.

---

<sup>3</sup> This is a weak assumption for low algebraic degrees. For instance  $s \leq 3$  gives  $d \equiv s \pmod 4$ ,  $s \leq 7$  gives  $d \equiv s \pmod 8$ , etc. Note that the complexity comparison in Sect. 7 does not take this assumption into account.

<sup>4</sup> Note that in Step 4 the additions must be computed from left to right in order to ensure the probing security.

---

**Algorithm 2.** Secure evaluation of a quadratic function

---

**Input** : the  $d$ -sharing  $(x_1, x_2, \dots, x_d)$  of  $x$  in  $\mathbb{F}_{2^n}$

**Output**: a  $d$ -sharing  $(y_1, y_2, \dots, y_d)$  of  $y = h(x)$

```

1 for  $i = 1$  to  $d$  do
2   for  $j = i + 1$  to  $d$  do
3      $r_{i,j} \leftarrow^{\$} \mathbb{F}_{2^n}; r'_{i,j} \leftarrow^{\$} \mathbb{F}_{2^n}$ 
4      $r_{j,i} \leftarrow r_{i,j} + h(x_i + r'_{i,j}) + h(x_j + r'_{i,j}) + h((x_i + r'_{i,j}) + x_j) + h(r'_{i,j})$ 
5 for  $i = 1$  to  $d$  do
6    $y_i \leftarrow h(x_i)$ 
7   for  $j = 1$  to  $d, j \neq i$  do
8      $y_i \leftarrow y_i + r_{i,j}$ 
9 if  $d$  is even then  $y_1 = y_1 + h(0)$ 
10 return  $(y_1, y_2, \dots, y_d)$ 

```

---

This algorithm is actually already known from [8]. However the authors only suggest to use it for the secure evaluation of a multiplication of the form  $x \cdot \ell(x)$  where  $\ell$  is a degree-1 function (linear or affine). We show here that this algorithm can actually be used to securely compute *any* degree-2 function. The only difference is that one must add  $h(0)$  to one output share whenever  $d$  is even (this term does not appear in [8] since by definition of  $h : x \mapsto x \cdot \ell(x)$  one always get  $h(0) = 0$ ). The probing security of Algorithm 2 holds from the security proof provided in [8].

**Complexity.** The following table summarizes the complexity of Algorithm 2 in terms of additions, evaluations of  $h$ , and sampled random values (we consider the worst case of  $d$  being even). For comparison, we also give the complexity of the ISW scheme for a single multiplication.

	# add	# eval <sub><math>h</math></sub>	# mult	# rand
Algorithm 2	$\frac{9}{2} d(d - 1) + 1$	$d(2d - 1)$	-	$d(d - 1)$
ISW multiplication	$2d(d - 1)$	-	$d^2$	$\frac{1}{2}d(d - 1)$

As explained in Sect. 2.3, for some values of  $n$ , a lookup table of  $2^n$  entries might be affordable while a lookup table of  $2^{2n}$  entries is not (typically for  $n = 8$  giving 256 bytes *vs.* 64 kilobytes). In such a situation, the cost of one evaluation of  $h$  is expected to be significantly lower than the cost of a multiplication. We hence expect Algorithm 2 to be more efficient than the ISW scheme. Moreover, as shown above, Algorithm 2 can be used to securely evaluate a degree-2 function with a polynomial representation possibly involving *many* multiplications, whereas the ISW scheme securely evaluates a *single* multiplication.

## 5 Another Method to Secure Low-Degree Functions

In this section we introduce another new scheme to secure the evaluation of any function  $h \in \mathcal{B}_{n,m}$  of given algebraic degree  $s$ . The method only involves additions and evaluations of  $h$  (that may be tabulated depending on the context). As the previous method, its complexity is exponential in  $s$  which makes it suited for low-degree functions only. We start by introducing the core ideas of our method with the simple case of a 2-probing secure evaluation (*i.e.* taking a 3-shared input) of a degree-2 function. Then, we show how to extend the approach to achieve arbitrary probing security. The study is completed in the full version of this paper where we generalize our result to functions of any algebraic degree.

### 5.1 Two-Probing Security for Quadratic Functions

Let  $h \in \mathcal{B}_{n,m}$  be a degree-2 function. We present hereafter a new method to securely construct a 3-sharing  $(y_1, y_2, y_3)$  of  $y = h(x)$  from a 3-sharing  $(x_1, x_2, x_3)$  of  $x$ . Since  $h$  is quadratic, its third-order derivatives are null (see Definition 1) which implies the following equality for every  $x \in \mathbb{F}_{2^n}$  and every triplet  $(r_1, r_2, r_3) \in \mathbb{F}_{2^n}^3$ :

$$h(x) = \sum_{1 \leq i \leq 3} h(x + r_i) + \sum_{1 \leq i < j \leq 3} h(x + r_i + r_j) + h(x + r_1 + r_2 + r_3), \quad (10)$$

or equivalently  $h(x) = \sum_{i=1}^7 h(x + e^{(i)})$ , where  $e^{(i)}$  denotes the scalar product  $\omega_i \cdot (r_1, r_2, r_3)$  with  $\omega_i$  being the binary representation of the index  $i$  (*e.g.*  $\omega_3 = (0, 1, 1)$ ). We shall say in the following that the family  $(e^{(i)})_{i \in [1,7]}$  is 3-spanned from  $(r_1, r_2, r_3)$ . Replacing  $x$  by the sum of its shares then leads to:

$$h(x_1 + x_2 + x_3) = \sum_{i=1}^7 h(x_1 + x_2 + x_3 + e^{(i)}). \quad (11)$$

It may be checked that the tuple  $(h_i)_{i \in [1,7]} = (h(x + e^{(i)}))_{i \in [1,7]}$  is a  $(7, 3)$ -sharing of  $h(x)$  (this holds since the rank of  $(e^{(i)})_{i \in [1,7]}$  is at most 3). However, a direct evaluation of the  $h_i$  would yield an obvious second-order flaw. Indeed, for every  $i \in \{1, 2, 3\}$ , the evaluation of at least one of the terms in (11) implies the computation of  $x + r_i$  which can be combined with  $r_i$  to recover  $x$ . A natural solution to avoid such a second-order flaw is to split the processing of each  $x + e^{(i)}$  into several parts, which actually amounts to randomly split each  $e^{(i)}$  into 3 shares  $e_1^{(i)}, e_2^{(i)}, e_3^{(i)}$ . This leads to the following expression:

$$h(x) = \sum_{i=1}^7 h\left((x_1 + e_1^{(i)}) + (x_2 + e_2^{(i)}) + (x_3 + e_3^{(i)})\right). \quad (12)$$

It then just remains to securely turn the  $(7, 3)$ -sharing  $(h_i)_{i \in [1,7]}$  into a 3-sharing  $(y_i)_{i \in [1,3]}$ . This is done by using the following sharing compression procedure specific to the case 7 to 3:

1.  $(m_1, m_2, m_3) \leftarrow^{\$} \mathbb{F}_{2^n}^q$
2.  $y_1 \leftarrow (h_1 + m_1) + (h_4 + m_2) + h_7$
3.  $y_2 \leftarrow (h_2 + m_1) + (h_5 + m_3)$
4.  $y_3 \leftarrow (h_3 + m_2) + (h_6 + m_3)$

### 5.2 Arbitrary Probing Security for Quadratic Functions

The idea in previous section can be extended to any security order  $d$  by starting from the following equation which generalizes (12):

$$h(x) = \sum_{i_1=1}^7 \sum_{i_2=1}^7 \cdots \sum_{i_t=1}^7 h\left(\sum_{j=1}^d x_j + e_j^{(i_1)} + e_j^{(i_1, i_2)} + \cdots + e_j^{(i_1, i_2, \dots, i_t)}\right), \quad (13)$$

where for every  $(j, q) \in \llbracket 1, d \rrbracket \times \llbracket 1, t \rrbracket$  and every tuple  $(i_1, \dots, i_{q-1}) \in \llbracket 1, 7 \rrbracket^{q-1}$  the family  $(e_j^{(i_1, \dots, i_{q-1}, i_q)})_{i_q \in \llbracket 1, 7 \rrbracket}$  is 3-spanned from fresh random values. Let us denote by  $h^{(i_1, i_2, \dots, i_t)}$ , with  $(i_1, \dots, i_t) \in \llbracket 1, 7 \rrbracket^t$ , the terms in the right-hand sum in (13). It may be checked that the family  $\{h^{(i_1, i_2, \dots, i_t)} \mid (i_1, i_2, \dots, i_t) \in \llbracket 1, 7 \rrbracket^t\}$  forms a  $(7^t, 3^t)$ -sharing of  $h(x)$ . To turn the latter into a  $3^t$ -sharing (which leaves us with a  $d$ -sharing of  $h(x)$  taking  $t = \log_3(d)$ ) the sharing compression procedure is recursively applied. This leads to the following recursive algorithm.

---

**Algorithm 3.** TreeExplore

---

```

Input: a  $d$ -sharing  $(z_1, z_2, \dots, z_d)$  of  $z \in \mathbb{F}_{2^n}$ , a depth parameter  $k$ 
Output: a  $3^k$ -sharing of  $h(z)$ 
1 if  $k = 0$  then
2   return  $h(z_1 + z_2 + \dots + z_d)$ 
3 for  $j = 1$  to  $d$  do
4    $(e_j^{(1)}, e_j^{(2)}, \dots, e_j^{(7)}) \leftarrow \text{RandSpan}(3)$ 
5 for  $i = 1$  to  $7$  do
6    $(h_{(w)}^{(i)})_{w \in \{1,2,3\}^{k-1}} \leftarrow \text{TreeExplore}(z_1 + e_1^{(i)}, z_2 + e_2^{(i)}, \dots, z_d + e_d^{(i)}, k - 1)$ 
7 for  $w$  in  $\{1, 2, 3\}^{k-1}$  do
8    $(h_{(w||1)}, h_{(w||2)}, h_{(w||3)}) \leftarrow \text{SharingCompress}(h_{(w)}^{(1)}, h_{(w)}^{(2)}, \dots, h_{(w)}^{(7)})$ 
9 return  $(h_{(w)})_{w \in \{1,2,3\}^k}$ 

```

---

A detailed and didactic description of the method together with a proof of probing security is provided in the full version of this paper.

**Complexity.** The following table summarizes the complexity of Algorithm 3 in terms of additions, evaluation of  $h$ , and random generation over  $\mathbb{F}_{2^n}$ .

	#add	#eval <sub>h</sub>	# rand
Exact count	$3d \cdot 7^t - 2d + \frac{5}{2}(7^t - 3^t)$	$7^t$	$\frac{d}{2}7^t - \frac{d}{2} + \frac{3(7^t - 3^t)}{4}$
Approximation	$3d^{2.77}$	$d^{1.77}$	$\frac{1}{2}d^{2.77}$

## 6 Adapting the CRV Method to Low Algebraic Degrees

This section proposes an adaptation of Coron-Roy-Vivek’s method (CRV) with improved complexity for functions with given (low) algebraic degree. We start by recalling the CRV method [9].

### 6.1 The CRV Method

In the following, we shall view functions over  $\mathcal{B}_{n,m}$  as polynomials over  $\mathbb{F}_{2^n}[x]$ . Let  $h(x) = \sum_{j=0}^{2^n-1} a_j x^j$  be the function that must be securely evaluated. To find a representation of  $h(x)$  that minimizes the number of nonlinear multiplications, CRV starts by building the union set  $\mathcal{L} = \bigcup_{i=1}^{\ell} \mathcal{C}_{\alpha_i}$  where  $\mathcal{C}_{\alpha_i}$  is the cyclotomic class of  $\alpha_i$  defined as  $\mathcal{C}_{\alpha_i} = \{2^j \cdot \alpha_i \bmod (2^n - 1) ; j \in \llbracket 0, n - 1 \rrbracket\}$  such that  $\alpha_1 = 0, \alpha_2 = 1$ , and  $\alpha_{i+1} \in \bigcup_{j=1}^i \mathcal{C}_{\alpha_j} + \bigcup_{j=1}^i \mathcal{C}_{\alpha_j}$  for every  $i$ . The elements in  $\{x^\alpha ; \alpha \in \mathcal{L}\}$  can then be processed with only  $\ell - 2$  nonlinear multiplications.<sup>5</sup> The set  $\mathcal{L}$  must satisfy the constraint  $\mathcal{L} + \mathcal{L} = \llbracket 0, 2^n - 1 \rrbracket$  and, if possible, the  $\ell$  classes  $\mathcal{C}_{\alpha_i}$  in  $\mathcal{L}$  are chosen such that  $|\mathcal{C}_{\alpha_i}| = n$ .

Let  $\mathcal{P} \subseteq \mathbb{F}_{2^n}[x]$  be the subspace spanned by the monomials  $x^\alpha$  with  $\alpha \in \mathcal{L}$ . The second step of CRV consists in randomly generating  $t - 1$  polynomials  $q_i(x) \in \mathcal{P}$  and in searching for  $t$  polynomials  $p_i(x) \in \mathcal{P}$  such that

$$h(x) = \sum_{i=1}^{t-1} p_i(x) \times q_i(x) + p_t(x). \tag{14}$$

This gives a linear system with  $2^n$  equations (one for each  $x \in \mathbb{F}_{2^n}$ ) and  $t \times |\mathcal{L}|$  unknowns (the coefficients of the  $p_i$ ). Such a system admits a solution for every choice of  $h$  if its rank is  $2^n$ , which leads to the necessary condition  $t \times |\mathcal{L}| \geq 2^n$ . Finding such a solution provides a method to evaluate  $h$  involving  $\ell + t - 3$  multiplications:  $\ell - 2$  multiplications to generate the monomial  $(x^j)_{j \in \mathcal{L}}$ , from which  $p_i(x)$  and  $q_i(x)$  are computed as linear combinations for every  $i \leq t$ , and  $t - 1$  multiplications to evaluate (14). In order to optimize the number of linear operations, the polynomials  $p_i$  can be represented as  $p_i(x) = \sum_{\alpha_j \in \mathcal{L}} \ell_{i,j}(x^{\alpha_j})$  where the  $\ell_{i,j}$  are linearized polynomials (*i.e.* polynomials of algebraic degree 1) that might be tabulated.

**Complexity.** Assuming that all the cyclotomic classes in  $\mathcal{L}$  except  $\mathcal{C}_0$  have maximum size  $n$  (*i.e.*  $|\mathcal{L}| = 1 + n \times (\ell - 1)$ ) and that the lower bound  $t \times |\mathcal{L}| \geq 2^n$  is reached, it is argued in [9] that the complexity of CRV is minimized for  $t = \lceil \sqrt{2^n/n} \rceil$  and  $\ell = \lceil \sqrt{2^n/n} - 1/n + 1 \rceil$ . Moreover, it is empirically shown in [9] that these lower bounds are often achieved for  $n \leq 12$ . Using ISW to secure nonlinear multiplications we get the following complexity (where #eval LP denotes the number of evaluations of a linearized polynomial):

<sup>5</sup> For  $\alpha_1 = 0$  and  $\alpha_2 = 1$  the building requires no nonlinear multiplication.



#add	#eval LP	#rand	#mult
$2d^2(t + \ell - 3) + d(t\ell - 2t - 3\ell + 5)$	$d\ell t$	$\frac{d(d-1)(t+\ell-3)}{2}$	$d^2(t + \ell - 3)$

*Remark 4.* Some of the  $\ell - 2$  nonlinear multiplications used to generate the set of powers  $\{x^\alpha; \alpha \in \mathcal{L}\}$  might take the form  $x^{\alpha_i} \cdot (x^{\alpha_i})^{2^j}$  with  $\alpha_i \in \mathcal{L}$ . In that case, the CPRR scheme (Algorithm 2) may be preferred to ISW. Indeed, as discussed in Sect. 4.2, this algorithm may be more efficient than ISW when  $x \mapsto x \cdot x^{2^j}$  can be tabulated whereas  $(x, y) \mapsto x \cdot y$  cannot. The same observation applies for the tweaked CRV method proposed in the next section for low-degree functions. In the complexity comparisons discussed in Sect. 7, this optimization is used (we have actually observed that it was always possible to entirely build  $\mathcal{L}$  based on  $\ell - 2$  multiplications of the form  $x \mapsto x \cdot x^{2^j}$ ).

### 6.2 The CRV Method for Degree- $s$ Functions

We propose here an adaptation of the CRV method for functions with (low) algebraic degree  $s$ . For the generation of  $\mathcal{L}$ , the constraint becomes  $\mathcal{L} + \mathcal{L} = \{\alpha \in \llbracket 0, 2^n - 1 \rrbracket; \text{HW}(\alpha) \leq s\}$ . When  $s$  is even, we impose the additional condition that every cyclotomic class  $\mathcal{C}_\alpha \subseteq \mathcal{L}$  verifies  $\text{HW}(\alpha) \leq \frac{s}{2}$  (the odd case is addressed hereafter). Then, as in the original method, we randomly generate  $t - 1$  polynomials  $q_i(x)$  in the subspace  $\mathcal{P}$  (*i.e.* the subspace of polynomials spanned by the monomials  $x^j$  with  $j \in \mathcal{L}$ ), and we try to solve a linear system obtained from (14). The difference is that the obtained system is of rank at most  $\sum_{r=0}^s \binom{n}{r}$ , *i.e.* the maximum number of non-null coefficients in a degree- $s$  function. Here again if this maximal rank is achieved, then the system has a solution for every target vector *i.e.* for every degree- $s$  function  $h(x)$ . The necessary condition to get a full-rank system then becomes  $t \times |\mathcal{L}| \geq \sum_{r=0}^s \binom{n}{r}$ . Assuming that the cyclotomic classes in  $\mathcal{L}$  except  $\mathcal{C}_0$  have maximum size  $n$ , this gives

$$t \geq \frac{\sum_{r=0}^s \binom{n}{r}}{n(\ell - 1) + 1}.$$

If this bound is reached, the number  $t + \ell - 3$  of nonlinear multiplications is minimized by taking  $t = (\ell - 1) = (\sum_{r=0}^s \binom{n}{r} / n)^{1/2}$ , and we get  $t + \ell - 3 = 2(\sum_{r=0}^s \binom{n}{r} / n)^{1/2} - 2$ . When  $s$  is odd, the method is similar but  $\mathcal{L}$  must contain some cyclotomic classes  $\mathcal{C}_\alpha$  such that  $\text{HW}(\alpha) \leq \frac{s+1}{2}$  and the  $q_i$  are constructed from powers  $x^\alpha$  such that  $\text{HW}(\alpha) \leq \frac{s-1}{2}$  (this ensures that the algebraic degree of  $p_i(x) \cdot q_i(x)$  is at most  $s$ ).

The complexity for this method is the same as for CRV (see table in the previous section), but for low-degree functions, the obtained parameters  $(t, \ell)$  are significantly smaller. The obtained number of nonlinear multiplications are compared in the following table.

	$n = 4$	$n = 5$	$n = 6$	$n = 7$	$n = 8$	$n = 9$	$n = 10$
Original CRV	2	4	5	7	10	14	19
CRV for $s = 2$	2	2	2	3	3	3	3
CRV for $s = 3$	2	3	4	5	5	6	7

## 7 Comparison

In this section, we first study the practical complexity of the methods introduced in Sects. 4, 5, and 6 to secure functions of low algebraic degrees (specifically of degrees  $s = 2$  and  $s = 3$ ). Then, we compare our algebraic decomposition method exposed in Sect. 3 with the CRV method (which is the best known alternative). The comparisons are done on specific examples where the dimension  $n$  fits with classical symmetric ciphers’  $s$ -boxes (*i.e.*  $n \in \{4, 8\}$ ) and the number  $d$  of shares ranges over  $\llbracket 2, 9 \rrbracket$ . An asymptotic analysis w.r.t parameters  $n$ ,  $d$ , and  $s$ , is further provided in the full version of the paper.

**Low-Degree Functions.** For the case  $s = 2$ , we compare Algorithm 2 (generalization of the CPRR scheme – see Sect. 4), Algorithm 3 (aka `TreeExplore` – see Sect. 5), and the tweaked CRV method for low-degree functions (aka `CRV-LD` – see Sect. 6). For the case  $s = 3$ , our first method (Algorithm 1) must be combined with a third-order secure evaluation method (primitive `SecureEval`) of degree-3 functions. For such a purpose, we either use Algorithm 3 (`TreeExplore`) or the tweaked CRV method (`CRV-LD`). The exact operations counts for these methods are plotted in Figs. 1 and 2.<sup>6</sup> In our comparisons, we assumed that an addition, a table lookup and a random generation of  $n$  bits have the same cost 1.<sup>7</sup> For the multiplication, we considered various possible costs  $C$ . The case  $C = 1$  corresponds to a device where the multiplication of two elements in  $\mathbb{F}_{2^n}$  can be precomputed and stored in a table, hence taking  $n2^{2n}$  bits of memory. When this is not possible (in a constraint environment and/or for too large values of  $n$ ), the multiplication must be implemented and its cost essentially depends on the device architecture.<sup>8</sup> We believe to encompass most realistic scenarios by considering  $C \in \{5, 10, 20\}$ . Note that for the case  $s = 3$ , we used the improved CRV method based on CPRR for the generation of powers as suggested in Remark 4. We did not use it for  $s = 2$  since a CPRR multiplication is similar to one call to Algorithm 2.

**Case ( $s = 2$ ).** Figure 1 clearly illustrates the superiority of Algorithm 2 for the secure evaluation of degree-2 functions at any order. The ranking between our

<sup>6</sup> Note that when  $d \leq s$ , Algorithm 1 is not specified and therefore cannot be applied.

<sup>7</sup> In the context of side-channel countermeasures, generating  $n$  random bits usually amounts to read a  $n$ -bit value in a TRNG register.

<sup>8</sup> In [12], the authors explain that the processing of a field multiplication with the CPU instructions set requires between 20 and 40 cycles and they give some examples of implementations.

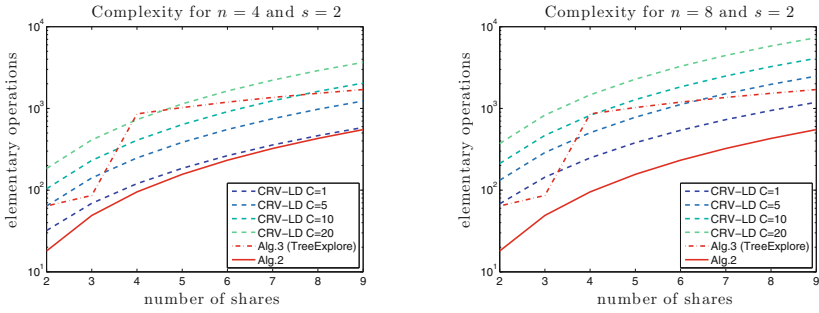


Fig. 1. Secure evaluation of quadratic functions

second method (Algorithm 3 aka *TreeExplore*) and the tweaked CRV method (CRV-LD) depends on the sharing order  $d$  and the multiplication cost ratio  $C$ . For high values of  $C$  (i.e. when the multiplication cannot be tabulated), *TreeExplore* outperforms the tweaked CRV method. It is particularly interesting for a sharing order  $d$  lower than 4. However, due to its tree structure of depth  $\log_3(d)$ , it becomes slower as soon as  $d$  reaches 4. Moreover for higher values of  $d$ , it would not be competitive since its asymptotic complexity is more than quadratic in the sharing order.

**Case (s = 3).** Fig. 2 show the superiority of the tweaked CRV method for sharing orders greater than 3 and even for costly multiplications (i.e.  $C = 20$ ). For small sharing orders  $d \in \{2, 3\}$ , Algorithm 3 (*TreeExplore*) is competitive (for the same reasons as for the degree-2 case). It appears that the use of our first method (Algorithm 1) for lowering the sharing order down to the algebraic degree is never interesting in this setting. We however think that this is not a dead-end point for this method and that the ideas used to obtain Algorithm 2 from the general description of the method could be used to get an improved version for the cubic case as well. We let this question open for further research on this subject.

**High-Degree Functions.** We now consider the algebraic decomposition method described in Sect. 3 and compare it to the CRV method. The following

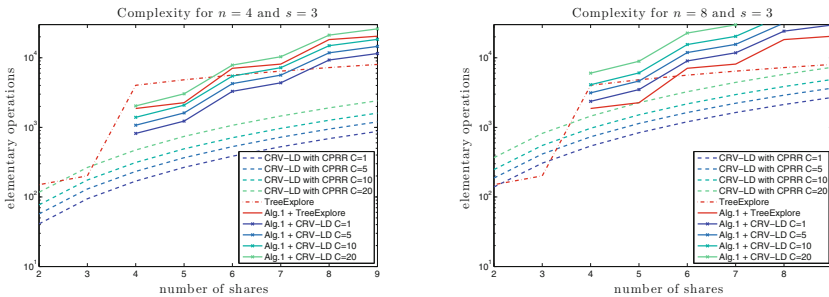


Fig. 2. Secure evaluation of cubic functions

table summarizes the number  $(\ell - 2) + (t - 1)$  of secure nonlinear multiplications involved in CRV, as well as the numbers  $r$  and  $t$  of secure evaluations of degree- $s_1$  functions and degree- $s_2$  functions in the algebraic decomposition method.

	This Paper					CRV		
	#eval-2	#eval-3	$(s_1, s_2)$	$r$	$t$	#mult	$\ell - 2$	$t - 1$
$n = 4$	3	-	(2, 2)	1	2	2	1	1
	1	1	(2, 3)	1	1			
	-	2	(3, 3)	1	1			
$n = 6$	5	-	(2, 2)	2	3	5	3	2
	1	2	(2, 3)	1	2			
	-	3	(3, 3)	1	2			
$n = 8$	11	-	(2, 2)	2	9	10	5	5
	2	3	(2, 3)	2	3			
	-	4	(3, 3)	1	3			

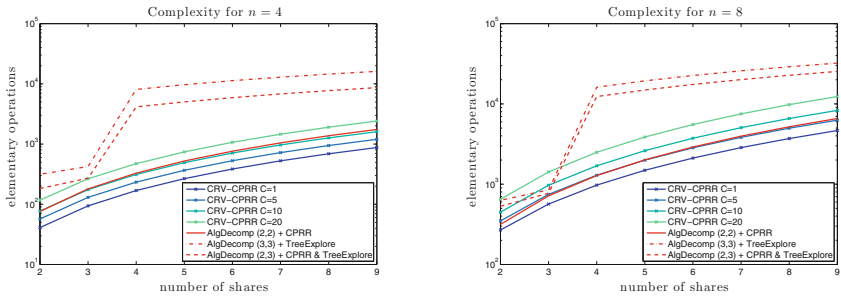


Fig. 3. Secure evaluation of nonlinear functions (arbitrary degree)

Figure 3 gives the overall cost of the CRV method and of our algebraic decomposition method for various values of  $n, d$  and  $C$  (these are the cross marked blue to green curves). We used the improved version of CRV suggested in Remark 4 (i.e. using CPRR multiplications for the first phase and ISW multiplications for the second phase). For our method, we considered quadratic decomposition (i.e.  $s_1 = s_2 = 2$ ) combined with Algorithm 2 for secure quadratic evaluations, as well as cubic decomposition ( $s_1 = s_2 = 3$ ) with TreeExplore for secure cubic evaluation. We further considered hybrid decomposition combined with both Algorithm 2 and TreeExplore.

We observe that the quadratic decomposition is always more efficient than the cubic and hybrid decompositions. This is due to the gap of efficiency between the secure quadratic evaluation (Algorithm 2) and the secure cubic evaluation (TreeExplore). Compared to CRV, the quadratic decomposition offers some efficiency gain depending on the multiplication cost. For  $n = 4$ , we observe that it is more efficient than CRV whenever a multiplication takes at least 10 elementary operations. For  $n = 8$ , the quadratic decomposition is better than

CRV whenever the multiplication cost exceeds 5 elementary operations. This shows that our algebraic decomposition approach is the best known method for the probing secure evaluation of nonlinear functions (such as s-boxes) in a context where the field multiplication is a costly operation.

## References

1. Balasch, J., Faust, S., Gierlichs, B.: Inner product masking revisited. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part I. LNCS, vol. 9056, pp. 486–510. Springer, Heidelberg (2015)
2. Barthe, G., Belaïd, S., Dupressoir, F., Fouque, P.-A., Grégoire, B., Strub, P.-Y.: Verified proofs of higher-order masking. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part I. LNCS, vol. 9056, pp. 457–485. Springer, Heidelberg (2015)
3. Bilgin, B., Nikova, S., Nikov, V., Rijmen, V., Stütz, G.: Threshold implementations of all  $3 \times 3$  and  $4 \times 4$  S-boxes. In: Prouff, E., Schaumont, P. (eds.) CHES 2012. LNCS, vol. 7428, pp. 76–91. Springer, Heidelberg (2012)
4. Bilgin, B., Nikova, S., Nikov, V., Rijmen, V., Tokareva, N., Vitkup, V.: Threshold implementations of small S-boxes. *Crypt. Commun.* **7**(1), 3–33 (2015)
5. Carlet, C.: Vectorial boolean functions for cryptography, Chap. 9. In: Crama, Y., Hammer, P. (eds.) *Boolean Models and Methods in Mathematics, Computer Science, and Engineering*, pp. 398–469. Cambridge University Press, Cambridge (2010)
6. Carlet, C., Goubin, L., Prouff, E., Quisquater, M., Rivain, M.: Higher-order masking schemes for S-boxes. In: Canteaut, A. (ed.) FSE 2012. LNCS, vol. 7549, pp. 366–384. Springer, Heidelberg (2012)
7. Coron, J.-S.: Fast Evaluation of Polynomials over Finite Fields and Application to Side-channel Countermeasures. Personal Communication, February 2014
8. Coron, J.-S., Prouff, E., Rivain, M., Roche, T.: Higher-order side channel security and mask refreshing. In: Moriai, S. (ed.) FSE 2013. LNCS, vol. 8424, pp. 410–424. Springer, Heidelberg (2014)
9. Coron, J.-S., Roy, A., Vivek, S.: Fast evaluation of polynomials over binary finite fields and application to side-channel countermeasures. In: Batina, L., Robshaw, M. (eds.) CHES 2014. LNCS, vol. 8731, pp. 170–187. Springer, Heidelberg (2014)
10. Duc, A., Dziembowski, S., Faust, S.: Unifying leakage models: from probing attacks to noisy leakage. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 423–440. Springer, Heidelberg (2014)
11. Genelle, L., Prouff, E., Quisquater, M.: Thwarting higher-order side channel analysis with additive and multiplicative maskings. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 240–255. Springer, Heidelberg (2011)
12. Grosso, V., Prouff, E., Standaert, F.-X.: Efficient masked S-boxes processing – a step forward –. In: Pointcheval, D., Vergnaud, D. (eds.) AFRICACRYPT. LNCS, vol. 8469, pp. 251–266. Springer, Heidelberg (2014)
13. 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)
14. Kim, H.S., Hong, S., Lim, J.: A fast and provably secure higher-order masking of AES S-box. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 95–107. Springer, Heidelberg (2011)

15. 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)
16. 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)
17. Kutzner, S., Nguyen, P.H., Poschmann, A.: Enabling 3-share threshold implementations for all 4-bit S-boxes. In: Lee, H.-S., Han, D.-G. (eds.) ICISC 2013. LNCS, vol. 8565, pp. 91–108. Springer, Heidelberg (2014)
18. Moradi, A., Poschmann, A., Ling, S., Paar, C., Wang, H.: Pushing the limits: a very compact and a threshold implementation of AES. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 69–88. Springer, Heidelberg (2011)
19. Nikova, S., Rijmen, V., Schl affer, M.: Secure hardware implementation of nonlinear functions in the presence of glitches. In: Lee, P.J., Cheon, J.H. (eds.) ICISC 2008. LNCS, vol. 5461, pp. 218–234. Springer, Heidelberg (2009)
20. Nikova, S., Rijmen, V., Schl affer, M.: Secure hardware implementation of nonlinear functions in the presence of glitches. *J. Cryptology* **24**(2), 292–321 (2011)
21. Poschmann, A., Moradi, A., Khoo, K., Lim, C., Wang, H., Ling, S.: Side-channel resistant crypto for less than 2, 300 GE. *J. Cryptology* **24**(2), 322–345 (2011)
22. Prouff, E., Rivain, M.: Masking against side-channel attacks: a formal security proof. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 142–159. Springer, Heidelberg (2013)
23. Prouff, E., Roche, T.: Higher-order glitches free implementation of the AES using secure multi-party computation protocols. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 63–78. Springer, Heidelberg (2011)
24. Rivain, M., Prouff, E.: Provably secure higher-order masking of AES. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 413–427. Springer, Heidelberg (2010)
25. Roy, A., Vivek, S.: Analysis and improvement of the generic higher-order masking scheme of FSE 2012. In: Bertoni, G., Coron, J.-S. (eds.) CHES 2013. LNCS, vol. 8086, pp. 417–434. Springer, Heidelberg (2013)
26. Satoh, A., Morioka, S., Takano, K., Munetoh, S.: A compact Rijndael hardware architecture with S-box optimization. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, p. 239. Springer, Heidelberg (2001)

# Consolidating Masking Schemes

Oscar Reparaz<sup>(✉)</sup>, Begül Bilgin, Svetla Nikova, Benedikt Gierlichs,  
and Ingrid Verbauwhede

ESAT/COSIC and iMinds, KU Leuven, Leuven, Belgium  
`oscar.reparaz@esat.kuleuven.be`

**Abstract.** In this paper we investigate relations between several masking schemes. We show that the Ishai–Sahai–Wagner private circuits construction is closely related to Threshold Implementations and the Trichina gate. The implications of this observation are manifold. We point out a higher-order weakness in higher-order Threshold Implementations, suggest a mitigation and provide new sharings that use a lower number of input shares.

**Keywords:** Masking · Private circuits · Ishai–Sahai–Wagner · Threshold implementations · Trichina gate · Higher-order DPA

## 1 Introduction

Side-channel cryptanalysis allows to break implementations of mathematically secure cryptographic algorithms running on embedded devices. Shortly after the introduction of a particularly powerful branch of side-channel attacks, namely Differential Power Analysis (DPA) by Kocher et al. [15], different countermeasures were proposed. An especially popular countermeasure used today is masking, introduced in [7, 12], mainly due to its theoretical soundness. Contrary to other heuristic, ad-hoc approaches, masking carries a proof of security [7]. A  $d^{\text{th}}$ -order secure masking works by splitting every sensitive variable (i.e. that depends on the key) into  $s$  shares, such that an adversary probing at most  $d$  values during the computation gets no information about any sensitive variable. This adversarial model is relevant in practice since the adversary is not weaker than a  $d^{\text{th}}$ -order DPA attack [8, 11]. The advantage of a properly masked implementation is that it forces the adversary to use higher-order DPA attacks in order to break it. Higher-order DPA attacks are substantially harder to launch, both in terms of data complexity and computational resources [7, 19, 24]. Masking, however, comes with a cost. Performing operations in the masked domain increases the computational requirements on the target platform (area, time and randomness, among others), thus in practice, it is crucial to design countermeasures that have a limited cost impact.

In this paper, we focus on Boolean masking, i.e. the intermediates are split additively in a given finite field. The difficulty is then reduced to masking functions that are not linear with respect to addition.

## 1.1 Related Works

There have been several efforts for constructing masking schemes — algorithms to compute on masked data. Some early development came from practitioners which produced designs mostly oriented to fit in real-world constraints: Trichina presents in [29] a masked AND gate resistant to first-order DPA attacks (first-order secure masking).

A generic algorithm for the masked AND computation at any security level is given by Ishai, Sahai and Wagner (here ISW) in [14], together with a convenient theoretical framework to prove the security of such a scheme. It is, however, well known that early theoretical concepts of masking schemes rely on assumptions that do not necessarily hold in practice. This is true for both the hardware and the software side. A common problem for the latter is that Hamming distance leakage, which is typically visible in memory-element transitions, may invalidate the assumption that leakages from each share are independent [1]. For the former, glitches (in static CMOS, a spurious transition of nodes in a combinational circuit within one clock cycle, resulting from different arrival times of the input signals) were shown to be a source of exploitable leakage [17, 18], enabling successful DPA attacks against theoretically sound masked implementations due to unsatisfactory leakage modeling.

The mitigation of glitches is a well-studied problem in digital design, since they are not only inconvenient from a security point of view. Glitches are useless transitions that consume extra energy and thus digital designers tend to minimize them to achieve low-power and high-speed circuits. There are strategies to reduce glitches (e.g., balancing the path delay using combinational tree-like structures) or fully eliminate them (e.g. using dynamic logic styles, such as Domino or dynamic differential such as SABL [27] or WDDL [28]).

Alternatively, a specific strand of masking schemes, namely Threshold Implementations (TIs), were introduced in [21] to address the aforementioned model limitations. TIs are designed to deal with non-idealities in hardware (glitches) at a higher level of abstraction, and can provide strong security guarantees that may be relevant in practice. While ISW requires first to decompose a circuit into (exclusively) AND, XOR and NOT gates and then masking those, TI has the advantage that any function can be shared directly, which typically results in more compact designs. Recently TIs were extended to provide not only first-order but also higher-order security [3].

## 1.2 Our Contribution

The discussion provided in this paper is threefold. First, we point out the similarities and differences between ISW, TI and the Trichina gate when the function to mask is an AND gate. We gain deeper understanding about masking schemes from these relations and use it to provide a generalized masking scheme (Sect. 3).

In the second part of the paper, we show how this generalization is mutually beneficial to all three masking schemes mentioned above. We show a weakness in the recently proposed higher-order extension of TI and suggest a fix using



ideas from the generalized scheme in Sect. 4. In addition, we discuss how ISW and the Trichina masked AND-gate can be implemented securely in logic styles that glitch.

Finally, we focus on constructive applications. In Sect. 5.1, we discuss under which conditions a TI function provides security against  $d^{\text{th}}$ -order attacks using only  $d + 1$  shares instead of the usual  $td + 1$  bound. We end the paper by describing how ideas from TI could be inherited in software-oriented schemes to provide security in a distance-based leakage model (Sect. 5.2).

## 2 Preliminaries

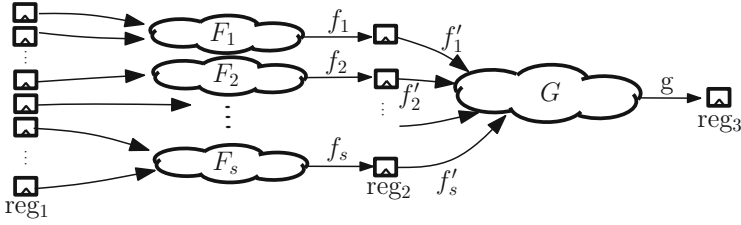
We begin with standard definitions and descriptions of the masking schemes that we consider. Lower-case characters refer to elements in a field with characteristic two. An element  $a$  is split into  $s$  shares  $a_i$ , where  $i \in 1, 2, \dots, s$  by means of Boolean masking. Namely, without loss of generality  $s - 1$  random shares  $a_1, \dots, a_{s-1}$  are drawn from the uniform distribution, then  $a_s$  is calculated such that  $a = \bigoplus_s a_i$ . Bold characters refer to a valid shared vector  $\mathbf{a} = a_1, \dots, a_s$ . We use the term  $s$ -share representation ( $s$ -sharing) of  $a$  to emphasize the number of shares. Note that the sharing  $\mathbf{a}$  generated as detailed above is uniform [4]. Moreover, the sharing  $\mathbf{a}_i = a_1, \dots, a_{i-1}, a_{i+1}, \dots, a_s$  is independent of the unshared value  $a$  for any choice of  $i$ . It is hence an  $(s, s)$  secret sharing.

We use upper-case characters to denote functions. For a given unshared function  $b = F(a)$ , we generate a shared vector  $\mathbf{F} = F_1, \dots, F_s$  of component functions  $F_i$  in order to perform the shared calculation. The sharing  $\mathbf{F}$  is *correct* if  $b = \bigoplus_s b_i$  for  $b_i = F_i(\mathbf{a})$ . The algebraic degree of a function is denoted with  $t$ .

*Adversarial Model.* We use  $d$  probing as our adversarial model which we define as follows. The adversary is allowed to probe  $d$  wires in the circuit within a certain time window. Each probed wire  $g$  calculating a function  $G$  gives information about all the inputs of  $G$  up to the latest synchronization point<sup>1</sup>. This definition directly implies that the adversary can derive all the intermediate values during the computation of  $G$  and hence the output of  $G$ . To clarify, let us refer to Fig. 1. An attacker probing the output of the function  $G$  (that is, wire  $g$ ) can observe all the inputs to  $G$  up to, and including,  $\text{reg}_2$ ; can generate all the intermediate values used during the calculation of  $G$  (including the outputs of  $G$  that are stored in  $\text{reg}_3$ ); but can not directly learn all the values stored in  $\text{reg}_1$  or any intermediate values occurring during the calculations of each  $F_i$ .

We note that this is a theoretical model stronger than a real-world attacker since a real-world attacker can only get a subset of the mentioned information. Moreover, it is slightly different than the conventional  $d$ -probing model [14]. Nevertheless, it is advantageous since being able to see the inputs of the gates used during the calculation implies the ability to observe real world effects such as glitches. Hence, we gain the flexibility to work also with non-ideal (glitchy)

<sup>1</sup> One of the many ways of synchronization is storing elements in registers which we inherit throughout this paper without loss of generalization.



**Fig. 1.** Exemplary circuit to aid the explanation of the adversarial model adopted in this paper.  $F_i$  and  $G$  are combinational logic blocks,  $reg_i$  are register stages and  $f_i$  and  $g$  are wires that compute the  $F_i$  (resp.  $G$ ) functions.

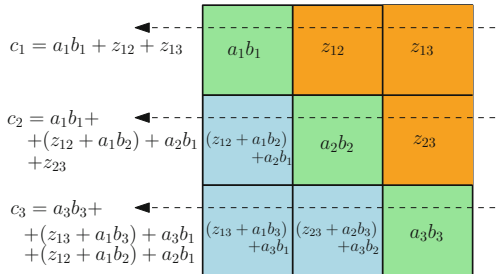
gates. If the usage of ideal gates is assumed, working with the conventional model is typically sufficient.

This model matches quite nicely with  $d^{\text{th}}$ -order DPA attacks, which consider a noisy function of intermediates’ leakage [11]. There are certainly other adversarial models that are even more powerful, in which the attacker has the ability to adaptively move the probes between time periods (but not within a time period). We note that this “adaptive-probes” model is stronger than  $d^{\text{th}}$ -order DPA model and we do not consider moving probes in this paper.

```

Require:  $s$ -shares  $\mathbf{a}$  and  $\mathbf{b}$ 
Ensure:  $s$ -shares  $\mathbf{c}$  satisfying  $c = ab$ 
for  $i$  from 1 to  $s$  do
  for  $j$  from  $i + 1$  to  $s$  do
     $z_{ij} \leftarrow \text{rnd}()$ 
     $z_{ji} \leftarrow (z_{ij} \oplus a_i b_j) \oplus a_j b_i$ 
  end for
end for
for  $i$  from 1 to  $s$  do
   $c_i \leftarrow a_i b_i$ 
  for  $j$  from 1 to  $s$ ,  $j \neq i$  do
     $c_i \leftarrow c_i \oplus z_{ij}$ 
  end for
end for
    
```

**Fig. 2.** ISW algorithm.



**Fig. 3.** Intermediate state of the ISW computation for  $s = 3$ .

*Ishai–Sahai–Wagner Scheme.* Private circuits [14] provide a procedure for computation on masked data. They give a construction for NOT and AND gates, and prove the security against  $d$  probes of any circuit composed of these secure gates (“gadgets”) which are in turn built from logic gates that do not glitch. To compute  $c = F(a, b) = ab$  while providing security against  $d$  probes, ISW

takes  $s = 2d + 1$  shares  $\mathbf{a}$  and  $\mathbf{b}$  of each input and consumes  $\binom{s}{2}$  bits of randomness. We exemplify the computation of a masked AND gate providing security against adversaries using one ( $d = 1, s = 3$ ) and two ( $d = 2, s = 5$ ) probes in Eqs. (1), (2) and (3) respectively. An intermediate state of the computation is shown in Fig. 3.

$$\begin{aligned}
 z_{21} &= (z_{12} \oplus a_1b_2) \oplus a_2b_1, & c_1 &= a_1b_1 \oplus z_{12} \oplus z_{13}, \\
 z_{31} &= (z_{13} \oplus a_1b_3) \oplus a_3b_1, & c_2 &= a_2b_2 \oplus z_{21} \oplus z_{23}, \\
 z_{32} &= (z_{23} \oplus a_2b_3) \oplus a_3b_2, & c_3 &= a_3b_3 \oplus z_{31} \oplus z_{32}.
 \end{aligned}
 \tag{1}$$

First, three (resp. ten) bits of randomness  $z_{ij}$  where  $1 \leq i < j \leq s$  are drawn i.i.d. uniformly random. Then the intermediates  $z_{ji}$  are computed as shown in the left column of Eq. (1) (resp. Eq. (2)). The last step xors the intermediates  $z_{ij}$  and the products  $a_i b_i$  to compute the  $s$  output shares  $\mathbf{c}$  (right column of Eqs. (1) and (3) respectively).

$$\begin{aligned}
 z_{21} &= (z_{12} \oplus a_1b_2) \oplus a_2b_1, & z_{31} &= (z_{13} \oplus a_1b_3) \oplus a_3b_1, \\
 z_{41} &= (z_{14} \oplus a_1b_4) \oplus a_4b_1, & z_{51} &= (z_{15} \oplus a_1b_5) \oplus a_5b_1, \\
 z_{32} &= (z_{23} \oplus a_2b_3) \oplus a_3b_2, & z_{42} &= (z_{24} \oplus a_2b_4) \oplus a_4b_2, \\
 z_{52} &= (z_{25} \oplus a_2b_5) \oplus a_5b_2, & z_{43} &= (z_{34} \oplus a_3b_4) \oplus a_4b_3, \\
 z_{53} &= (z_{35} \oplus a_3b_5) \oplus a_5b_3, & z_{54} &= (z_{45} \oplus a_4b_5) \oplus a_5b_4.
 \end{aligned}
 \tag{2}$$

$$\begin{aligned}
 c_1 &= a_1b_1 \oplus z_{12} \oplus z_{13} \oplus z_{14} \oplus z_{15}, & c_4 &= a_4b_4 \oplus z_{41} \oplus z_{42} \oplus z_{43} \oplus z_{45}, \\
 c_2 &= a_2b_2 \oplus z_{21} \oplus z_{23} \oplus z_{24} \oplus z_{25}, & c_5 &= a_5b_5 \oplus z_{51} \oplus z_{52} \oplus z_{53} \oplus z_{54}. \\
 c_3 &= a_3b_3 \oplus z_{31} \oplus z_{32} \oplus z_{34} \oplus z_{35},
 \end{aligned}
 \tag{3}$$

Extensions to higher orders are similarly generated using the algorithm in Fig. 2.

It is well known that the ISW algorithm can work in larger finite fields by building upon field multiplications instead of AND gates. In the case of AES, there is a significant performance gain if ISW operates in  $\text{GF}(2^8)$ , due to the algebraic structure of the AES S-box [25]. We refer to [14] for a variant of this method using  $s = d + 1$  shares.

*Threshold Implementations.* TI provides provable security against  $d^{\text{th}}$ -order DPA even in a circuit with glitches according to [3]. In addition, it is also advantageous since any degree  $t$  function can be securely implemented using at least  $s \geq td + 1$  shares.

The security of a single function relies on the satisfaction of  $d^{\text{th}}$ -order *non-completeness*: any combination of up to  $d$  component functions  $F_i$  of  $\mathbf{F}$  must be independent of at least one input share. It is shown that such a sharing can always be constructed using  $s_{in} = td + 1$  and  $s_{out} = \binom{s_{in}}{t}$  shares. Examples for the function  $d = F(a, b, c) = c \oplus ab$  are given in Eqs. (4) and (5) for  $d = 1$  and  $d = 2$  respectively.

$$\begin{aligned}
 d_1 &= c_2 \oplus a_2b_2 \oplus a_1b_2 \oplus a_2b_1, \\
 d_2 &= c_3 \oplus a_3b_3 \oplus a_3b_2 \oplus a_2b_3 \\
 d_3 &= c_1 \oplus a_1b_1 \oplus a_1b_3 \oplus a_3b_1.
 \end{aligned}
 \tag{4}$$

Notice that  $s_{out} > s_{in}$  for  $d > 1$ . In order to avoid further increase of the number of shares when several functions are cascaded, some of the output shares are typically xored. It is important that this reduction is performed only after the  $s_{out}$ -sharing  $\mathbf{d}$  is stored in the registers in order to satisfy the non-completeness property and to avoid glitches depending on all shares of a variable.

$$\begin{aligned}
 d_1 &= c_2 \oplus a_2b_2 \oplus a_1b_2 \oplus a_2b_1, & d_2 &= c_3 \oplus a_3b_3 \oplus a_1b_3 \oplus a_3b_1, \\
 d_3 &= c_4 \oplus a_4b_4 \oplus a_1b_4 \oplus a_4b_1, & d_4 &= c_1 \oplus a_1b_1 \oplus a_1b_5 \oplus a_5b_1, \\
 d_5 &= a_2b_3 \oplus a_3b_2, & d_6 &= a_2b_4 \oplus a_4b_2, \\
 d_7 &= c_5 \oplus a_5b_5 \oplus a_2b_5 \oplus a_5b_2, & d_8 &= a_3b_4 \oplus a_4b_3, \\
 d_9 &= a_3b_5 \oplus a_5b_3, & d_{10} &= a_4b_5 \oplus a_5b_4.
 \end{aligned} \tag{5}$$

In order to provide security when several functions are cascaded, (i.e. the output of  $\mathbf{F}$  is used as the input to another shared nonlinear-function  $\mathbf{G}$ ), the shared function and its output should satisfy *uniformity* [4]. Several methods to achieve uniformity have been proposed [5, 6, 16, 22]. It is advised to use re-masking [4, 20] in case these methods do not provide a solution.

When a single AND gate is considered, it has been shown that there exists no 3-sharing satisfying both uniformity and first-order non-completeness [5]. Therefore, the output shares of the 3-share AND gate must be re-masked (refreshed). Moreover, the sharing in Eq. (4) considering an AND and XOR gate instead of a single AND gate satisfies all TI properties.

*Trichina AND-gate.* Unlike ISW and TI which can be applied both at the algorithm and at the gate level, Trichina [29] investigates how to implement a masked AND gate  $c = ab$  securely strictly at the gate level. The construction, which is described in Eq. (6), requires two 2-share inputs and uses 1-bit extra randomness  $z_{12}$  to generate a 2-share output. The security relies strictly on the order of the operations to avoid unmasking certain bits, on the ideality of the cells and on the assumption that the sharing  $\mathbf{a}$  of  $a$  is independent from  $\mathbf{b}$ .

$$c_1 = (((z_{12} \oplus a_1b_2) \oplus a_2b_1) \oplus a_2b_2) \oplus a_1b_1, \quad c_2 = z_{12}. \tag{6}$$

### 3 Conciliation

In this section we mainly describe a generalized masking scheme and argue its security. In order to do that, we first relate the ISW scheme with TI and the Trichina gate using elementary transformations. We then use ingredients from all three schemes in our generalized construction. As a case study, we consider a first-order sharing of an AND gate.

#### 3.1 From ISW to TI

Consider the ISW construction with  $s = 3$  input shares, providing first-order security as depicted in Fig. 4. It is equivalent to the computation in Eq. (1) and to Fig. 2. In Fig. 4, the computation flows from the outside towards the center.

It begins with deriving all the cross products  $a_i b_j$ . Then three random values  $z_{ij}$  are added to some of the cross products. The terms are finally xored together in three groups to generate the output shares  $c_i$ .

In the following, we perform several elementary transformations on this circuit to arrive to a typical re-masked 3-share TI of an AND gate.

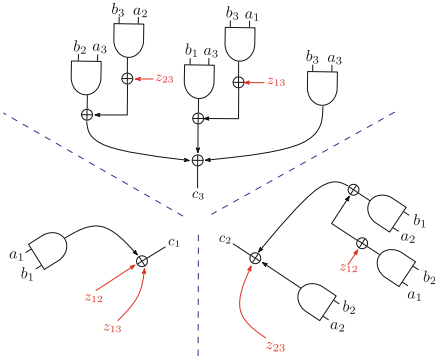


Fig. 4. Original ISW scheme.

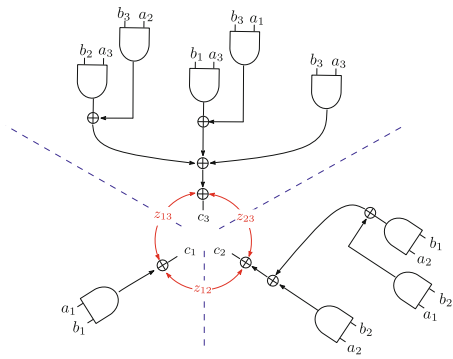


Fig. 5. After first transformation.

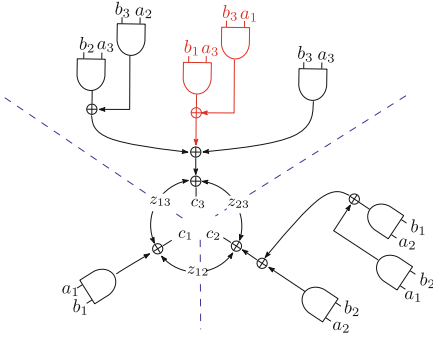
*First Transformation: Moving Random Bits.* Delaying the injection of randomness using the random bits  $z_{ij}$  closer to the center (towards the end of the calculation) as depicted in red yields the construction in Fig. 5. Of course, this operation preserves the correctness of the output. It is already possible to recognize a refreshing operation in the inner ring where  $z_{12}$ ,  $z_{13}$  and  $z_{23}$  are involved. Note that the security of this intermediate construction highly depends on the order of computation of the XOR gates and the ideality (glitching behavior) of the gates.

*Second Transformation: Moving AND Gates.* The next modification transforms the circuit of Fig. 6 into Fig. 7. It simply moves around the two red AND gates  $a_1 b_3$  and  $a_3 b_1$  together with the XOR gate from the upper to the lower-left branch.

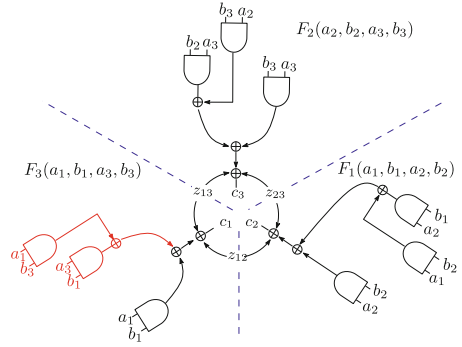
This second transformation also preserves the correctness at the output. Notice that after this transformation each branch of the circuit sees at most two shares of each input. For example, the upper branch sees only  $a_2$ ,  $a_3$ ,  $b_2$  and  $b_3$ . We can absorb the computation from each branch (3 ANDs and 2 XORs) into its respective component function  $F_i$  as shown in Eq. (7).

$$\begin{aligned}
 F_1(a_1, a_2, b_1, b_2) &= a_2 b_2 \oplus a_1 b_2 \oplus a_2 b_1, \\
 F_2(a_2, a_3, b_2, b_3) &= a_3 b_3 \oplus a_2 b_3 \oplus a_3 b_2, \\
 F_3(a_1, a_3, b_1, b_3) &= a_1 b_1 \oplus a_3 b_1 \oplus a_1 b_3.
 \end{aligned}
 \tag{7}$$

The reader will recognize that the resulting sharing  $\mathbf{F}$  is a TI (satisfying first-order non-completeness) followed by a refreshing (resulting from the first



**Fig. 6.** Before second transformation.



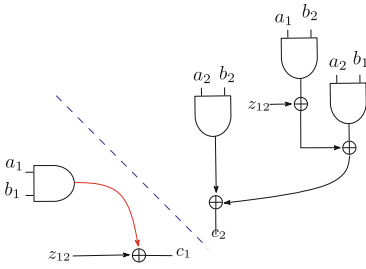
**Fig. 7.** After second transformation.

transformation.) Note that one could also equivalently see this refreshing as an addition with the uniform shares of the constant value 0.

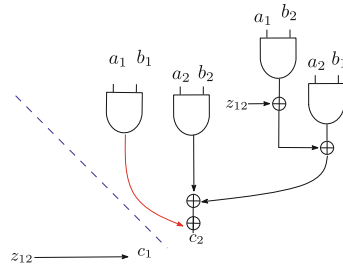
The security of this construction follows from the fact that it is a TI, followed by a refreshing. In particular, this construction is secure even in the presence of glitches. Therefore, we link the  $s = 3$  ISW scheme to first-order TI.

### 3.2 From ISW to the Trichina AND-gate

In Fig. 8, we draw the ISW computation<sup>2</sup> of an AND gate for  $s = 2$ . In Fig. 9, we have the Trichina AND gate. Similar to Sect. 3.1, we can transform the ISW construction  $s = 2$  to the Trichina gate by rearranging the term  $a_1 b_1$ . It is noteworthy that the Trichina gate resembles a simplified ISW, and thus can be seen as the practitioners version.



**Fig. 8.** ISW with  $s = 2$ .



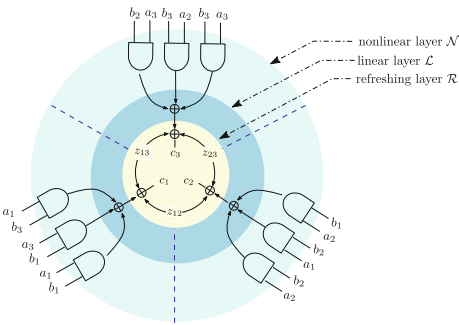
**Fig. 9.** Trichina AND gate.

<sup>2</sup> We stress that ISW with  $s = 2$  is neither strictly defined nor proven secure in the ISW simulation model. We are extending the algorithm in Fig. 2 in a straight forward way to any  $s$ .

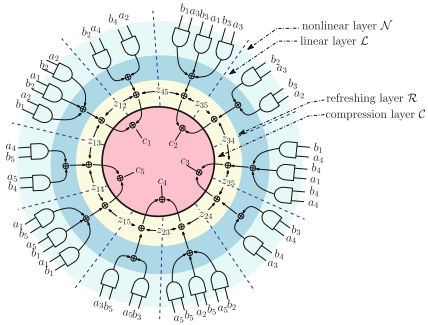
### 3.3 Generalizing and Inducing a Structure

We can generalize the masked AND-gate transformations from Sects. 3.1 and 3.2 to the general case of higher orders. In addition, we can construct variants that compute logic gates with more than two inputs or more sophisticated functions.

In order to induce a structure to the mentioned generalization, we decompose the resulting construction into four layers as exemplified in Figs. 10 and 11 for first- and second-order security respectively. Specifically, we notice a non-linear layer  $\mathcal{N}$ , followed by a linear layer  $\mathcal{L}$  and a refreshing layer  $\mathcal{R}$ . In certain cases where we want to reduce the number of shares, we add a linear compression layer  $\mathcal{C}$ . Below we detail the functionality of each layer.



**Fig. 10.** First-order secure ( $s = 3$ ) after transformation.



**Fig. 11.** Second-order secure ( $s = 5$ ) after transformation.

*The Non-linear Layer  $\mathcal{N}$ .* This layer is responsible for the bulk of the computation, corresponding to the cross products  $a_i b_j$ . For example, in the first-order secure construction of a 3-share 2-input AND gate,  $\mathcal{N}(\mathbf{a}, \mathbf{b}) = (a_1 b_1, a_1 b_2, \dots, a_3 b_3)$  maps 6 input bits to 9 output bits ( $a_i b_j$ ). Note that the set of cross products calculated in this layer is defined by the number of shares and the function itself.<sup>3</sup>

In order to generalize the construction such that a function other than  $c = ab$  is computed (such as  $d = abc$ ,  $d = a \oplus bc$ ,  $d = a \oplus bc \oplus abc$ ),  $\mathcal{N}$  needs to be modified accordingly. To be specific, all the shared terms (cross products and linear terms) should be placed in  $\mathcal{N}$  to be used in the following steps.

*The Linear Layer  $\mathcal{L}$ .* This layer is an XOR net that reduces the number of shares without modifying the unshared value. In the AND-gate example, it maps 9 input bits (output of  $\mathcal{N}$ ) to 3 output bits for the first-order case and 25 input bits to 10 output bits in the second-order case. The linear layer  $\mathcal{L}$  of TI is responsible for preserving non-completeness. Failure to achieve non-completeness can cause

<sup>3</sup> It is possible to add other terms to  $\mathcal{N}$  (as long as they are inserted an even number of times), such as virtual variable pairs [6], in order to increase the flexibility for generating a uniform sharing.

sensitive information leakage in a glitchy circuit as in the case of the original ISW scheme (see Sect. 4.2).

The reduction of the number of shares performed by  $\mathcal{L}$  partially limits the exponential blow-up of shares otherwise caused by the non-linear layer  $\mathcal{N}$  alone. We point out that the output of  $\mathcal{N}$  is already a valid, non-complete sharing when each cross product is considered as an output share. However, cascading several sharings without  $\mathcal{L}$  increases the number of shares rapidly, making such an implementation impractical except for circuits with very shallow logic depth.

*The Refreshing Layer  $\mathcal{R}$ .* This layer is applied in order to re-mask the output of  $\mathcal{L}$ . It is shown in several prior works on first-order TI that this layer can be avoided if the output of  $\mathcal{L}$  already satisfies uniformity. However,  $\mathcal{R}$  is critical in order to provide higher-order security as will be discussed in Sect. 4.1. In ISW, each output of each masked AND gate is refreshed, which clearly increases the randomness requirements compared to the generalized sharing of a more complex function with several terms (such as  $d = a \oplus bc \oplus abc$ ).

*The Compression Layer  $\mathcal{C}$ .* While designing the  $\mathcal{L}$  layer, there is a natural tension between two desirable properties, namely satisfying  $d^{\text{th}}$ -order non-completeness and having a small number of output shares. Normally, one designs  $\mathcal{L}$  to have a small number of output shares yet satisfying  $d^{\text{th}}$ -order non-completeness. One example of this issue is the second-order masking of an AND-gate depicted in Fig. 11, where the number of shares at the output of  $\mathcal{L}$  (10 shares) is considerable larger than the number of shares of each input variable (5-share  $\mathbf{a}$  and  $\mathbf{b}$ ).

If it is desired to decrease the number of output shares further, the compression layer  $\mathcal{C}$  is applied. This layer is composed of XOR gates only. In order to satisfy non-completeness and avoid glitches causing leakage of more than the intended number of shares, it is crucial to isolate the  $\mathcal{R}$  and  $\mathcal{C}$  layers using registers.

Note that in typical TIs, the layers  $\mathcal{N}$  and  $\mathcal{L}$  are combined and absorbed into the component functions without registers between these layers as drawn in Fig. 11. An additional challenge is to design  $\mathcal{L}$  so that it simultaneously satisfies non-completeness and uniformity.

### 3.4 Security Arguments for Generalized Scheme

In this section, we argue the security of the generalized structure. We start by showing the security of a 2-input AND gate (2AND) against a  $d$ -probing adversary and inductively continue to a function of degree  $t$ . We assume that inputs to  $\mathcal{N}$  are uniformly shared and synchronous. This discussion enables us to relate the number of required shares in TI with that in ISW.

*2-input AND gate.* Let us consider a set of information  $I$  (based on indices) gathered by the attacker by probing  $d$  wires. Specifically, if a wire corresponding to  $a_i, b_i$  or  $a_i b_i$  is probed, the index  $i \in I$ . If the wire corresponding to  $a_i b_j$  is probed, both  $i, j \in I$ . This implies that a probed wire at the output of the layer  $\mathcal{N}$  can give information about at most two indices. Therefore, the cardinality



of  $I$  is at most  $2d$  when  $d$  wires are probed in  $\mathcal{N}$ . It follows that using at least  $2d+1$  shares is required to provide security up to  $\mathcal{L}$ . However, an attacker is not limited to probing certain layers. Notice that the attacker probing closer to the end of the calculation of the component functions, i.e. just before the register between  $\mathcal{R}$  and  $\mathcal{C}$ , gains more information. By the definition of the linear layer  $\mathcal{L}$ , the component functions should be formed such that any combination of up to  $d$  of them should be independent of at least one share, i.e. one index, when a  $d$ -probing secure circuit is considered. Hence, we know that if it is possible to construct  $\mathcal{L}$  with the given restriction, the attacker probing  $d$  wires never has all the indexes in  $I$ . Since the input shares are uniformly shared and the vectors  $\mathbf{a}_i, \mathbf{b}_i, \dots$  are independent from the unshared values  $a, b, \dots$ , we achieve security at the end of  $\mathcal{L}$ . Moreover, knowing the randomness used in  $\mathcal{R}$  does not yield additional information to the attacker. At this point the possibility of generating  $\mathcal{L}$  with  $2d+1$  shares becomes the question. It has been shown in [3] with a combinatorial argument that this is possible if the linear layer  $\mathcal{L}$  is divided into  $\binom{2d+1}{2}$  component functions. Namely, each component function uses at most two input shares and hence at most two indices are put in  $I$  for each probing. This gives the cardinality of at most  $2d$  when  $d$  probes are used.

Notice that the security discussion provided so far considers only one AND gate. However, the security of the generalized scheme also holds for the composition of several AND gates. Namely, the refreshing layer  $\mathcal{R}$  and the register afterwards impose independence of the composed elements and non-completeness respectively. Hence, the union of the gathered information does not give an additional advantage to the attacker.

In the case of a single-probe adversary, we can relax the requirements on  $\mathcal{R}$ . As long as the next nonlinear function sees uniformly shared inputs, one can simplify the construction of  $\mathcal{R}$  and even in some cases avoid  $\mathcal{R}$ . This result follows from the fact that an attacker using a single probe is unable to combine information from more than one function.

*3-input AND gate.* The security argument for a 3-input AND gate ( $F(a, b, c) = abc$ ) follows the same lines as for the 2-input AND gate. The nonlinear layer  $\mathcal{N}$  calculates  $a_i b_j c_k$  terms. In order to keep the number of shares small, we need to make sure that each component function uses variables with at most 3 different indices. Then, an attacker probing  $d$  wires can only gather information from at most  $3d$  indices. The question if it is possible to arrange  $\mathcal{L}$  such that this restriction is respected is answered positively in [3]. It can be done by dividing  $\mathcal{L}$  into  $\binom{3d+1}{3}$  component functions. Note that for a full proof of security, the insertion of randomness (the  $\mathcal{R}$  layer) and registers become critical in order to provide higher-order security of the composition of such gates.

Naturally, it is also possible to generate a shared 3AND gate by composing two shared 2AND gates. This requires usage of registers after both the first and the second 2AND-gate calculation. However, the construction described above which performs the 3AND gate calculation at once is typically more efficient.

*t*-input AND gate. We can inductively apply the arguments for 2AND and 3AND gates to the *t*-input AND gate. This implies the sufficient lower bound of  $s_{in} = td + 1$  input shares. The shared function should be split into at least  $\binom{s_{in}}{t}$  component functions in  $\mathcal{L}$  and satisfy  $d^{\text{th}}$ -order non-completeness.

*Degree  $t$  Boolean Functions.* The above inductive argument does not exclude functions composed of more than one degree  $t$  term. To clarify, the generation of  $\mathcal{N}$  is performed by straight-forward calculation of all cross products using  $s_{in} = td + 1$  shares. The linear layer is split into  $\binom{s_{in}}{t}$  component functions, each of which sees  $t$  indices as described above for a *t*-input AND gate. Any shared term of degree  $\leq t$  can be placed to at least one existing component function since any cross product of the shared  $\leq t$  term uses at most  $t$  indices, which concludes the argument.

*Degree  $t$  Functions in Other Finite Fields.* A careful investigation of the above arguments shows that they are independent of the used field. Namely, it is enough to replace the AND gates in GF(2) with multiplication in the required field in order to provide security for a degree  $t$  function.

We conclude the security argument of the generalized masking scheme by noting that  $s_{in}$  can be chosen to be greater than  $td + 1$  in order to achieve flexibility without invalidating the security arguments.

### 3.5 Wrapping up

In this section, we provided a generalized scheme which extends ISW and TI like masking schemes. Specifically, unlike the ISW scheme which builds up on AND gates or field multiplications; the generalized scheme allows to implement any function directly, enabling the usage of less compositions. The generalized scheme inherits the ability to operate on larger fields and security against  $d$ -probing adversary. In addition, it offers protection for composition of gates.

## 4 What can Go Wrong?

In Sect. 3 we constructed a generalized masking structure, and assigned precise requirements and functions to each of its layer. In this section, we show how small deviations from this generalized scheme can cause vulnerable implementations. In particular, in Sect. 4.1 we analyze the cost of lacking a refreshing layer  $\mathcal{R}$ . We use the recently proposed higher-order TI as our case study to show a higher-order flaw, then we suggest a generic fix. In Sect. 4.2, we elaborate on the insecurity that deviating from the structure especially on  $\mathcal{L}$  brings in the presence of glitches using the ISW and Trichina scheme as our case study.

### 4.1 Higher-Order TI is Not so Higher-Order Secure

The higher-order TI proposed in [3] fits to our generalized structure as follows.  $\mathcal{N}$  and  $\mathcal{L}$  together enforce a correct and  $d^{\text{th}}$ -order non-complete implementation.

However, unlike the generalized scheme, the refreshing layer  $\mathcal{R}$  is not performed in TI when the uniformity of the shared output of  $\mathcal{C}$  can be satisfied without  $\mathcal{R}$ . This difference becomes important since as we shall see in the sequel, it can induce a higher-order security flaw when composing several sharings, even if these sharings are uniform.

For simplicity, we use a second-order TI of a *mini-cipher* construction and show a second-order leakage.

*Description of the Mini-Cipher.* Let us consider a minimal non-linear feedback shift register. This mini-cipher comes from an extreme simplification of the KATAN block cipher for which a higher-order TI was given in [3]. We consider a 4-bit state  $S[i]$ ,  $i \in 0, \dots, 3$  for which the taps are at the state bits with indices  $i = 1, 2, 3$  and the feedback is plugged at position 0. This state is a “sensitive variable<sup>4</sup>”. The feedback function (“round function” of an extremely unbalanced Feistel)  $F = F(S[3], S[2], S[1])$  is the same AND-XOR feedback function as in KATAN, namely  $d = F(a, b, c) = ab \oplus c$ .

*Shared Version of the Mini-Cipher.* The shared version of this mini-cipher (non-complete sharing in the  $\mathcal{N}$  and  $\mathcal{L}$ ) follows the lines of [3]. The feedback function  $F$  is shared as Eq. (5). In particular, to provide security against glitches, the state bit  $\mathbf{S}[0]$ , in which the output of  $\mathbf{F}$  is stored, is composed of 10 shares, whereas  $\mathbf{S}[1]$ ,  $\mathbf{S}[2]$ ,  $\mathbf{S}[3]$  are composed of 5 shares. The conversion from 10 to 5 shares is done as suggested in [3]. That is, the fifth share of  $\mathbf{S}[1]$  sees the xor of the last six shares of  $\mathbf{S}[0]$  when the cipher is clocked (Fig. 12).

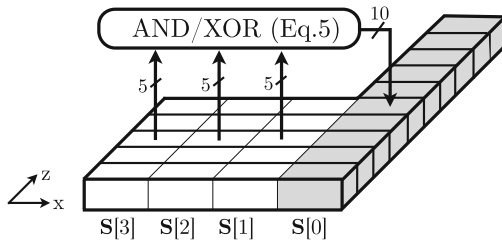


Fig. 12. Diagram of the shared version of the mini-cipher.

*A Second-Order Leakage.* Some lengthy, albeit straightforward, calculations show that the covariance (second mixed statistical moment) between the fifth share of  $\mathbf{S}[1]$  after the first cycle and the fourth share of  $\mathbf{S}[1]$  after the seventh cycle depends on the unshared initial value  $S[2]$ . Thus, there is a second-order flaw that invalidates the security claims of the scheme.<sup>5</sup>

<sup>4</sup> The goal is to show leakage in this construction. To simplify and keep the essentials, we do not explicitly inject the key in this mini-cipher construction, but assume that the initial state is secret (sensitive).

<sup>5</sup> This result had been previously reported in [23] and experimentally confirmed in [26].

*Mitigation.* The direct mitigation is to refresh the shares after each shared function computation, for example by adding fresh shares of the null vector. In other words, the refreshing layer  $\mathcal{R}$  should be implemented in TI when higher-order security is considered. The idea here is to isolate the intermediates occurring within each computation stage from intermediates of another stage, so that combining intermediates from different stages no longer reveals information about a secret unshared value. With this argument, we fix the flaw in [3] using the conciliation of masking schemes.

Note that this fix naturally increases area and randomness requirements and we do not claim that it is the optimal solution. We foresee that this solution may be an overkill in many situations, and a careful analysis can save significant amount of randomness. This is especially true since the existence of a second-order dependency between two variables does not necessarily imply an easy key-extraction by DPA. In particular, if there is a second-order flaw between two intermediates for which there is enough key-diffusion between them, key recovery exploiting the joint leakage of intermediates becomes difficult. The exact minimum amount of  $\mathcal{R}$  layers needed to make the whole implementation secure against higher-order attacks may depend from design to design.

## 4.2 ISW and Trichina in the Presence of Glitches

ISW scheme implicitly considers a logic gate that does not glitch. Thus, a straightforward translation of ISW into standard CMOS technology can result in a vulnerable implementation. To see this, observe that in Eq. (8)  $c_3$  breaks the non-completeness property:

$$\begin{aligned} c_1 &= a_1b_1 \oplus z_{12} \oplus z_{13}, \\ c_2 &= a_2b_2 \oplus ((z_{12} \oplus a_1b_2) \oplus a_2b_1) \oplus z_{23}, \\ c_3 &= a_3b_3 \oplus ((z_{13} \oplus a_1b_3) \oplus a_3b_1) \oplus (z_{23} \oplus a_2b_3) \oplus a_3b_2. \end{aligned}$$

The trivial fix here is to register signals that otherwise could result in undesired (and pernicious) glitches. More precisely, if during the ISW computation in logic, the intermediate values  $z_{ji}$  where  $i < j$  (outputs in Eq. (1), left column; and Eq. (2)) are stored in registers together with the intermediate values  $a_i b_i$  before further XOR combinations, the circuit becomes secure even if there are glitches. This follows since non-completeness holds between register stages. The caveat of this fix is the significant increase in area (and latency) due to extra registers. Note that this extra layer of registers is prevented by careful selection of the layer  $\mathcal{L}$ .

Similar observations apply to the Trichina construction. Trichina also imposes restrictions on the logic gates, especially on the order of evaluation of these gates. It is implicitly assumed that signals are registered or latched in order to avoid glitches. The case where first-order security fails due to glitches is studied in [18].

## 5 Applications

Here we introduce two additional constructive applications. In the first one, we focus on optimizing the generalized scheme further such that it uses less input shares per function. The second application analyses software-like implementations in a distance-based leakage model.

### 5.1 Using $d + 1$ Input Shares

As described in Sect. 3, the generalized scheme uses at least  $td + 1$  input shares to protect a function with degree  $t$  against  $d$ -probing attacks. Here, we improve the scheme such that it uses less input shares, specifically,  $d + 1$  shares to achieve  $d$ -probing security. As a trade-off, however, this sharings are more restrictive with the requirements of *independent* input sharings. We illustrate with single-probe secure examples the design process of such sharings and the construction of layers. We provide a security argument and discuss connections with prior works.

*First Crack.* We start with the first-order sharing of  $c = ab$  with  $s_{in} = 2$  and  $s_{out} = 4$  given in Eq. (8). The sharing  $\mathbf{c}$  is only composed of the crossproducts  $a_i b_j$ . Hence, it can be seen as the output of  $\mathcal{N}$  which is already a correct sharing for  $c$ . Moreover, if the sharing of  $a$  is independent than that of  $b$  then each share  $c_i$  is independent of at least one input share of each variable. In other words, non-completeness is satisfied. This implies the independence of  $\mathbf{c}$  from the unmasked variables  $a$  and  $b$  providing security under a single probe.

$$c_1 = a_1 b_1, \quad c_2 = a_1 b_2, \quad c_3 = a_2 b_1, \quad c_4 = a_2 b_2. \tag{8}$$

Note that in this simple sharing, if the sharings of  $a$  and  $b$  were dependent (for example,  $\mathbf{a} = \mathbf{b}$ ), then the second output share  $a_1 b_2 = a_1 a_2$  would depend on all shares of  $a$  (breaking non-completeness) and this would clearly leak information about  $a$ . During the construction of layers in the following, we assume that the sharings of each input variable is independent from all others.

*Construction of  $\mathcal{N}$  and  $\mathcal{L}$ .* The increase of number of variables in the input increases the number of cross products and hence the number of output shares of  $\mathcal{N}$  exponentially. For example, if we consider the sharing of  $d = a \oplus ac \oplus bc$  with  $s_{in} = 2$ , we end up with 10 terms ( $a_1, a_2, a_1 c_1, a_1 c_2, a_2 c_1, a_2 c_2, b_1 c_1, b_1 c_2, b_2 c_1, b_2 c_2$ ) in  $\mathcal{N}$ . Notice that it is possible to reduce the number of output shares using a careful selection of a linear layer  $\mathcal{L}$  as shown in Eq. (9) while satisfying the non-completeness property.

$$\begin{aligned} d_1 &= a_1 \oplus a_1 c_1 \oplus b_1 c_1, & d_3 &= a_2 \oplus a_2 c_1 \oplus b_2 c_1, \\ d_2 &= a_1 c_2 \oplus b_1 c_2, & d_4 &= a_2 c_2 \oplus b_2 c_2. \end{aligned} \tag{9}$$

The number of output shares of  $\mathcal{L}$  also changes significantly depending on the function itself in addition to the number of input shares. To clarify, let us

consider the sharing of  $d = a \oplus ac \oplus bc \oplus ab$  which differs from the prior unshared function in the additional term  $ab$ . The terms  $(a_1b_1, a_1b_2, a_2b_1, a_2b_2)$  should be added to Eq. (9) for a correct implementation. Even if we place the additional terms  $a_1b_1$  and  $a_2b_2$  to the first and the last component functions in Eq. (9) respectively, the remaining terms  $a_1b_2$  and  $a_2b_1$  can not be placed in these four component functions without breaking the non-completeness property. Hence, we need to increase the number of shares. One option to obtain non-completeness is increasing the number of output shares of  $\mathcal{L}$  as shown in the equation below.

$$\begin{aligned}
 d_1 &= a_1 \oplus a_1c_1 \oplus b_1c_1 \oplus a_1b_1, & d_4 &= a_2c_2 \oplus b_2c_2 \oplus a_2b_2, \\
 d_2 &= a_1c_2 \oplus b_1c_2, & d_5 &= a_1b_2, \\
 d_3 &= a_2 \oplus a_2c_1 \oplus b_2c_1, & d_6 &= a_2b_1.
 \end{aligned}
 \tag{10}$$

*Construction of  $\mathcal{R}$  and  $\mathcal{C}$ .* It is clear that if  $n \times s_{in} < m \times s_{out}$ , the output sharing can not be uniform. Even if  $n \times s_{in} \geq m \times s_{out}$  the uniformity is not guaranteed. The output sharing described in Eqs. (8), (9) and (10) are such non-uniform examples which require refreshing ( $\mathcal{R}$  layer). An alternative approach for the first-order case only is to decrease the number of shares in order to achieve uniformity after storing the output of the mentioned sharings in registers (prior to the  $\mathcal{C}$  layer). To exemplify, consider the following sharing of  $d = ab \oplus c$  with 2 input shares of each variable.

$$d_1 = a_1b_1 \oplus c_1, \quad d_2 = a_1b_2, \quad d_3 = a_2b_1 \oplus c_2, \quad d_4 = a_2b_2. \tag{11}$$

The sharing  $\mathbf{d}$  is a nonuniform 4-sharing. However, the 2-sharing  $\mathbf{e}$  generated by  $e_1 = d_1 \oplus d_2$  and  $e_2 = d_3 \oplus d_4$  is uniform. Moreover, if the sharing  $\mathbf{d}$  is stored in registers before decreasing the number of shares, as implied by the registers between the  $\mathcal{R}$  and  $\mathcal{C}$  layers in the generalized construction, any glitch during the calculation of  $e_i$  does not reveal information about the input values. Note that the selection of the xored terms is not random at all and must be performed with extreme care. A bad choice for a compression layer would be  $e_1 = d_1 \oplus d_3$  and  $e_2 = d_2 \oplus d_4$ , since  $e_2 = (a_1 \oplus a_2)b_2 = ab_2$  obviously reveals information on  $a$ .

*Security Argument of the Improved Bound on the Number of Shares.* It is noteworthy that the security of the improved scheme is not proven using indices as for the generalized scheme described in Sect. 3.4. Instead, since we assume that each input sharing is independent of the others, we ensure that any combination of  $d$  probes miss at least one share of each input variable. Therefore  $d + 1$  input shares are sufficient in order to provide non-completeness in  $\mathcal{N}$  hence independence of the output shares from each unmasked input. As discussed above the requirements that should be satisfied by the  $\mathcal{L}$  and  $\mathcal{C}$  layers in order to provide the claimed security highly depends on the function. Therefore, we avoid to give a generic construction besides imposing  $d^{\text{th}}$ -order non-completeness in  $\mathcal{L}$  and extreme care not to unmask in  $\mathcal{C}$ . The extension to higher orders is straightforward under given exceptions.

*Application to 4-bit Quadratic Permutations.* Due to space restrictions, this section is not present in this version. The interested reader can find it in the extended version<sup>6</sup> of this paper.

*Connections with Software ISW.* In [25], a fast masked AES at any order is given. The authors improve the security guarantees with respect to the number of shares from  $s = 2d + 1$  to  $s = d + 1$ . This improvement actually resembles to the contribution of this section. The improvement was later shown to be slightly flawed by [10]. However, we observe here that the refreshing from [25] is not exactly the same as the layer  $\mathcal{R}$  presented in Sect. 3.3 (operating in  $\text{GF}(2^8)$ ). Namely, the refreshing from [25] uses 1 unit of randomness (elements in  $\text{GF}(2^8)$ ) less than  $\mathcal{R}$ . Using a refreshing that mimics the layer  $\mathcal{R}$  makes the second-order flaw disappear [2].

## 5.2 Resistance Against Distance Leakage

There are many applications of the ISW scheme for masked software implementations [13, 25]. In the case of AES, there is a significant performance gain if the ISW operates in  $\text{GF}(2^8)$ , due to the algebraic structure of the AES Sbox. A common problem with ISW-based software implementations is the mismatch between the probing model in which ISW is proven secure and the leakage behavior of the device that runs the implementation. For instance, typical processors can be approximately modeled by a distance-based leakage behavior (Hamming distance) rather than value-based one (Hamming weight). Thus, a straightforward implementation of ISW without a careful prior profiling of the device leakage behavior will likely lead to an insecure implementation. This is because, even if ISW is secure in weight-based leakage behavior, it is not in a distance-based one.

There are already some theoretical solutions for this problem, although they come with a great cost [1, 9]. We point out here that it is possible to minimize the exposure to this issue with the same modification performed in Sect. 3, i.e. bringing the non-completeness condition.

The generalized scheme (e.g. after the second modification in Fig. 7) computes sequentially each branch (component function)  $F_i, i = 1, 2, 3$  and then performs a refreshing. This scheme is secure even if during the computation of each branch  $F_i$  the device leaks distances (or a more complex leakage function of several values). Contrary to the ISW, we do not impose specific constraints on the order of evaluation of intermediates (within each  $F_i$ ). This result immediately follows from non-completeness of each branch  $F_i$ . It is required, however, to make sure that there is no distance leakage between an intermediate appearing in  $F_i$  and another in  $F_j$ , for  $i \neq j$ . It is noteworthy that the randomness requirement, running time and memory requirements stay the same as in the original algorithm.

<sup>6</sup> <http://www.reparaz.net/oscar/crypto2015/>.

## 6 Conclusion

In this paper, we explored the connections, similitudes and differences between several masking schemes, both from theoretical domains and from practitioners working under real-world constraints. It is remarkable how two substantially disparate communities arrive to essentially similar designs. This perhaps builds even more confidence on the underlying techniques.

There are certainly many future avenues of research. For example, it would be desirable to have explicit and tight bounds on the randomness requirements to achieve efficient masked implementations.

**Acknowledgement.** The authors would like to thank the CRYPTO 2015 reviewers for their valuable comments, as well as Fre Vercauteren and Vincent Rijmen for stimulating discussions. This work has been supported in part by the Research Council of KU Leuven (OT/13/071 and GOA/11/007), by the FWO G.0550.12N and by the Hercules foundation (AKUL/11/19). Oscar Reparaz is funded by a PhD fellowship of the Fund for Scientific Research - Flanders (FWO). Benedikt Gierlichs is a Postdoctoral Fellow of the Fund for Scientific Research - Flanders (FWO). Begül Bilgin is partially supported by the FWO project G0B4213N.

## References

1. Balasch, J., Gierlichs, B., Grosso, V., Reparaz, O., Standaert, F.-X.: On the cost of lazy engineering for masked software implementations. In: Joye, M., Moradi, A. (eds.) CARDIS 2014. LNCS, vol. 8968, pp. 64–81. Springer, Heidelberg (2015)
2. Barthe, G., Belaïd, S., Dupressoir, F., Fouque, P.-A., Grégoire, B., Strub, P.-Y.: Verified proofs of higher-order masking. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 457–485. Springer, Heidelberg (2015)
3. Bilgin, B., Gierlichs, B., Nikova, S., Nikov, V., Rijmen, V.: Higher-order threshold implementations. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014, Part II. LNCS, vol. 8874, pp. 326–343. Springer, Heidelberg (2014)
4. Bilgin, B., Gierlichs, B., Nikova, S., Nikov, V., Rijmen, V.: A more efficient AES threshold implementation. In: Pointcheval, D., Vergnaud, D. (eds.) AFRICACRYPT. LNCS, vol. 8469, pp. 267–284. Springer, Heidelberg (2014)
5. Bilgin, B., Nikova, S., Nikov, V., Rijmen, V., Stütz, G.: Threshold implementations of all  $3 \times 3$  and  $4 \times 4$  S-boxes. In: Prouff, E., Schaumont, P. (eds.) CHES 2012. LNCS, vol. 7428, pp. 76–91. Springer, Heidelberg (2012)
6. Bilgin, B., Nikova, S., Nikov, V., Rijmen, V., Tokareva, N., Vitkup, V.: Threshold implementations of small S-boxes. *Crypt. Commun.* **7**(1), 3–33 (2015)
7. Chari, S., Jutla, C.S., Rao, J.R., Rohatgi, P.: Towards sound approaches to counteract power-analysis attacks. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 398–412. Springer, Heidelberg (1999)
8. Coron, J.-S.: Higher order masking of look-up tables. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 441–458. Springer, Heidelberg (2014)
9. Coron, J.-S., Giraud, C., Prouff, E., Renner, S., Rivain, M., Vadnala, P.K.: Conversion of security proofs from one leakage model to another: a new issue. In: Schindler, W., Huss, S.A. (eds.) COSADE 2012. LNCS, vol. 7275, pp. 69–81. Springer, Heidelberg (2012)



10. Coron, J.-S., Prouff, E., Rivain, M., Roche, T.: Higher-order side channel security and mask refreshing. In: Moriai, S. (ed.) FSE 2013. LNCS, vol. 8424, pp. 410–424. Springer, Heidelberg (2014)
11. Duc, A., Dziembowski, S., Faust, S.: Unifying leakage models: from probing attacks to noisy leakage. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 423–440. Springer, Heidelberg (2014)
12. Goubin, L., Patarin, J.: DES and differential power analysis the duplication method. In: Koç, Ç.K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, p. 158. Springer, Heidelberg (1999)
13. Grosso, V., Leurent, G., Standaert, F.-X., Varıcı, K.: LS-designs: bitslice encryption for efficient masked software implementations. In: Cid, C., Rechberger, C. (eds.) FSE 2014. LNCS, vol. 8540, pp. 18–37. Springer, Heidelberg (2015)
14. 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)
15. 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)
16. Kutzner, S., Nguyen, P.H., Poschmann, A.: Enabling 3-share threshold implementations for all 4-bit S-boxes. In: Lee, H.-S., Han, D.-G. (eds.) ICISC 2013. LNCS, vol. 8565, pp. 91–108. Springer, Heidelberg (2014)
17. Mangard, S., Popp, T., Gammel, B.M.: Side-channel leakage of masked CMOS gates. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 351–365. Springer, Heidelberg (2005)
18. Mangard, S., Schramm, K.: Pinpointing the side-channel leakage of masked AES hardware implementations. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 76–90. Springer, Heidelberg (2006)
19. Messerges, T.S.: Using second-order power analysis to attack DPA resistant software. In: Paar, C., Koç, Ç.K. (eds.) CHES 2000. LNCS, vol. 1965, pp. 238–251. Springer, Heidelberg (2000)
20. Moradi, A., Poschmann, A., Ling, S., Paar, C., Wang, H.: Pushing the limits: a very compact and a threshold implementation of AES. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 69–88. Springer, Heidelberg (2011)
21. Nikova, S., Rechberger, C., Rijmen, V.: Threshold implementations against side-channel attacks and glitches. In: Ning, P., Qing, S., Li, N. (eds.) ICICS 2006. LNCS, vol. 4307, pp. 529–545. Springer, Heidelberg (2006)
22. Poschmann, A., Moradi, A., Khoo, K., Lim, C.-W., Wang, H., Ling, S.: Side-channel resistant crypto for less than 2,300 GE. *J. Cryptology* **24**(2), 322–345 (2011)
23. Reparaz, O.: A note on the security of higher-order threshold implementations. Cryptology ePrint Archive, Report 2015/001
24. Reparaz, O., Gierlichs, B., Verbauwhede, I.: Selecting Time samples for multivariate DPA attacks. In: Prouff, E., Schaumont, P. (eds.) CHES 2012. LNCS, vol. 7428, pp. 155–174. Springer, Heidelberg (2012)
25. Rivain, M., Prouff, E.: Provably secure higher-order masking of AES. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 413–427. Springer, Heidelberg (2010)
26. Schneider, T., Moradi, A.: Leakage assessment methodology - a clear roadmap for side-channel evaluations. In: CHES. LNCS (2015)

27. Tiri, K., Akmal, M., Verbauwhede, I.: A dynamic and differential CMOS logic with signal independent power consumption to withstand differential power analysis on smart cards. In: ESSCIRC, pp. 403–406 (2002)
28. Tiri, K., Verbauwhede, K.: A logic level design methodology for a secure DPA resistant ASIC or FPGA implementation. In: DATE (2004)
29. Trichina, E.: Combinational logic design for aes subbyte transformation on masked data. Cryptology ePrint Archive, Report 2003/236

## Author Index

- Abdalla, Michel I-388  
Agrawal, Shashank I-538  
Alkhzaimi, Hoda I-95  
Alwen, Joël II-763  
Ananth, Prabhanjan I-308, II-657  
Andrychowicz, Marcin II-379  
Benhamouda, Fabrice I-388, II-107  
Bilgin, Begül I-764  
Biryukov, Alex I-116  
Blondeau, Céline I-455  
Boyle, Elette II-742  
Brakerski, Zvika II-657  
Broadbent, Anne II-609  
Canetti, Ran II-3, II-43  
Carlet, Claude I-742  
Catalano, Dario II-254  
Cheng, Lei I-95  
Chitambar, Eric II-443  
Chung, Kai-Min I-287, II-742  
Clear, Michael II-630  
Cogliati, Benoît I-189  
Cohen, Asaf II-3  
Coron, Jean-Sébastien I-247, I-267  
Couteau, Geoffroy II-107  
Dinur, Itai I-433  
Dodis, Yevgeniy II-463  
Döttling, Nico I-329  
Dov Gordon, S. II-63  
Dunkelman, Orr I-433  
Dziembowski, Stefan II-379, II-585  
Elias, Yara I-63  
Espitau, Thomas I-683  
Faust, Sebastian II-585  
Fehr, Serge II-403  
Fillinger, Max II-403  
Fiore, Dario II-254  
Fischlin, Marc II-545  
Fortescue, Benjamin II-443  
Fouque, Pierre-Alain I-43, I-561, I-683  
Fuchsbauer, Georg I-601, II-233  
Garg, Sanjam II-191  
Gay, Romain II-485  
Gaži, Peter I-368  
Genkin, Daniel II-721  
Gentry, Craig I-247  
Gierlichs, Benedikt I-764  
Gilbert, Henri I-475  
Gorbunov, Sergey II-503  
Goyal, Vipul II-23, II-43  
Gu, Dawu II-209  
Günther, Felix II-545  
Guo, Qian I-23  
Gupta, Divya I-538, II-23, II-701  
Haitner, Iftach II-173  
Hajiabadi, Mohammad I-224  
Halevi, Shai I-247  
Hamburg, Mike I-705  
Hanser, Christian II-233  
Hoang, Viet Tung I-493  
Hsieh, Min-Hsiu II-443  
Hu, Zhangxiang II-150  
Huang, Jialin I-141  
Huang, Ming-Deh A. I-581  
Ishai, Yuval II-173, II-191, II-359, II-701, II-721  
Jafargholi, Zahra I-601  
Jain, Abhishek I-308, II-43  
Jeffery, Stacey II-609  
Johansson, Thomas I-23  
Kalai, Yael Tauman II-422  
Kapron, Bruce M. I-224  
Karpman, Pierre I-623, I-683  
Keller, Marcel I-724  
Keller, Nathan I-433  
Kerenidis, Iordanis II-485  
Kiltz, Eike II-275  
Kirchner, Paul I-43  
Kiyoshima, Susumu II-85  
Kölbl, Stefan I-161  
Kolmogorov, Vladimir II-585  
Kosters, Michiel I-581  
Kowalczyk, Lucas II-524  
Kumaresan, Ranjit II-359  
Kushilevitz, Eyal II-191, II-359

- Laarhoven, Thijs I-3  
 Lai, Xuejia I-141  
 Lallemand, Virginie I-663  
 Lampe, Rodolphe I-189  
 Lauter, Kristin E. I-63  
 Leander, Gregor I-161  
 Lee, Moon Sung I-561  
 Lepoint, Tancrede I-247, I-267, I-561  
 Lewko, Allison Bishop II-524  
 Li, Chao I-95  
 Li, Ruilin I-95  
 Li, Xiangxue II-209  
 Libert, Benoît II-296  
 Lin, Huijia I-287  
 Lindell, Yehuda II-3  
 Liu, Feng-Hao II-63  
 Liu, Zhiqiang I-95  
 Luykx, Atul I-209  
  
 Maji, Hemanta K. I-247, I-538, II-701  
 Mandal, Avradip I-518  
 Marson, Giorgia Azzurra II-545  
 McGoldrick, Ciarán II-630  
 Meier, Willi I-643  
 Miles, Eric I-247  
 Minaud, Brice I-351  
 Mohassel, Payman II-150  
 Mouha, Nicky I-209  
  
 Naor, Moni II-565  
 Naya-Plasencia, María I-663  
 Nikova, Svetla I-764  
 Nizzardo, Luca II-254  
 Nyberg, Kaisa I-141  
  
 Omri, Eran II-173  
 Orsini, Emanuela I-724  
 Ostrovsky, Rafail II-130, II-191, II-339,  
 II-763  
 Ozman, Ekin I-63  
  
 Pan, Jiaxin II-275  
 Pandey, Omkant I-538  
 Paskin-Cherniavsky, Anat II-359  
 Pass, Rafael I-287, II-742  
 Passelègue, Alain I-388  
 Paterson, Kenneth G. II-545  
 Perrin, Léo I-116  
 Persiano, Giuseppe II-130  
 Peters, Thomas II-296  
 Peyrin, Thomas I-455, I-623  
  
 Pietrzak, Krzysztof I-368, I-601, II-585  
 Pinkas, Benny II-319  
 Plût, Jérôme I-475  
 Pointcheval, David II-107  
 Polychroniadou, Antigoni II-721  
 Prabhakaran, Manoj I-538  
 Prouff, Emmanuel I-742  
  
 Raykova, Mariana I-247  
 Reparaz, Oscar I-764  
 Reyhanitabar, Reza I-493  
 Richelson, Silas II-339  
 Rijmen, Vincent I-95  
 Rivain, Matthieu I-742  
 Roche, Thomas I-742  
 Rogaway, Phillip I-493  
 Rosulek, Mike II-150  
 Rothblum, Ron D. II-422  
 Roy, Arnab I-518  
  
 Sahai, Amit I-247, II-23, II-191, II-701  
 Scafuro, Alessandra II-339  
 Scholl, Peter I-724  
 Schröder, Dominique I-329  
 Segev, Gil II-657  
 Seurin, Yannick I-189, I-351  
 Shaltiel, Ronen II-173  
 Shamir, Adi I-433  
 Shi, Elaine II-63  
 Slamanig, Daniel II-233  
 Smart, Nigel P. II-319  
 Stange, Katherine E. I-63  
 Stankovski, Paul I-23  
 Stevens, Marc I-623  
 Sun, Bing I-95  
  
 Tessaro, Stefano I-368  
 Tibouchi, Mehdi I-247, I-267, I-561  
 Tiessen, Tyge I-161  
 Todo, Yosuke I-413  
 Treger, Joana I-475  
  
 Vaikuntanathan, Vinod II-503, II-657  
 Vaudenay, Serge I-141  
 Verbauwheide, Ingrid I-764  
 Visconti, Ivan II-130  
 Vizár, Damian I-493  
  
 Wang, Lei I-455  
 Wang, Qingju I-95  
 Waters, Brent II-678

- Wee, Hoeteck [II-107](#), [II-275](#), [II-485](#), [II-503](#)  
Weng, Jian [II-209](#)  
Xu, Chao [I-643](#)  
Yanai, Avishay [II-319](#)  
Yao, Yanqing [II-463](#)  
Yeo, Sze Ling [I-581](#)  
Yogev, Eylon [II-565](#)  
Yu, Yu [II-209](#)  
Yung, Moti [II-296](#)  
Zhang, Bin [I-643](#)  
Zhou, Hong-Sheng [II-763](#)  
Zikas, Vassilis [II-763](#)